THE UNIVERSITY
of EDINBURGH

EXPERIENCE-DRIVEN OPTIMAL MOTION SYNTHESIS IN COMPLEX AND SHARED ENVIRONMENTS

WOLFGANG XAVER MERKT

Doctor of Philosophy
School of Informatics
University of Edinburgh

2019

Wolfgang Xaver Merkt:
Experience-driven Optimal Motion Synthesis in Complex and Shared Environments
Doctor of Philosophy, 2019

SUPERVISOR:
Prof. Sethu Vijayakumar FRSE

ABSTRACT

Optimal loco-manipulation planning and control for high-dimensional systems based on general, non-linear optimisation allows for the specification of versatile motion subject to complex constraints. However, complex, non-linear system and environment dynamics, switching contacts, and collision avoidance in cluttered environments introduce non-convexity and discontinuity in the optimisation space. This renders finding optimal solutions in complex and changing environments an open and challenging problem in robotics. Global optimisation methods can take a prohibitively long time to converge. Slow convergence makes them unsuitable for live deployment and online re-planning of motion policies in response to changes in the task or environment. Local optimisation techniques, in contrast, converge fast within the basin of attraction of a minimum but may not converge at all without a good initial guess as they can easily get stuck in local minima. Local methods are, therefore, a suitable choice provided we can supply a good initial guess.

If a similarity between problems can be found and exploited, a memory of optimal solutions can be computed and compressed efficiently in an offline computation process. During runtime, we can query this memory to bootstrap motion synthesis by providing a good initial seed to the local optimisation solver. In order to realise such a system, we need to address several connected problems and questions: First, the formulation of the optimisation problem (and its parametrisation to allow solutions to transfer to new scenarios), and related, the type and granularity of user input, along with a strategy for recovery and feedback in case of unexpected changes or failure. Second, a sampling strategy during the database/memory generation that explores the parameter space efficiently without resorting to exhaustive measures—i.e., to balance storage size/memory with online runtime to adapt/repair the initial guess. Third, the question of how to represent the problem and environment to parametrise, compute, store, retrieve, and exploit the memory efficiently during pre-computation and runtime.

One strategy to make the problem computationally tractable is to decompose planning into a series of sequential sub-problems, e.g., contact-before-motion approaches which sequentially perform goal state planning, contact planning, motion planning, and encoding. Here, subsequent stages operate within the null-space of the constraints of the prior problem, such as the contact mode or sequence.

This doctoral thesis follows this line of work. It investigates general optimisation-based formulations for motion synthesis along with a strategy for exploration, encoding, and exploitation of a versatile memory-of-motion for providing an initial guess to optimisation solvers. In particular, we focus on manipulation in complex environments with high-dimensional robot systems such as humanoids and mobile manipulators.

The first part of this thesis focuses on collision-free motion generation to reliably generate motions. We present a general, collision-free inverse kinematics method using a combination of gradient-based local optimisation with random/evolution strategy restarting to achieve high success rates and avoid local minima. We use formulations for discrete collision avoidance and introduce a novel, computationally fast continuous collision avoidance objective based on conservative advancement and harmonic potential fields. Using this, we can synthesise continuous-time collision-free motion plans in the presence of moving obstacles. It further enables to discretise trajectories with fewer waypoints, which in turn considerably reduces the optimisation problem complexity, and thus, time to solve.

The second part focuses on problem representations and exploration. We first introduce an efficient solution encoding for trajectory library-based approaches. This representation, paired with an accompanying exploration strategy for offline pre-computation, permits the application of inexpensive distance metrics during runtime. We demonstrate how our method efficiently re-uses trajectory samples, increases planning success rates, and reduces planning time while being highly memory-efficient. We subsequently present a method to explore the topological features of the solution space using tools from computational homology. This enables us to cluster solutions according to their inherent structure which increases the success of warm-starting for problems with discontinuities and multi-modality.

The third part focuses on real-world deployment in laboratory and field experiments as well as incorporating user input. We present a framework for robust shared autonomy with a focus on continuous scene monitoring for assured safety. This framework further supports interactive adjustment of autonomy levels from fully teleoperated to automatic execution of stored behaviour sequences. Finally, we present sensing and control for the integration and embodiment of the presented methodology in high-dimensional real-world platforms used in laboratory experiments and real-world deployment. We validate our presented methods using hardware experiments on a variety of robot platforms demonstrating generalisation to other robots and environments.

LAY SUMMARY

Designing motions for robots with many joints is a challenging problem, particularly when non-intuitive requirements have to be taken into account. These include, for instance, maintaining balance on a legged platform, minimising energy consumption to increase battery lifetime, and avoiding collisions with itself, bystanders, or objects in the environment. One popular approach to satisfy these requirements while achieving desired characteristics is to apply numerical optimisation. Here, requirements can be given as constraints (e.g., *maintain balance*, *do not collide*), while desired characteristics become objectives or defects to be minimised (e.g., *minimise energy consumption*, *increase robustness to perturbation*). Due to its use in many other fields—mathematics, physics, business operations, and finance to name a few—large-scale numerical optimisation algorithms have become widely available.

They come with two challenges, though: First, they need to find a possible (*feasible*) solution, i.e., one motion where all requirements are fulfilled irrespective of the defects or objectives. Second, to find the optimal value of the objectives, while still satisfying all requirements and not ending the search prematurely. These challenges determine both the success or failure and time to finish the optimisation process. This is a particular issue in robotics as an additional requirement for motion synthesis is to obtain a new motion quickly in response to changes in the problem or environment (e.g., when working collaboratively with a human or when the sensing of the situation is uncertain).

To address this, we propose several contributions for rapidly obtaining motion plans for complex robotic systems in challenging and changing environments in this thesis. These include (1) a generic framework for optimisation-based motion planning, (2) a novel, computationally efficient approach for continuous collision avoidance enabling the use of fewer waypoints while ensuring safe trajectories, (3) a method for encoding and retrieving prior experiences quickly from a trajectory library to initialise optimisation, (4) a topological approach for analysing the underlying structure of a set of trajectories to guide machine learning approaches, and (5) the demonstration of these contributions in an integrated system with input from human operators for increased flexibility and automation.

Optimisation solvers work very similar to trying to reach the lowest point while taking a hike in hilly terrain and having to stay below a certain altitude. First,

finding a path through the landscape that remains below the maximum elevation and maximum slope one can hike may be tricky if one does not have a good idea where to start or access to a map (the *feasibility problem*). Second, one may reach a valley where to all sides the slopes increase: One appears to be at the lowest point. However, this valley may be just one of many—and by far not the one with the lowest elevation (*local minima*). As such, without a good hunch of which path to take, one can easily find the task impossible to solve or to get stuck along the way.

The analogy of the difficult terrain is very similar to the cost landscape on which the optimisation solvers operate. As motion designers formulate and combine requirements and challenges intuitively, they often create challenging landscapes for the optimisation solvers to go through. They are, however, similar to some degree for comparable problems. In this thesis, we leverage this insight and use large amounts of offline exploration to build experience (in this example, a map) to be able to provide a good enough hunch at runtime when presented with a new motion problem.

To this end, we focus on three areas in this thesis: (a) formulations of motion synthesis which take collision avoidance into account, (b) efficient methods for exploring, encoding, and retrieving motion samples to guide optimisation solvers, and (c) deployment on real-world robotic platforms in collaboration with human operator input and guidance.

In the first part, we introduce a general non-linear optimisation-based formulation for planning single configurations for a robot and discuss and compare different ways of including collision avoidance across a large number of optimisation algorithms. We then introduce a novel way of avoiding collision on the interpolation between two robot configurations. We leverage insights from computer graphics (inspired by *when will the bullet hit the wall*) and physics (the electric field of an electrically charged surface) to come up with a fast and efficient way of how to avoid collision with moving obstacles without resorting to having to check lots of intermediate configurations. Our new approach permits to reduce the size of the optimisation problem by requiring fewer waypoints, and as such, speeds up optimisation while providing certainty on being collision-free.

In the second part, we focus on how to formulate problems such that we can efficiently explore examples of movements, store them, and learn from this experience. We first introduce an efficient data structure that makes use of every small element within a sample motion and also allows for figuring out how to enlarge the area for which this example is a good initialisation for the optimiser. We demonstrate

this on a humanoid robot performing pick-and-place from a shelf: Here, every problem is slightly different (with different locations on the bookshelf)—but the motion required to not bump into the shelf itself and to maintain balance largely the same. Subsequently, we focus on exploring the underlying structure of the samples obtained during exploration. In many cases, there is only a limited number of possibilities for solving a task: For instance, one can walk around a lake using a path on the left or the right-hand side. During exploration, we are likely to find many trails that go on either side—however, to provide a good initialisation for the optimiser it is enough to be able to show a sample each using the left and right paths. We use tools from computational topology (which studies the underlying structure of and relationships within spaces) to separate the samples into distinct ways of solving a task, and then apply tools from machine learning to each. This has shown to work better than blindly trying to learn without paying attention to structure (which, in the above example, would suggest a path through the water—clearly violating the feasibility requirement of walking on firm ground). These insights also allow us to vastly reduce the memory size required to store examples in a trajectory library as one sample of each class is usually enough.

In the third part, we focus on incorporating the developed algorithms in full systems with perception (sensing of the environment) and control algorithms for deployment. First, we extend on a concept allowing sliding control between teleoperation where a human operator controls every movement of a robot to full autonomy where the robot takes all decisions by itself (*shared autonomy*). We add a method that continuously monitors changes in the environment and decides whether it is safe to continue with the planned motion, whether it needs to be interrupted, and whether it can be restarted or replanned automatically after the change has cleared. Finally, we combine the insights into a system for quickly deploying new applications that combine forward motion of a mobile manipulator together with robot arm motion to carry out actions. This makes it possible to manipulate objects without having to stop leading to further efficiency increases. Similarly, we can now carry out tasks that require a large workspace to be covered. Finally, this capability also enables the rapid transfer of motions developed in simulation on a computer to real-world deployment.

DECLARATION

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Edinburgh, 2019

Wolfgang Xaver Merkt
April 15, 2020

PUBLICATIONS

Some ideas and figures have previously appeared in the following publications:

Ferrolho, H., Merkt, Wolfgang, Yang, Y., Ivan, V., and Vijayakumar, S. (2018). Whole-Body End-Pose Planning for Legged Robots on Inclined Support Surfaces in Complex Environments. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 944–951.

Ivan, V., Yang, Y., Merkt, Wolfgang, Camilleri, M. P., and Vijayakumar, S. (2019). EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control. In Koubaa, A., editor, *Robot Operating System (ROS): The Complete Reference (Volume 3)*, pages 211–240. Springer International Publishing, Cham.

Mastalli, C., Budhiraja, R., Merkt, Wolfgang, Saurel, G., Hammoud, B., Naveau, M., Carpentier, J., Righetti, L., Vijayakumar, S., and Mansard, N. (2020). Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*.

Mower, C. E., Merkt, Wolfgang, and Vijayakumar, S. (2019). Comparing Alternate Modes of Teleoperation for Constrained Tasks. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 1497–1504.

Merkt, Wolfgang, Ivan, V., and Vijayakumar, S. (2018). Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5877–5884.

Merkt, Wolfgang, Ivan, V., and Vijayakumar, S. (2019a). Continuous-Time Collision Avoidance for Trajectory Optimization in Dynamic Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7248–7255.

Merkt, Wolfgang, Ivan, V., Yang, Y., and Vijayakumar, S. (2019b). Towards Shared Autonomy Applications using Whole-body Control Formulations of Locomanip-

ulation. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 1206–1211.

Merkt, Wolfgang, Yang, Y., Stouraitis, T., Mower, C. E., Fallon, M., and Vijayakumar, S. (2017). Robust Shared Autonomy for Mobile Manipulation with Continuous Scene Monitoring. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 130–137.

Yang, C., Yuan, K., Merkt, Wolfgang, Komura, T., Vijayakumar, S., and Li, Z. (2018a). Learning Whole-Body Motor Skills for Humanoids. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 270–276.

Yang, Y., Ivan, V., Merkt, Wolfgang, and Vijayakumar, S. (2016). Scaling sampling-based motion planning to humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1448–1454.

Yang, Y., Merkt, Wolfgang, Ferrolho, H., Ivan, V., and Vijayakumar, S. (2017). Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps. *IEEE Robotics and Automation Letters*, 2(4):2279–2286.

Yang, Y., Merkt, Wolfgang, Ivan, V., Li, Z., and Vijayakumar, S. (2018b). HDRM: A Resolution Complete Dynamic Roadmap for Real-Time Motion Planning in Complex Scenes. *IEEE Robotics and Automation Letters*, 3(1):551–558.

Yang, Y., Merkt, Wolfgang, Ivan, V., and Vijayakumar, S. (2018c). Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1–9.

ACKNOWLEDGEMENTS

The research conducted over the past few years leading up to this thesis would not have been possible without the thoughts, advice, and support of many.

First and foremost, I am extremely thankful to my advisor, Professor Sethu Vijayakumar, for his encouragement and advice throughout my research endeavours, and for creating an environment that fosters the freedom to explore, learn, and grow. I am very grateful for our insightful and invigorating discussions, and his support and excellent guidance throughout the years as well as my bachelor's, master's, and doctoral degrees.

I wish to express my gratitude to my examiners—Professor Tamim Asfour, Dr Steve Tonneau, and Dr Mustapha Suphi Erden—and committee chair, Professor Robert Fisher, for their insightful feedback which exposed further application areas and helped to improve this thesis. I am also grateful to the members of my annual review committee—Dr Michael Mistry and Professor Barbara Webb—for their valuable feedback and encouragement over the years.

I thank my research group and co-authors for their curious questions, refreshing conversations, productive collaboration, and the many light-hearted moments. Hat tip to Traiko Dinev and Henrique Ferrolho for being awesome mentees.

I am also thankful for the research collaborations and projects that I had the opportunity to be involved in and from which I took away invaluable skills. These include the joint work with the NASA Johnson Space Center, the Florida Institute for Human and Machine Cognition, and the Massachusetts Institute of Technology on the Valkyrie humanoid platform, and with the partners of the European Union Horizon 2020 project Memory-of-Motion. Among many others, I especially thank Dr Nicolas Mansard and Dr Justin Carpentier for the eye-opening research discussions and from whom I had the pleasure of learning a great deal.

I am indebted to my friends for making the past few years a wonderful experience and playing a vital role in striking a balance outside the beautiful world of robotics. I particularly wish to recognise the beautiful friendships formed through the GSP14 and ExO communities, as well as Henrique Ferrolho, Dr Vladimir Ivan, Laura Landau, and Xinnuo Xu. A special mention and thanks to Dr Rui Li and Marissa Putri—thank you for enriching my life with countless wonderful memories.

Finally, I am grateful to my family for their everlasting unwavering love, support, and encouragement at every step of the way.

To my family—past, present, and future.

CONTENTS

Part I

PRELIMINARIES

1

INTRODUCTION

Automated machinery and robotic systems have become more pervasive as we strive both for higher labour productivity as well as to reduce human exposure to repetitive and strenuous tasks and hazardous environments. Over the past decade, predominantly driven by commoditised component cost, standardisation, as well as an open-source robotics movement, many new applications have materialised. One hallmark is the emergence of *collaborative robots* that interact safely with people without the requirement to be physically separated by security equipment. Collaborative robots at the same time have ushered in a paradigm where non-expert end-users are empowered to set-up and reconfigure the robots for new tasks. Human and robots working side-by-side and hand-in-robot-hand is in contrast to the dawn of industrial robots where the motion paths were programmed for speed and throughput by an expert during assembly line commissioning.

Some tasks, however, can only be achieved by repositioning the robot—for instance, to have a more extensive range of motion in parts of the workspace that are relevant to the execution of the task. Mobile robots using wheels or tracks can navigate built and outdoor environments at high speeds. However, they lack the versatility of humans and animals to climb varied and irregular terrain by using limbs and their body to make and break contact. In robotics research, we call this *whole-body multi-contact locomotion*.

Legged robots

Legged robots have a morphology that resembles humans and animals. They use actuated limbs to traverse terrain and have the ability to operate both in environments built for people as well as in challenging terrain. While initially most of an academic (and cinematic) interest, significant progress has recently been demonstrated. Impressive—and often acrobatic—advancements in the field of legged robots by companies such as Boston Dynamics and ANYbotics have captured the attention and imagination of many.

Degrees-of-freedom

A large number of actuated motors power this extreme versatility: A quadruped commonly uses 12 and humanoids over 30 motors. Each motor represents a degrees of freedom (DoF) that we can control. Legged robots need to maintain balance and might have further un-actuated DoF from their placement in the world, i.e., they are *under-actuated*. We can control these DoFs through interaction with the environment

in the form of forces at the contacts. Designing motions for robots with this many DoF—a quadruped commonly has 18 DoF and a humanoid 30–40—is a challenge as the size of the configuration space grows exponentially with every degree. This is referred to as the *curse-of-dimensionality* and highlights the need for efficient methods for planning and control.

Curse-of-dimensionality

Exploratory, or sampling-based, motion planning is focused on searching the space, for instance, through random samples. Seminal algorithms include the Probabilistic Roadmap (PRM) [Kavraki et al., 1996] which uses search algorithms on a connected graph of configurations and the Rapidly-Exploring Random Tree (RRT) [LaValle, 1998] algorithm which grows a tree structure biased towards the goal. If a final goal configuration is known, we can employ efficient bi-directional variants of these algorithms (e.g., RRT-Connect Kuffner and LaValle [2000]) to find a solution. However, obtaining a valid, collision-free goal configuration satisfying multiple constraints can be challenging. In Chapter 2, we review and compare a general formulation for obtaining valid goal configurations. A challenge with solutions obtained from exploratory planning is that solutions may include artefacts from the random sampling process: Erratic and unnecessary movements. These are not only inefficient but also unexpected to users in a shared workspace. As thus, sampling-based planning methods commonly apply path shortening or smoothing in a post-processing step.

Task representation

In practice, it is unintuitive and difficult to express a task for a high-dimensional system in terms of the configuration, i.e., the set of joint positions. Goals such as reaching a target with a hand and complex constraints such as maintaining balance can be expressed much more readily in the corresponding task-space. Many variants of sampling-based planning algorithms exist (see Elbanhawi and Simic [2014] for a recent review). Some allow to directly plan on the constraint manifold (see Kingston et al. [2018] for an overview), while others use a sample adaptation using an optimisation problem to ensure constraint satisfaction (e.g., [Yang et al., 2016b]). Methods of the former category work well if the constraint can be expressed as an *equality* constraint, i.e., an algebraic expression that evaluates to be precisely equal to a value for the constraint to hold (e.g., an end-effector task to reach an object). To handle *inequality* constraints (e.g., for maintaining balance or a minimum distance), sample adaptation methods in the latter category require numerical optimisation. In either case, sampling-based planners often require heuristics to be carefully designed or tuned. Optimal sampling-based planning

algorithms, e.g., RRT-Star [Karaman and Frazzoli, 2011] and BIT-Star [Gammell et al., 2015], can take a very long time to converge.

Redundancies

Often, the number of DoF is higher than what is required to achieve a particular task, i.e., the robot has redundancies. These redundancies can be exploited in motion planning to satisfy desired optimality criteria and by the controller to achieve robustness against model error and perturbations. We can also motivate this biomimetically: Humans show considerable variability in their movements unless in the vicinity of the movement goal [Tassa, 2011].

Optimisation-based motion planning

When tasks can be formulated using smooth functions, optimisation-based methods can be applied to achieve one or multiple optimality criteria while satisfying complex constraints, see Figure 1.

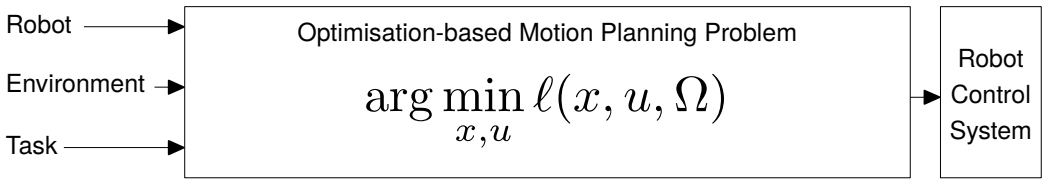


Figure 1: Idealised optimisation-based motion planning pipeline

Where gradients can be derived, efficient first- and second-order methods can be applied (Moré [1978], Liu and Nocedal [1989], Ratliff et al. [2009], Toussaint [2009, 2014]). However, these methods perform *local optimisation* and only optimise within the vicinity of their starting point (initial seed). Providing a good seed is a challenge, and the algorithms may get stuck in *local minima*. A proper initialisation is particularly essential in motion planning as both kinematics and dynamics of robot systems are non-linear, and often also non-convex. An optimiser is said to be *warm-started* if the provided initial seed is the output of a previous optimisation. If one cannot obtain gradients, an option is to apply Monte Carlo methods which use roll-outs (e.g., Theodorou et al. [2010], Kalakrishnan et al. [2011]). While trivially parallelisable, where roll-outs are expensive, they tend to be sample inefficient.

A key difference in motion planning is whether a sequence of joint configurations is optimised (*kinematic trajectory optimisation*) or whether the dynamics of the robot, e.g., how fast it can accelerate, are taken into account (*dynamic trajectory optimisation*). In the latter case, the control sequence—i.e., the motor commands such as current or torque—is optimised. Kinematic trajectory optimisation is fast due to its simplicity but does not guarantee the satisfaction of dynamics constraints, e.g., whether a real robot could achieve the desired motions.

Dynamic trajectory optimisation

Algorithms in dynamic trajectory optimisation generally fall in two families:[1]

INDIRECT TRAJECTORY OPTIMISATION methods optimise the control sequence and obtain the state trajectory (positions and velocities) by performing integration using the nominal model of the robot dynamics. As they thus fold the dynamics into the optimisation, they ensure that the resulting state trajectories are always strictly feasible [Tassa et al., 2014]. Indirect methods are straightforward to formulate; and in the absence of control limits, they benefit from entirely unconstrained optimisation. They are faster (per iteration) and better suited for warm-starting, yet very sensitive to local minima. Additionally, state constraints cannot be incorporated directly.

DIRECT TRAJECTORY OPTIMISATION methods, on the other hand, discretise over both the state and the control trajectories and constrain subsequent states to satisfy the robot dynamics. As a result, one can directly formulate and include state and control constraints with ease. Unlike indirect methods, they discard the temporal structure, and thus sparsity, of the problem (treating dynamics as a constraint), hence requiring search in a constrained space. This necessitates optimisation solvers that can handle constraints explicitly. Direct trajectory optimisation approaches also result in much larger optimisation problems as both states and controls are programme variables. They, however, can find better optima through continuation [Tassa et al., 2014].

Direct trajectory optimisation methods are particularly popular for planning motions for legged robots which inherently interact with the environment for locomotion and multi-contact manipulation. However, making and breaking contact violates the assumption of smooth and continuously differentiable functions.

Methods for planning motions on legged robots that include changing contacts and full dynamics models in a single optimisation problem (see Figure 1) can take prohibitively long to converge (on the order of minutes to hours), if at all. As thus, previous work proposed a range of simplifications of either dynamics models or contact locations and phases. Contact-Invariant Optimisation [Mordatch et al., 2012] uses additional continuous variables to indicate whether an end-effector is in contact as well as pre-specified contact phases. To be able to use unconstrained optimisation,

1 While this categorisation reflects whether the state trajectories are optimised *directly* (i.e., simultaneously) or *indirectly*, there is a further distinction in numerical optimisation; discussed, for instance, in [Betts, 2010, Section 4.3]. For the remainder of this thesis, we will describe methods optimising the controls and obtaining the state through integration as *indirect* (or shooting methods) and methods which simultaneously optimise both states and controls as *direct* (or simultaneous).

simplifications and relaxations to constraints arising from physics and dynamics are applied. Contact-Implicit Optimisation [Posa et al., 2014] models contacts as inelastic collisions with complementarity constraint and optimises over mode changes. However, due to the non-convexity of the problem, proper initialisation or expert-tuned heuristics are required for convergence. Winkler et al. [2018] optimise over timing and gait sequence by applying phase-based parametrisation and a simplified centroidal dynamics model. The hybrid formulation proposed in Stouraitis et al. [2018] optimises over both discrete contact locations and their timing as well as continuous force profiles. To make the problem tractable, they use a simplified single rigid body dynamics model and continuously-parametrised environments. Independent of the particular formulation, all of the above methods result in large Non-linear Programming (NLP) problems.

Collision avoidance

A challenge for all of these methods is the requirement to avoid *unwanted contact* (collisions) as these may perturb the system or cause injury, damage to the environment, or the robot itself. Exploratory methods use rejection sampling to discard configurations which are in a collision state. Including collision information into trajectory optimisation, however, requires smooth and differentiable proxy metrics. In practice, these are often expensive to compute and highly non-linear complicating the optimisation problem. Previous work proposed multiple direct approximations based on the collision shape geometry. We review and compare these in Chapter 2 on an goal state planning task. Other approaches which require preprocessing of the robot or environment model, such as the *Euclidean Distance Transforms* [Ye, 1988] or sphere approximations and signed distance fields, exist. Their description and comparison, however, is out of the scope of this thesis. When extending these collision avoidance proxies to trajectory optimisation, there is no guarantee that the transition between two discretised configurations is also collision-free. Seminal work by Schulman et al. [2014] introduced a convex penalty based on the swept volume between two configurations. For efficiency, they created the convex swept volume directly from the support mappings of the collision shapes. However, the resulting penalty requires the (expensive) computation of signed distances between bodies and has not been extended to dynamic environments. We describe an alternative novel and efficient formulation, based on harmonic potential fields, which works in the presence of moving obstacles in Chapter 3.

Pipeline approach

As an alternate solution to joint optimisation, decomposed pipeline approaches with several subsequent sub-problems have proven practical and reliable [Carpentier et al., 2017], see Figure 2. The advantage of pipeline approaches lies in the

improved tractability of the individual decomposed sub-problems. As a result, previous work proposed several interchangeable approaches for each stage. This further benefits the adaptability of a robotic system by combining different methods for the separate sub-problems as fit for the targeted application.

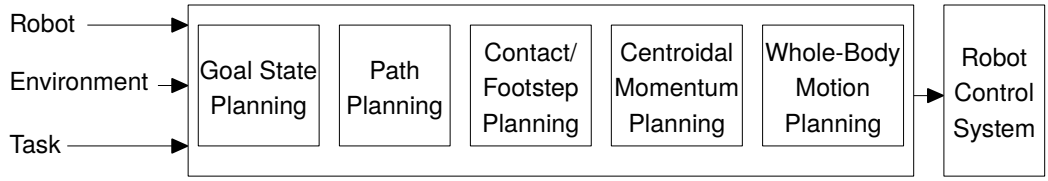


Figure 2: Common pipeline approach: The multi-contact motion planning problem is decomposed and solved by several, subsequent sub-problems.

Here, after obtaining a final pose and configuration (*Goal State Planning*), a possible path for a simplified lumped collision model is found (*Path Planning*). The contact planning phase then establishes reachable contact configurations (and often also a trajectory for the Centre-of-Mass (CoM) of the robot) along the guide path obtained by path planning. Using only a convex decomposition of support surfaces and a final footstep location, Deits and Tedrake [2014] formulate a Mixed-Integer Quadratically Constrained Quadratic Programme (MIQCQP) which finds an optimal contact sequence without the requirement for a guide path. Tonneau et al. [2018a] introduce a reachability-based contact planner able to synthesise acyclic contact alleviating the need to specify a gait mode. To efficiently generate feasible CoM trajectories that satisfy kinematic constraints from contact sequences, Tonneau et al. [2018b] present a convex formulation based on Bézier curve parametrisation. To plan CoM motions for dynamic behaviours from contact sequences which satisfy angular momentum constraints, Carpentier et al. [2016] introduced a walking pattern generator sufficiently fast for online replanning. To also optimise the timing of phases, Ponton et al. [2018] proposed a convex relaxation to achieve real-time performance. If only provided with task-space trajectories by centroidal momentum planning, second-order inverse kinematics or task-space inverse dynamics is used to execute on a robot [Del Prete et al., 2015, Farshidian et al., 2017]. Based on a given contact sequence and timing, Dai et al. [2014] optimises a full-body motion using centroidal dynamics while retaining a full kinematics model.

Contact planning

Walking pattern generation

Whole-body motion planning

The above all assumed that *sense*, *plan*, and *act* are separate and subsequent stages. This separation is only valid as long as no change to the assumptions of the previous stage occurs—or if one can execute the three steps sufficiently fast to respond to change. However, in the real world and especially in shared environments,

Unmodelled/ unexpected changes

unmodelled changes are commonplace; sensed information and models are imperfect. Model errors can impact assumptions about the physical properties of a robot such as its moments of inertia or the friction of contact interaction. Additionally, as robots enter new applications, the traditional assumptions that it is possible to control environments, and that all dynamics can be modelled (or cancelled out via stiff, high-gain position control) no longer hold. Finally, interacting with people and the environment introduces uncertainty and requires flexibility from the systems to adapt to changes. With uncertainty and change, the underlying assumptions no longer hold, and a quicker response is required.

Adaptability to change

Where algorithms can converge fast enough, online replanning can be applied on-the-fly in response to change. *Model-Predictive Control (MPC)* describes a control paradigm which does not require convergence of the optimisation [Wieber, 2006]. Instead, the updated sensor information is used as the initial state for optimisation with a limited time budget (usually to a sub-optimal solution within one or a handful of iterations). The first control of the obtained sequence within the preview horizon is applied, a new state sensed, and the process repeated at every control step. MPC commonly uses indirect/shooting methods due to the strict dynamic feasibility of the obtained solutions and favourable time per iteration compared with direct trajectory optimisation methods. For low-dimensional systems that have analytic dynamics models, e.g., quadrotor control, previous work has successfully applied direct multiple-shooting approaches for MPC, in this case, referred to as Non-linear Model-Predictive Control (NMPC). If the solution is within the same local minima, this receding horizon replanning scheme can be employed efficiently for online control. Koenemann et al. [2015] successfully applied this concept to a full-size humanoid robot and discussed relevant implementation details and how to deal with computation delay. Such an MPC scheme fails if the magnitude of the change in the initial condition exceeds the solver's ability to adapt the prior solution using local optimisation. One example is the requirement to take extra steps to prevent a fall in response to a strong push (i.e., impulse). In this case, the updated initial condition requires a significant change to the previous solution (which an MPC scheme uses as the initial guess) such that it is not possible to reach a feasible solution from inside the current local minima. However, if it is possible to provide a new suitable initial seed or warm-start, such an operating scheme can be applied robustly in response to change. One way is to provide a warm-start seed based on prior experience, see Figure 3.

Online replanning/ Model-Predictive Control

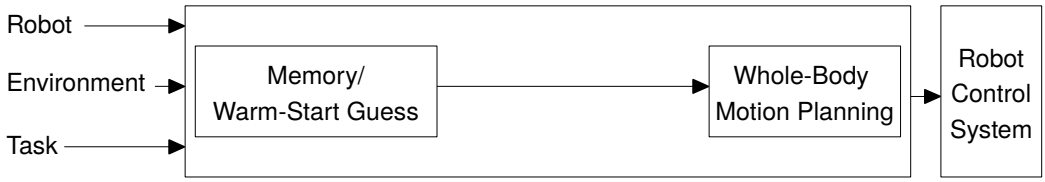


Figure 3: Conceptual idea: A memory-of-motion is used to provide an initial guess to an optimisation solver to speed up convergence and avoid getting stuck in local minima.

Trajectory libraries

Trajectory library approaches exploit similarity to previous problems or situations (e.g., initial states). They store sets of trajectory samples and retrieve an initial guess through look-up methods, e.g., nearest neighbour [Stolle and Atkeson, 2006, Stolle et al., 2007, Tassa et al., 2008]. An advantage of trajectory library approaches is the ability to support different solution lengths: In order to reach a task goal which is close to the initial state, fewer waypoints are required compared with queries where the initial state is far from the target. Approximation schemes commonly require a uniform length for prediction. Key challenges for implementing trajectory libraries include the definition of suitable distance metrics for look-up, transfer across environments, and the trade-off between generalisation/coverage and memory size. As trajectory libraries can contain multiple potentially applicable warm-starts, it is also possible to use ranked lists. Here, if optimisation from an initial seed fails, the next ranked can be tried. Dey et al. [2013] applied exhaustive offline learning to learn a ranked classification of trajectory solutions based on a parametrised problem input. While this—unlike look-up with simple distance metrics—ensures that the best possible samples from the library are selected, it is only applicable to limited-size datasets and—unlike standard trajectory library approaches—does not support incremental addition of new samples online.

In Chapter 4, we follow a trajectory library approach and propose an efficient representation and indexing scheme that allows initialisation of high-dimensional kinematic trajectory optimisation in complex collision environments while using inexpensive distance metrics.

However, trajectory library approaches come with two challenges: First, they make inefficient use of the data as they do not generalise over or interpolate between samples. Second, storing sufficient trajectory samples on high-dimensional robots requires much memory and slows down look-up. In many cases, it is not necessary to store the entire dataset: The large numbers of raw samples contain few representative classes or local minima solutions which share similarities among

them. Efficient (e.g., second-order) optimisation methods can adapt them quickly to new problem queries. Thus, often only a few *representative* samples would be required. Identifying and selecting these is a challenge, however.

Previous work proposed several ways of extracting insights and the underlying structure of the trajectory samples. One way of compressing the original dataset is to use tools from machine learning to approximate a function generating initial guesses using low-dimensional models.

Programming-by-Demonstration

Similar to humans who learn by observing and repeated trials, Programming by Demonstration (PbD) and Learning from Demonstration (LfD) build on the idea that tasks can be learnt, i.e., their objectives and constraints extracted, directly from human demonstrations [Billard et al., 2008]. A model can approximate the provided sample demonstrations—e.g., using a Bayesian Gaussian Mixture Model [Pignat and Calinon, 2019]—or used to train a stable, dynamic system [Schaal, 2006, Khansari-Zadeh and Billard, 2011].

Alternately, demonstrations can be used to extract the underlying objectives (or rewards), e.g., using Inverse Reinforcement Learning (IRL) [Ziebart et al., 2008, Mori et al., 2011]. Similarly, Inverse Optimal Control (IOC) (Mombaur et al. [2010], Levine and Koltun [2012]) can be used to learn value functions which are less expensive to evaluate.

The availability of large-scale motion capture datasets such as the CMU Graphics Lab Motion Capture Database[2] and the KIT Motion Database [Mandery et al., 2015] have enabled considerable progress in the field of computer graphics. More recently, human demonstrations have further been automatically extracted from widely available online video footage (e.g., from YouTube) using 2D [Cao et al., 2019] and 3D [Mehta et al., 2018] pose reconstruction algorithms. Holden et al. [2017] introduced *Phase-Functioned Neural Networks* for real-time character control by conditioning the weights of a small neural network on the insight that human gait is periodic. Similarly, Zhang et al. [2018] applied a similar concept for quadruped character control by conditioning the neural network on the gait mode. While resulting in naturally looking impressive movements, neither method took physical or dynamics constraints into account. Peng et al. [2018] addressed this by adapting a Reinforcement Learning (RL) approach for interactive character control in physical simulation. To bootstrap training, the authors used 3D pose reconstruction from video footage available online.

2 http://mocap.cs.cmu.edu/

Using a custom simulator for fast simulation with stable contact models and learnt dynamics, Hwangbo et al. [2019] recently demonstrated versatile locomotion behaviour in hardware experiments learnt entirely using reinforcement learning.

Other work has focused on combining Optimal Control (OC) for obtaining optimal sample trajectories with learning using neural networks [Levine and Koltun, 2013, Carius et al., 2020].

Iterative exploration and learning

Mansard et al. [2018] proposed an iterative exploration scheme which combines a kino-dynamic PRM with trajectory prediction. The advantage of this method is that exploration is guided and terminates once the trajectory prediction outperforms the roadmap-based baseline. The authors applied this concept to low-dimensional dynamic systems, including quadrotors, while extensions of this approach to complex environments and legged robots have yet to be shown. A challenge here is that standard function approximators assume uni-modal distributions, i.e., that every input can only have one output. Due to this, Mansard et al. [2018] enforced uni-modality during the exploration. However, complex robotic systems often have multiple ways of achieving a goal (e.g., one can walk around an obstacle from two sides) or discontinuous solutions induced by the dynamics or environment.

Discontinuity-sensitive learning

To address these discontinuities, Tang and Hauser [2019] used the expertise and insights of a system designer to split the original data in clusters and apply a Mixture-of-Experts (MoE) approach. This strategy, in contrast to Mansard et al. [2018], requires all data to be available during training time. In Chapter 5, we follow a similar approach to address discontinuity and multi-modality. Instead of relying on an expert system designer, we use tools from computational topology to inform clustering and warm-start decisions.

While the previous approaches used locally optimal samples to learn a global policy, Deits et al. [2019] recently proposed a fascinating approach to directly learn the value function (which, unlike the policy function, is unique). Here, the authors obtain samples with bounded global sub-optimality from early terminated mixed-integer programming. They demonstrate their approach in a predictive controller on multi-contact problems on a planar humanoid as well as a cart-pole with walls.

Transfer across environments

Beyond system dynamics, the complexity of the environments robots operate in has a significant impact on the transferability of a learnt policy. Most of the above work did not consider complex or changing collision environments. The key challenge here is to create a representative and transferable descriptor/encoding for the state of general collision objects with respect to the robot. A challenge is that this encoding needs to be able to be efficiently explored by randomly generating

(realistic) environments. Jetchev and Toussaint [2013] in their seminal work investigated the ability to learn sparse feature descriptors to generalise across environments. This approach still relies on the ability to provide a way to generate random environments and optimal solutions for a specific domain, and subsequently learn the sparse feature encoding before learning the trajectory prediction. In roadmap-based methods, an opposite approach is taken: Rather than representing the environment collision obstacles in a feature descriptor, Dynamic Roadmaps (DRMs) [Leven and Hutchinson, 2002]—a variant of PRM—encode the workspace occupation of configuration samples. Presented with a new static environment, samples/vertices that would be in-collision can be efficiently filtered, resulting in a reduced, collision-free graph. The essential advantage here is that collision checking is off-loaded into an offline computation phase with only filtering taking place online. The disadvantage is that storing the collision information for samples (vertices) and the connecting motions (edges) quickly exhausts available memory, limiting the number of samples that can be stored. Recently, the Hierarchical Dynamic Roadmap (HDRM) [Yang et al., 2018a] has been proposed which does not need to store occupancy information. However, it has not shown to scale to systems of higher dimensionality or that involve dynamics.

Shared autonomy

We have now discussed formulations and methods to warm-start optimisation solvers. Here, an experienced operator selected and designed the constraints and objectives based on the particular task at hand. For robots to be flexibly deployed and operated by end-users, the human-robot interface to formulate tasks and to interact with the system plays a significant role. *Shared autonomy* allows users sliding control between full remote control (assisted by easy-to-use perception and planning interfaces) and full autonomy. In Chapter 6, we introduce a robust shared autonomy system which automatically monitors and evaluates changes in the environment to stop, replan, and resume operation. We evaluate the system in real-world experiments and field deployment.

Whole-body control

While the previous paragraphs focused on motion planning, an essential factor for the success of deploying and translating research results from simulation to real-world experiments are the control systems at hand. On legged robots, Quadratic Programming (QP) based inverse dynamics control that operates on synchronised sub-systems is the de facto standard. Most mobile manipulation platforms and laboratory equipment, however, are delivered with unsynchronised sub-systems. As a result, many of the assumptions from the motion planning stage are invalid, and often the redundancy built-in to these systems gets reduced as locomotion

and manipulation are treated separately. Inspired by whole-body control on legged robots, we introduce a full system for mobile loco-manipulation in Chapter 7.

1.1 PROBLEM STATEMENT

In the previous section, we reviewed state-of-the-art methods following the concept of using offline generated optimal samples to bootstrap trajectory optimisation online in response to change across a variety of approaches and fields. The discussed work differs across multiple dimensions: In the complexity of the dynamics, the dimensionality of the systems, and the environment complexity considered. We illustrate the systems and scenarios considered in this thesis against these dimensions in Figure 4.

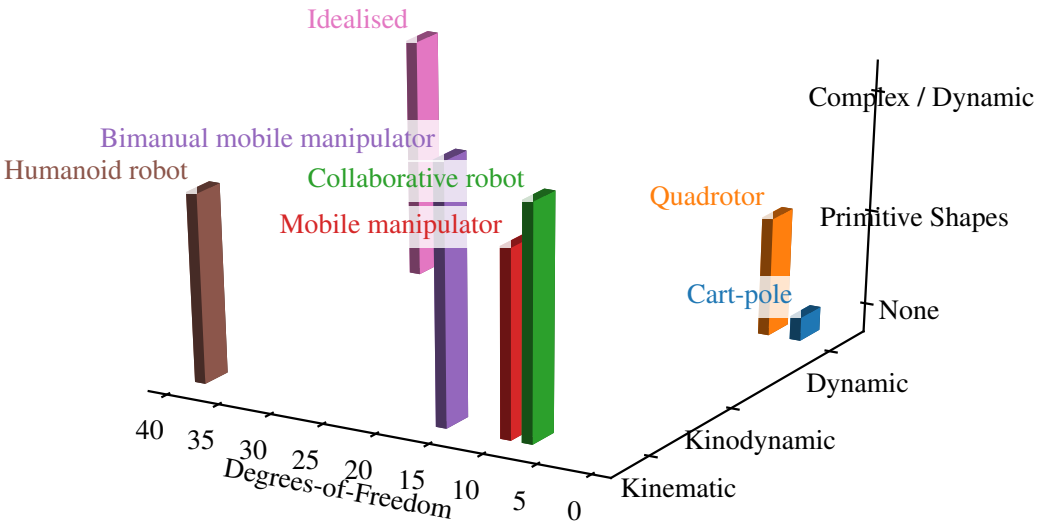


Figure 4: Lattice showing the complexity of problems considered in this thesis categorised by (a) dimensionality, (b) dynamics model considered, and (c) complexity of environment.

The key open—and closely related—questions include:

PROBLEM FORMULATION This relates to the assumptions and models used as well as the set of representations, constraints, and objectives. One important decision surrounds whether to use soft (penalties) or hard constraints. Similarly, the decision whether to use a uniform running cost or a final cost influences the susceptibility of a formulation to approximation error in an initial guess.

EXPLORATION The offline exploration process has a large impact on the success of being able to provide a suitable initialisation. Random sampling may lead to many samples in areas of the state space that are unlikely to be encountered during deployment. In contrast, exhaustive or directed sampling is likely to be untractable. As thus, a sampling scheme that weighs exploration with coverage of relevant areas of the state space is important.

ENCODING Low-dimensional problems in simple environments can be encoded in their respective state space. To the best of our knowledge, the question for a generalisable and transferable representation for high-dimensional systems in interaction with complex environments which can be explored easily remains an open area of ongoing research.

1.2 CONTRIBUTIONS

The main contributions of our work presented in this thesis are:

CHAPTER 3 We introduce a novel formulation for efficient continuous-time collision avoidance among moving obstacles using harmonic potential fields and conservative advancement collision checks. To the best of our knowledge, this is the first method to demonstrate general, continuous-time collision avoidance in dynamic environments.

CHAPTER 4 Following a trajectory library approach, we introduce an efficient encoding and indexing scheme along with a tailored exploration strategy. We demonstrate our approach on a high-dimensional kinematic manipulation task on a humanoid robot in a complex environment. We show that the indexing scheme allows for memory-efficient storage as well as for inexpensive distance metrics to achieve a high warm-start success rate and coverage.

CHAPTER 5 Using computational homology to extract underlying features of the solution space, we show that the extracted clusters can be used to select representative sample trajectories and improve warm-start success rates.

CHAPTER 6 We present an extension to a shared autonomy framework with continuous scene monitoring for deploying mobile manipulation systems with sliding control between operator and autonomous behaviours. We demonstrate our approach in laboratory experiments and field trials with remote operation.

In this thesis, we incorporate the following publications: Merkt et al. [2019a] (Chapter 3), Merkt et al. [2018] (Chapter 4), Merkt et al. [2017] (Chapter 6), Merkt et al. [2019b] (Chapter 7).

The technical aspects have been implemented and open-sourced in an extensible framework for prototyping and benchmarking planning and control algorithms [Ivan et al., 2019]. Chapter 6 builds and extends on a software framework evolved from the work of the Massachusetts Institute of Technology (MIT) DARPA Robotics Challenge (DRC) team [Marion et al., 2017].

1.3 THESIS STRUCTURE

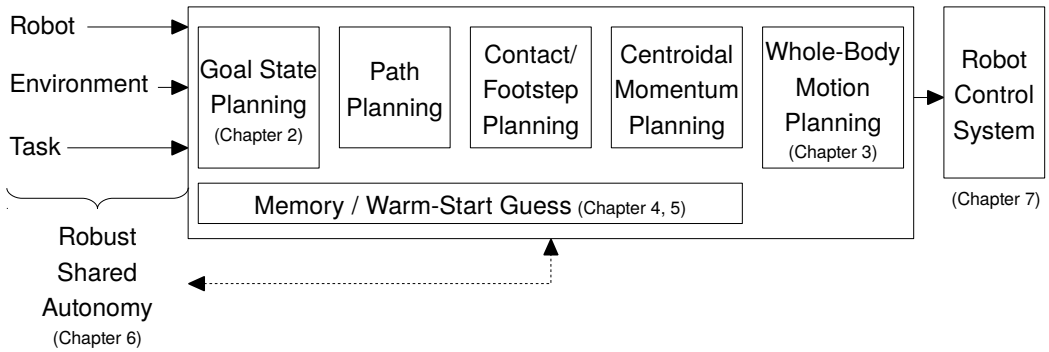


Figure 5: Structure of this thesis

Figure 5 shows the organisation of this thesis: In Part II, we focus on the motion synthesis problem. We describe a general formulation for collision-free goal state planning and discuss different approaches for including collision avoidance (Chapter 2). In Chapter 3, we extend the formulation to trajectories and introduce a novel formulation for continuous collision avoidance. Part III focuses on using offline computation to bootstrap trajectory optimisation during runtime. We introduce an efficient indexing scheme in Chapter 4 for high-dimensional, kinematic trajectory optimisation in complex environments. In Chapter 5, we explore the topological structure of the solution space to improve warm-start success in dynamic optimal control settings. Part IV focuses on embodiment and deployment for real-world scenarios. In Chapter 6, we provide a general framework for shared autonomy with semantic perception and continuous scene monitoring. A key feature is an ability for a user to specify high-level commands to be relayed to the underlying algorithms. In Chapter 7, we describe a full system architecture for translating

whole-body manipulation from visualisation to real-world experiments. Finally, we provide a summary in Chapter 8 and outline future directions in Chapter 9.

An overview of robot platforms to evaluate the methods presented in this thesis is given in Table 1.

Table 1: Overview of the robot platforms used in this thesis: We give degrees of freedom as well as the state (NX) and control (NU) space dimensions. As collaborative robots, we used the Kuka LWR3 and Franka Emika Panda, as a mobile manipulation platform the Adabotics Ada500, as a bimanual mobile manipulation platform the Clearpath Husky, and as a humanoid robot the NASA Valkyrie.

Platform	DoF	NX	NU	Robot Model	Environment	Chapter	Deployment
Cart-pole	2	4	1	Dynamic	None	5	Simulation
Quadrotor	6	12	4	Dynamic	Shape primitives	5	Simulation
Collaborative robot	7	14	7	Kinematic	Complex, dynamic	3, 6	Visualisation (3), real-world experiments (6)
Mobile manipulator	9	18	9	Kinematic	Complex	3, 7	Visualisation (3), real-world experiments (7)
Bimanual mobile manipulator[3]	15	30	15	Kinematic	Complex, dynamic	6	Real-world experiments
Humanoid robot[4]	38[5]	76	32	Kinematic	Complex	2, 3, 4	Real-world experiments

3 As this platform uses individual controllers for each sub-system, it will only be considered as either three, six, or twelve DoF simultaneously for motion planning, and 15 DoF for goal state planning.

4 Here, as for the quadrotor, we account for the rigid body transformation of the floating-base ($\mathbb{SE}(3)$) using Euler angles instead of quaternions.

5 Excludes hands.

Part II

MOTION SYNTHESIS

2

COLLISION-FREE GOAL STATE PLANNING

Being able to generate collision-free whole-body configurations is an important capability in a wide range of settings in robotics. One example is end-to-end motion and manipulation planning systems as required for generating large datasets of optimal motions in Part III of this thesis. Frequently used pipeline approaches separate the motion planning problem into staged/subsequent sub-problems (see Figure 2), where finding a collision-free goal configuration—unless specified by a user operator as in Chapter 6—is the first stage and dramatically impacts overall planning success. Within global, exploration-based methods, the obtained end-poses are used to initialise bi-directional sampling-based motion planning algorithms such as RRT-Connect [Kuffner and LaValle, 2000] or Hierarchical Dynamic Roadmap (HDRM) [Yang et al., 2018a]. Similarly, they are also used directly to provide trajectory optimisation methods such as Covariant Hamiltonian Optimisation for Motion Planning (CHOMP) [Zucker et al., 2013], Stochastic Trajectory Optimisation for Motion Planning (STOMP) [Kalakrishnan et al., 2011], Trajectory Optimisation for Motion Planning (TrajOpt) [Schulman et al., 2014], and Approximate Inference COntrol (AICO) [Toussaint, 2009] with a straight-line initialisation, i.e., by providing a seed trajectory consisting out of the linear interpolation between a start and goal state. Beyond dataset generation, this capability can further be used to achieve higher levels of autonomy or support the operator in interactive approaches such as the ones discussed in Part IV of this thesis.

In this chapter, we first introduce a flexible and general non-linear optimisation-based formulation to generate optimal whole-body configurations and placements subject to complex objectives and constraints (Section 2.1). We then formulate and compare several ways to encode collision avoidance in optimisation problems using soft and hard constraints (i.e., penalties and inequality constraints; the *formulation dimension*) in Section 2.2. Finally, we detail a benchmark and compare them across a variety of optimisation solvers and algorithms with regards to their success rate and optimisation time (the *solver dimension*) in Section 2.3.

2.1 GENERAL OPTIMISATION-BASED GOAL STATE PLANNING

We define goal state planning as the process which returns a goal state $\mathbf{x}_{goal}$ that satisfies operational space targets $\mathbf{y}^*$, where the operational space $\mathbf{y}$ is a user-defined mapping of the robot state: $\mathbf{y} = \phi(\mathbf{x})$, which we refer to as *task-map*. Task-maps can, for instance, represent the task-space position and orientation of an end-effector to grasp an object (as in classical inverse kinematics, see Figure 6). However, a task-map can also more generally encode other criteria such as quasi-static balance, relative or absolute distances between links, gaze targets to keep a workspace region within the field of view of a sensor, or alternate space representations such as *Interaction Mesh* [Ivan et al., 2013] or *Distance Mesh* [Yang et al., 2015]. Goal state planning is thus a general form of optimisation-based inverse kinematics/inverse geometry for floating-base systems. Here, we explicitly consider the floating-base placement as part of the state during the planning and focus on a more general task-space goal set $\phi(\mathbf{x})$ using multi-objective optimisation.

We describe the motion of a robotic system with respect to (w.r.t.) an inertial, fixed reference frame I. The position of the base (or root link) w.r.t. the inertial frame I expressed in the inertial frame is denoted as ${}_{I}\mathbf{r}_{IB} \in \mathbb{R}^3$, and the orientation of the free-floating base w.r.t. the inertial frame with $\boldsymbol{\psi}_{IB}$. The orientation of the free-floating base can be parametrised using rotation matrices, Euler angles, or Hamiltonian unit quaternions, among others. The translations and rotations of a frame are direct Euclidean isometries and form part of the special Euclidean group. The placement, position and orientation, of the base frame T_{base} is an element of the special Euclidean group $\mathbb{SE}(3)$. In our implementation, we support several representations which can be chosen at runtime and use Lie algebra for operations on them. We capture the configuration of all n_j joints in a joint configuration vector $\mathbf{q}_j \in \mathbb{R}^{n_{q,j}}$. In systems that only use revolute or prismatic joints, $n_{q,j}$ is equal to the number of joints n_j. We describe the configuration of joints with a continuous or spherical range of motion using the special orthogonal groups $\mathbb{SO}(2)$ ($\mathbb{R}^2$) and $\mathbb{SO}(3)$ ($\mathbb{R}^3$), respectively. Using this, we can write the generalised coordinates vector $\mathbf{q}$ and the generalised velocities vector $\boldsymbol{\nu}$ of a floating-base system as

$$\mathbf{q} = \begin{bmatrix} {}_{I}\mathbf{r}_{IB} \\ \boldsymbol{\psi}_{IB} \\ \mathbf{q}_j \end{bmatrix} \in \mathbb{SE}(3) \times \mathbb{R}^{n_{q,j}}, \quad \boldsymbol{\nu} = \begin{bmatrix} \boldsymbol{\nu}_B \\ \dot{\mathbf{q}}_j \end{bmatrix} \in \mathbb{R}^{n_\nu}, \tag{1}$$

where $n_v = 6 + n_{q,j}$ and the twist $\boldsymbol{\nu}_B = [{}_I\boldsymbol{\nu}_B \quad {}_B\boldsymbol{\omega}_{IB}] \in \mathbb{R}^6$ encodes the linear and angular velocities of the base B w.r.t. the inertial frame expressed in the I and B frames, respectively. We consider $\mathbf{x}$ to fully capture the state of a robotic system comprised of its generalised configuration and generalised velocities vector:

$$\mathbf{x} = (\mathbf{q}, \boldsymbol{\nu}). \tag{2}$$

Goal state planning thus is the process that returns a fully defined goal state $\mathbf{x}_{goal}$ given a task-space target $\mathbf{y}^*$ and an environment Ω, starting from an initial state $\mathbf{x}_0$:

$$\mathbf{x}_{goal} = \mathtt{GoalStatePlanning}(\mathbf{x}_0, \mathbf{y}^*, \Omega) \tag{3}$$

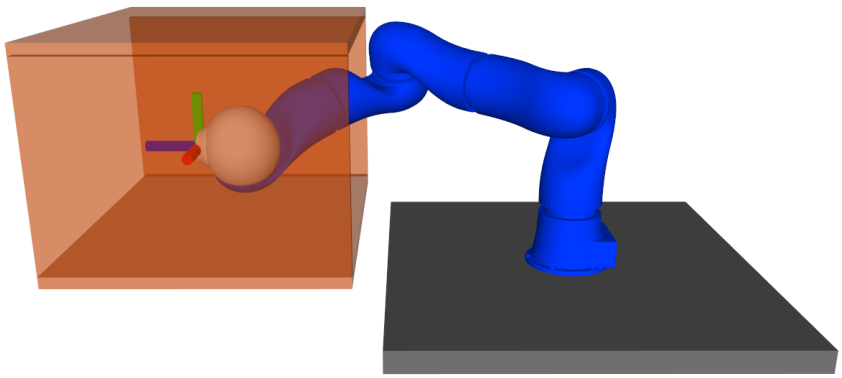


Figure 6: Inverse kinematics task with collision avoidance on a 7-DoF manipulator. We use the redundancy in the task null-space to resolve collision avoidance objectives.

We formulate (3) as a single optimisation problem minimising a general/non-linear objective $\ell(\mathbf{x})$ subject to bound constraints on the variables $\mathbf{x}$ and general equality and inequality constraints $h(\mathbf{x})$ and $g(\mathbf{x})$:

$$\mathbf{x}_{goal} = \arg\min_{\mathbf{x}} \ \ell(\mathbf{x}) \tag{4}$$

$$\text{s.t.: } g(\mathbf{x}) \leqslant 0 \tag{5}$$

$$h(\mathbf{x}) = 0 \tag{6}$$

$$\mathbf{x}_{lb} \leqslant \mathbf{x} \leqslant \mathbf{x}_{ub} \tag{7}$$

and use the concept of task-maps to flexibly encode and recombine objectives and constraints of arbitrary complexity for $\ell(\mathbf{x})$, $g(\mathbf{x})$, and $h(\mathbf{x})$. We apply the specific algebra of the corresponding task-spaces (e.g., Lie algebra for rotations) during the difference (denoted as $\boxminus$) and integration ($\boxplus$) operations. As a result, the dimension of $\mathbf{y}^*$ and $\phi(\mathbf{x})$, expressed in the task-space, and $\mathbf{y}_{error} = \mathbf{y}^* \boxminus \phi(\mathbf{x})$,

a vector in the tangent space of the task-space, may differ. For task-maps expressed in Euclidean spaces, these operations simplify to Euclidean difference ($-$) and integration ($+$) operations. We follow the operator notation of Bloesch et al. [2016], which provide further detail on the differential calculus of 3D orientations. An important note to make here is that while a task-map performs the mapping from the configuration space $\mathbf{x}$ to the arbitrary task-space $\mathbf{y}$, it does not carry a notion of how these task-space error terms $\mathbf{y}_{error}$ are composed into the individual objective and constraint functions. Arbitrary cost landscapes can be designed by using the concept of an activation function $a(\circ)$, for instance, a sum-of-squares or logarithmic function. The gradients for these can then be obtained by application of the chain rule in combination with the analytic Jacobian of the task-maps and the geometric Jacobian of the robot. For simplicity, we often use sum-of-squares for objective terms and linear activations for constraints. Note, however, that this flexibly allows the recombination of a task-map commonly used as a constraint into a cost term (e.g., through the use of a *log-barrier* or *relaxed-log-barrier* [Feller and Ebenbauer, 2017] as an activation function for inequality constraints and a *Lagrange multiplier* for equality constraints akin to an Augmented Lagrangian scheme).

In the following, we consider the ability to express any frame with respect to any other arbitrary frame during the forward kinematic mapping. Examples include expressing the position of the hand with respect to the head or constraining the relative distance between two feet. As such, all task-maps can be expressed either in relative or absolute terms through reconfiguration in the configuration file of the planning problem within our optimisation framework.

Most goal state planning tasks can be realised by recombination of a few frequently used task-maps. We will describe these in the following subsections.

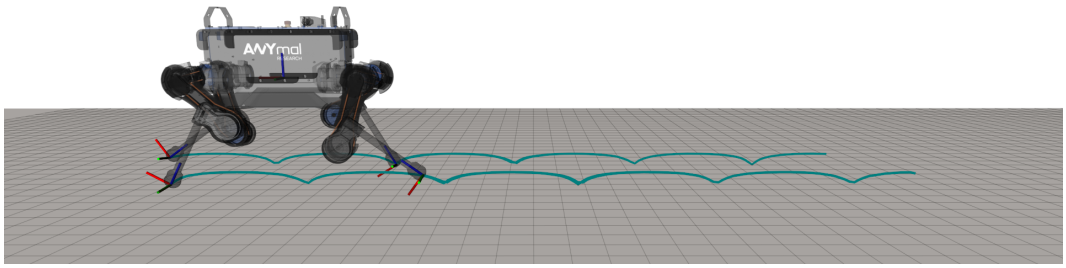


Figure 7: Visualisation of real-time whole-body remapping of task-space commands by using the presented goal state planning formulation showing the target frames and traces over multiple time steps.

2.1.1 *Frame Position*

Frame Position is a three-dimensional task-map representing the position (x, y, and z coordinates) of a link/frame A expressed in frame B as obtained from the forward kinematic mapping:

$$\phi_{\text{position}} = {}_B\mathbf{r}_{BA}. \tag{8}$$

2.1.2 *Frame Orientation*

Frame Orientation is a task-map representing the orientation of a link/frame A w.r.t. a frame B using the chosen parametrisation:

$$\phi_{\text{orientation}} = \boldsymbol{\psi}_{BA}. \tag{9}$$

The selected parametrisation—rotation matrix, Euler angles, or Hamiltonian unit quaternion—determines the dimensionality of the task-map. Differences from the desired target orientation expressed as a twist are computed using Lie algebra.

2.1.3 *Frame Placement*

Frame Placement is a task-map combining both the position and orientation task-maps to fully describe the rigid body transformation of a frame as obtained from the forward kinematic mapping:

$$\phi_{\text{placement}} = \begin{bmatrix} {}_B\mathbf{r}_{BA} \\ \boldsymbol{\psi}_{BA} \end{bmatrix}. \tag{10}$$

It supports different orientation representations and uses Lie algebra to perform difference and integration operations.

2.1.4 *Frame Axis Alignment*

Frame Axis Alignment is a one-dimensional task-map representing the angle between the vector representing an axis of a frame and the desired target direction as computed by the dot product between the two vectors (see Figure 8). We use it, for instance, to ensure flat contact between the foot and supporting region (to restrict foot roll and pitch w.r.t. the plane's normal vector to zero) or as a relaxed

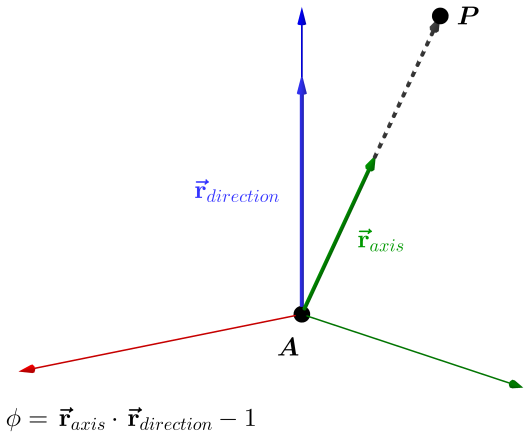


Figure 8: Illustration showing the working principle of the axis alignment task-map.

orientation constraint that keeps one degree of freedom (yaw) unconstrained (e.g., to keep a target level/horizontal). We denote the target direction as a normalised vector $\vec{\mathbf{r}}_{\mathrm{direction}}$, and the axis $\vec{\mathbf{r}}_{\mathrm{axis}}$ of a frame A to be the normalised vector from the origin of the frame A to a point P expressed in A:

$$\phi_{\mathrm{axis-alignment}} = \vec{\mathbf{r}}_{\mathrm{axis}} \cdot \vec{\mathbf{r}}_{\mathrm{direction}} - 1. \tag{11}$$

2.1.5 *Joint Configuration*

Joint Configuration is a linear map to represent the difference from the desired target value, e.g., a nominal configuration for regularisation or to fix specific joints. It supports subsets of the joint configuration vector using a selection matrix S and can also be used to specify a goal configuration:

$$\phi_{\mathrm{configuration}} = S\mathbf{q}. \tag{12}$$

2.1.6 *Distance to Plane*

Distance to Plane is a one-dimensional task-map representing the shortest distance between a plane and the position of a frame A (see Figure 9). We use the dot product between the normal vector of a plane (here, defined as the unit vector of the z-axis of a frame B) and the position of the frame A expressed in B, ${}_{\mathrm{B}}\mathbf{r}_{\mathrm{BA}}$:

$$\phi_{\mathrm{distance-to-plane}} = {}_{\mathrm{B}}\vec{\mathbf{z}} \cdot \vec{\mathbf{r}}_{\mathrm{BA}}. \tag{13}$$

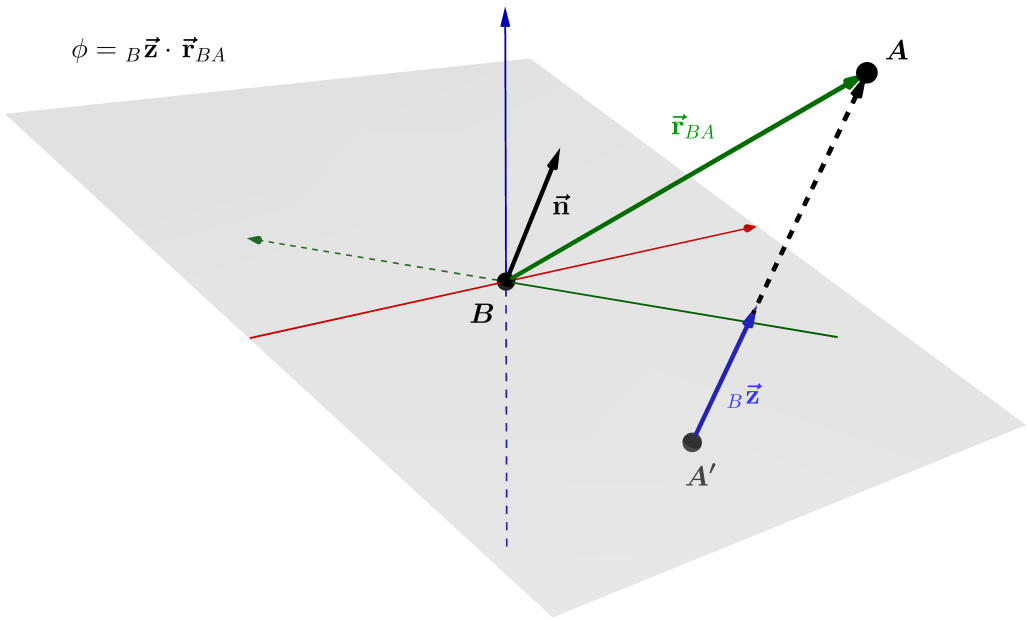


Figure 9: Illustration showing the working principle of the distance to plane task-map.

We use this task-map for instance to constrain the foot to be in contact with the supporting surface or to ensure contact between an end-effector and a surface in wiping tasks.

2.1.7 *Quasi-Static Balance*

Quasi-Static Balance is a one-dimensional task-map representing whether the projection of the Centre-of-Mass (CoM) of the robot falls within the convex hull of a set of support contact points. In particular, we use a harmonic potential field-based metric defining the potential to the absolute minimum with the boundary of the support region at zero potential. Using a harmonic potential field ensures a smooth gradient with a unique minimum throughout the task-space.

2.1.8 *Use of the task-maps within our optimisation framework*

The implementation of most of these task-maps allows them to be flexibly used as either a cost or constraint in the optimisation framework. We provide an overview in Table 2. Figure 10 shows an application using a combination of the above-described task-maps to realise an interactive whole-body inverse kinematics solutions that automatically adjusts the support contacts to the environment.

Task-map	Cost	Equality constraint	Inequality constraint
Frame Position	✓	✓	✓
Frame Orientation	✓	✓	✓
Frame Placement	✓	✓	✓
Frame Axis Alignment	✓	✓	✓
Joint Configuration	✓	✓	✓
Distance to Plane	✓	✓	✓
Quasi-Static Balance	✓	✗*	✓

Table 2: Overview of task-maps and how they can be used in the optimisation framework. Where a task-map can be both an equality and inequality constraint, the inequality option allows to specify an allowed tolerance—or desired minimum deviation in conjunction with a negative weight value. Some task-maps, e.g., quasi-static balance, come with configuration options to switch between use as a cost or constraint.
* It is possible to use the quasi-static balance task-map as an equality constraint, however, it is illogical.

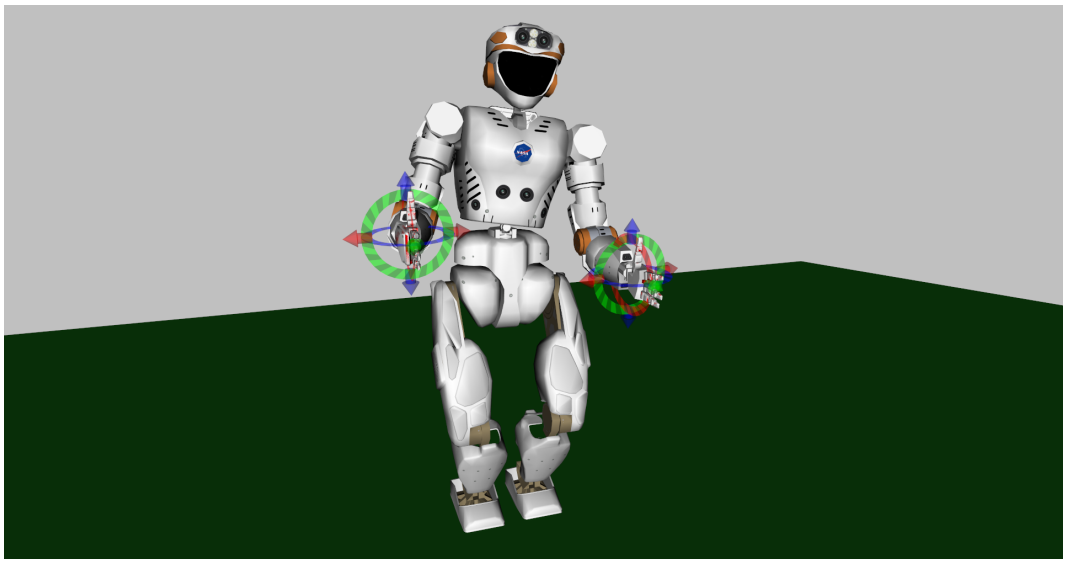


Figure 10: Interactive whole-body goal state planning with the presented framework using tasks for quasi-static balance, minimum and maximum distance between the feet, contact (and normal alignment) with the support surface/plane, collision avoidance, and two end-effector targets.

2.2 COLLISION AVOIDANCE

The task-maps in the previous section are solely derived from the rigid body transformations of the frames associated with the links in the scene. Collision avoidance, on the other hand, requires to take the geometric shapes of the collision objects attached to links in the scene into account. Hence, formulating penalty and constraint terms is more involved, and we discuss common approaches below.

2.2.1 *Collision Proxy*

A state is said to be in a collision if any of the collision shapes (which can be primitive shapes such as spheres, cylinders, boxes, and capsules as well as triangulated meshes) attached to the robot's links occupy the same workspace as obstacles in the environment. While efficient methods to determine intersection—the binary collision check—exist, defining a continuous, smooth, and differentiable metric is not as straight-forward. In order to obtain a smooth, differentiable penalty for collision avoidance, we compute a collision proxy $P_{\mathcal{A},\mathcal{B}}$ for every pair of collision objects $\mathcal{A}, \mathcal{B}$ which contains (a) a signed distance $d_{\mathcal{A},\mathcal{B}}$ for the closest distance or deepest penetration between the objects, (b) the closest points between or the deepest penetration point on either body $\mathbf{p}_{\mathcal{A}}, \mathbf{p}_{\mathcal{B}}$ (also referred to as witness points), and (c) the normals $\hat{\mathbf{n}}_{\mathcal{A}} = \mathbf{p}_{\mathcal{B}} - \mathbf{p}_{\mathcal{A}}, \hat{\mathbf{n}}_{\mathcal{B}} = \mathbf{p}_{\mathcal{A}} - \mathbf{p}_{\mathcal{B}}$ between these two virtual points:

$$P_{\mathcal{A},\mathcal{B}} = (d_{\mathcal{A},\mathcal{B}}, \mathbf{p}_{\mathcal{A}}, \mathbf{p}_{\mathcal{B}}, \hat{\mathbf{n}}_{\mathcal{A}}, \hat{\mathbf{n}}_{\mathcal{B}}) \tag{14}$$

The concept of using virtual collision proxies is illustrated in Figure 11 for two spheres as well as the shelf scenario considered (only external/interaction proxies are visualised).

Using the collision proxies for pairs of collision bodies, we can now formulate a set of penalties and constraints in the following sections.

2.2.2 *Collision Distance*

The most direct use of a set of obtained collision proxies is to constrain the distance between every pair of collision objects to be greater than zero, i.e., to add general/non-linear inequality constraints:

$$g_i(\mathbf{x}) = -d_{\mathcal{A},\mathcal{B}} \; \forall \mathcal{A} \in \text{Robot links}, \; \mathcal{B} \in \text{Environment links} \tag{15}$$

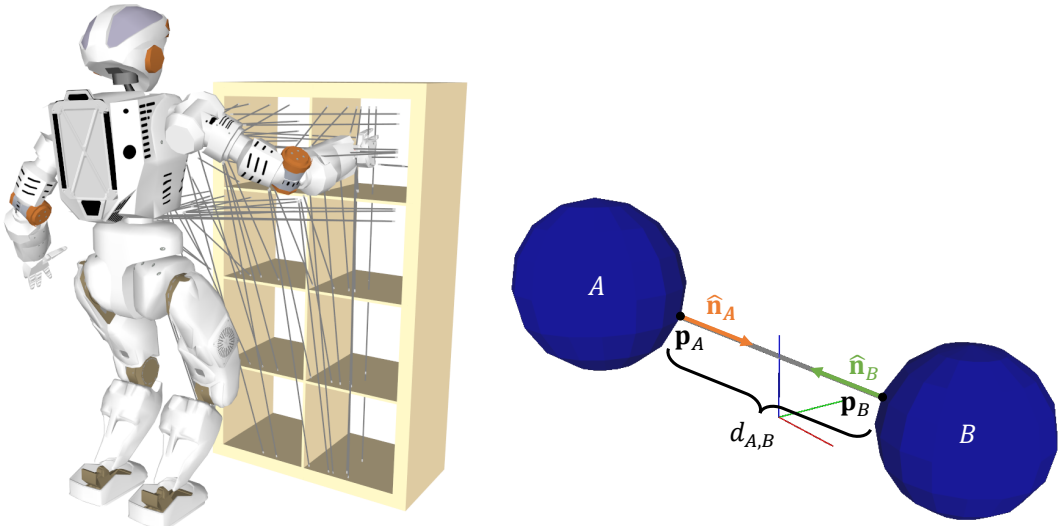


Figure 11: Collision proxies: The grey line visualises the normal between the nearest points on either collision body along which the virtual separation and penetration will move. Illustrated example for a sphere-sphere distance on the right and visualisation of all considered collision proxies for a motion planning problem on the NASA Valkyrie robot platform on the left.

Assuming the number of collision shapes for the robot is n_R, and the number of environment shapes is n_E, this would in the worst case create $n_R \times n_E$ inequality constraints for collision avoidance with the environment and $n_R \times (n_R - 1)$ collision constraints for self-collision avoidance not regarding excluded collision pairs from an *Allowed Collision Matrix*.[1] Alternately, as often the size of the constraint matrices are pre-allocated, and the number of environment obstacle shapes is not known a priori, n_R constraints can be added where only the closest proxy for each of the robot links is considered. Note, however, that this leads to sudden jumps in the gradient as the closest object is switching which has an impact on the numerical stability of optimisation solvers.

Using the normals of the witness points as introduced above, we can define an approximate metric for the change of the distance between the objects as movement along the normal connecting the witness points. By using the geometric Jacobian

1 Using a data structure which indicates whether collision shapes are in a collision by default (e.g., because the collision shapes are enlarged compared to the exact geometry of the robot), cannot be in a collision (e.g., because the shapes are out of kinematic reach of each other or restricted by joint limits), or cannot be in a collision because they belong to the same link, this worst-case estimate can often be reduced significantly in practice.

of the witness points relative to the corresponding links and applying the chain rule, we can derive an approximate gradient:

$$\frac{\delta g_i(\mathbf{x})}{\delta \mathbf{x}} = -\hat{\mathbf{n}}_{\mathcal{A}}^{\mathsf{T}} \cdot \mathrm{J}_{\mathbf{p}_{\mathcal{A}}}(\mathbf{x}) + \hat{\mathbf{n}}_{\mathcal{B}}^{\mathsf{T}} \cdot \mathrm{J}_{\mathbf{p}_{\mathcal{B}}}(\mathbf{x}) \tag{16}$$

2.2.3 *Smooth Collision Distance*

The previous section considered the use of inequality constraints of variable dimension. However, many optimisation solvers are not able to directly handle inequality constraints, and the use of barrier methods comes with numerical challenges if no good seed is available. Thus, this section focuses on describing a smooth, one-dimensional penalty term. Using the collision proxies as defined above, a smooth penalty is defined for when the bodies are closer than a threshold ϵ and zero otherwise:[2]

$$\ell(\mathbf{x}) = \sum_{\mathrm{P}} \begin{cases} \left(1 - \frac{\mathrm{d}}{\epsilon}\right)^2 & \text{if } \mathrm{d} < \epsilon \\ 0 & \text{if } \mathrm{d} \geqslant \epsilon \end{cases} \tag{17}$$

Suitable values for ϵ depend on the time discretisation, the velocity limit, the accuracy of the tracking controller, and the target application. Assuming that moving along the normal between the contact/nearest points increases/decreases separation or penetration as in the previous section, we can formulate a derivative of the cost using the geometric Jacobian of the witness points on the collision bodies and their normals:

$$\frac{\delta \ell(\mathbf{x})}{\delta \mathbf{x}} = \sum_{\mathrm{P}} \begin{cases} \frac{2}{\epsilon^2} \hat{\mathbf{n}}_{\mathcal{A}}^{\mathsf{T}} \cdot \mathrm{J}_{\mathbf{p}_{\mathcal{A}}}(\mathbf{x}) - \frac{2}{\epsilon^2} \hat{\mathbf{n}}_{\mathcal{B}}^{\mathsf{T}} \cdot \mathrm{J}_{\mathbf{p}_{\mathcal{B}}}(\mathbf{x}) & \text{if } \mathrm{d} < \epsilon \\ 0 & \text{if } \mathrm{d} \geqslant \epsilon \end{cases} \tag{18}$$

Note, while introduced as a penalty term, this *smooth collision distance* metric can also be used as a fixed-dimensional general constraint. Here, a linear version without the square in (17) can be used.

2 This proxy cost term and its first-order derivative are derived from the source code implementation of AICO as used in Ivan et al. [2013].

2.2.4 *Sum of Penetrations*

An alternate term proposed for instance by Jetchev and Toussaint [2013] is to use the sum of all penetrations as a collision metric:

$$\ell(\mathbf{x}) = \sum_{P} \begin{cases} |d_{\mathcal{A},\mathcal{B}}| & \text{if } d_{\mathcal{A},\mathcal{B}} < 0 \\ 0 & \text{if } d \geqslant 0 \end{cases} \tag{19}$$

As this term is equal to zero when there are no collisions, this can be used either as a constraint or penalty term. Here, we use finite differencing for obtaining its derivative but note that an approximate analytic gradient could be derived similarly to the previous sections. As this is a one-dimensional penalty term based on all collision proxies, it suffers from the same limitations as the *Smooth Collision Distance* in the previous section: Two individual collision instances may have opposing (i.e., cancelling) gradients causing an optimiser to get stuck in a collision state.

2.2.5 *Collision Check as Binary Step Cost*

All the previous formulations made use of the information contained within the collision proxy, particularly the signed distance between two geometric shapes. This information is commonly computed using the Gilbert–Johnson–Keerthi (GJK) [Gilbert et al., 1988] and/or Expanding Polytope Algorithm (EPA) [Van Den Bergen, 2001] algorithms. While efficient for extracting the distance/penetration depth and closest points, they require traversal of vertices. As such, these algorithms are orders of magnitude slower compared to a binary check whether two shapes are in a collision based on their support mappings. The following, simplistic metric exploits this by simply using a step cost if in collision:

$$\ell(\mathbf{x}) = \begin{cases} 1 & \text{if in collision}(\mathcal{A},\mathcal{B}) \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

This penalty term is neither smooth nor continuous but can be used in stochastic and derivative-free optimisation methods. For use in gradient-based approaches, we use finite differencing to obtain a (non-smooth and discontinuous) gradient.

2.3 BENCHMARK

We have now formulated a generic goal state planning problem (Section 2.1) and described several methods to incorporate collision avoidance (Section 2.2). A variety of non-linear optimisation algorithms can solve the problem in Equations (4)–(7). The selection of a specific algorithm (and its particular implementation) has a major impact on the success of the goal state planning stage. Some solver or algorithm implementations—beyond their classification in active-set or barrier methods for handling general constraints—come with more powerful heuristics (e.g., for line search) allowing them to deal better with non-smoothness or different scaling of terms. In the following, we describe a benchmark protocol (Section 2.3.1), a set of formulations using the above-introduced collision avoidance penalties (Section 2.3.2), and a set of solvers and restarting strategies (Section 2.3.3) to evaluate planning success and times with regards to both formulation and chosen solver.

2.3.1 *Benchmark protocol*

In order to test the formulations and solvers, we create a benchmark using the 7-degrees of freedom (DoF) Kuka LWR3+ robot model. After sampling two random, self-collision-free configurations $\mathbf{q}_A$ and $\mathbf{q}_B$ along with a random path connecting the two, we randomly populate the environment with obstacles (here: cubes/voxels with side length 10 cm) up to the desired obstacle density (here: 1 %). When creating the benchmark, we enforced that two valid configurations (start and goal) as well as a collision-free motion path connecting the two exist. We now define a benchmark problem to consist out of one configuration as the initial state as well as the target end-effector placement as obtained from the forward kinematic mapping of the other configuration. Using 1000 distinct benchmark environments with two configuration pairs, we thus have 2000 goal state planning problems with known, existing solutions (see Figure 12). The benchmark environments used here correspond to the ones used in the 1 % benchmark in Yang et al. [2018a]—with the task shifted from finding a collision-free motion plan to finding the required start and goal configurations.

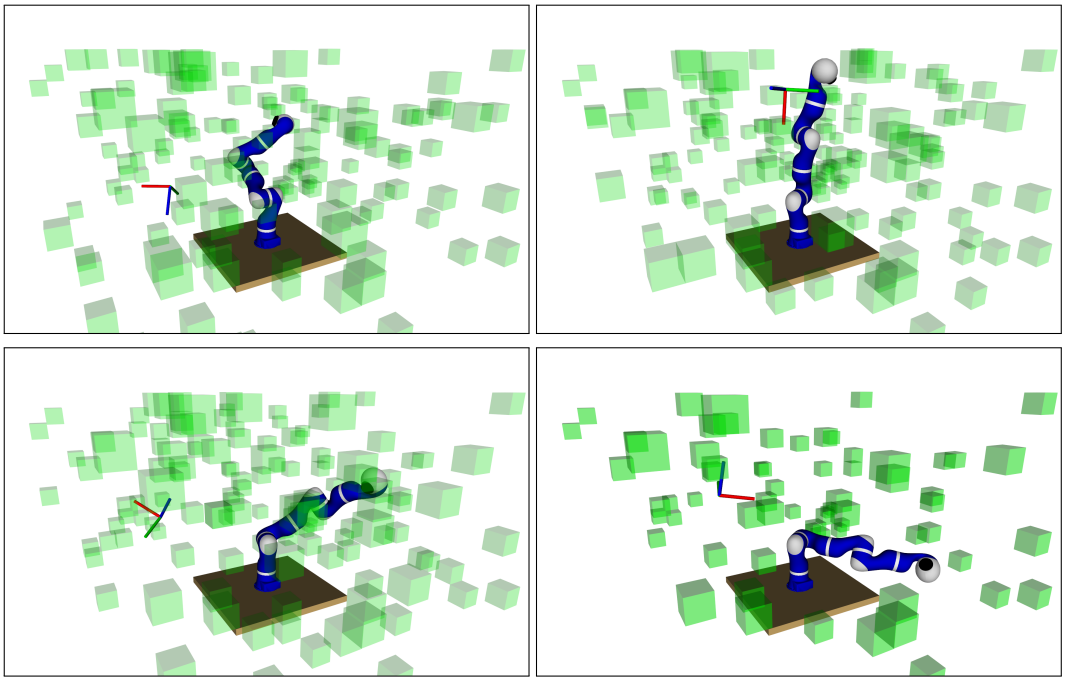


Figure 12: Visualisation of four out of 2000 test cases: The axis indicates the reach frame target (6D) for the end-effector. The manipulator is shown in its initial configuration (seed). We test 1000 different environments across a range of formulations and solvers.

2.3.2 *Formulations*

The generic framework in Equations (4)–(7) assumes that solvers can handle non-linear objectives, equality, and inequality constraints as well as bounds on the optimisation variables. As some methods cannot take these into account directly, we formulate a series of problems belonging to three categories. *Soft-constrained* formulations handle all constraints, including variable bounds, as cost terms. *Bound constrained* problems handle variable bounds explicitly while considering all other tasks as penalties. We refer to formulations for which solvers handle all constraints explicitly as *hard constrained*. We denote problems in these categories with $\mathcal{U}$, $\mathcal{B}$, and $\mathcal{H}$, respectively.

For the unconstrained and bound-constrained problems, we compare three different collision avoidance objectives: Smooth Collision Distance (SCD), Sum-of-Penetrations (SoP) with finite differencing for the gradient, and binary Collision Check (CC) as a step-change cost with finite differencing for the gradient to not considering any collision avoidance objective (None). For the constrained problems, we compare per link closest collision distance as an inequality constraint (Collision

Distance (CD)), smooth collision distance (SCD), as well as no collision avoidance objective (None). An overview of tested formulations is shown in Table 3.

Formulation	Joint Limits	End-Effector Target	Collision Avoidance				
			CD	SCD	SoP	CC	None
$\mathcal{U}_A$	Cost, $\rho_{jl} = 10^4$	Cost, $\rho_{ee} = 10^5$		Cost, $\rho_{ca} = 10^2$			
$\mathcal{U}_B$	Cost, $\rho_{jl} = 10^4$	Cost, $\rho_{ee} = 10^5$			Cost, $\rho_{ca} = 10^2$		
$\mathcal{U}_C$	Cost, $\rho_{jl} = 10^4$	Cost, $\rho_{ee} = 10^5$				Cost, $\rho_{ca} = 10^2$	
$\mathcal{U}_D$	Cost, $\rho_{jl} = 10^4$	Cost, $\rho_{ee} = 10^5$					✓
$\mathcal{B}_A$	Explicit	Cost, $\rho_{ee} = 10^5$		Cost, $\rho_{ca} = 10^2$			
$\mathcal{B}_B$	Explicit	Cost, $\rho_{ee} = 10^5$			Cost, $\rho_{ca} = 10^2$		
$\mathcal{B}_C$	Explicit	Cost, $\rho_{ee} = 10^5$				Cost, $\rho_{ca} = 10^2$	
$\mathcal{B}_D$	Explicit	Cost, $\rho_{ee} = 10^5$					✓
$\mathcal{H}_A$	Explicit	Equality	Inequality				
$\mathcal{H}_B$	Explicit	Equality		Cost, $\rho_{ca} = 10^2$			
$\mathcal{H}_C$	Explicit	Cost, $\rho_{ee} = 1$	Inequality				
$\mathcal{H}_D$	Explicit	Equality					✓

Table 3: Overview of formulations tested in benchmark. Note, we compare formulations that do not explicitly include collision avoidance against several collision avoidance terms. This helps to validate whether the benchmark is too easy (can be solved without explicitly taking collision avoidance into account, or does not have enough redundancy).

2.3.3 *Optimisation solvers*

In the following, we provide an overview of the solvers we used to benchmark and reference the particular implementation of the algorithm. We use the suffix *RR* for a random restart strategy (re-initialisation drawn from a uniform distribution across the configuration space) and *CMAES-R* for a restart strategy in combination with the state-of-the-art, stochastic Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm [Hansen, 2006] for finding a good initial seed for fine-tuning. The application of gradient-free, stochastic optimisation methods, and in particular CMA-ES, have recently seen increased applications, e.g. in trajectory optimisation [Gehring et al., 2016, Agrawal and van de Panne, 2013, Radulescu et al., 2017] or in lieu of gradient estimation in differential dynamic programming [Rajamäki et al., 2016]. We benchmark using the following solvers:

PINV Solves a linear, unconstrained system using the Jacobian pseudo-inverse method.

GN Gauss Newton algorithm for solving non-linear least squares problems.

LM Levenberg-Marquardt algorithm [Moré, 1978] with an initial damping value of $\lambda = 0.01$.

AICO Uses probabilistic inference to solve the goal state problem as a one-step look-ahead version of the trajectory optimisation method described in Toussaint [2009]. Also referred to as *BayesianIK*.

TNEWTON Truncated Newton algorithm [Dembo and Steihaug, 1983]. We use the implementation provided in *NLopt* [Johnson, 2014].

SNOPT A commercial large-scale, sparse optimisation solver, internally using Limited-memory BFGS (L-BFGS) for minimisation of unconstrained objectives and sequential quadratic programming for constrained problems [Gill et al., 2002].

IPOPT An open source large-scale, sparse optimisation solver using the interior-point method for constrained optimisation [Wächter and Biegler, 2006].

KNITRO A commercial large-scale, sparse optimisation solver automatically selecting the most appropriate algorithm for handling constraints (barrier, active-set, or sequential quadratic programming) [Byrd et al., 2006].

2.4 RESULTS

We perform a benchmark evaluation across the formulation (Section 2.3.2) and solver (Section 2.3.3) dimensions and report all results in Table 6 in Appendix A. All evaluations were carried out on a computer with an Intel Core i7-6700K CPU with 4 GHz base frequency and 32 GB 2133 MHz memory, with four individual benchmark evaluations ran in parallel for a total computation time of 17.34 h. Unless otherwise stated, we run the solvers with default parameters. Note, that in the case of IPOPT this uses a less powerful and slower linear algebra solver as with a custom compilation and setting, and for KNITRO requires an initial evaluation to determine the chosen algorithm and initial penalty parameters automatically.

In this section, we show our results using bar plots that indicate the mean algorithm time along with standard deviation and outliers. We sort the results in

descending order of success rate. Here, we define success as returning a configuration which satisfies the end-effector and bound constraints and is collision-free. Including the standard deviation and outliers helps to identify an algorithm that satisfies the particular requirements of an application (e.g., predictable runtime).

2.4.1 *Results for soft-constrained formulations*

The results for the soft-constrained formulations $\mathcal{U}_A$–$\mathcal{U}_D$ are shown in Figures 13, 14, 15, and 16. As expected the random restart variants outperform the single

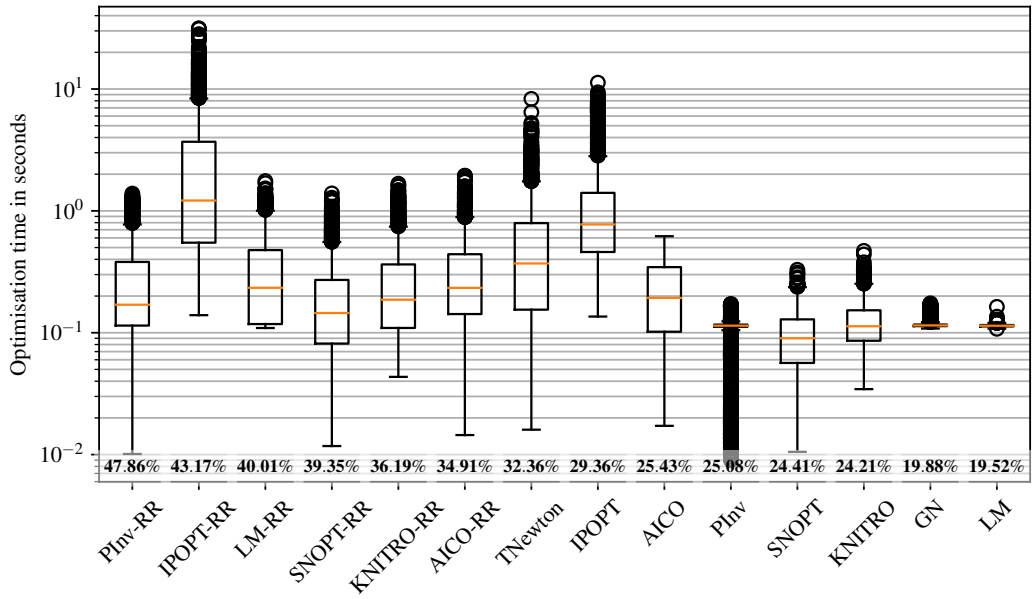


Figure 13: $\mathcal{U}_A$: Joint limits ($\rho_{jl} = 10^4$), end-effector target ($\rho_{ee} = 10^5$), and obstacle avoidance (SCD, $\rho_{ca} = 10^2$) handled as cost terms.

initialisation versions as this allows them to escape potential local minima. We limited the number of random restarts to 10 and note that parallelisation and a higher number of restarts can result in higher success rates for these variants. Surprisingly, the SCD penalty with its approximate gradient has a lower success rate compared with the penalty terms that use finite differencing to estimate the gradient. This observation suggests that the approximate gradient may poorly reflect the actual change in distance. Looking at the failure cases in Table 6, however, paints a different picture: $\mathcal{U}_A$ rarely failed as a result of a collision with most failure stemming from divergence. This result suggests that this penalty is very useful in avoiding collisions (with sharply increasing cost), however, increases the challenge of relative weight tuning in multi-objective optimisation.

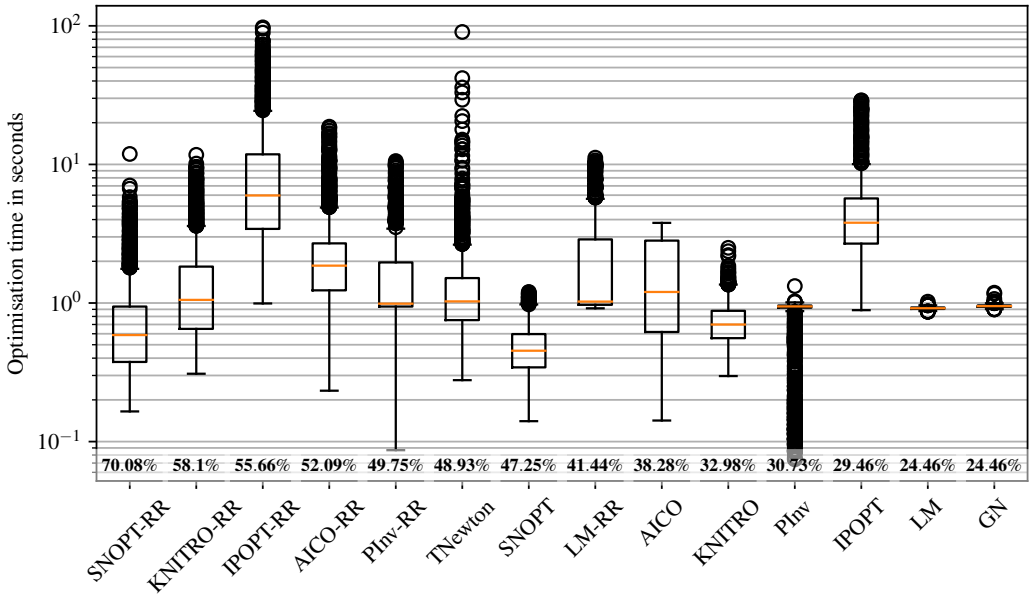


Figure 14: $\mathcal{U}_B$: Joint limits ($\rho_{jl} = 10^4$), end-effector target ($\rho_{ee} = 10^5$), and obstacle avoidance (SoP, $\rho_{ca} = 10^2$) handled as cost terms.

It is important to note that a pure random restart strategy without explicitly taking into account collision avoidance ($\mathcal{U}_D$, Figure 16) performed almost as well as the best collision avoidance term ($\mathcal{U}_B$, Figure 14) and better than SCD and CC. This can also be seen across all later benchmark results suggesting that the benchmark may have been over-constrained and left little ambiguous redundancy to resolve collision avoidance.[3]

In Table 6 in Appendix A, we detail the source of failures. A key observation is that—despite the penalty term for exceeding joint limits—a substantial percentage of failures apart from divergence are due to violation of joint limits. When using solvers that explicitly handle variable bounds as in the following formulations, this failure mode disappears.

2.4.2 *Results for bound-constrained formulations*

The results of the bound-constrained formulations $\mathcal{B}_A$–$\mathcal{B}_D$ are shown in Figure 17, 18, 19, and 20. $\mathcal{B}_D$, which does not explicitly take collision avoidance into account, achieves the highest success rate followed by $\mathcal{B}_B$ which uses SoP. This is in line with the results using the soft-constrained formulations. Analysing the failure cases again shows that SCD is effective at avoiding collisions, however, at the expense

3 In fact, we use a six-dimensional position and orientation task on a 7-DoF robot arm.

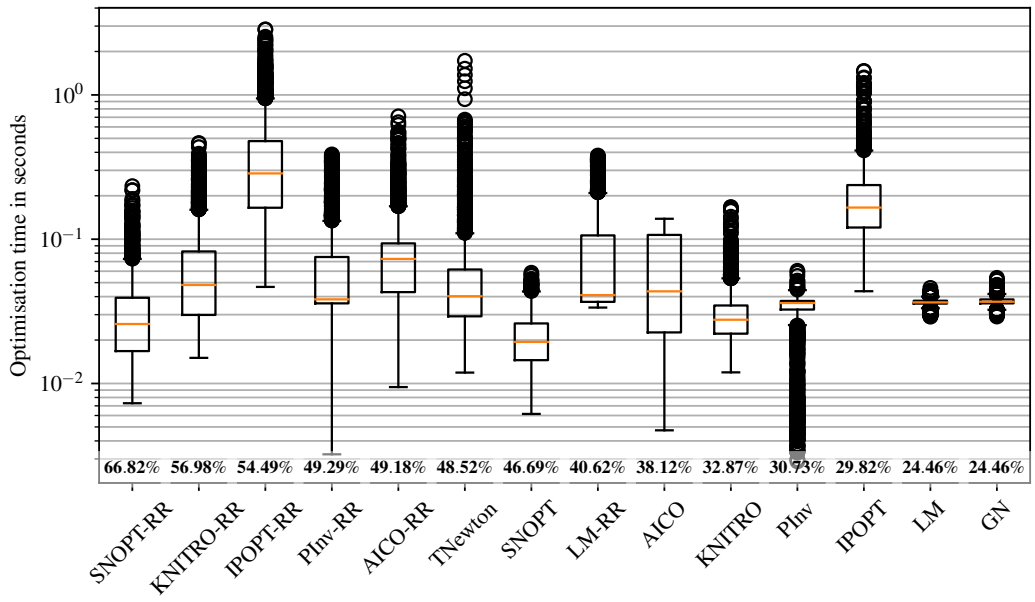


Figure 15: $\mathcal{U}_C$: Joint limits ($\rho_{jl} = 10^4$), end-effector target ($\rho_{ee} = 10^5$), and obstacle avoidance (CC, $\rho_{ca} = 10^2$) handled as cost terms.

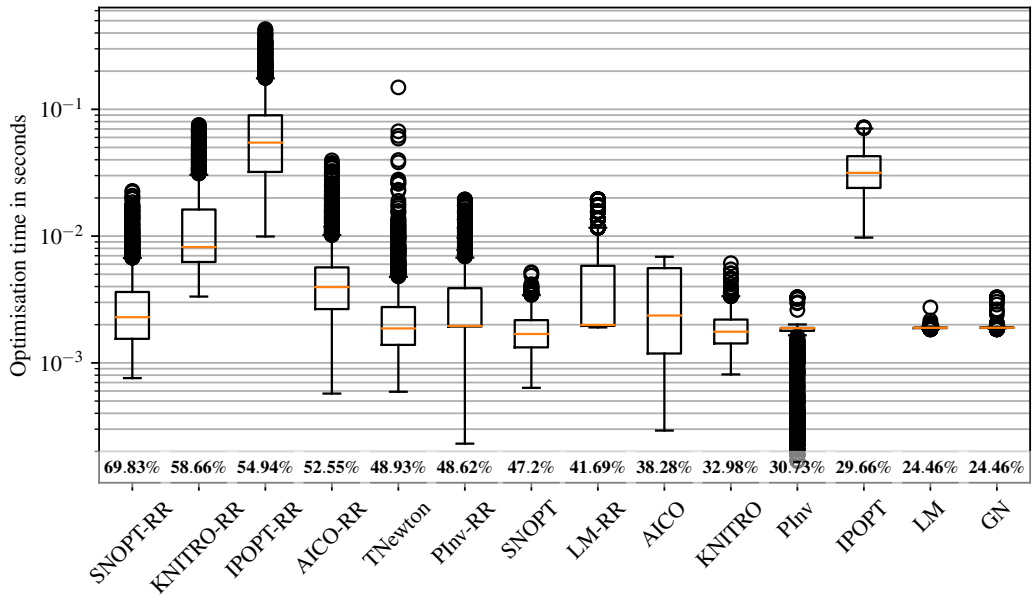


Figure 16: $\mathcal{U}_D$: Joint limits ($\rho_{jl} = 10^4$) and end-effector target ($\rho_{ee} = 10^5$) handled as cost terms. This formulation includes no collision avoidance term.

of poor convergence. It is thus a good choice when a good initialisation can be provided. One example is for instance in interactive goal state planning.

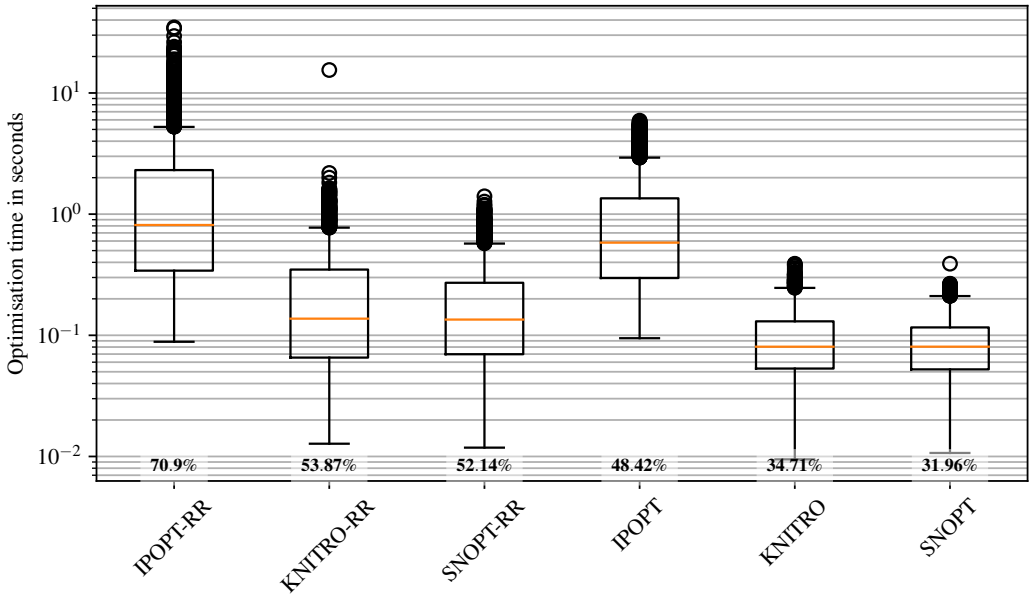


Figure 17: $\mathcal{B}_A$: End-effector target ($\rho_{ee} = 10^5$) and obstacle avoidance (SCD, $\rho_{ca} = 10^2$) handled as cost terms. Joint limits are handled explicitly.

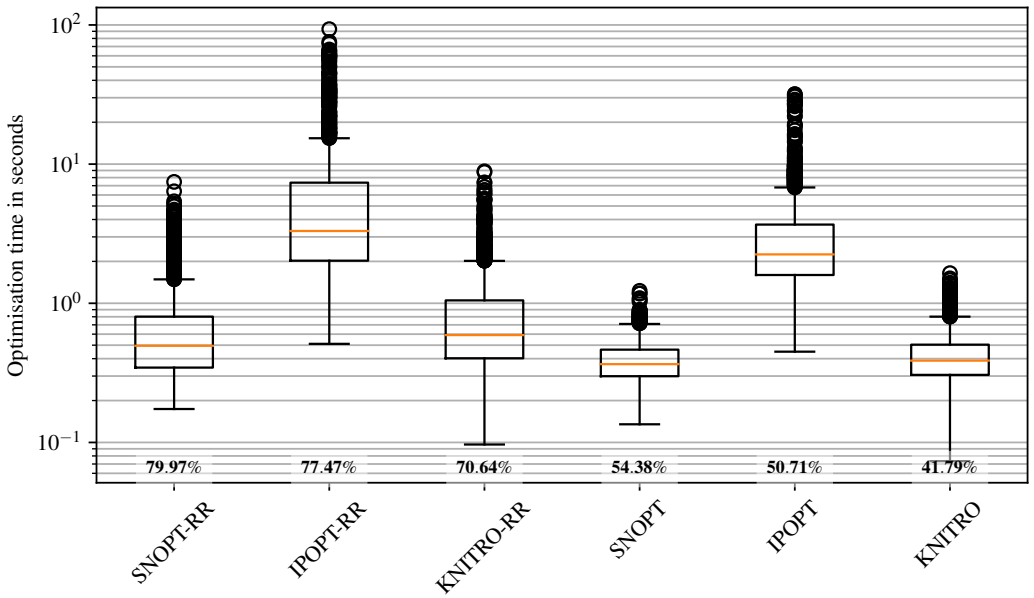


Figure 18: $\mathcal{B}_B$: End-effector target ($\rho_{ee} = 10^5$) and obstacle avoidance (SoP, $\rho_{ca} = 10^2$) handled as cost terms. Joint limits are handled explicitly.

2.4.3 *Results for hard-constrained formulations*

Success rates improve significantly when all tasks are explicitly handled as constraints ($\mathcal{H}_A$). Here, directly using the closest distance between the witness points

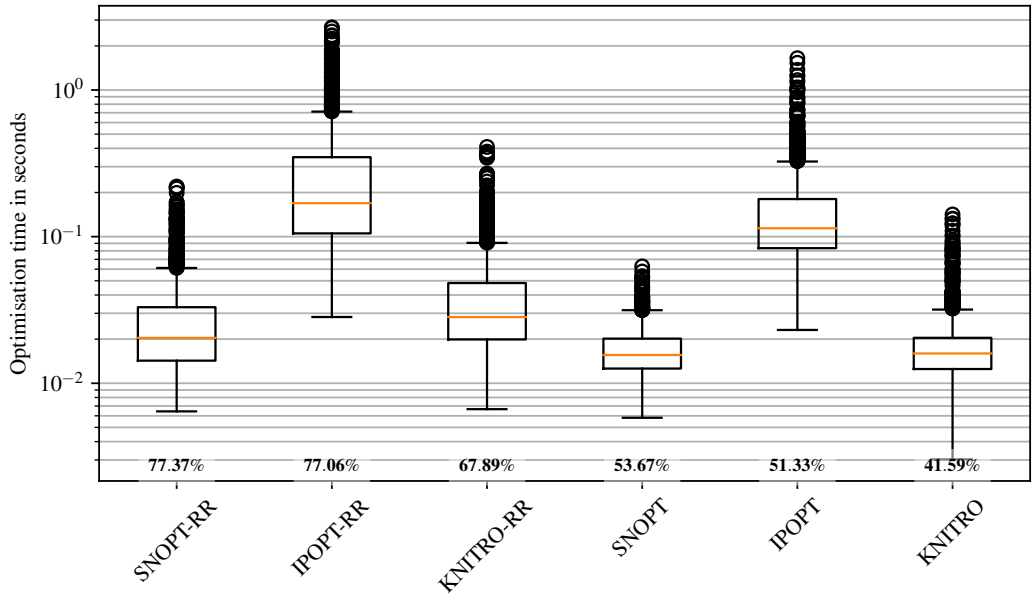


Figure 19: $\mathcal{B}_C$: End-effector target ($\rho_{ee} = 10^5$) and obstacle avoidance (CC, $\rho_{ca} = 10^2$) handled as cost terms. Joint limits are handled explicitly.

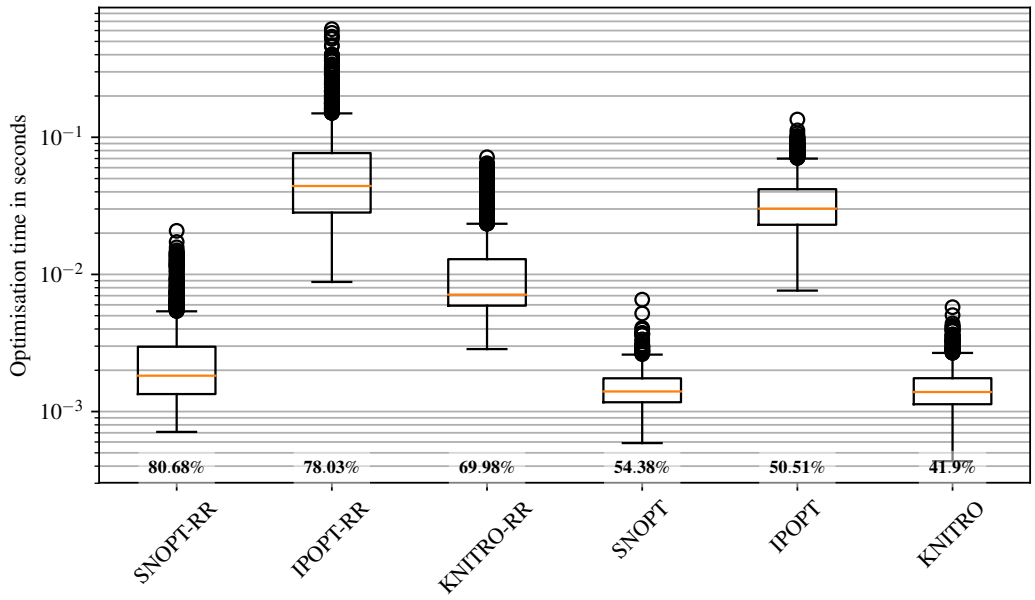


Figure 20: $\mathcal{B}_D$: The end-effector target ($\rho_{ee} = 10^5$) is handled as a cost term and no collision avoidance penalty is considered. Joint limits are handled explicitly.

as an inequality constraint resulted in the highest success rate. When starting from an infeasible initialisation, interior-point methods with a random restart strategy outperform a Sequential Quadratic Programming (SQP) solver with CMA-ES and random restarting. Similar results are obtained when using SCD as a cost with

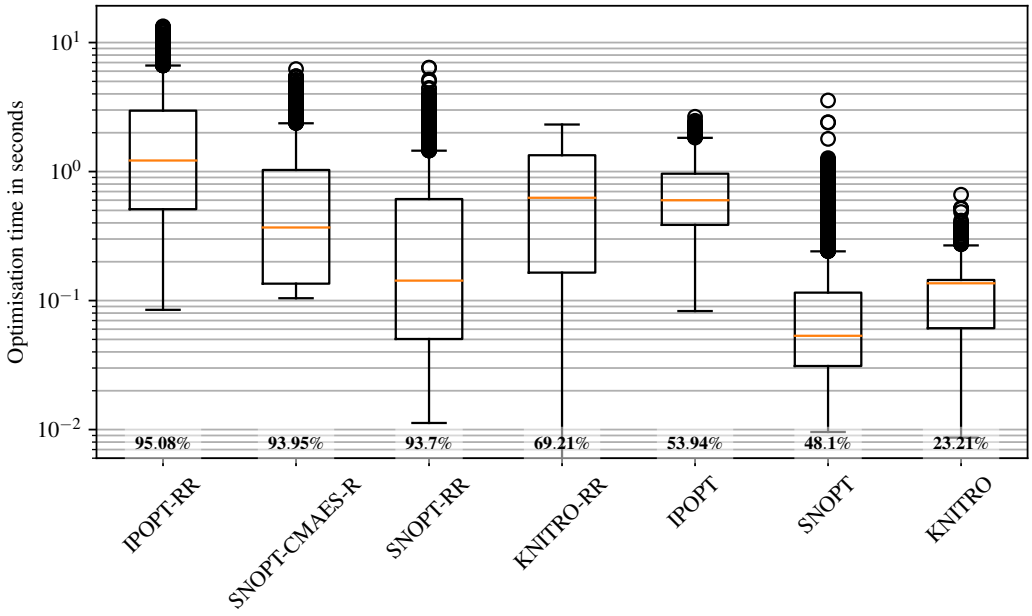


Figure 21: $\mathcal{H}_A$: The end-effector target is handled as an equality constraint and the collision avoidance (CD) as an inequality constraint. Joint limits are handled explicitly.

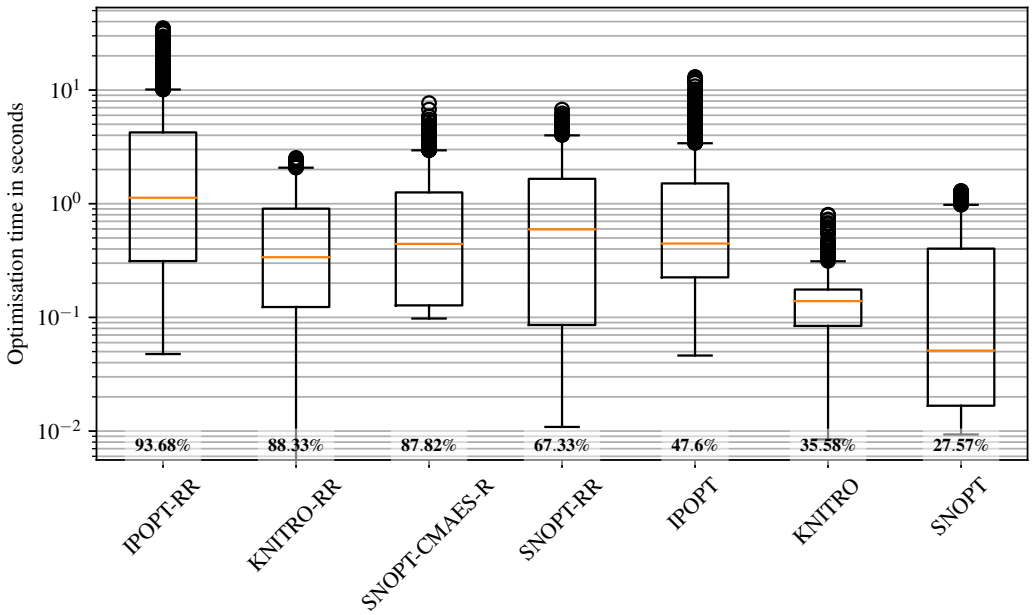


Figure 22: $\mathcal{H}_B$: The end-effector target is handled as an equality constraint and the collision avoidance (SCD) as a cost ($\rho_{ca} = 10^2$). Joint limits are handled explicitly.

the end-effector target handled explicitly as an equality constraint ($\mathcal{H}_B$). This confirms the results from the soft- and bound-constrained benchmarks that SCD is a good penalty for avoiding collisions. Finally, $\mathcal{H}_D$ which did not consider collision avoidance explicitly but used random restarting has a comparable success rate to

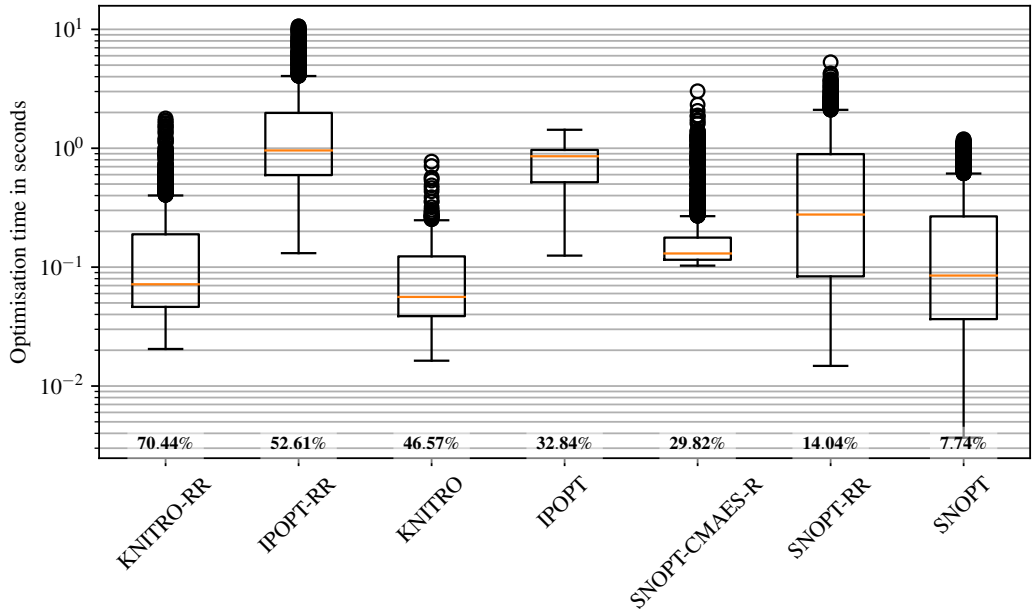


Figure 23: $\mathcal{H}_C$: The end-effector target is specified as a cost ($\rho_{ee} = 1$) and the collision avoidance (CD) as an inequality constraint. Joint limits are handled explicitly.

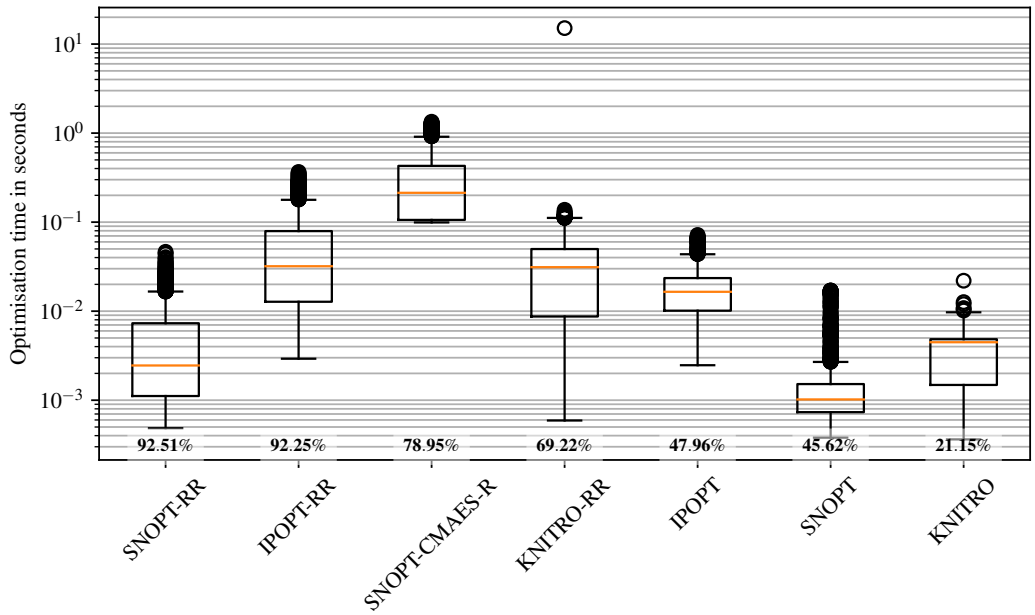


Figure 24: $\mathcal{H}_D$: The end-effector target is handled as an equality constraint. No collision avoidance penalty is included and joint limits are handled explicitly.

the best performing formulation ($\mathcal{H}_A$)—however, with three orders of magnitude lower computation times as no collision distances need to be computed using the GJK and EPA algorithms. This discrepancy in runtime is also partially due to the

best-performing algorithm for $\mathcal{H}_A$, IPOPT, being generally slower compared to SNOPT, the best-performing algorithm for $\mathcal{H}_D$.

2.5 DISCUSSION

2.5.1 *Problem formulations*

Comparing the tested formulations, our benchmark results highlight the importance of handling task success criteria as hard constraints. While high success rates can be demonstrated using expert tuning using multi-objective optimisation, generalisation across a diverse benchmark is a challenge.

In general, soft-constrained formulations with a sum-of-squares cost require careful selection of the weights for the individual cost terms. While we hand-tuned the relative weights for a few examples, the generalisation across the benchmark was poor. Although SoP and finite-differencing on a step cost (CC) appear to outperform SCD as a collision penalty in terms of their overall success rate, an analysis of the failure modes shows that SCD is a suitable collision penalty with failures due to divergence from the end-effector target. This behaviour can be observed frequently when the target is in close vicinity to an obstacle. The challenge of the relative tuning of cost weights can be simplified or addressed by exploring activation functions. One such function is the *groove loss* introduced by Rakita et al. [2018]. It normalises cost terms to a uniform range and provides steeper gradients around the desired value.

A large number of failures can be attributed to the violation of joint limits as confirmed by the breakdown of failure causes in Table 6 in Appendix A. Correspondingly, we observe an increase in planning success rates comparing the soft- and bound-constrained formulations as variable bounds are handled explicitly.

Unsurprisingly, formulations which explicitly handle task constraints using hard constraints show high success rates. Particularly, formulation $\mathcal{H}_A$ which considers end-effector targets, joint limits, and collision avoidance directly has a high success rate and fast runtime if provided with a good initial guess. This initial seed can, for instance, be stored in offline computed maps capturing robot capability and workspace occupation. Work by Yang et al. [2016a, 2017] and Ferrolho et al. [2018] applied Dynamic Roadmap (DRM) methods to provide initial seeds to similar goal state planning formulations. However, as their work did not explicitly consider collision avoidance in the optimisation, they required large numbers

of samples to be stored to achieve high success rates. This limitation poses a challenge both in terms of memory usage and required extensive parallelisation for responsive operation. Additionally, as their goal state planning formulation did not include collision avoidance, they had to resort to reject-and-retry after optimisation if the optimised goal state resulted in a collision. This limitation of inverse Dynamic Reachability Map (iDRM)-based methods can be addressed by the flexible formulation described in this chapter. As a result, smaller datasets can be used in iDRM, which, in turn, can provide a suitable and efficient (re-)initialisation strategy for the formulations explored in this chapter.

2.5.2 *Optimisation solvers*

Regarding the choice of solvers, our results show that commercial implementations of algorithms frequently outperform the same algorithms in open-source libraries. We mainly attribute this to the advanced sets of heuristics they include, e.g., for automatically scaling values and gradients. This effect can for instance be seen in benchmarks that consider a metric with discontinuous values and non-smooth derivatives (e.g., $\mathcal{U}_B$, $\mathcal{U}_C$, and $\mathcal{B}_C$). Many open-source implementations are available for the soft-constrained formulations which use unconstrained optimisation. Their success and numerical stability depends more strongly on the care that has been taken during design of the penalty terms (see $\mathcal{U}_A$) as well as the initial scaling of the cost terms. A majority of observed planning failures stem from failing to satisfy the end-effector target rather than being in collision with an obstacle. This observation motivates the exploration of projection-based optimisation methods which project the optimisation of an augmented objective (i.e., the cost augmented with the inequality constraints through a barrier method) into the null-space of the equality constraints.

Our benchmarks further show that IPOPT, a popular open-source interior-point Non-linear Programming (NLP) solver, is a powerful, general choice when used with random restarting (see $\mathcal{U}_A$, $\mathcal{B}_A$, $\mathcal{H}_A$, and $\mathcal{H}_B$) or where a feasible initial guess can be provided. At comparable success rates, IPOPT is, however, significantly—at times up to an order of magnitude—slower when compared to the commercial SNOPT and KNITRO solvers (see $\mathcal{U}_B$, $\mathcal{U}_C$, $\mathcal{U}_D$, $\mathcal{B}_B$, $\mathcal{B}_C$, $\mathcal{B}_D$, and $\mathcal{H}_D$). During our exploration, we have also empirically observed IPOPT to be more susceptible to effects of scaling compared to SNOPT.

2.5.3 *Restarting strategies*

The results in this chapter show that—in line with expectation—restarting strategies improve the success rates of optimisation-based goal state planning dramatically. Smart restarts, such as through evolutionary strategies, bring a similar improvement in success rates and outperform random restarts depending on the formulation—however, in our benchmark only slightly. This marginal advantage may be in part due to the particular implementation as our initial experiments suggested greater performance differences as in the presented results. In literature, hybridisations of evolutionary methods with efficient first-order optimisation algorithms such as L-BFGS have been proposed with remarkable success rates and runtimes. Starke et al. [2017] uses Particle Swarm Optimisation (PSO) and highly optimised parallel implementations of forward kinematics to achieve real-time performance on high-dimensional systems. However, neither random restarting nor evolutionary strategies utilise prior information on the robot's capabilities, the task, or environment. This information can for instance be encoded in capability maps expressing manipulability information [Vahrenkamp et al., 2015] or including workspace occupation [Yang et al., 2016a, 2017, Ferrolho et al., 2018], and then be used to provide a (re-)starting strategy to the afore-studied formulations.

2.5.4 *Benchmark*

The observation that the majority of observed planning failures stems from failing to satisfy the end-effector target rather than being in collision with an obstacle further suggests that the benchmark is too heavily constrained. Due to the high number of small, unconnected obstacles—which is not representative of realistic environments—a difficult cost/constraint landscape is created with little redundancy left to avoid collisions. As at least one collision-free configuration is known to exist, random restarting without considering collision avoidance proves more successful than to be expected for a realistic benchmark on a higher-dimensional system. As one example, Ferrolho et al. [2018] describe a benchmark protocol for general humanoid goal state planning for shelf manipulation in complex terrain. A more detailed benchmark would further consider planning queries that are known to not have a valid solution. This aspect would allow to test the ability of solvers to quickly determine infeasibility.

2.6 CONCLUSIONS

In this chapter, we introduced a general and flexible framework for optimisation-based goal state planning (Section 2.1) and looked at different formulations for including collision avoidance in optimisation problems (Section 2.2).

While the benchmark comparison in Section 2.3 and Section 2.4 focused on the example of optimising a single configuration, the concepts, framework, and formulations are general. Motion planning for longer horizons in trajectory optimisation is similar to optimising a sequence of individual configurations with transition costs and constraints. In this case, task-maps as described in this chapter are applied at every waypoint of a discretised trajectory.

Collision avoidance, however, becomes more complex when considered across a trajectory. The formulations we looked at in this chapter check collisions within a specified margin around the robot. As robots can move through (*sweep*) large areas of workspace while operating within their velocity limits $\boldsymbol{v}_{\mathrm{max}}$, a challenge arises between the size of this margin and how densely a trajectory is discretised. Large margins frequently conflict with task requirements. As a consequence, this imposes the requirement to have a densely discretised trajectory—and thus large optimisation problem. Nonetheless, a dense discretisation does not provide any guarantees that transitions between waypoints are collision-free. We will address this issue in the following chapter.

3

CONTINUOUS-TIME COLLISION AVOIDANCE IN DYNAMIC ENVIRONMENTS

Common formulations to consider collision avoidance in trajectory optimisation often use either preprocessed environments or only check and penalise collisions at discrete time steps as considered in the previous chapter. However, when motion planning only checks collisions at discrete states, either large safety margins or dense time discretisation are required to ensure that the trajectory and its transitions are collision-free. Large safety margins prevent manipulation close to obstacles or in tight spaces. Dense discretisation in time increases the size, and thus complexity, of the optimisation problem. However, with either approach, collisions may still occur in the interpolation/transition between two valid states, or in environments with thin obstacles, as shown in Figure 25. Here, the workspace swept by the robot in a single transition within the maximum joint velocity is greater than the width of the obstacle.

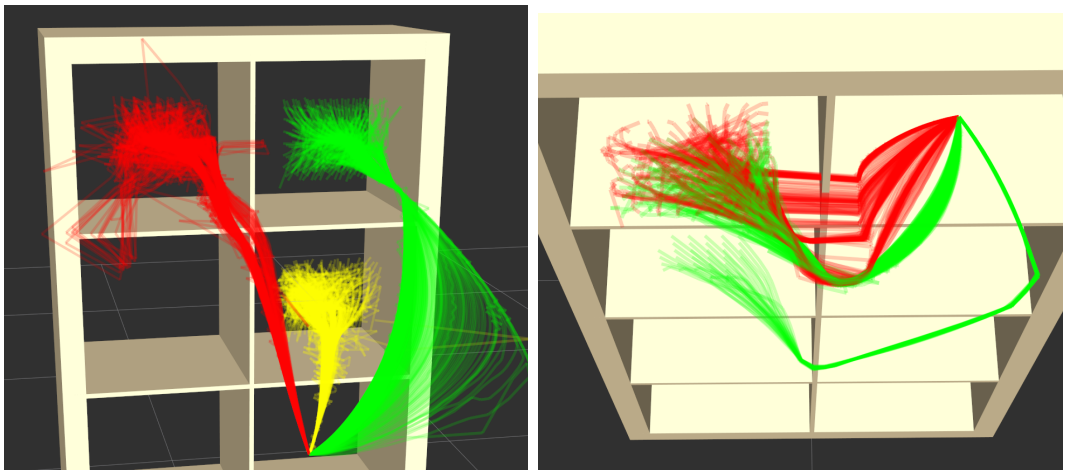


Figure 25: Examples of trajectory samples that are valid with a discrete collision avoidance term, but are in collision during the interpolation (particular in the red cluster).

In this chapter, we introduce a computationally inexpensive continuous-time collision avoidance term in the presence of static and moving obstacles. Our penalty is based on conservative advancement and harmonic potential fields. It can be used as either a cost or constraint in off-the-shelf Non-linear Programming (NLP)

solvers without changes to the optimisation scheme (e.g., outer loop changes to cost scheduling or interactive addition/removal of constraints between iterations). Due to the use of conservative advancement (collision checks) instead of distance computations, our method outperforms discrete collision avoidance based on signed distance constraints resulting in smooth motions with continuous-time safety while planning over discrete states. We evaluate our proposed continuous collision avoidance on scenarios including manipulation of moving targets, loco-manipulation on mobile robots, manipulation trajectories for humanoids, and quadrotor path planning and compare penalty terms based on harmonic potential fields with ones derived from contact normals.

3.1 BACKGROUND

Collision avoidance in motion optimisation primarily focuses on defining penalty terms or non-linear inequality constraints based on signed distance information between pairs of primitive shapes or polyhedral objects $\mathcal{A}$ and $\mathcal{B}$, with their transforms obtained from forward kinematics at the current configuration $\mathbf{q}$. In order to define gradients, local approximations based on motion along the normal defined by the witness points on the respective objects are derived (see Section 2.2). However, in practice this approach suffers from two limitations: (a) numerical instability related to the implementations of the Gilbert–Johnson–Keerthi (GJK) [Gilbert et al., 1988] and Expanding Polytope Algorithm (EPA) [Van Den Bergen, 2001] algorithms in commonly available software libraries, and (b) fast motion of these witness points when surfaces are parallel and the resulting discontinuity in the gradients for non-strictly-convex shapes [Escande et al., 2014]. Additionally, the computation of the signed distance depends on the number and complexity of the collision shapes (number of vertices) and, generally, is expensive to compute.

Many approaches to ensure smooth gradients have thus focused on offline or online preprocessing of the collision model and environment. Stasse et al. [2008] used strict-convexity bounding volumes based on patches of spheres and tori (Sphere-Torus-Patches Bounding Volume, STP-BV, [Escande et al., 2014]) to guarantee the continuity of the proximity distance gradient for real-time collision avoidance on a full-size humanoid robot. Covariant Hamiltonian Optimisation for Motion Planning (CHOMP) [Ratliff et al., 2009] and Stochastic Trajectory Optimisation for Motion Planning (STOMP) [Kalakrishnan et al., 2011] replace the robot collision model with overlapping sphere approximations and use Euclidean Distance Transforms for

the environment. Similarly, for self-collisions only, Sugiura et al. [2006] proposed self-collision-avoidance using an artificial force changing the desired posture target in the null-space of the main tasks using a sphere-swept line bounding volume. In general, simplifications based on replacing polyhedra with enclosing primitive shapes, e.g., the use of cylinders and spheres for a redundant manipulator [Patel et al., 2005] or sphere swept volumes on a humanoid robot [Sugiura et al., 2006], make use of the availability of inexpensive to compute analytic closed-form solutions. However, fitting approximations is a time-consuming (and often manual) process and requires attention to convexity, or, otherwise, local minima may arise [Escande et al., 2014]. Riemannian Motion Optimisation (RieMo) [Ratliff et al., 2015], on the other hand, builds on Riemannian geometry allowing motion near thin or long obstacles which are common failure cases for pairwise signed distance constraints. Alternatively, approaches to limit the computational cost associated with collision queries have been proposed; for instance, adaptive collision checking densities with more checks closer to obstacles [Pavlichenko and Behnke, 2017].

However, these approaches only enforce the collision constraint as an inequality bound on the signed distance at discrete waypoints to a safety margin of ϵ. While there is a direct relationship between the selected safety margin, the maximum joint velocity, and the time discretisation of the trajectory, there is no guarantee to obtain a continuous-time collision-free trajectory.

Sampling-based planners use continuous collision checks that check at distinct interpolations. In practice, some number of sub-samples between two configurations $\mathbf{q}_t$ and $\mathbf{q}_{t+1}$ are evaluated—if one is in collision, the transition (edge) is considered invalid. Dynamic Roadmaps (DRMs) explicitly encode the swept volume of an edge and update the graph based on environment collision information. Available memory mainly limits the number of edges that can be stored. Recently, Yang et al. [2018a] proposed a novel hierarchical variant which explicitly resolves the required ϵ for discrete distance checking. Their method further encodes a mapping for configuration-to-workspace-occupation information which ensures that two adjacent vertices fully envelop the edge connecting them. As a result, one can guarantee that a trajectory is continuous-time collision-free if all of its individual, discrete waypoints are.

However, for trajectory optimisation, continuous-time collision avoidance remains an open challenge and has received less attention compared to discrete-time constraints. Instead, denser discretisation of the time horizon is frequently applied; however, this results in larger optimisation problems. Alternately, a common ap-

proach is to use greater safety margins. However, this restricts close interactions, such as reaching into deep boxes or through narrow gaps.

A notable exception—and similar to our approach—which considers continuous-time safety by constraining the minimum signed distance between swept volumes of convex shapes and non-convex shapes using sequential convex optimisation is Trajectory Optimisation for Motion Planning (TrajOpt) [Schulman et al., 2014]. TrajOpt uses convex-convex collision checking between approximately convex static objects and swept-out (cast) volumes of convex collision bodies to formulate a maximum penetration penalty term as a convex hinge loss. Instead of using the obstacle shapes directly, Deits and Tedrake [2015] apply greedy convex segmentation and mixed-integer optimisation for dynamic quadrotor path planning. This approach ensures that the motion lies within convex subregions of obstacle-free space.

Recently, Hauser [2018] proposed a trajectory optimisation formulation based on Semi-Infinite Programming (SIP) which handles both non-convex obstacles and continuous-time collision avoidance by interactively adding constraints during the optimisation. However, these methods depend on more complex optimisation methods compared with off-the-shelf NLP solvers or require the computation of signed distances, which are an order of magnitude more expensive than a simple, binary collision check.

Continuous-time collision detection, however, is a well-studied problem in computer graphics to address the *tunneling problem* where collisions occur between two simulation time steps (e.g., a fast travelling bullet may otherwise pass through a wall). As a result, methods such as conservative advancement (CA) [Mirtich, 1996, Redon et al., 2005] and continuous-collision detection (CCD) are based on bounding volume hierarchies for polyhedral models. They are an efficient way to compute the time of contact/impact between two objects under motion while guaranteeing not to miss any collisions.

3.2 METHODOLOGY

3.2.1 *Problem formulation*

We consider path planning and motion planning as a constrained minimisation of a canonical optimality criterion (e.g., minimum time, minimum torque, or a higher order smoothness term) subject to bound, equality, and inequality constraints:

$$\begin{aligned} X^* = &\arg\min_X \ \ell(X) \\ \text{s.t.: } &g(X) \leqslant 0 \\ &h(X) = 0 \\ &X_{lb} \leqslant X \leqslant X_{ub} \end{aligned} \tag{21}$$

where a trajectory of length T with uniform time discretisation Δt is represented as the sequence of state vectors $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$.

3.2.2 *Discrete-time collision avoidance*

Traditionally, collision avoidance is incorporated as non-linear inequality constraints of the closest signed distance to the collision shapes of the actuated links at the discrete time points t, subject to a safety margin ϵ: $g_i(\mathbf{q}) = sd(\mathbf{q}_i) - \epsilon \leqslant 0$ (see Section 2.2.2). Similarly, a smooth cost and gradient for unconstrained optimisation can be formulated (see Section 2.2.3).

Determining a suitable safety margin ϵ can prove tricky: While intuitive for the discrete-time case, one needs to choose an appropriately sized ϵ in order for it to capture potential collisions in the continuous-time transition between two waypoints. This ϵ needs to be large enough such that the two collision shapes enlarged by their safety margins overlap (see Figure 26). In particular, without an explicit joint velocity limit, solutions may become discontinuous—they will be valid at the discrete waypoints, but standard transition between two configurations (e.g., straight line interpolation) will result in a collision (see Figure 30).

In order to develop a metric for scaling ϵ for each actuated link i attached to a kinematic chain, an upper bound on the change between two states should be set (i.e., a joint velocity limit $\boldsymbol{\nu}_{max}$). Using this, an approximate scaling can be developed for each link $i \in 1..N$:

$$\epsilon_i = \epsilon_{i-1} + \frac{l_i \cos(\boldsymbol{\nu}_{max} \Delta t)}{2}, \text{ with } \epsilon_0 = 0 \tag{22}$$

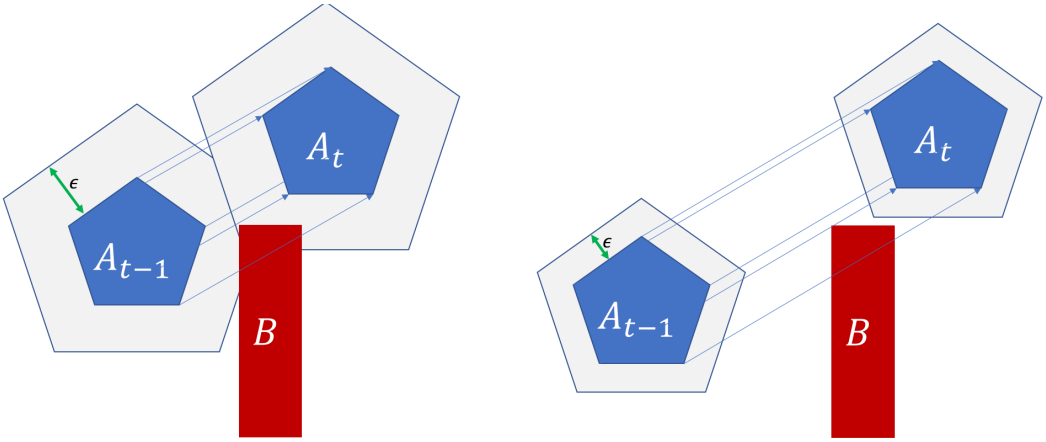


Figure 26: Left: A wide safety margin ϵ along with a dense time discretisation and limited maximum step capture the continuous-time transition between $t-1$ and t as the enlarged collision bodies overlap. Right: In contrast, a small ϵ or large permissible state transition stemming from the time discretisation/joint velocity limit result in the interpolation from $t-1$ to t being in collision with $\mathcal{B}$ while the signed distance constraints at $\mathcal{A}_{t-1}$ and $\mathcal{A}_t$ are both satisfied.

where l_i is the length of the i-th link. As is evident, the ϵ has to be greater for links further removed from the root of the kinematic tree, which makes fine-grained manipulation or interaction in confined spaces impossible.

In contrast, in order to specify a maximum workspace distance, Yang et al. [2018a] developed a relationship between a workspace resolution and a corresponding required configuration space discretisation to guarantee a collision-free edge (i.e., by ensuring the workspace occupation of two subsequent states/vertices overlap). Their work further provided a resolution completeness proof for general deterministically-sampled roadmaps. We can alternately use this approach to determine individual maximum joint velocities given a desired workspace resolution/collision avoidance margin ϵ. Such an approach, however, can easily result in the requirement for a fine discretisation of the time horizon (due to increasingly smaller allowable joint deltas). The method's current form further does not extend to moving multiple joints at the same time as continuous variables, as is required in motion optimisation.

3.2.3 *Continuous collision avoidance*

In practice, it is easy to implement continuous collision checking for sampling-based algorithms by sub-sampling intermediated interpolated states between two waypoints or by applying conservative advancement collision checks. The challenge arises when defining a differentiable cost or constraint term for use in optimisation-based algorithms.

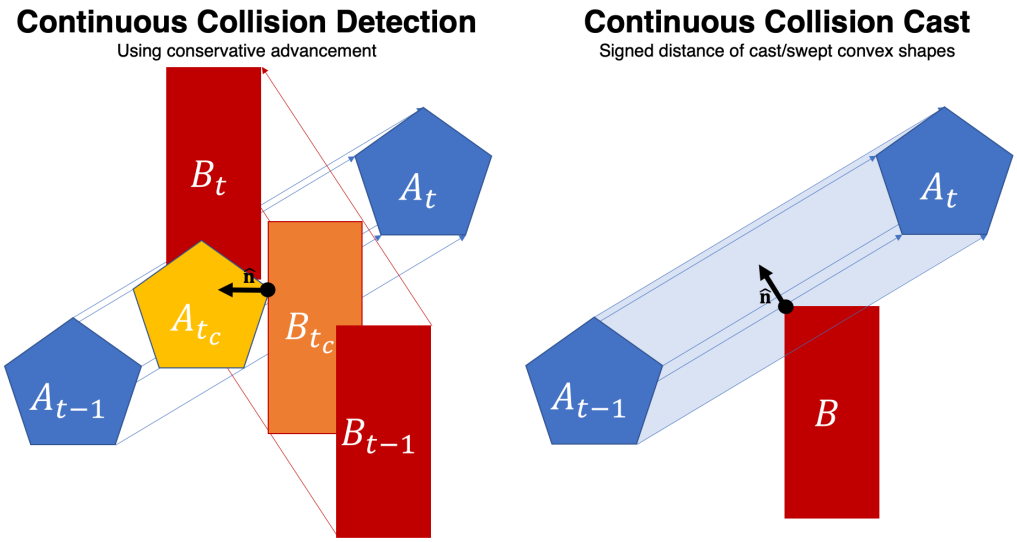


Figure 27: The figure above highlights the two different continuous collision detection modes: conservative advancement (left) and convex collision shape casting (right, e.g., as in Schulman et al. [2014]). Note, the meaning and direction of the collision normal differs significantly. Further, the motion of the objects $\mathcal{A}$ and $\mathcal{B}$ in the continuous collision detection on the left can follow arbitrary, non-linear motion models while the continuous collision cast assumes linear motion (translation only).

In order to define a penalty term, we introduce the concept of a *continuous collision proxy* (CCP) which contains a binary flag c on whether the two objects are in collision along the interpolation, the time of contact t_c if they are in collision, their transforms $T_{\mathcal{A},t=c}, T_{\mathcal{B},t=c}$ at time of contact, the penetration depth between the objects d_p, the contact position $\mathbf{p}_c$ and the contact normal $\hat{\mathbf{n}}_c$:

$$\text{CCP} = (c, t_c, T_{\mathcal{A},t=c}, T_{\mathcal{B},t=c}, d_p, \mathbf{p}_c, \hat{\mathbf{n}}_c) \tag{23}$$

We calculate this information for every pair of robot and environment links by performing continuous collision detection (CCD) for two objects $\mathcal{A}$ and $\mathcal{B}$ given

their initial and final transforms expressed in the fixed reference frame, and a selected motion interpolation model:

$$\text{CCP} = \text{CCD}(\mathcal{A}, \mathcal{B}, \mathsf{T}_{\mathcal{A},t=0}, \mathsf{T}_{\mathcal{B},t=0}, \mathsf{T}_{\mathcal{A},t=T}, \mathsf{T}_{\mathcal{B},t=T}) \tag{24}$$

Here, we use *Conservative Advancement* [Mirtich, 1996] to perform CCD. We note that we can also compute the same information by considering the approximate swept-out volume of the edge as a convex body computed from its support mappings as used in Schulman et al. [2014]. Using a swept-out volume represents a geometrically more realistic interpretation of the direction and depth between the edge and a static obstacle (and additionally includes the signed distance for objects at a distance). In this chapter, we use continuous collision detection as it supports dynamic shapes with arbitrary motion models. As a result, we cannot expect to be able to use the computed normal information directly. The difference between the two approaches is visualised in Figure 27.

If CCD returns that a collision with an actuated link occurs, we formulate a penalty using the continuous collision proxy with the largest penetration depth for the respective link:

$$\phi(\mathbf{q}_{t-1}, \mathbf{q}_t) = \begin{cases} d_p & \text{if } d_p \geqslant 0 \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

with a corresponding, approximate derivative:

$$\frac{\delta \phi}{\delta \mathbf{q}_t}(\mathbf{q}_{t-1}, \mathbf{q}_t) = \begin{cases} -\hat{\mathbf{n}}_c \cdot J_{\mathbf{p}_c} & \text{if } d_p \geqslant 0 \\ \mathbf{0} & \text{otherwise ,} \end{cases} \tag{26}$$

where $J_{\mathbf{p}_c}$ is the geometric Jacobian of the contact point expressed in the local frame.

We note that this derivative is approximate and inaccurate at the boundary of collision (while the scalar metric is not). However, due to the use of continuous collision checks instead of distances to swept volumes (the former of which is computationally significantly faster), we obtain a very fast metric that in practice works well when initialised from a collision-free guess in complex environments. We can provide this guess, for instance, from sampling-based planning or memory-of-motion. To maintain a Markovian structure in our optimisation problem, we further formulate the derivative with respect to (w.r.t.) a single waypoint assuming the previous state fixed.

3.2.4 *Harmonic potential field-based continuous collision avoidance in dynamic environments*

One challenge with conservative advancement is that the computed contact information is the first contact between two moving collision bodies (i.e., the bodies are touching). The returned contact normal is often not representative of the normal vector of the maximum penetration of the cast/swept volume and in practice unstable and hugely dependent on the implementation and its numerical stability.

In order to provide a more robust gradient for the contact penalty term, we utilise a harmonic potential field in place of the penetration depth in (25). Such fields exhibit local minima (saddle points) only due to the topology of the obstacle. This property makes them ideal for gradient descent and optimisation. Indeed, harmonic potential fields have been widely used for avoiding static obstacles [Kim and Khosla, 1991] and for navigation problems [Daily and Bevly, 2008]. A detailed introduction to harmonic potential fields and analysis of their properties is given by Kim and Khosla [1991].

A harmonic potential field around a 3D shape can be derived from equations for electric potential around a uniformly charged object [Wang et al., 2013]. The electric potential arising from a point charge q, at a distance r from the charge is defined as $V = \frac{1}{4\pi\epsilon_0}\frac{q}{r}$. However, we are interested in computing the potential over a triangulated surface of the obstacle's surface.[1] Calculating the approximate harmonic potential field based on the potentials of their constituent triangles and using the principle of superposition enables us to interact with generic shapes. As this is an approximation of the actual harmonic potential field of the original shape, local minima may potentially arise. Goto et al. [1992] analysed the resulting approximation error, and Wang et al. [2013] introduced a linear programming-based method to readjust the local surface charge distribution to achieve a uniform field.

1 To further improve the performance of this method, one can use closed-form solutions for the potential and field of primitive shapes (e.g., boxes, cylinders, and spheres).

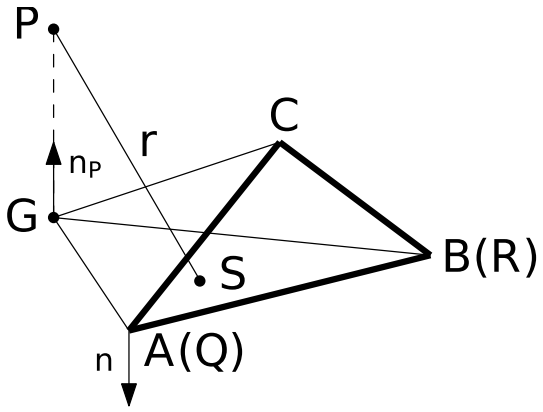


Figure 28: Electrostatic potential due to a charged triangle. Figure reproduced from Wang et al. [2013].

The integral of the potential V over the surface of an uniformly charged triangle ABC (see Figure 28 for definitions of variables) has been derived in Goto et al. [1992] as:

$$V = \frac{1}{4\pi\epsilon_0}\frac{q}{\triangle ABC} I \tag{27}$$

$$I = \pm I_{\triangle ABG} \pm I_{\triangle BCG} \pm I_{\triangle CAG} \tag{28}$$

$$\begin{aligned} I_{\triangle QRG} = &\frac{\sigma}{|\overrightarrow{RQ}|}\log\frac{\overrightarrow{RQ}\cdot\overrightarrow{RG}+|\overrightarrow{RQ}||\overrightarrow{RP}|}{-\overrightarrow{QR}\cdot\overrightarrow{QG}+|\overrightarrow{QR}||\overrightarrow{QP}|} \\ &+ h\arctan\frac{\sigma\overrightarrow{RQ}\cdot\overrightarrow{RG}(h-|\overrightarrow{RP}|)}{\sigma^2|\overrightarrow{RP}|+h(\overrightarrow{RQ}\cdot\overrightarrow{RG})^2} \\ &+ h\arctan\frac{\sigma\overrightarrow{QR}\cdot\overrightarrow{QG}(h-|\overrightarrow{QP}|)}{\sigma^2|\overrightarrow{QP}|+h(\overrightarrow{QR}\cdot\overrightarrow{QG})^2} \end{aligned} \tag{29}$$

$$\sigma = (\overrightarrow{QG}\times\overrightarrow{QR})\cdot\frac{\overrightarrow{AC}\times\overrightarrow{BC}}{\|\overrightarrow{AC}\times\overrightarrow{BC}\|}, h = \|\overrightarrow{QP}\|, \tag{30}$$

where ABC are the vertices of the triangle, P is the query point and G its projection onto the plane of the triangle ABC, ϵ_0 is the permittivity of vacuum, and q is the charge of the triangle. Since the physical meaning of equation (27) is irrelevant in the context of trajectory optimisation, the charge q can be chosen to arbitrarily scale the potential to improve the numerical stability of the solver or as a relative weight against other cost terms. To calculate I, we substitute the permutations of the triangle vertices into equation (29). We then sum the potentials of all triangles to calculate the potential over the whole surface of the collision mesh. We obtain the derivative of the potential, also called the *electric field*, using the chain rule (see Wang et al. [2013] for full derivation and formula).

For computational efficiency, we create and cache a harmonic potential field for each collision body as a set of triangles and transform all queries into the local frame of the collision body. We query the potential of the collision contact point on the surface of the active/moving robot shape within the harmonic potential field of the obstacle. For the moving obstacles, we transform the collision mesh of the moving obstacle halfway between its start and end position. This approximation allows us to avoid computing the mesh of the swept volume of the moving obstacle. We then use the geometric Jacobian of the relative point in the link of the robot to compute a derivative w.r.t. the control variables. Finally, we use the resulting term as a constraint or a cost term in an optimisation problem (Equation 21).

3.3 EVALUATION

We have implemented the proposed collision avoidance method in the prototyping and benchmarking library for motion planning Extensible Optimisation Toolset (EXOTica) [Ivan et al., 2019] using FCL [Pan et al., 2012] for continuous collision detection (Equation (24)) and the Bullet physics engine [Coumans, 2003–2019] for continuous collision casts.[2] Using EXOTica comes with the advantage that all benchmarks use the same forward kinematics, collision checking, objective and Jacobian computation and thus performance differences are attributable to internal optimisation solver computations and the number of problem updates they require to converge to a solution. All evaluations were carried out on a computer with an Intel Core i7-6700K CPU with 4 GHz base frequency and 32 GB 2133 MHz memory in a single thread. In this chapter, we report our results using the commercial sparse, constrained NLP solvers SNOPT [Gill et al., 2002] and KNITRO [Byrd et al., 2006] with default parameters.

All motion problems include the end-effector constraints and goal states with equality constraints and the collision avoidance terms with inequality constraints. The cost function is a smoothness criterion penalising the sum of weighted changes between subsequent states.

2 While we choose to rely principally on FCL for this work, the flexible framework of EXOTica allows for the easy switching of collision solver plug-ins. This reconfigurability opens the opportunity for comparative evaluation of the proposed metrics using other libraries, e.g., NVIDIA PhysX, in future work. In this chapter, we use the ROS-Industrial middleware Tesseract as an interface to the Bullet physics engine as it implements the continuous collision casting as in Schulman et al. [2014].

A video explaining our approach is available at https://youtu.be/tHn9E10zDWo. We describe the test scenarios in the following sections, with the results of our evaluation presented in Table 4.

3.3.1 *Quadrotor path planning*

In this example, we optimise a sparsely discretised collision-free trajectory for a quadrotor (6-degrees of freedom (DoF)) in the presence of obstacles. We constrain the maximum task-space velocities and provide a zero-motion initialisation. The collision environment consists of a wide and thin obstacle which divides the space as depicted in Figure 29. We present the benchmark results in the first and second row of Table 4. It is critical to note that the discrete collision penalty converges quicker than the formulations using continuous collision avoidance penalties. However, the trajectory obtained using the discrete collision penalty is not continuous-time collision-free: It would pass through the thin obstacle as subsequent states are just before and immediately after the obstacle: An interpolation between them would result in a collision. The harmonic potential field-based method converges quickly to a continuous-time collision-free path (shown in Figure 29). At the same time, the contact normal-based penalties result in divergence or termination from numerical instability.

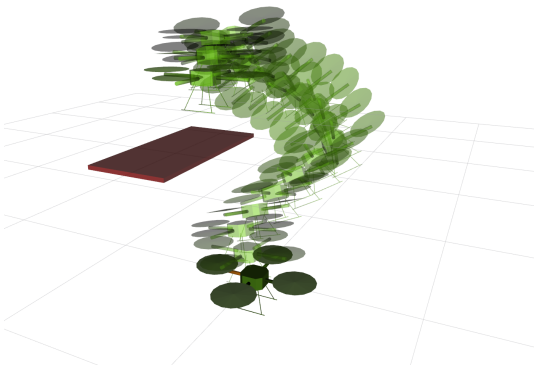
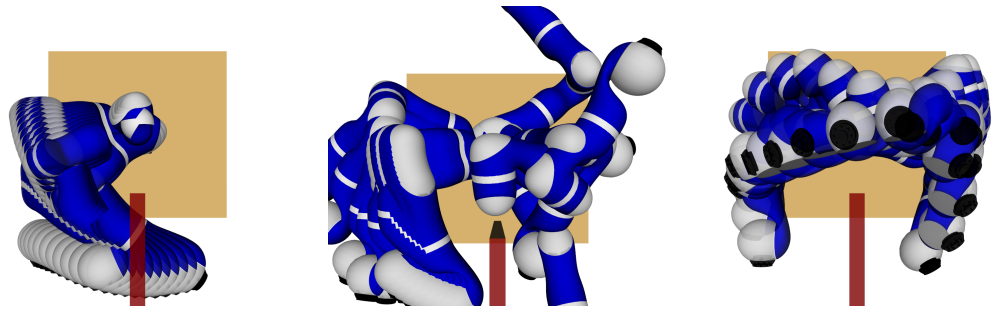


Figure 29: Collision-free path for a quadrotor (zero-motion initialisation, 8.19 ms).

3.3.2 *Motion planning near thin obstacles*

As thin obstacles pose particular problems for collision avoidance constraints at discrete waypoints, we demonstrate a sparsely discretised motion plan (10–20

waypoints) on a 7-DoF Kuka LWR3+ in Figure 30 with benchmark results given in rows 3 and 4 of Table 4. Figure 30a highlights the optimal motion from a start state on the left side of the obstacle to an end-effector constraint on the right side without considering collision avoidance. Figure 30b depicts the optimal solution using a distance-based discrete collision constraint which is valid at each waypoint, but discontinuous and results in collisions during its transition: The motion trail depicts the individual (discrete-time valid) states in the trajectory. As they are spread all across the joint range by up to the maximum allowed joint velocity, connecting subsequent states passes through the thin obstacle. Figure 30c shows the optimal trajectory computed using the proposed continuous collision avoidance constraint—note, the computation times are significantly lower than for the discrete-time penalty.



(a) No collision avoidance (0.05 s) (b) Discrete-time collision avoidance (2.37 s) (c) Continuous-time penalty (0.68 s)

Figure 30: Sparsely discretised motion on a 7-DoF manipulator near a thin obstacle: The discrete collision avoidance inequality constraint results in a discontinuous and infeasible path (although valid as per the discrete objective) while taking longer (due to computation of distances) compared with the proposed continuous collision task which results in a smooth, continuous-time collision-free trajectory.

3.3.3 *Moving environment obstacles*

Collision avoidance in dynamic environments is a challenging problem in motion planning, and in particular for trajectory optimisation. Assuming the trajectory of all objects in the scene is known a priori, time-configuration space sampling or optimisation methods can be employed. As directly solving the full problem with a single optimisation problem was considered intractable with local optimisation methods, Yang et al. [2018b] split the planning problem into a sequence of

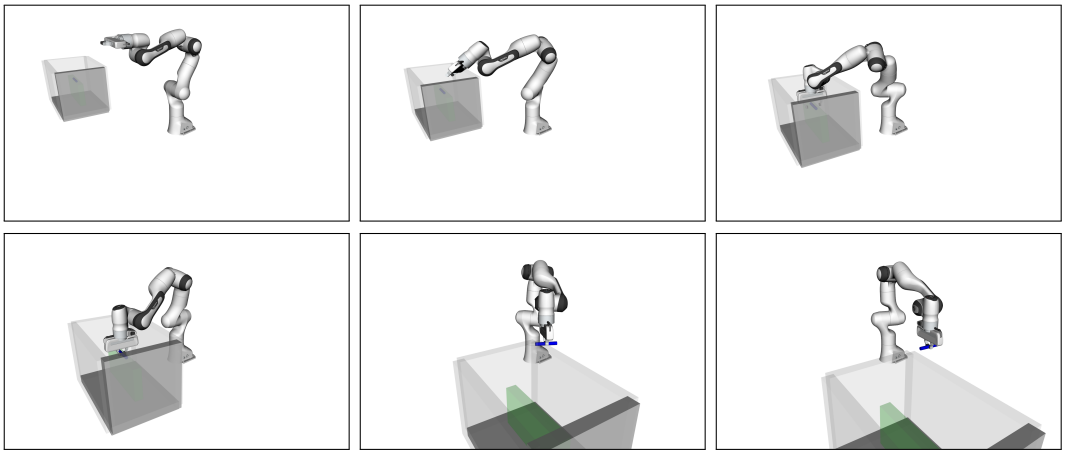


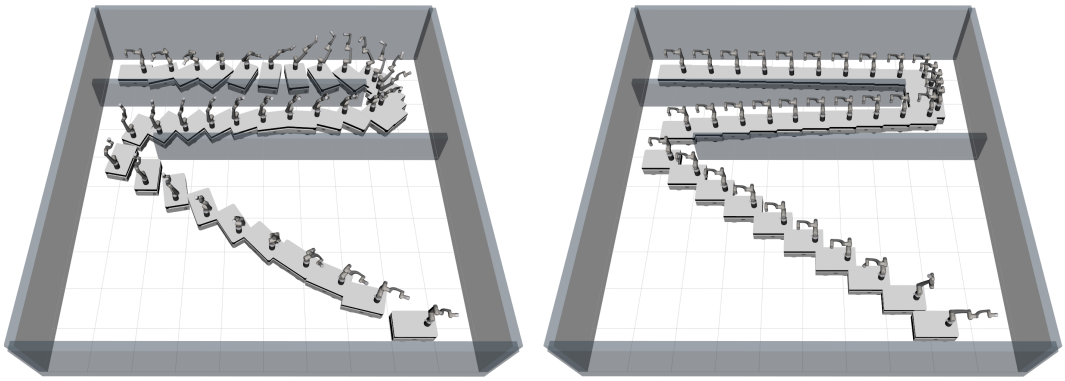
Figure 31: Collision-free pick-and-place of a moving target with a 7-DoF Franka Emika Panda manipulator. The solution has been synthesised from a zero-motion initialisation using the proposed harmonic potential field-based continuous collision avoidance constraint in 7.55 s.

sampling- and optimisation-based sub-problems: Reaching (collision-free, bidirectional sampling-based motion planning), grasping (trajectory optimisation), and placing (collision-free, bidirectional sampling-based motion planning). Here, we demonstrate a scenario similar to those presented in Yang et al. [2018b] as a single optimisation problem using harmonic potential field-based continuous collision avoidance and solve it from zero-motion initialisation. The 7-DoF Franka Emika Panda robot has to reach into a box which moves on a trajectory at a velocity of $0.2\,\mathrm{m\,s^{-1}}$, follow a target object while the fingers are closing, and place it to the side. The proposed method solves the problem with a time horizon of 10 s at a 0.1 s discretisation in 7.55 s (see Table 4, rows 5 and 6, and Figure 31). Using a higher relative function tolerance, a first feasible solution can be found in 1.77 s. The penalties which are based on contact normals and using the information returned from continuous collision casts fail for this application as they were not designed to handle moving obstacles.

3.3.4 *Loco-manipulation*

Loco-manipulation considers navigation/locomotion and manipulation as a unified problem. Loco-manipulation is frequently split into pipeline-based approaches to make the problem tractable to solve. Pipeline-based approaches treat the base placement, navigation, and manipulator motion planning as separate problems

resulting in sub-optimal motions. We consider loco-manipulation planning for a 9-DoF mobile manipulation consisting out of an omnidirectional base (3-DoF) with a 6-DoF manipulator. In a bug trap-like scenario, sampling-based planners for the entire motion deliver sub-optimal results (see Figure 32a). We use the collision-free path as an initial solution (RRT-Connect with dense interpolation between two states to validate edges, 0.35 s) and optimise using the proposed continuous collision avoidance metrics for a smooth whole-body motion (0.40 s, see Figure 32b and Table 4, rows 7 and 8).



(a) Collision-free initialisation/feasible guess computed by RRT-Connect (0.35 s).

(b) Optimised, continuously collision-free loco-manipulation trajectory (0.40 s).

Figure 32: Experiment scenarios using a mobile manipulation platform (9-DoF).

3.3.5 *Shelf manipulation using a humanoid robot*

Planning smooth collision-free manipulation motion on bipedal robots in the vicinity of obstacles is challenging due to the requirement to maintain balance while avoiding obstacles and remain within actuation limits. We focus on manipulation on a shelf with multiple cabinets which introduce local minima and non-convexity. We initialise our trajectory optimisation problem from a global sampling-based planner (RRT-Connect, 2.40 s) and validate the resulting trajectory on the 38-DoF NASA Valkyrie humanoid platform (see Figure 33), with results given in row 9 of Table 4 (0.90 s optimisation time).

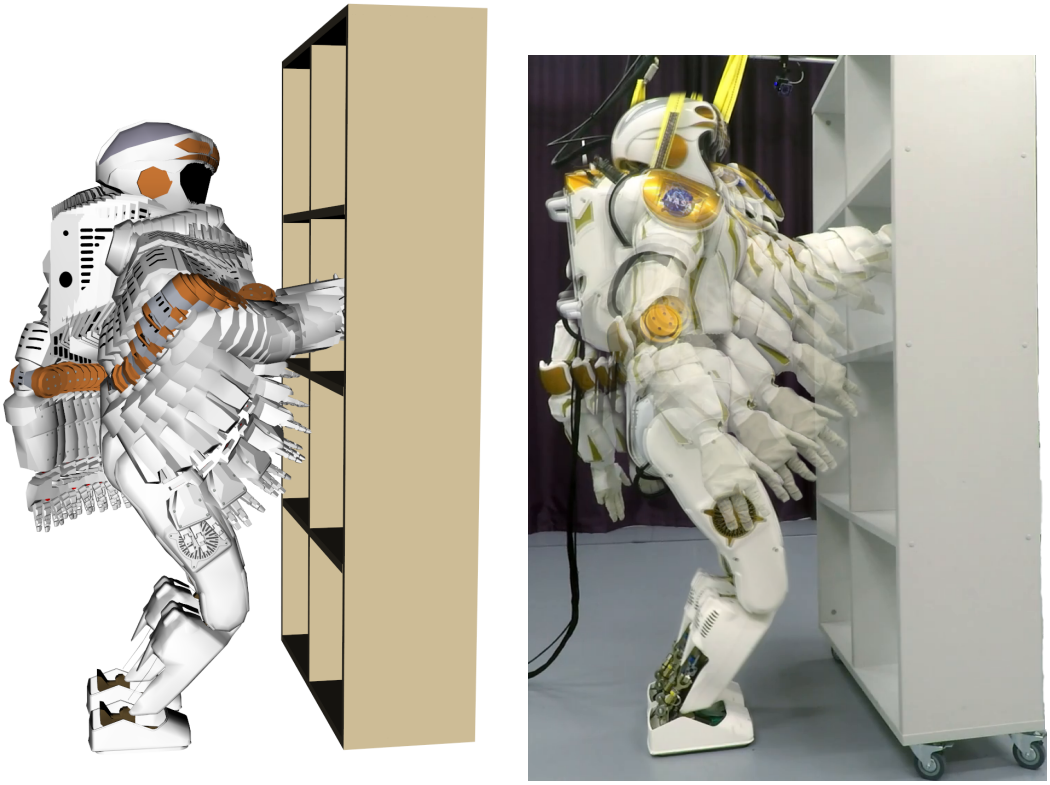


Figure 33: Continuous time collision avoidance during whole-body manipulation execution on the 38-DoF NASA Valkyrie humanoid platform: Discrete metrics will consider a coarsely discretised trajectory to be valid while a transition between two states can be in collision with the environment. Continuous collision avoidance for discrete trajectories successfully avoids the obstacle and guarantees continuous-time collision safety.

3.3.6 *Step-up swing trajectory planning for legged robots*

For legged robots, the swing trajectories of the end-effectors transitioning between different contact configurations, e.g., footholds, are commonly parametrised using low-dimensional polynomials, splines, or three-point interpolations. While this is efficient and often works in practice, it comes with no guarantee that the swing trajectory is collision-free, particularly when stepping up big steps or in cluttered environments—which depending on the planned footsteps can lead to clipped steps and falls. Here, we consider the problem of planning a collision-free trajectory for the swing leg while satisfying the quasi-static balance constraint given swing and double support duration and show the obtained collision-free swing trajectory in Figure 34.

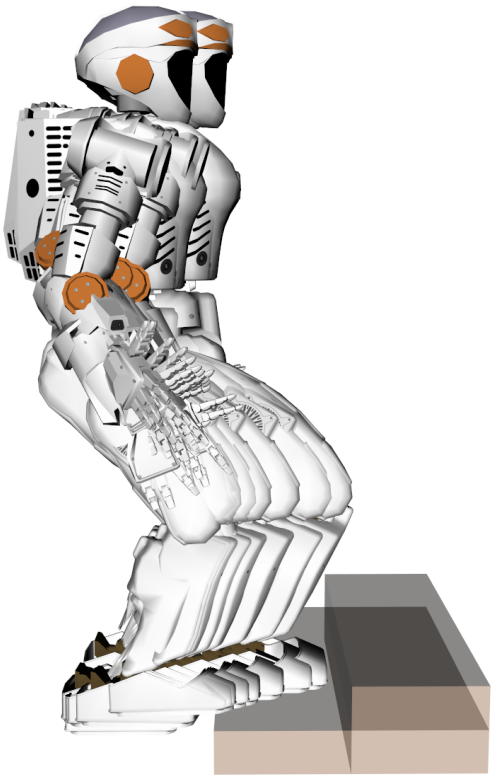


Figure 34: Collision-free, quasi-statically balanced whole-body step-up/swing trajectory (38-DoF).

	Experiment	DoF	Initialisation	Solver	HPF (FCL)	Contact normal (FCL)	HPF (Tesseract/Bullet)	Contact normal (Tesseract/Bullet) Schulman et al. [2014]	Discrete (FCL)
1	UAV	6	Zero-motion	SNOPT	✓**0.008 ± 0.001**	✗0.053 ± 0.001	✗0.019 ± 0.001	✗0.021 ± 0.001	✗0.003 ± 0.001
2				KNITRO	✓0.075 ± 0.006	✗0.581 ± 0.007	✓**0.068 ± 0.001**	✗0.162 ± 0.002	✗0.008 ± 0.001
3	LWR, thin obstacle	7	Zero-motion	SNOPT	✓0.170 ± 0.001	✗0.769 ± 0.011	✗0.786 ± 0.014	✓**0.104 ± 0.002**	✗3.124 ± 0.060
4				KNITRO	✓1.803 ± 0.025	✓3.158 ± 0.024	✓2.293 ± 0.012	✓**1.457 ± 0.029**	✗5.679 ± 0.049
5	Panda, moving obstacle	7	Zero-motion	SNOPT	✓**7.557 ± 0.089**	✓14.178 ± 0.119	✗475.403 ± 4.172	✗105.864 ± 0.126	✗11.528 ± 0.016
6				KNITRO	✗126.665 ± 1.017	✗252.125 ± 3.806	✗400.606 ± 4.264	✗211.840 ± 0.235	✗252.471 ± 1.890
7	Loco-manipulation with mobile manipulator	9	Collision-free (RRT-Connect)	SNOPT	✓**0.396 ± 0.019**	✓0.400 ± 0.029	✓0.431 ± 0.025	✓0.429 ± 0.008	✓7.604 ± 0.049
8				KNITRO	✓**1.855 ± 0.089**	✓7.753 ± 0.233	✓11.625 ± 0.579	✗16.245 ± 0.592	✗839.747 ± 23.034
9	Valkyrie: shelf manipulation	10	Collision-free (RRT-Connect)	SNOPT	✓**0.904 ± 0.228**	✗1.308 ± 0.858	✗1.835 ± 1.721	✗3.877 ± 0.693	✗1.624 ± 0.145

Table 4: Computation times for a selection of motion planning problems indicating their degrees of freedom (DoF), initialisation strategy (zero-motion or collision-free from a global, sampling-based algorithm), and non-linear programming solver. We compare two continuous-time metrics (contact normal and harmonic potential field, HPF) for two different collision solvers (FCL and Bullet), as well as a discrete distance-based nonlinear inequality constraint (using FCL). All computation times are given in seconds and averaged over 10 runs. The best-performing algorithm with a valid continuous-time collision-free solution is highlighted in bold. We indicate whether a trajectory is valid satisfying all constraints and is continuous-time collision-free (as validated by using very dense interpolation) using ✓ and ✗ otherwise.

3.4 DISCUSSION

This chapter considered the formulation of fast continuous collision avoidance penalty terms in the presence of moving obstacles which can be incorporated directly into formulations solved with off-the-shelf NLP solvers. To the best of our knowledge, this is the first work to address continuous collision avoidance for trajectory optimisation in dynamic environments.

In particular, we combine conservative advancement with harmonic potential fields around collision shapes to obtain a smooth, continuously differentiable proxy metric for continuous collision avoidance. We highlighted the versatility of the proposed method on a variety of high-dimensional trajectory optimisation problems from discrete collision-free and zero-motion initialisation. We further validated our motion plans using hardware experiments on the NASA Valkyrie humanoid.

As a limitation, the conservative advancement implementation we deployed in this work relies on linear or screw interpolation between initial and final transforms as a motion model. Due to this, the method cannot handle arbitrarily large transitions. While other motion models could be chosen, they also form an approximation to the actual non-linear motion of the collision body rigidly attached to a kinematic chain. For the case of a continuous collision cast, Schulman et al. [2014] present an analysis of the difference between a linear sweep/cast and the exact non-linear shape and state that, in practice, the difference can often be neglected as it is contained within the safety threshold ϵ.

As we used local non-linear optimisation, our method still frequently requires collision-free (but not necessarily feasible) initialisation in order to converge. It performs well in combination with sampling-based planning, e.g., as part of a hybrid planner where a collision-free guess can be provided quickly from sampling-based or roadmap-based methods and then refined to fit optimality criteria using trajectory optimisation. Alternately, it can be used within a memory-of-motion framework for warm-start initialisation as we describe in Chapter 4. The favourable computation time of the harmonic potential field-based term compared with constraints based on distance computations permits online deployment. This online planning capability is not limited to trajectory optimisation but also applies to path simplification and motion smoothing as commonly employed in sampling-based motion planning. In this application, our novel continuous collision avoidance term enforces the validity of continuous-time motion constraints while resulting in optimal trajectories.

An interesting approach to consider in future work is convex formulations for continuous collision avoidance. Schulman et al. [2014] use a convex hinge penalty on the signed distance obtained from a continuous collision cast in combination with a custom sequential convex optimisation solver and can synthesise motion from an infeasible initialisation. Deits and Tedrake [2015] apply greedy subdivision of the space into convex subregions and use mixed-integer optimisation to plan globally optimal continuously-collision-free trajectories for the dynamic model of a quadrotor in complex environments. However, the scalability of the approach to articulated robots has not yet been investigated. Considering the properties of convex optimisation with regards to convergence and optimality, formulating a convex continuous-collision avoidance constraint in the presence of moving obstacles following these lines of work presents an exciting area of study.

All of the discussed approaches required computing signed distances between polyhedra or preprocessed proxies such as Euclidean Distance Transforms. Campana et al. [2016] avoid the need to compute distance information altogether by iteratively adding (or modifying existing) linear constraints whenever a collision is detected. This approach requires a custom optimisation scheme as the number of constraints differs between iterations. Nonetheless, it would be interesting to explore whether this scheme could be extended to continuous-time collision avoidance due to its low computational cost and numerical stability.

A potential challenge with our method is the case switching in Equation (25) due to the binary collision indicator c. For motion which is continuous-time collision-free, continuous collision detection does not return any contact information. As such, we set the Jacobian to zero when not in a collision which is discontinuous at the boundary in contrast to signed distance-based approaches. Nonetheless, it worked well in practice, particularly from collision-free initialisation. However, further investigating the use of approximate harmonic potential fields for a fully continuous metric is an exciting avenue for future work.

We do not currently explicitly handle self-collisions as part of the continuous collision avoidance term. Instead, we incorporate them through a combination of joint limits and discrete collision costs, which in future work can be learnt as a robot model-specific term.

Finally, we considered path planning and high-dimensional kinematic trajectory optimisation in this chapter. When extending our approach to dynamic trajectory optimisation, care needs be taken for handling inevitable collision states arising from a link's velocity and the system's control limits.

3.5 CONCLUSIONS

In this chapter, we have developed a novel and computationally fast method to account for potential collisions in the continuous transition between discrete waypoints in trajectories. As our method is based on continuous collision detection, it is computationally fast and supports arbitrary, non-linear motion models. We use harmonic potential fields to overcome the challenges of using information from continuous collision detection in motion optimisation directly and pairing harmonic potential fields with continuous collision detection results in a fast and numerically stable continuous collision avoidance metric. Accounting for continuous collision avoidance enables us to use fewer waypoints for discretising trajectories, and thus smaller optimisation problems. Furthermore, our method applies to scenes in which both the robot and objects in the environment are moving. To the best of our knowledge, this has not been shown with continuous collision avoidance for trajectory optimisation before.

Concluding, in this part of the thesis, we introduced methods for (1) finding collision-free goal states, and (2) obtaining optimal collision-free trajectories. However, the convergence time and success rate largely depend on the initial guess. To address these challenges, we will focus on efficient exploration, encoding, and exploitation of a memory-of-motion in the next part of this thesis.

Part III

EXPLORATION, ENCODING, AND EXPLOITATION

4

BOOTSTRAPPING TRAJECTORY OPTIMISATION FROM MEMORIES-OF-MOTION

In the previous part of this thesis, we looked at formulations for collision-free motion synthesis to compute original solutions to parametrised goal state, trajectory optimisation, and optimal control problems. However, both runtime and success rates of this motion synthesis process depend in no small extent on the initial seed used for the optimisation, carefully designed pipelines with sequentially more accurate models and hand-designed heuristics. This strong dependence on a proper initialisation is due to the highly non-linear cost and constraint landscapes, which are often also non-convex and can include discontinuities introduced by the environment or system dynamics. The solutions obtained from the optimisation are further local to their initial conditions. They can become non-adaptable in light of changes or perturbations—exacerbated by uncertainty with regards to the underlying models as well as perception. As thus, the deployment of these methods on platforms for applications with real-time motion planning requirements, e.g., in response to changes in the task, user input, or environment in online replanning or Model-Predictive Control (MPC) fashion is limited.

However, if the parametrised problem space can be sampled during an offline computation phase, a large number of optimal solutions can be explored. Based on a dataset of samples, a variety of approaches can be taken in order to build a memory from which to bootstrap trajectory optimisation as we have outlined in Chapter 1.

In this part of this thesis, we exploit similarity to previously solved problems by creating a memory or knowledge-base through offline pre-computation and fast online look-up for retrieving a similar solution. In particular, we focus on how to efficiently *explore* parametrised optimal control problems during a pre-computation phase, how to *encode* a set of solutions efficiently in a memory-of-motion, and how to *exploit* this to bootstrap trajectory optimisation.

In this chapter, we focus on providing good initialisation seeds for on-the-fly optimal, collision-free motion synthesis on high-dimensional systems in complex and changing environments. We exploit the fact that repeated tasks are similar according to some metric and introduce a problem encoding to build a trajectory

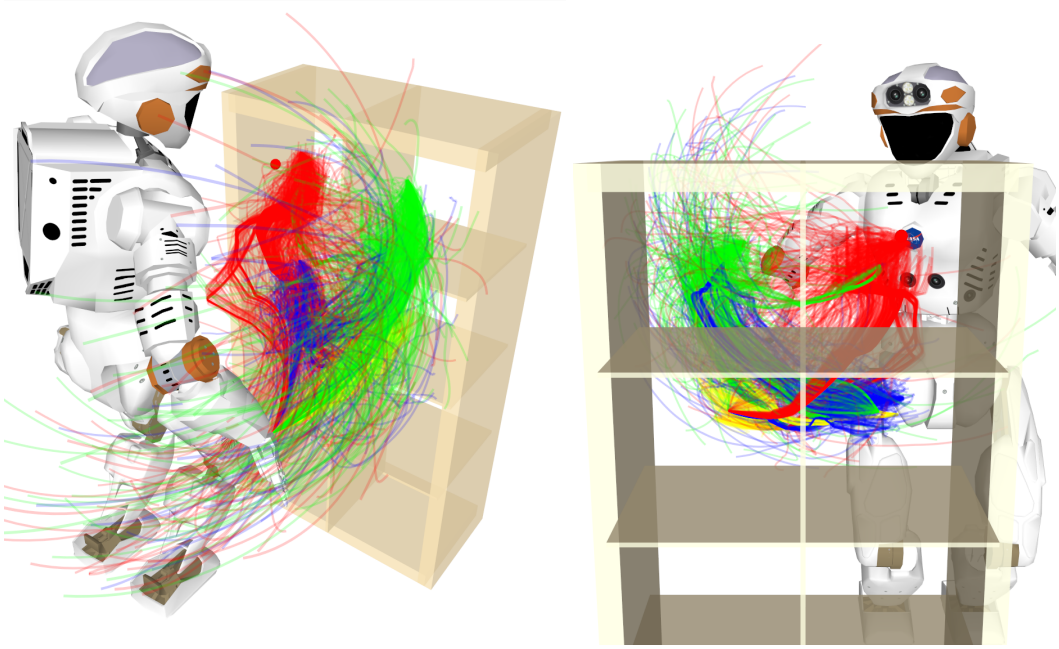


Figure 35: Pick-and-place scenario in a complex environment with collision avoidance. A humanoid robot has to retrieve an item from a shelf, the target of which may change based on perception information or user input. In order to synthesise optimal motion online in a fast manner, we investigate different warm-start strategies for trajectory optimisation methods.

library offline, into which we can index online to retrieve an initial guess. We detail our exploration strategy and explain how solution sub-indexing and goal region growing can be incorporated for efficient solution reuse and to reduce pre-computation and online memory requirements. We compare how different initialisation strategies affect the global convergence and runtime of quasi-Newton and probabilistic inference solvers. Our analysis on the 38-degrees of freedom (DoF) NASA Valkyrie robot shows that efficient and optimal planning in high-dimensional state spaces is possible despite the presence of globally non-smooth and discontinuous constraints such as the ones imposed by collision avoidance. A supplementary video is available at https://youtu.be/Omg3FhVi_Tk.

4.1 BACKGROUND

Common trajectory optimisation formulations differ in their use of derivative and environment information as well as their initialisation strategies. Transcription methods convert the continuous optimal control problem into a Non-linear Pro-

gramming (NLP) subject to equality and inequality constraints. If a derivative can be defined or approximated, unconstrained NLPs can be solved efficiently by Newton or quasi-Newton method algorithms which use second- and first-order information, respectively. Similarly, for formulations with hard constraints, Augmented Lagrangian, interior point, or Sequential Quadratic Programming (SQP) methods can be applied. Recent gradient-based frameworks include Covariant Hamiltonian Optimisation for Motion Planning (CHOMP) [Ratliff et al., 2009], Trajectory Optimisation for Motion Planning (TrajOpt) [Schulman et al., 2014], k-Order Motion Optimisation (KOMO) Toussaint [2014], and Riemannian Motion Optimisation (RieMo) [Ratliff et al., 2015]. Stochastic, derivative-free algorithms which apply Monte Carlo methods on roll-outs include Stochastic Trajectory Optimisation for Motion Planning (STOMP) [Kalakrishnan et al., 2011] and Policy Improvement with Path Integrals (PI^2) [Theodorou et al., 2010].

Approximate Inference COntrol (AICO) [Toussaint, 2009] formulates a probabilistic trajectory model and uses iterative message passing and approximate inference techniques to solve the non-linear stochastic optimal control problem. It is highly efficient in solving unconstrained minimisation problems as the variational message passing allows to update single states repeatedly without having to roll-out the entire trajectory.

As discontinuities and non-smooth cost gradients are difficult to handle, the way collision avoidance is included in the cost and constraint functions has received additional attention. TrajOpt uses approximate convex decomposition and convex-convex collision checking penalising penetrations with a hinge loss. CHOMP and STOMP use the Euclidean Distance Transform and overlapping sphere approximations for the robot body. As a consequence, these approaches require online or offline preprocessing of the environment and robot model. RieMo applies Riemannian geometry to the workspace in order to plan motion in the presence of thin or long obstacles which translate into many local minima. As collision queries are expensive, Pavlichenko and Behnke [2017] proposed an adaptive collision checking density with more checks closer to obstacles. The collision avoidance terms are still in general very likely to create local minima. In order to address this locality issue, an initial trajectory should, therefore, lie in the same neighbourhood as the global minimum.

In practice, motion optimisation frameworks commonly use zero motion initialisation (e.g., RieMo), or straight-line interpolation between start and goal configurations assuming knowledge of a final configuration for instance from inverse

kinematics (e.g., CHOMP, STOMP). These initialisations are often infeasible, and a potentially lengthy stochastic search may be required to find a solution. Alternatively, two-phase optimisation has been proposed where a first result is obtained using a reduced cost and the full optimisation warm-started using this result [Pavlichenko and Behnke, 2017]. Similar to MPC, planning and execution can be interleaved by providing a fixed time budget for planning and warm-starting subsequent optimisations with the suboptimal solution from the previous timestep. Park et al. [2012] follow this principle while updating an estimated conservative-bound location of dynamic obstacles.

In order to speed up and ensure convergence of local optimisation solvers, near-optimal initialisation has been proposed. Here, the similarity to previously solved problems is used instead of planning from scratch. For instance, trajectory libraries and function approximation have been considered to speed up online optimisation [Atkeson and Morimoto, 2003, Stolle and Atkeson, 2006]. Tassa et al. [2008] use a Euclidean nearest neighbour look-up from a library of local control trajectories in a receding horizon Differential Dynamic Programming (DDP) scheme and demonstrate scalability to high-dimensional state and action spaces. To predict warm-start seeds in cluttered environments, Jetchev and Toussaint [2013] focussed on learning expressive task and environment descriptors. Work by Mansard et al. [2018] iteratively approximates the expensive-to-evaluate value and policy function using a neural network-based regression model employed as a distance metric and to evaluate transitions in a kino-dynamic Probabilistic Roadmap (PRM).

On the other hand, Sampling-based Planning (SBP) methods such as Rapidly-Exploring Random Trees (RRTs) or PRMs have been demonstrated to produce feasible motion plans in less than one second for high-dimensional systems in cluttered environments [Elbanhawi and Simic, 2014]. Constrained SBP algorithms can, for instance, take balancing into account and compute whole-body motion plans on humanoid robots in the order of seconds [Yang et al., 2016b]. Recently, Hierarchical Dynamic Roadmap (HDRM) which uses a configuration-to-workspace-occupation encoding to offload collision checking to an offline pre-computation phase has been shown to compute feasible trajectories for industrial manipulators in 1 ms to 2 ms [Yang et al., 2018a]. These results encourage the use of SBPs for warm-starting optimisation-based methods. However, by definition, geometric SBP methods produce a sequence of *kinematically feasible configurations*. These are often long, sub-optimal paths and may contain abrupt changes in velocities and accelerations. In order to be executable, i.e., dynamically feasible with respect to

(w.r.t.) actuation constraints, post-processing such as time spacing, short-cutting, and smoothing are required. This post-processing does not, however, take into account *optimality* such as smoothness in higher-order terms (velocity, acceleration, jerk) or the dynamics of the plant (e.g., torque limits). Previous work explored kino-dynamic sampling-based planners as well as optimal sampling-based planners (such as RRT*). These methods can take a cost function and steering method for connecting samples into account, however, come with challenges regarding the efficient exploration of the vastly increased state-space, and thus long runtimes. In order to address the challenge of sampling valid states, hybridisations of optimisation- and sampling-based planners have been proposed. However, this added sample adaptation step can limit exploration when considering temporal motion constraints such as balance and end-effector orientation during the motion. We argue that SBP may provide a feasible initialisation in a local minimum for an optimisation-based algorithm for one class of problems, but also propose an alternate method exploiting a database of previous solutions. To this end, we analyse different initialisation strategies.

4.2 METHODOLOGY

4.2.1 *Problem formulation*

We consider a trajectory optimisation problem with first-order Markov dynamics, similar to the unconstrained KOMO [Toussaint, 2014, 2017]. Here, we use a pseudo-dynamically weighted state transition cost $\ell_x(t) = (\mathbf{x}_t - \mathbf{x}_{t-1})^T W (\mathbf{x}_t - \mathbf{x}_{t-1})$ where W is a relative measure of the mass ratio of the kinematic chain that is moved by the respective joint: $W_{j,j} = \frac{\sum_{i=j}^{N} m_i}{M}$. After transcription, the synthesis of an optimal motion X^* in an environment Ω is an unconstrained non-linear minimisation problem over the evolution of system states $X = (\mathbf{x}_1, \cdots, \mathbf{x}_T)$, where the time horizon is uniformly discretised into T time steps:

$$X^* = \arg\min_X \sum_{t=1}^{T} [\ell_x(t) + \ell_y(t, \mathbf{y}^*)] \tag{31}$$

$$\text{with } \ell_y(t, \mathbf{y}^*) = \ell_{CoM} + \ell_{JointLimits} + \ell_{Position} + \\ + \ell_{Orientation} + \ell_{CollisionAvoidance} \tag{32}$$

Each term in (32) is defined as a square cost $\rho_t \|\mathbf{y}_t^* - \phi(\mathbf{x}_t)\|^2$. Here, ϕ is a mapping from the configuration space to a task-space with a task-space goal $\mathbf{y}^*$ as introduced

in Section 2.1 and ρ a mixing weight. The task-space is commonly composed of forward kinematics, Centre-of-Mass (CoM), or alternate spaces such as distance or interaction meshes [Yang et al., 2015, Ivan et al., 2013]. We use the following four types of task-spaces in our experiments:

CENTRE-OF-MASS (COM) To maintain static stability on flat terrain, the projection of the CoM has to fall within the convex hull of its contact points (referred to as the support polygon). For quasi-static motions and to account for state estimation error, this support polygon is shrunk by a safety margin γ. In order to favour stable configurations, we apply a quadratic penalty for the deviation of the CoM-projection from the centre of the shrunk support polygon.

JOINT LIMITS To avoid joint limits, we add a quadratic barrier (squared hinge loss) to configurations that are within a set percentage of the bounds of the joint range.

PALM POSITION AND ORIENTATION We penalise the twist from the desired reaching goal and the position and orientation of the end-effector as obtained through a forward kinematic map.

COLLISION AVOIDANCE We use the smooth, differentiable penalty term and its approximate Jacobian as described in Section 2.2.3.

Combinations of the first three terms in the objective function are non-linear, but they are smooth, continuous, and with relatively few local minima. On the other hand, the collision avoidance cost is highly discontinuous because it depends heavily on the geometry of the environment. This frequently gives rise to multiple local minima. Initialising the optimisation in the correct part of the space is therefore crucial.

4.2.2 *Warm-start initialisation*

Since most of the complexity arises from the collision cost, we begin by finding a collision-free trajectory that does not have to satisfy any other optimality criteria. We use a SBP algorithm to obtain a sequence of valid, i.e., collision-free and balanced configurations. We apply path smoothing and short-cutting in a post-processing step and adjust the time parametrisation to adhere to the velocity limit

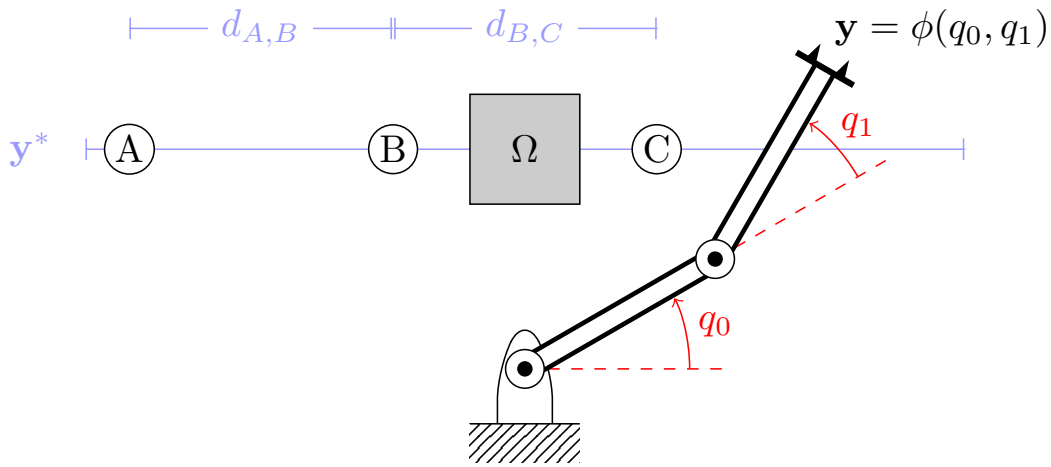


Figure 36: Planar reaching task: Given a start state $\mathbf{x} = (q_0, q_1)$ (which maps to task-space $\mathbf{y} = \phi(\mathbf{x})$), a robot has to reach a task-space goal $\mathbf{y}^* = B$ while avoiding obstacle Ω. The warm-start solutions for task-space goals A and C are equidistant in $\mathbf{y}^*$ from B. Thus, given the same $\mathbf{x}$, both trajectories appear to be equally suitable warm-starts to a simple distance metric.

constraints. Afterwards, the time-spaced sequence of configurations is sub-sampled using spline interpolation to create a uniformly spaced trajectory. In order to verify that the interpolated trajectory continues to be collision-free, we perform additional collision-checks before using the resulting trajectory as an initial seed. Thus, the feasible seed trajectory time horizon T is determined by the maximum joint space velocity and original feasible joint space path length. In order to compare costs across variable-length time horizons, we normalise the cost defined in Equation (31) for the trajectory duration. These initialisations can be generated online (at query time) at the cost of computing a feasible plan from scratch. However, where a task (or a family of tasks) is repeated frequently, a similar or same query has to be computed over and over. An efficient solver would compute these solutions offline and store them in a library. We begin creating a library of motion by defining an indexing scheme and pre-computation strategy.

4.2.3 *Problem encoding*

A task in an environment Ω is fully described by its initial state $\mathbf{x}_0$ and task-space goals $\mathbf{y}^*$. The relationship between the task-space $\mathbf{y}$ and environment Ω can hereby be captured and encoded through parametrised obstacles, topological

[Ivan et al., 2013] or geodesic [Sugiyama et al., 2008] coordinates, relative distances [Yang et al., 2015], or learnt descriptors [Jetchev and Toussaint, 2013]. In this chapter, we focus on initialisation strategies in a known environment leaving considerations regarding generalisation across environments to future work. The start state $\mathbf{x}_0$, task-space goals $\mathbf{y}^*$, and environment parametrisation Ω together form a fully specified planning problem $(\mathbf{x}_0, \mathbf{y}^*, \Omega)$ which maps to an optimal trajectory $X^* = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T)$ if it is solvable:

$$(\mathbf{x}_0, \mathbf{y}^*, \Omega) \mapsto (\mathbf{x}_1, \cdots, \mathbf{x}_T) \tag{33}$$

For a robot with a free-floating base, we capture the relative relation between the robot and the environment by expressing the encoding in an environment frame (e.g., a landmark). In this case we adjust the notation:

$$(\mathbf{x}_{0,\Omega}, \mathbf{y}^*, \Omega) \mapsto (\mathbf{x}_{1,\Omega}, \cdots, \mathbf{x}_{T,\Omega}), \tag{34}$$

where $\mathbf{x}_{0,\Omega}$ has the free-floating base frame expressed with respect to the environment frame.

As the problem encoding captures the relative aspect between the robot and an encoded environment, we can augment the dataset by reusing sub-trajectories of the original solutions as any part of an optimal trajectory can form a near-optimal solution for a similar problem. In particular, we consider two cases for sub-indexing:

1. The sub-trajectory starting from state $\mathbf{x}_{0<t<T} \in X$ to the original task-space goal $\mathbf{y}^*$ forms a near-optimal solution for a problem described by $(\mathbf{x}_{t,\Omega}, \mathbf{y}^*, \Omega)$.

2. The sub-trajectory from state $\mathbf{x}_{0<t<T}$ to the task-space goal defined by a subsequent state $\mathbf{y}^*_t = \phi(\mathbf{x}_{t<i<T})$.

Algorithm 1 shows the procedure for creating these sub-indices.[1] We also visualise the concept of sub-indexing in Figure 37. The originally synthesised optimal solution shown in black can be sub-indexed in three different ways: In blue, we highlight sub-paths from intermediate waypoints to the original task-space goal $\mathbf{y}^*$. The green and orange paths use the forward mapping $\phi(\mathbf{x})$ to obtain the task-space representation of waypoints as intermediate goals. As we only store the problem-solution indices and pairs and not the sub-trajectories on their own, this

1 In Merkt et al. [2018], we included a condensed procedure for computing the green and orange sub-paths shown in Figure 37 within a single `for`-loop. In this chapter, we differ to match with the illustrative example.

Algorithm 1 Create Sub-indexing in Problem-Solution-Map

Require: Pre-computed Library
Ensure: Augmented Library (with sub-indexing)

```
for all Sample ∈ Library do
    for t = 1 to t = T − 1 do
        Problem = (x_{t,Ω}, y*, Ω)
        Start = t
        Length = T − t
        StoreSubIndex(Problem, Start, Length)
    for t = 1 to t = T − 1 do
        Problem = (x_{0,Ω}, φ(x_t), Ω)
        Start = 0
        Length = t
        StoreSubIndex(Problem, Start, Length)
    for t_start = 1 to t_start = T − 2 do
        for t_end = 2 to t_end = T − 1 do
            if t_start ≠ t_end then
                Problem = (x_{t_start,Ω}, φ(x_{t_end}), Ω)
                Start = t_start
                Length = t_end − t_start
                StoreSubIndex(Problem, Start, Length)
```

represents an efficient way of augmenting the available solutions. Note, that this is suitable for look-up methods as the returned trajectory seeds will be of different waypoint lengths. For function approximation, they would have to be interpolated to a uniform length. Further, depending on the formulation (e.g., whether uniform task running cost or only final state costs are used), the sub-indexed paths may not be optimal, but merely feasible. Using this sub-indexing method, along with the goal region growing, we obtain a dense problem-solution-map.

4.2.4 *Trajectory library pre-computation*

It is important to note that the task-space goals $\mathbf{y}^*$ as a parameter of (31) may change the landscape of the objective function significantly and discontinuously. A good initial guess should be local in the value function (have a similar cost landscape); however, this may not translate to closeness in the chosen problem

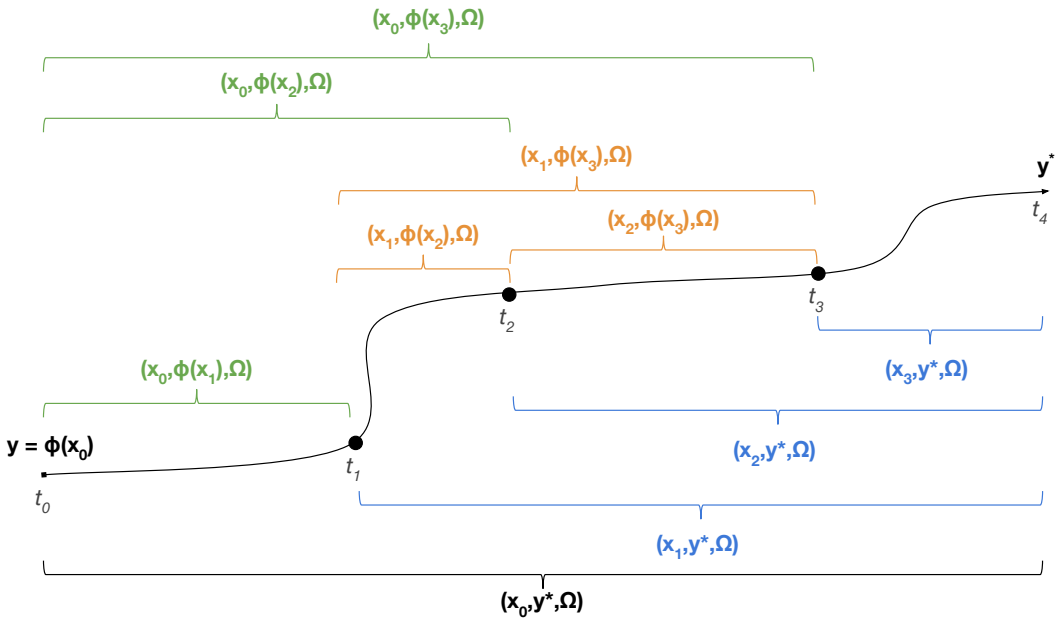


Figure 37: Data augmentation of an original solution (black) using sub-indexing: The blue solutions are optimal sub-paths to the original goal, while the green and orange paths are to task-space goals defined by the way points.

encoding space. Consider the two-link reaching task depicted in Figure 36, where the end-effector has to repeatedly reach different positions $\mathbf{y}^*$ along the blue line. By solving the problem from varying start configurations $\mathbf{x}_0 = (q_0, q_1)$ for changing task-space goals $\mathbf{y}^*$, we can create a library mapping problem encodings to optimal trajectories. A key consideration at this moment is the trade-off between pre-computation time and storage size and online retrieval, convergence time, and adaptation success rate. Exhaustively enumerating the problem encoding (and thus the value function) is intractable for higher-dimensional problems. Finding representative sample trajectories and determining a sufficient sample density is non-trivial as the cost landscape (value function) of the task is not known a priori.

Random sampling in the space of the problem encoding is unlikely to achieve sufficient coverage in task-relevant parts of the space, with many samples prone to being unused and possibly resulting in a low warm-start success rate. Deterministic sampling in a discretised problem encoding space can result in the issue highlighted in Figure 36: Depending on the resolution of the discretisation, two warm-start seeds (A and C) may appear equidistant to a simplistic distance metric. However, this does not necessitate a similarity in the optimal solutions (e.g., mode changes introduced by the environment or dynamics). On the other hand, the Euclidean metric in the problem encoding space is fast to compute, and scaling of the distance

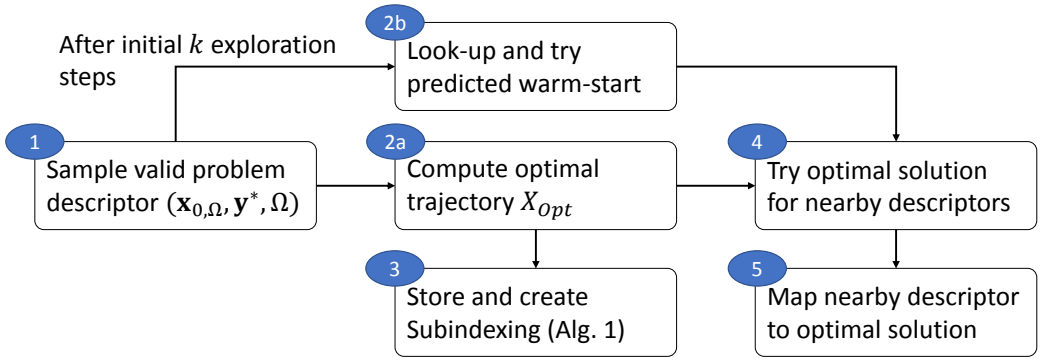


Figure 38: The pre-computation procedure for building the trajectory library: First, a valid problem descriptor is sampled in the space of the problem encoding and an optimal solution computed offline. The optimal trajectory is stored and sub-indexed using Algorithm 1. Finally, goal region growing is performed by testing the adaptability to nearby descriptors.

metric can be used to improve the quality of the retrieved initial guess trajectories. Thus, with a focus on a suitable sampling bias and strategy, efficient look-up can be performed using inexpensive distance metrics.

Our pre-computation and exploration strategy is shown in Figure 38. In general, we aim for density in the problem encoding space while storing only a limited number of solutions as computing new optimal trajectories from scratch is expensive, while adapting existing solutions is comparatively cheap. As such, upon obtaining a new optimal solution, we attempt to explore its region of validity by sampling in the neighbourhood of the problem encoding and running an adaptation optimisation step with a given time and iteration budget. We refer to this as *goal* or *validity region growing*. If the solver converges, we add a mapping from the new sampled problem descriptor to the original optimal solution. Thereby, we achieve density in the problem encoding for a small number of original solutions. This process can also be seen as creating a basin of attraction around the original sample.

In our approach, we use a combination of task-space goal seeds, random exploration, and goal region growing. This process results in an encoded trajectory library mapping problem descriptors to variable-length optimal trajectories. At query time, we predict the best warm-start seed given a problem encoding by similarity look-up.

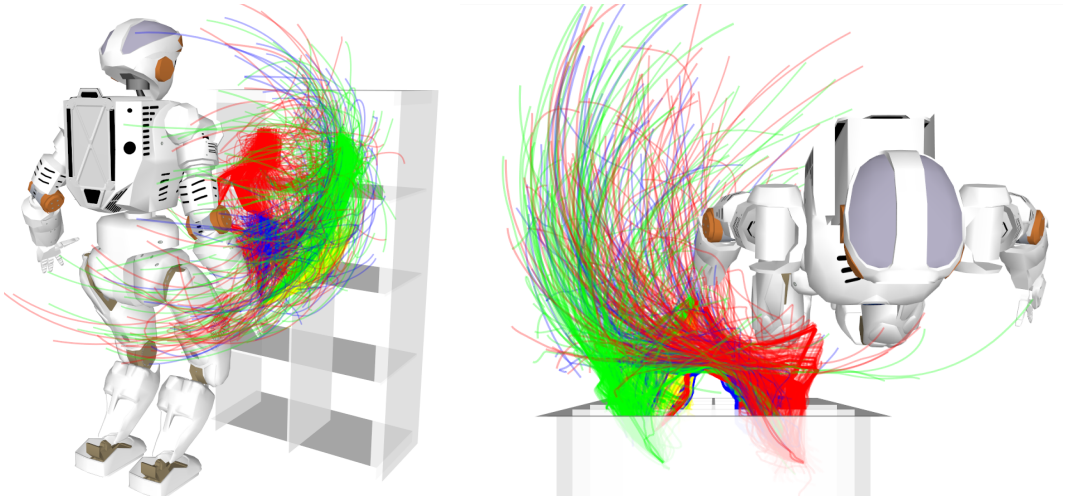


Figure 39: Visualisation of the original optimal trajectories in the library. The class labels (color) have been assigned based on closest task-space goal $\mathfrak{y}^*$.

4.3 EVALUATION

Having pre-computed a library of warm-start trajectories, we now benchmark the proposed warm-start method against zero motion, straight-line linear interpolation, and an online RRT-Connect-based feasible initialisation. We also compare different NLP solvers. For a fair comparison, we use the same forward kinematics, collision checking, objective, and Jacobian computation with the planning prototype and benchmarking library Extensible Optimisation Toolset (EXOTica) [Ivan et al., 2019]. Performance differences between solvers thus correspond to the number of evaluations each algorithm requires and their internal updates. While we implemented the problem formulation and solvers in C++, we use Python for high-level logic and initial guess look-up. All evaluations are carried out on a computer with an Intel Core i7-6700K CPU with 4 GHz base frequency and 32GB 2133 MHz memory in a single thread, with several independent benchmark or pre-computation processes running in parallel. All our evaluations use densely sampled time horizons with $\Delta t = 0.05s$. It is important to note that the choice of Δt has a direct impact on the size of memory as well as the computation time. We discuss methods and embeddings for a sparse parametrisation in Section 4.4.

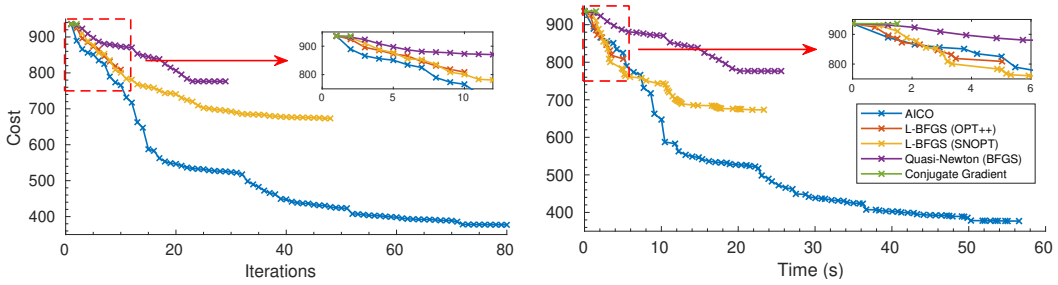


Figure 40: Cost evolution for different optimisation solvers using the same feasible initialisation seed. Left: Cost vs. iterations. Right: Cost vs. time. AICO initially improves the fastest and achieves the lowest overall cost. For quasi-Newton solvers, a commercial implementation with Limited-memory BFGS (L-BFGS) updates performs best (SNOPT, orange). For open source quasi-Newton solver implementations, a solver using Broyden-Fletcher-Goldfarb-Shanno (BFGS) updates for the Hessian achieves a lower cost than its limited memory variant (OPT++, red), however, at the cost of taking more time per step to estimate the full Hessian.

4.3.1 *Convergence using different solvers*

We evaluate the problem formalised in Section 4.2 with a probabilistic inference solver (AICO), quasi-Newton methods (BFGS and L-BFGS), as well as Conjugate Gradient (CG) descent for a humanoid shelf picking task. We use our own implementation of AICO, the open-source quasi-Newton and non-linear CG solvers from OPT++ [Meza et al., 2007], as well as a commercial L-BFGS implementation (SNOPT, [Gill et al., 2002]).

The cost evolution against time and iterations for optimisation until convergence from a feasible RRT-Connect initialisation are shown in Figure 40. AICO is quick to make progress and achieves the lowest overall cost (as it updates individual states repeatedly without a full roll-out). However, it may return invalid trajectories (i.e., those which diverge from the final target position or have a sample in a collision state). Quasi-Newton methods do not achieve a similarly low cost as AICO as they stay within the initially provided local optimum. This behaviour, however, leads to a higher success rate from feasible initialisations as they would not update individual states to be in collision in order to satisfy a smoother transition cost. As expected, full BFGS updates require more function and gradient evaluations to estimate the Hessian and are thus slower than their limited-memory variant. In turn, they can use a more accurate Hessian approximation to achieve a lower final cost. This performance gain of L-BFGS is more pronounced for longer time

horizons. Overall, as AICO and SNOPT have the highest convergence rates with the lowest costs, we will focus on them in the following benchmark experiments.

4.3.2 *Pick-and-place benchmark*

For the shelf benchmark, we compare different initialisation strategies for use with AICO and SNOPT's L-BFGS: Common (zero-motion and straight-line interpolation between the start and goal configurations), feasible (RRT-Connect), and the proposed trajectory library with problem encoding.

The trajectory library used in the benchmark contains 3,233 original solutions with 36,256 problem-encoding-to-solution-mappings after sub-indexing and goal region growing (24 MB, 12.6 h single thread pre-computation time). These original trajectories are shown in Figure 39. For feasible initialisations during offline pre-computation, we use a bidirectional SBP algorithm (RRT-Connect, [Kuffner and LaValle, 2000]) with L-BFGS as the optimisation and goal region growing/adaptation solver. We initialise the exploration with task-space goals to reach into each shelf compartment from a nominal whole-body posture as well as from within each compartment to every other compartment. We apply goal region growing and limit the number of additional random exploration steps to 500.

For fast indexing, we compare k-Nearest Neighbour (KNN) with an exhaustive search over the problem encoding as retrieval strategies in the following settings:

1. KNN with a Euclidean distance metric tested both with and without scaling by normalising on the range (rows 7–10). The normalisation is helpful as the problem encoding can include metric with non-metric information unknown to the look-up distance metrics.

2. Exhaustive search by retrieving the solution of the problem descriptor with the minimum norm, i.e., the closest in the problem encoding. We compare normalised and unscaled problem encodings (rows 11–14) with a task-informed distance metric (rows 15–16).

For comparability, all time horizons in a single benchmark request are equal and determined by the length of the time-spaced and interpolated RRT-Connect solution. On runtime, we sample uniformly in the range of the problem encoding constraining the task-space goal $\mathbf{y}^*$ to the task domain with added Gaussian noise and ensure that each problem is solvable through the existence of a feasible solution.

	Solver	Initialisation	Success Rate	Warm-Start Time (ms)	Residual Cost	Time to Convergence (s)	Iterations until Convergence
1	AICO	Zero-Motion	33.40%	–	1912.60 ± 2426.13	53.44 ± 56.03	139.32 ± 141.72
2	L-BFGS		11.60%		1729.23 ± 1209.04	34.09 ± 24.15	292.91 ± 166.14
3	AICO	Straight-Line	14.40%	–	1483.56 ± 3293.76	31.05 ± 36.02	129.82 ± 144.10
4	L-BFGS		14.80%		16923.00 ± 37558.22	17.64 ± 19.48	180.27 ± 210.52
5	AICO	RRT-Connect	94.80%	619.25 ± 359.07	927.83 ± 3203.49	34.62 ± 35.43	71.55 ± 70.62
6	L-BFGS		96.10%		909.81 ± 513.43	27.12 ± 24.99	166.01 ± 164.38
7	AICO	KNN	64.10%	0.87 ± 0.11	653.00 ± 550.46	34.70 ± 35.51	75.06 ± 73.68
8	L-BFGS		65.20%		866.54 ± 510.07	20.95 ± 17.65	120.63 ± 136.46
9	AICO	KNN (normalised problem encoding)	**96.00%**	1.18 ± 0.20	**496.51 ± 306.24**	19.87 ± 28.06	56.33 ± 63.18
10	L-BFGS		95.80%		667.50 ± 409.58	**14.17 ± 15.18**	152.49 ± 158.69
11	AICO	Exhaustive search	64.10%	3.72 ± 0.35	653.00 ± 550.46	34.60 ± 35.29	75.06 ± 73.68
12	L-BFGS		65.20%		866.54 ± 510.07	20.97 ± 17.76	120.63 ± 136.46
13	AICO	Exhaustive search (normalised problem encoding)	96.00%	3.73 ± 0.37	496.51 ± 306.24	19.89 ± 28.08	56.33 ± 63.18
14	L-BFGS		95.80%		667.50 ± 409.58	14.18 ± 15.21	152.49 ± 158.69
15	AICO	Exhaustive search (task-informed weights)	98.30%	3.71 ± 0.31	**416.02 ± 286.37**	35.37 ± 37.87	75.99 ± 69.20
16	L-BFGS		**99.50%**		741.85 ± 442.54	15.65 ± 20.44	110.64 ± 146.28
17	RRT-Connect	–	100.00%	–	5163.96 ± 26328.00	0.62 ± 0.36	–

Table 5: Overview of success rate, convergence time and iterations for different optimisation solvers and initialisation strategies across 1,000 benchmark trials with initial states uniformly sampled within the valid range and task-space goals sampled from a normal distribution centred at the shelf compartments.

4.3.3 *Analysis of results*

Averaged results over 1,000 benchmark requests are presented in Table 5. Based on these, we can draw several observations highlighting different properties of the tested initialisation strategies:

A. As expected, feasible RRT-Connect initialisation leads to convergence in the majority of cases: 96.1 % with L-BFGS (row 6) and 94.8 % with AICO (row 5). However, this comes at a 40 % to 80 % higher final cost as the feasible initialisation does not consider the other objectives of the cost function beyond collision avoidance and reaching the final configuration. The flexibility of a guaranteed feasible solution thus comes at the cost of a longer warm-start time as a new solution is computed from scratch.

B. Zero motion (which consequently has a large distance to the task-space goals) and straight-line interpolation (which may be infeasible and pass through obstacles) rarely succeed for quasi-Newton methods which require to be initialised in the basin of attraction of a local minimum (11.6 % and 14.8 % success rates, respectively). Where they converge on a solution, the solutions have final costs between 4 and 40 times higher than the best-performing initialisation strategy (row 15). This is expected as if in collision, quasi-Newton methods are trapped (in fact this is the source of all failures).

C. AICO manages to converge with 33.4 % and 14.4 % in these scenarios, although at an up to twice higher cost compared with being initialised with a feasible solution while requiring similar time or longer. When optimisation fails, it is primarily due to not achieving the final task-space goal for zero motion (74.2 %), and almost exclusively due to being stuck in a collision for straight-line initialisation (94.4 %).

D. An exhaustive exploration of the trajectory library using a normalised problem encoding yields a 96.0 % success rate along with the lowest overall final cost for an automatic distance metric suggesting the exploration strategy produced a library with good coverage on the task domain. Use of a task-informed, hand-tuned metric can boost success rates to 99.5 % while reducing mean final cost by a further 16.2 %. This improvement suggests that future investigations on automatically learning distance metrics or generalising classifiers can ensure more efficient use of the generated trajectory library.

E. Among the KNN variants, the one trained on the normalised problem encoding performed the best—with similar success rates for both AICO and L-BFGS; rivalling the success rates of feasible initialisation while obtaining a lower final cost with up to twice quicker convergence. Note, the results in rows 7–10 equal those in rows 11–14 as the nearest neighbour algorithm selected the same initial seeds as the exhaustive search while being more efficient in the look-up due to its data structure. The success rate with L-BFGS is marginally lower compared to being initialised with RRT-Connect. However, it achieves a lower cost twice as fast.

F. For our trajectory library, KNN is 3.1 times faster to retrieve an initial guess than searching the library exhaustively—this difference is expected to grow with an increasing number of stored problem-solution-mappings. The KNN retrieval features the fastest warm-start time (within approximately 1.2 ms), and either KNN or exhaustive search are at least two orders of magnitude faster than running SBP from scratch.

G. Normalising the problem encoding to account for different scaling provides better warm-start seeds and, as a result, higher convergence independent of the look-up method as it permits the successful use of simple distance metrics (49.8 % and 46.9 % increases in success rates, respectively).

Overall, the benchmark results suggest that given a known environment and the presented pre-computation strategy, the proposed problem encoding can be efficiently used to initialise trajectory optimisation solvers online using nearest neighbour look-up with inexpensive distance metrics.

4.4 DISCUSSION

This chapter considered the problem of providing a suitable initialisation for trajectory optimisation algorithms in complex environments which result in highly non-linear objective functions. We hereto proposed the use of an offline generated trajectory library with a problem encoding which can be used for fast online indexing based on inexpensive nearest neighbour metrics. We detailed strategies for efficient sample reuse and evaluated our method in a randomised benchmark on a shelf picking task. Our results demonstrate that our method achieves similar or higher success rates to initialisation with feasible solutions from sampling-based planners while converging faster with lower final costs. This advantage is in line

with expectation as, unlike with SBP, the seeds are optimal solutions to a similar trajectory optimisation problem and do not have to first optimise for the objectives (e.g., smoothness and balance).

While we considered only the closest candidate from indexing and look-up based on similarity metrics, other work has focused on training generalising classifiers for single or ordered list prediction. Dey et al. [2013], e.g., used exhaustive online training after library generation to obtain a series of classifiers for list prediction allowing a sequential initialisation with the next best candidate should previously-tried initialisations fail.

A key challenge for applying regression over a dataset of initial trajectories are discontinuities, e.g., introduced by dynamics or complex environments. While we focused on high-dimensional, kinematic planning with discontinuous objective functions introduced by complex environments, similar pre-computation and initialisation approaches have been presented, e.g., for dynamic lower-DoF optimal control problems in obstacle-free environments. Here, Mansard et al. [2018] presented an iterative approximation of the value and policy functions initialised from and used in building an optimal kino-dynamic PRM. While the nature of the underlying problems differs, we consider iterative approximation *during* offline pre-computation a highly exciting avenue for future work. Graph-based approaches, e.g., HDRM, PRM and its variant Experience Graphs [Phillips et al., 2012] are based on prior solutions with heuristics guiding search onto previously explored graphs. They are attractive due to fast exploitation using graph-search algorithms, but may limit exploration and are only optimal w.r.t. the roadmap.

One of the critical limitations of our formulation is the use of soft constraints, e.g., for collision and joint limit avoidance as well as task-space targets. While high penalties on these terms often lead to the desired behaviour, there are no guarantees on the satisfaction of these constraints. As a result, a final collision check of the resulting optimal trajectory is required. Furthermore, different cost terms contradict each other causing early, sub-optimal, or no convergence. In practice, using unconstrained optimisation is efficient and fast given a good initialisation, e.g., from a trajectory library as considered in this chapter. Additionally, using analytic second-order information where available and exploring a staged cost function or adaptation of weights in an outer loop can address poor initialisation and speed up convergence.

In our work, we use the signed distance between the robot and environment as a collision proxy without preprocessing. Preprocessing, e.g., convex decomposition

[Schulman et al., 2014], using signed distance fields and overlapping sphere approximation of the robot body itself [Ratliff et al., 2009, Kalakrishnan et al., 2011] could speed up collision queries which currently dominate the cost and prevent failures from discontinuities, e.g., when initialised in a collision. Additionally, evaluating collisions only at discrete time steps requires a dense discretisation of the trajectory while not guaranteeing continuous-time safety (e.g., as considered in Chapter 3 and Schulman et al. [2014]). Further, dense discretisation of the time horizon results in large optimisation problems. This presents opportunities to speed up convergence by using trajectory parametrisations and embeddings such as Gaussian Processes [Dong et al., 2016, Mukadam et al., 2016], B-splines, dynamical systems, kernel methods [Sugiyama et al., 2008, Rawlik et al., 2013, Marinho et al., 2016], or sub-sampling of an initial coarse time parametrisation [Mitrovic et al., 2010].

For trajectory libraries, defining the task-space encoding and distance metric for efficient look-up is vital. Avenues for future work include clustering of solutions into distinct classes (e.g., from topology as in Chapter 5), learning of feature descriptors (in place of the engineered task-space encoding) and distance metrics, as well as investigating function approximation of the value and policy functions. The idea of learning disentangled policy predictions that regress within distinct solution classes is particularly intriguing. Additionally, our current formulation only handles static obstacles. Providing initial seeds in dynamic scenes remains an open challenge.

As discretised or exhaustive sampling is prohibitive, future work should address exploration strategies for the offline dataset generation stage when posed with more complex problems. Beyond exploration, limitations on memory and online look-up runtime motivate information-theoretical considerations on how many samples are required and which ones can be discarded.

4.5 CONCLUSIONS

In this chapter, we considered high-dimensional problems in complex environments using multi-objective kinematic trajectory optimisation. In this setting, using good initial seeds allowed fast motion synthesis despite the many local minima introduced by the non-linearity and discontinuity of including collision avoidance.

However, the efficacy of warm-starts is more pronounced for tasks which include complex dynamics such as switching contacts or under-actuated systems. In these scenarios, computing an alternative initialisation from scratch is more expensive

(e.g., requiring pipeline approaches as outlined in Chapter 1). Multiple solution modes to solve the same problem may exist and be returned by the pipeline methods for pre-computation as these often include random sampling components. In order to successfully apply data compression and generalisation methods on large trajectory libraries, insights into the inherent structure of datasets are required.

We take steps towards this direction in the next chapter, where we focus on the analysis of the underlying structure of the solution space of dynamic optimal control problems.

5

CLUSTERING OF OPTIMAL TRAJECTORIES USING PERSISTENT HOMOLOGY

In the previous chapter, we observed two cases which make direct learning as a way of compressing and generalising across optimal solution samples challenging: First, *discontinuity*, where similar problems (as in closeness in the problem parametrisation space) yield vastly different optimal solutions. Examples for this can be seen in the phase space plot of optimal solutions for a pendulum swing-up task as well as with discontinuities in solution paths introduced by environment obstacles. Second, *multi-modality*, where multiple equally optimal solutions to a problem exist. A prominent example is the ability to traverse around an obstacle in two ways.

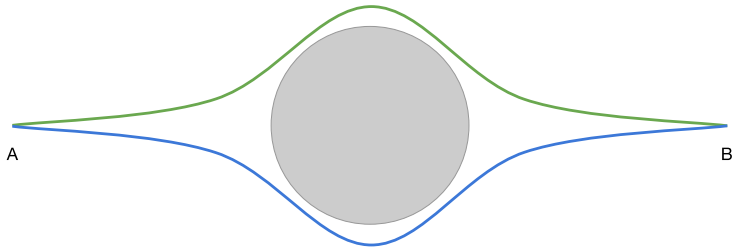


Figure 41: Illustration of discontinuity and multi-modality for simple problems: The green trajectory from A to B (top) cannot be continuously deformed into the blue trajectory (bottom). This violates the continuous map assumption between problem parametrisation and solution output $f : X \to Y$ which is core to function approximation.

Both discontinuity and multi-modality can significantly impact the quality of prediction obtained using function approximation as regressors smooth across the boundaries between samples from distinct clusters or modalities within the solution space. This is expected as efficient function approximation methods such as Locally Weighted Projection Regression (LWPR) [Vijayakumar and Schaal, 2000] and Gaussian Process Regression (GPR) [Williams and Rasmussen, 1996] in their standard formulation assume unimodal distributions and continuity (and in the case of GPR also homoscedasticity). One workaround to avoid multi-modality is to bias the sampling/exploration stage to include/enforce only one modality, as in *IREPA* [Mansard et al., 2018].

Another way is to use machine learning models that can handle multi-modality and discontinuity directly. These are often a combination of multiple local models that each model one continuous cluster of data well. In the previous chapter, we have used nearest-neighbour, a hyper-local model that returns the closest neighbour without any interpolation. Thus, the returned solution is valid (i.e., does not violate any constraints for the similar problem), but does not generalise between samples (and also does not compress the original dataset). In order to improve success rates for handling discontinuous and multi-modal distributions directly, Mixture-of-Experts (MoE) [Jacobs et al., 1991] and Product-of-Experts (PoE) [Hinton, 1999] systems have been proposed. For MoE, a gating function or network acts as a classifier/selector to determine which expert/model will provide the best output and query that model exclusively. The rationale is that a regressor trained exclusively on a continuous subset of the data will do well when queried within that subset but poorly outside. Traditionally, MoE systems first partition the input data and then train classifier and regressor individually on subsets of data. If a division into clusters/subtasks is known, joint training using a loss function that encourages both specialisation and cooperation between the local experts and thus automatically assigns samples to classes can be employed [Jacobs et al., 1991]. MoE has been used, for instance, with different experts modelling different characters in text recognition.

PoE methods, on the other hand, combine the output of multiple probabilistic models to form the prediction. One example are Gaussian Mixture Models (GMMs) which can capture multi-modality directly. However, they require information on the number of classes present or the tuning of hyperparameters. Pignat and Calinon [2019], for instance, use a Dirichlet process which can represent an infinite number of clusters, while Bishop [2006] introduces a Dirichlet distribution prior with a fixed number of clusters.

Using expert knowledge about the dataset, Tang and Hauser [2019] applied a MoE approach for discontinuity-sensitive learning of initialisation seeds. Here, they applied clustering using k-Means informed by a system designer's intuition on the input-output relationship such as the periodicity of angles or Lagrange multipliers of constraints. They subsequently learnt a MoE system with standard, fully connected neural networks; and demonstrated their approach on dynamic tasks in a minimum-time optimal control setting on a pendulum, 2D car, and a quadcopter with a single spherical obstacle.

We argue that while these approaches may work well on small problems and datasets where a system designer's intuition is readily available, it is often challenging to extract heuristics for labelling data for higher dimensional tasks. Further, due to the random nature of the exploration stage, the available dataset may not include samples of all homotopy classes or all modalities.

The authors of Pokorny et al. [2016] took a different approach to clustering trajectories. They used filtrations of simplicial complexes and persistent homology for modelling trajectories in configuration spaces. They then used the persistency to classify trajectories with fixed start and end points. This method looks at the changes in topology across different scales and identifies at which scale topological features (connected components and holes) appear and disappear. The theory behind this approach has been studied by Edelsbrunner and Harer [2008], and computationally efficient algorithms have been proposed by Carlsson [2009]. These were further improved for computational speed and memory efficiency by Chen and Kerber [2011] and Cavanna et al. [2015]. Motivated by this, we aim to extract information on the underlying solution space automatically.

While in the previous chapter we have employed deterministic sampling to achieve a desired density in the problem-solution map to employ inexpensive distance metrics for look-up, in this chapter we are interested in determining the underlying structure of the solutions. To this end, we combine persistent homology with clustering algorithms to identify multiple modes and discontinuities among continuous trajectories within a pre-computed dataset before applying machine learning tools for memory compression. To the best of our knowledge, we are the first to use the persistent homology tools on a dataset of motion of highly dynamic systems such as a cart-pole and a quadrotor as well as to warm-start optimisation solvers.

5.1 PROBLEM FORMULATION

In this chapter, we focus on discrete-time, finite-horizon non-linear optimal control problems. Consider a system with non-linear dynamics $\mathbf{f}$ for which we aim to find a policy $\mathbf{u} = \pi(\mathbf{x})$ that minimises a canonical cost function

$$J = \ell_f(\mathbf{x}_T) + \sum_{t=1}^{T-1} \ell(\mathbf{x}_t, \mathbf{u}_t) \tag{35}$$

starting from an initial state $\mathbf{x}_0$ subject to the dynamics constraints (i.e., transition from state $\mathbf{x}$ by applying controls $\mathbf{u}$):

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t), \tag{36}$$

where $\ell_f(\mathbf{x}_T)$ is the state cost at the end of the horizon and $\ell(\mathbf{x}_t, \mathbf{u}_t)$ the general running cost on state and controls.

We discretise the time horizon in T time steps and minimise J to obtain the sequence of controls $U^* = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{T-1}]$, where all controls are bounded with upper and lower limits. We denote the minimal cost for any given state $\mathbf{x}$ at timestep t as the cost-to-go $V(\mathbf{x}_t, T)$.

For this chapter, we will solve the above optimal control problem using indirect/shooting methods (see Section 5.1.3). As thus, we only consider general costs $\ell(\mathbf{x}_t, \mathbf{u}_t)$ and bounded control constraints. We do not explicitly use constraints on the states but include these as cost terms or enforce them in the forward pass/simulation. We will now detail Differential Dynamic Programming (DDP) and the dynamics systems considered in this chapter, before introducing our method in Section 5.2.

5.1.1 *Cart-pole*

The cart-pole is a canonical dynamic system used in non-linear optimal control. It features a pendulum mounted on a cart using a continuous, un-actuated hinge joint. The cart travels on a rail and can use horizontal forces as input to control the system. Due to control limits on the horizontal forces that can be applied, the cart-pole is a canonical task for non-linear optimal control as the preview horizon is required to build up enough energy to swing up to the unstable equilibrium at the top.

Following Tedrake [2014–2019], we model the system as follows: We denote the horizontal position on the slider as x and the angle between the pendulum and the cart as θ, where $\theta = 0$ is the stable equilibrium with the pendulum at rest at the bottom. Thus, the configuration is $\mathbf{q} = [x, \theta]^T$, the velocity $\dot{\mathbf{q}} = [\dot{x}, \dot{\theta}]^T$, and the state $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T = [x, \theta, \dot{x}, \dot{\theta}]^T$. The control input is $\mathbf{u} \in \mathbb{R}^1$ and is limited to $\mathbf{u} \in [-5, 5]$ N. The aim is to swing up the pendulum to the upright position with the horizontal position of the cart at zero and zero final velocity. This is commonly characterised as $\mathbf{x}_{goal} = [0, \pi, 0, 0]^T$. Note, however, that here we do not require it to reach the final value of π, but any upright configuration. The reason is that it is not relevant

to the task from which side the pole swings up—or whether it completes more than one full rotation before coming to rest. Therefore, we are modelling the continuous hinge joint using the special orthonormal group $\mathbb{SO}(2)$ and represent the goal state as $[x, \cos\theta, \sin\theta, \dot{x}, \dot{\theta}]^\mathsf{T}$. Finally, we obtain first- and second-order derivatives using symbolic differentiation.

5.1.2 *Quadrotor*

Quadrotors (or quadcopters) are agile multi-rotor aircraft which have become very popular in the past decade due to increased battery energy densities, more powerful brushless motors, and reduced component prices. They feature versatile use cases due to their ability to carry payloads (e.g., for videography, sensing, and mapping or delivery) and hover, while also being agile and powerful for carrying out dynamic manoeuvres.

To model the quadrotor dynamics, we follow the (canonical) dynamics model introduced by Mellinger and Kumar [2011] with minor modifications: We do not consider effects from air drag (or near ground effects) and control the rotor forces directly. As such, the control inputs are $\mathbf{u} \in \mathbb{R}^4$ (limited between $0\,\mathrm{N}$ to $5\,\mathrm{N}$ per rotor) and the state is $\mathbf{x} \in \mathbb{R}^{12}$ using Euler representation for the angular component of the free-floating base. We derive first- and second-order derivatives using symbolic differentiation.

5.1.3 *Differential dynamic programming*

Introduced in 1966 by Mayne, DDP is an indirect, second-order shooting method optimising only over the unconstrained control space displaying quadratic convergence. As a gradient descent method, it uses locally-quadratic approximations of the dynamics and cost functions [Mayne, 1966, Jacobson, 1968]. DDP alternates between a backward pass on the nominal trajectory in order to generate a new sequence of feedback control laws, and a forward pass computing the new state trajectory. State constraints are enforced as part of the forward pass.

Following Tassa et al. [2014], if Q is the variation of the cost-to-go V (with the subsequent cost-to-go noted as V′), its variation is expanded to second-order as:

$$Q(\delta \mathbf{x}, \delta \mathbf{u}) \approx \frac{1}{2} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 0 & Q_{\mathbf{x}}^{\mathsf{T}} & Q_{\mathbf{u}}^{\mathsf{T}} \\ Q_{\mathbf{x}} & Q_{\mathbf{x}\mathbf{x}} & Q_{\mathbf{x}\mathbf{u}} \\ Q_{\mathbf{u}} & Q_{\mathbf{u}\mathbf{x}} & Q_{\mathbf{u}\mathbf{u}} \end{bmatrix} \begin{bmatrix} 1 \\ \delta \mathbf{x} \\ \delta \mathbf{u} \end{bmatrix}, \tag{37}$$

where the individual terms are:

$$Q_{\mathbf{x}} = \ell_{\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V_{\mathbf{x}}' \tag{38}$$

$$Q_{\mathbf{u}} = \ell_{\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{x}}' \tag{39}$$

$$Q_{\mathbf{x}\mathbf{x}} = \ell_{\mathbf{x}\mathbf{x}} + \mathbf{f}_{\mathbf{x}}^{\mathsf{T}} V_{\mathbf{x}\mathbf{x}}' \mathbf{f}_{\mathbf{x}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{x}\mathbf{x}} \tag{40}$$

$$Q_{\mathbf{u}\mathbf{u}} = \ell_{\mathbf{u}\mathbf{u}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{x}\mathbf{x}}' \mathbf{f}_{\mathbf{u}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{u}\mathbf{u}} \tag{41}$$

$$Q_{\mathbf{u}\mathbf{x}} = \ell_{\mathbf{u}\mathbf{x}} + \mathbf{f}_{\mathbf{u}}^{\mathsf{T}} V_{\mathbf{x}\mathbf{x}}' \mathbf{f}_{\mathbf{x}} + V_{\mathbf{x}}' \cdot \mathbf{f}_{\mathbf{u}\mathbf{x}}. \tag{42}$$

Solving for the optimal change in control $\delta \mathbf{u}^*$ given a change in state $\delta \mathbf{x}$ we get

$$\delta \mathbf{u}^* = \arg\min_{\delta \mathbf{u}} Q(\delta \mathbf{x}, \delta \mathbf{u}) = \mathbf{k} + \mathbf{K} \delta \mathbf{x} = -Q_{\mathbf{u}\mathbf{u}}^{-1} (Q_{\mathbf{u}} + Q_{\mathbf{u}\mathbf{x}} \delta \mathbf{x}), \tag{43}$$

where $\mathbf{k}$ is the feed-forward term and $\mathbf{K}$ the feedback gain matrix.

Using this result, Equation 37 can be solved for the quadratic model of the value change

$$\Delta V = -\tfrac{1}{2} Q_{\mathbf{u}} Q_{\mathbf{u}\mathbf{u}}^{-1} Q_{\mathbf{u}} \tag{44}$$

$$V_{\mathbf{x}} = Q_{\mathbf{x}} - Q_{\mathbf{u}} Q_{\mathbf{u}\mathbf{u}}^{-1} Q_{\mathbf{u}\mathbf{x}} \tag{45}$$

$$V_{\mathbf{x}\mathbf{x}} = Q_{\mathbf{x}\mathbf{x}} - Q_{\mathbf{x}\mathbf{u}} Q_{\mathbf{u}\mathbf{u}}^{-1} Q_{\mathbf{u}\mathbf{x}}. \tag{46}$$

The backwards pass consists out of iteratively computing the local quadratic model of the value function and the control modifications recursively backwards in time. It alternates with a forward pass integrating the initial state with the control trajectory to form a new nominal trajectory until convergence or a maximum number of iterations.

To directly incorporate bound constraints on the control variables without sacrificing convergence, Tassa et al. [2014] introduced the use of a projected-Newton class active-set Quadratic Programming (QP) solver to take control variable inequalities into account explicitly. Regularisation is introduced to ensure that the Hessian does not lose its positive finiteness. Line search with variable step sizes is used to achieve convergence (e.g., by scaling the step length relative to the inverse of the iteration count with regards to the overall maximum iteration limit; or through backtracking line search).

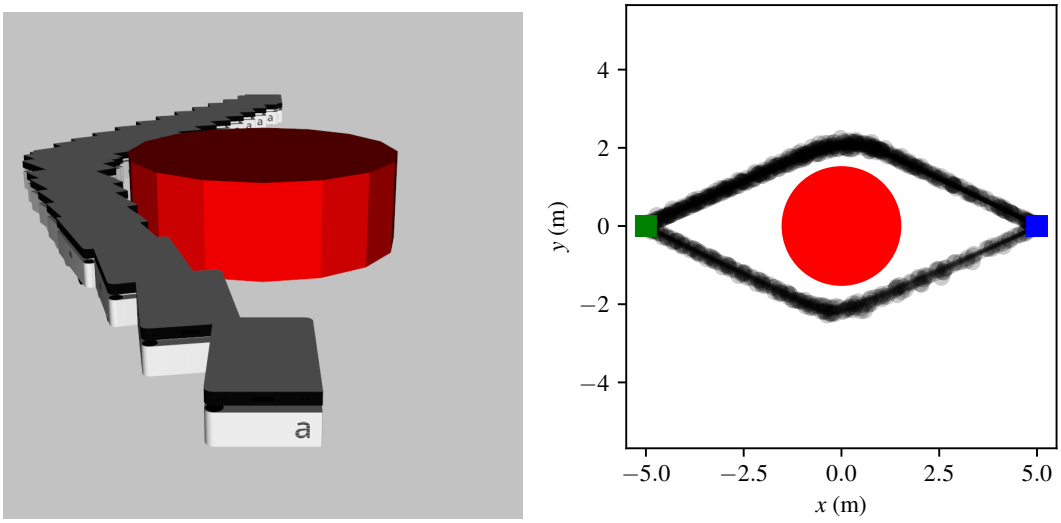
5.2 PERSISTENT HOMOLOGY

Having obtained a set of solutions to parametrised optimal control problems as a dataset, we are interested in learning a mapping $f : X \to Y$ from the problem encoding/parametrisation to output. However, as the data can include discontinuity and multi-modality, we are first interested in separating the available samples according to their homotopy class. A homotopy is the ability to continuously transform one function (in our case, trajectory/solution) into another [Weisstein, 1999–2019]. Computing homotopy directly is computationally very expensive, and in most cases, intractable. However, we can instead identify topological features such as clusters and holes in data by applying tools from persistent homology [Wright, 2015]. Here, efficient algorithms exist to compute homology groups of simplicial complices which allow reasoning about the global properties of a space based on local computations [Kaczynski et al., 2006].

In general, in order to analyse a dataset, we consider it as a set of points associated with a distance metric. We outline how to transform a set of trajectories into a dataset of points in Section 5.2.1. From the set of points associated with a distance metric, we are particularly interested in computing the invariant *cohomology* of the output space (i.e., the number of "holes"). A fundamental building block of algebraic topology are *simplicial complices*: Points are a 0-simplex, edges/lines between two points form a 1-simplex, three points forming a triangular face are a 2-simplex, and a solid tetrahedron is a 3-simplex. During filtration, starting from a small distance (or scale parameter) r, for each point in the dataset, we connect all points within distance r. We record when simplicial complices emerge (marked on the *Birth* axis in the homology diagram) and disappear (the *Death* axis). This process gets repeated for increasing values of r. Invariant features of the underlying dataset have a long persistence/lifetime (are far away from clusters and the diagonal) and can thus be read off the homology diagram. Alternatively, these features can be visualised using a barcode diagram [Ghrist, 2008]. Here, the rank of the zeroth dimensional homology group (H_0) corresponds to the number of connected components. In contrast, the rank of the first-dimensional homology group (H_1, the number of one-dimensional "holes") allows us to reason about the number of clusters in the data.

Following the conceptual example in Figure 41, we use RRT-Connect with smoothing to sample path plans for a holonomic mobile robot travelling in an environment

with a cylindrical obstacle (see Figure 42a). We sample 25 trajectories and interpolate each to uniform length $T = 25$, see Figure 42b.



(a) Visualisation of one trajectory sample.

(b) Sample trajectories ($N = 25$).

Figure 42: Sample path plans for a holonomic robot navigating in $\mathbb{SE}(2)$ from $(-5, 0, 0)$ to $(5, 0, 0)$. Two homotopy classes are clearly visible.

5.2.1 *Dealing with time-series data*

As the trajectories represent time-series data (the state or controls at time steps $t_1, \ldots, t_T$), we cannot directly pass them as a dataset to the computational algorithms. In this section, we describe how to transform a trajectory into a set of n-dimensional points.

Suppose we have N samples of M-dimensional time-series data with T time steps. As we are interested in the structure of the M-dimensional space, we stack them into a $M \times (N \times T)$-dimensional matrix.

We create a pairwise distance matrix (of dimension $(N \times T) \times (N \times T)$) and post-process it by incorporating explicit connectivity information from the trajectories: We explicitly set the distance for subsequent time steps to zero. We can also set the distance for connected start and end states to zero. We then apply filtration to the post-processed distance matrix to extract the persistent homology groups.

To illustrate, we apply this process to the navigation example and visualise the preprocessed distance matrix and the homology diagram in Figure 43. The distance matrix, displayed on the left, shows patterns from the preprocessing step. In the

filtration diagram, displayed on the right, we can clearly identify the one connected component (single blue dot of the H_0 group at the infinity line) as well as the hole introduced by the obstacle (single orange H_1 group outlier). We can additionally extract the separating distance as $4 > r > 1.5$.

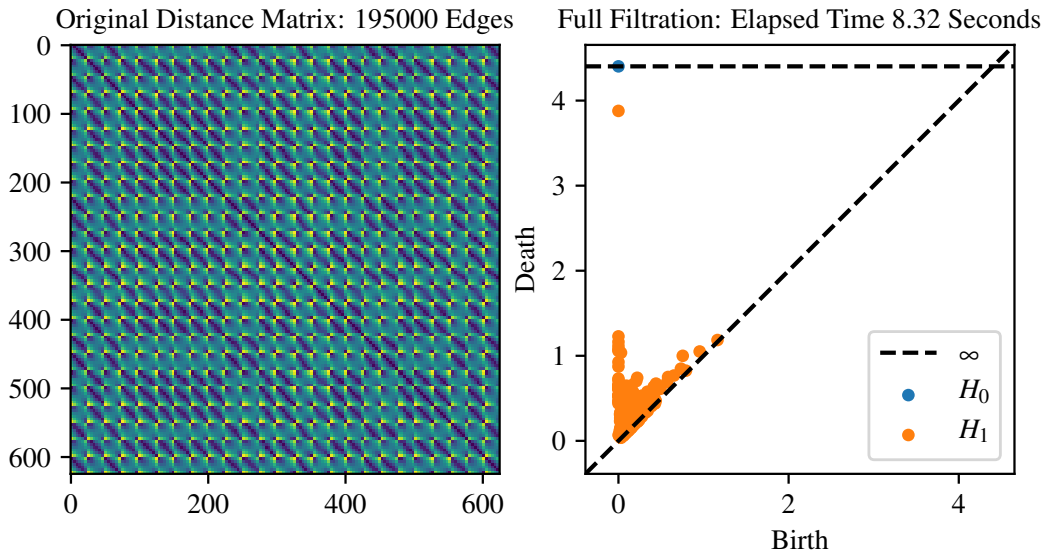


Figure 43: *Left:* Original distance matrix of the preprocessed dataset (195,000 edges). *Right:* Output of dense filtration (elapsed time 7.57 s. The invariant features of the dataset include one connected component (H_0, single blue dot at the infinity line) and one hole (H_1, one orange data point far removed from the infinity line), i.e., two classes/clusters.

Note, however, that the proposed process considers distance matrices in their dense form, which can very quickly exhaust memory during computation.[1] In practice, we frequently reduce the sampling frequency along the time dimension. An alternate approach would be to use dimensionality reduction tools or representation with embeddings before applying filtration.

5.3 EVALUATION

We test our methodology on optimal control tasks using the cart-pole and quadrotor dynamic models introduced above. We implement the optimal control problem,

1 We have explored to use filtration using sparse distance matrices. However, the results did not uncover any invariant features, even for relatively densely connected matrices. We suppose that the preprocessing for time-series data presented in this section may not work as intended for sparse matrices. We have attempted multiple small distances with similar results.

system dynamics, and DDP solver in Extensible Optimisation Toolset (EXOTica) [Ivan et al., 2019]. For topological analysis, we employ *Ripser.py* [Tralie et al., 2018], a Python interface to the highly efficient Ripser library [Bauer, 2017] for computing persistence barcodes, to compute the persistence cohomology of the dataset.

5.3.1 *Experiments on a cart-pole swing-up task*

We now consider a swing-up task on the cart-pole. From intuition, we can postulate that there are two modes: *swinging up from the left side* and *swinging up from the right side*. Indeed, these two solutions can be seen in Figure 44.

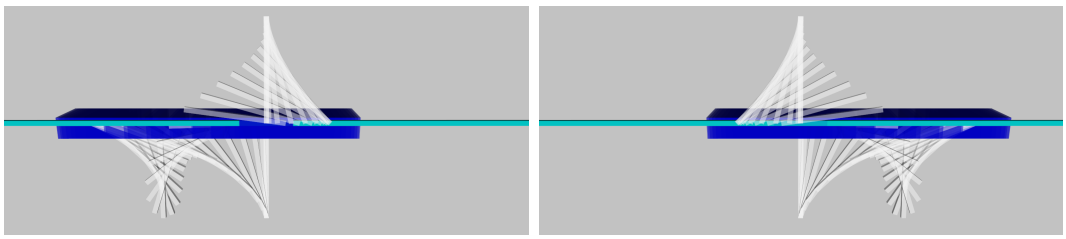


Figure 44: Visualisation of the two modes for the cart-pole swing-up task.

To build the dataset, we randomly sample start states between $(-1, -\pi, -1, -0.5\pi)$ and $(1, \pi, 1, 0.5\pi)$ and random control policies uniformly from $[-5, 5]$ N. We solve the optimal control problem using DDP until we have at least $N = 10$ solutions per random start state. The state trajectories in the dataset are shown in Figure 45.

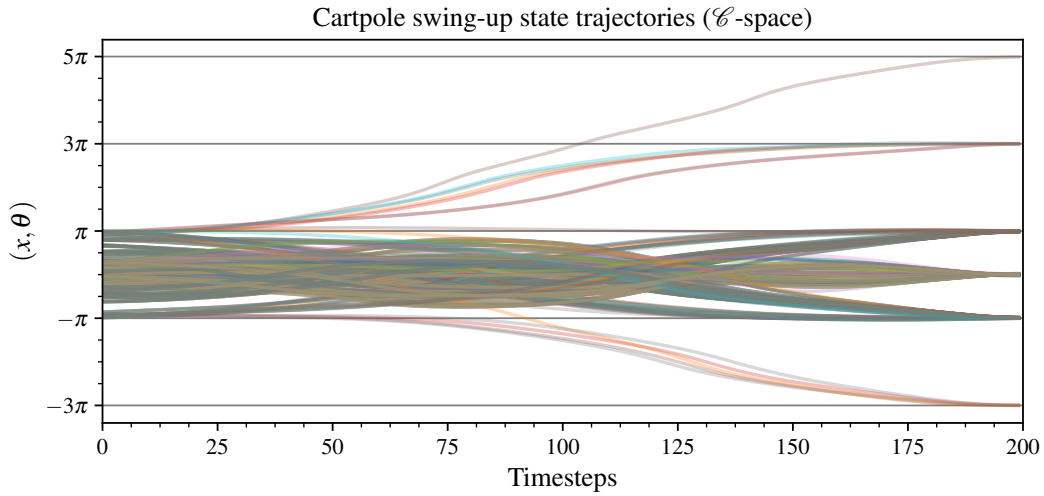


Figure 45: State trajectories (x, θ) for 500 swing-up policies from random initial states (50 random start states, 10 policies per start state). Note, depending on the initial velocity the pole may complete several full rotations before stabilising at the top.

This demonstrates that the solutions may complete several full rotations prior to reaching a zero velocity at the unstable, top equilibrium at the end of the horizon. However, as we aim to have the pole reach the upright position independent of the direction of swing-up (*from the left*, *from the right*) or the number of revolutions before stabilising to the upright position (it is a continuous hinge joint), we represent the task using the complex representation of the Special Orthogonal Group $SO(2)$: $(\cos\theta, \sin\theta)$.

We plot the state trajectories on the configuration manifold $(x, \cos\theta, \sin\theta)$ in Figure 46.

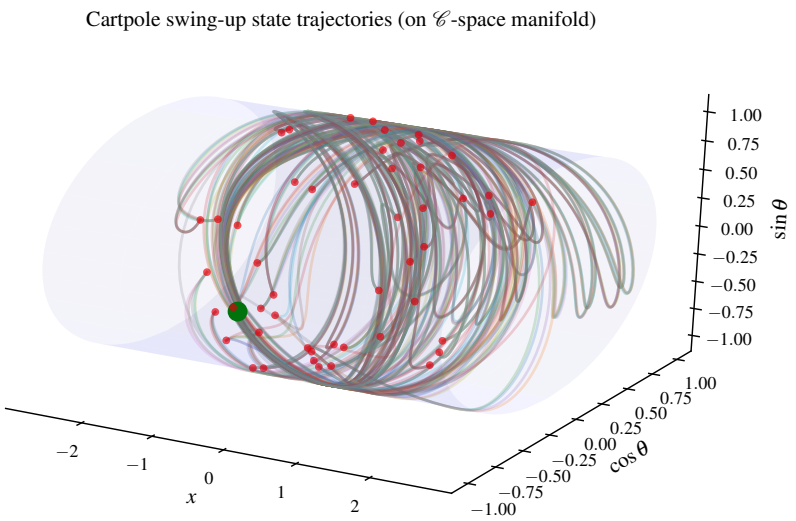


Figure 46: State trajectories visualised on the configuration space manifold.

Here, we annotate the random start states $\mathbf{x}_{rand} = (x, \cos\theta, \sin\theta, \dot{x}, \dot{\theta})$ with red circles and the swing-up goal $\mathbf{x}_{goal} = (0, -1, 0, 0, 0)$ in green. The associated control policies are shown in Figure 47.

Our aim now is to extract (a) the number of topological classes contained in the data and (b) assign the corresponding class labels to the raw data. We apply the methodology outlined in the previous section and show the preprocessed and downsampled dataset passed to filtration in Figure 48.

From the obtained persistent homology diagram in Figure 49, we can extract that (a) there is one connected set and (b) that there is one hole, i.e., that there are two classes. We can further read off a separating distance of ≈ 4.5.

Using this information, it is now possible to cluster the raw state and control trajectory data, see Figures 50 and 51.

We now investigate the impact that this separation by the modality of solutions has on interpolating or learning control trajectories from the raw control trajectory

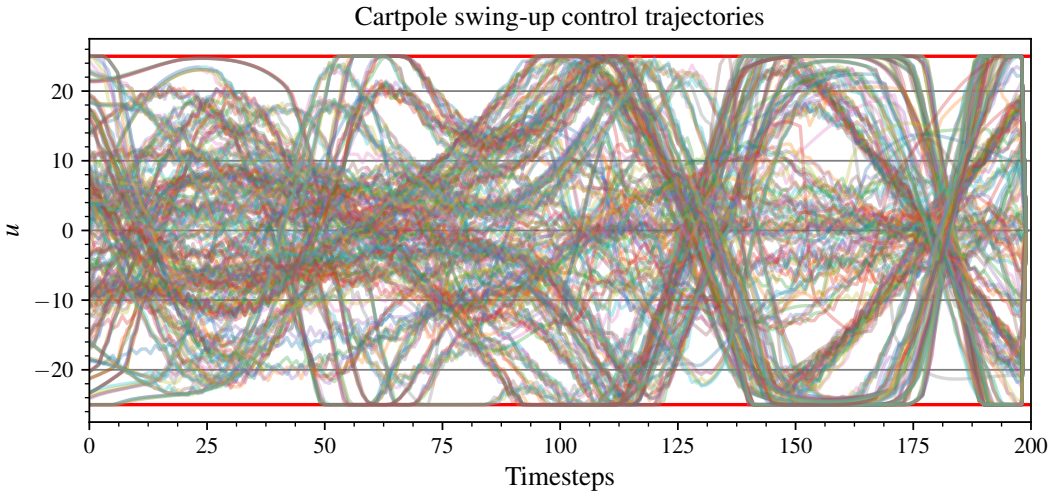


Figure 47: Control trajectories to achieve swing-up from random start states. We denote the control limits $(-25, 25)$ in red.

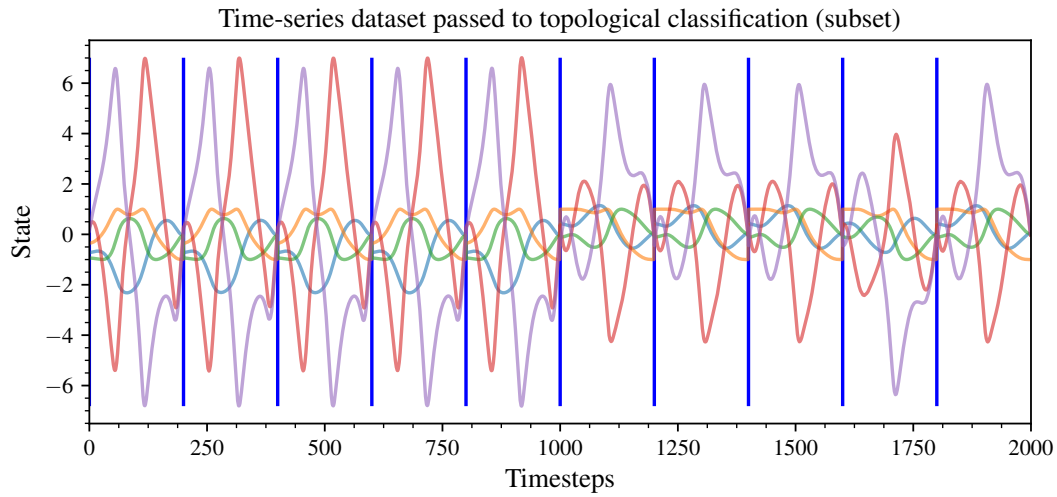


Figure 48: Subset of downsampled and preprocessed dataset as passed to homology filtration: The blue lines denote the boundary between different data samples. We stack the individual trajectories into vectors by dimension and preserve the time series nature by preprocessing the distance matrix to connect start, subsequent, and end states.

samples visualised in Figure 47 by looking at the mean and standard deviation of the control trajectories in the raw dataset as well as within each of the two identified and labelled classes (see Figure 52). The idea here is to take this as a proxy for function approximation. As shown in the Figure, if the multi-modality of the solutions is not taken into account, the information in the data effectively cancels. Note that this analysis applies to swinging up from $(0, 0, 0, 0)$ to $(0, \pi, 0, 0)$

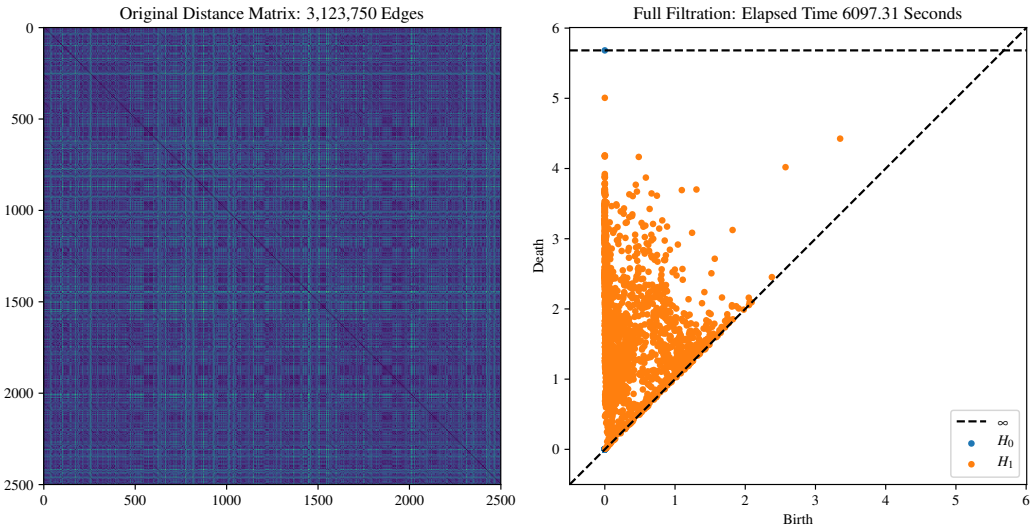


Figure 49: Left: preprocessed distance matrix, right: persistent homology diagram.

Cartpole swing-up state trajectories (on $\mathscr{C}$-space manifold)

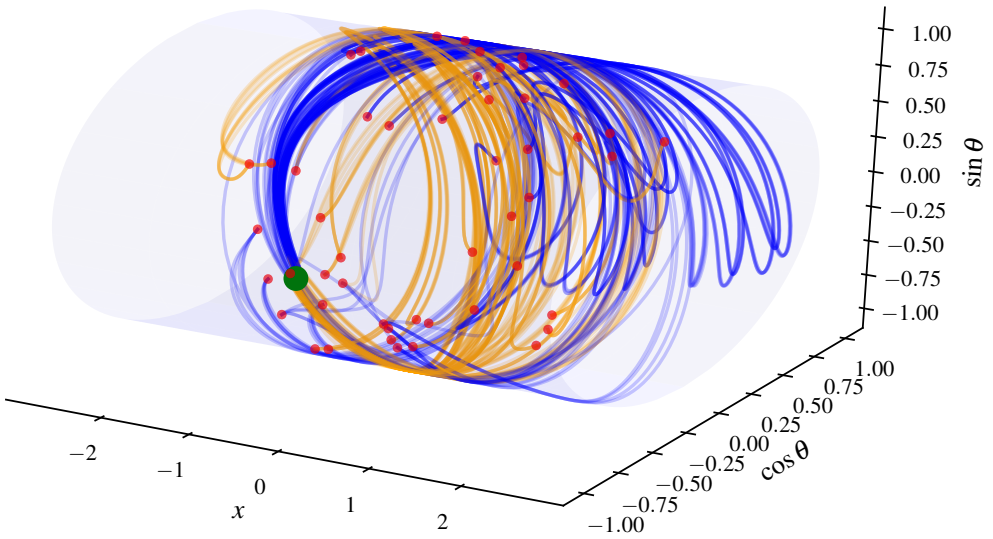


Figure 50: State trajectories visualised on the configuration space manifold. Class labels have been assigned using the output of the persistent homology filtration.

(same start and goal states, and thus same encoding). We recognise that the mean would not be an appropriate proxy for $f : \mathcal{X}_0 \to \mathcal{U}$ when considering multiple, different start states.

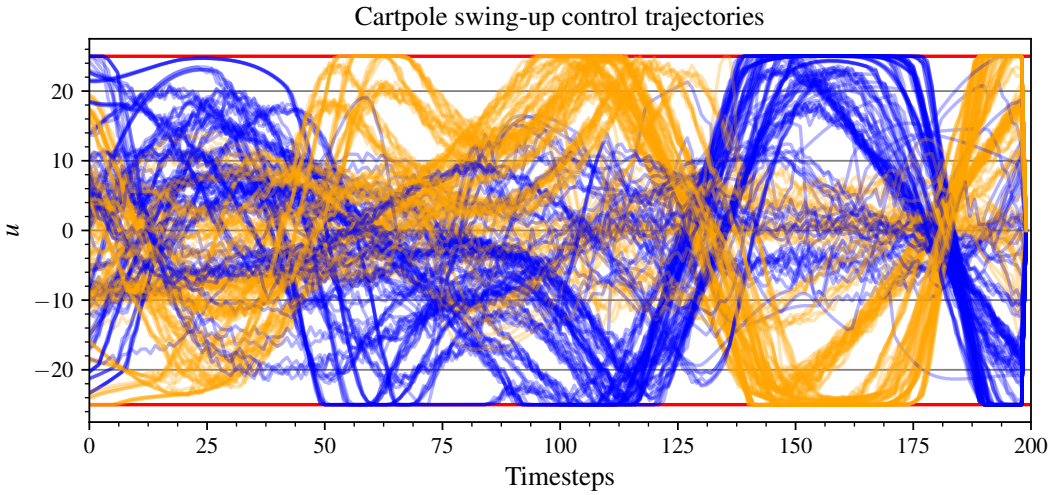


Figure 51: Control trajectories to achieve swing-up from random start states with class labels assigned from the output of the persistent homology filtration.

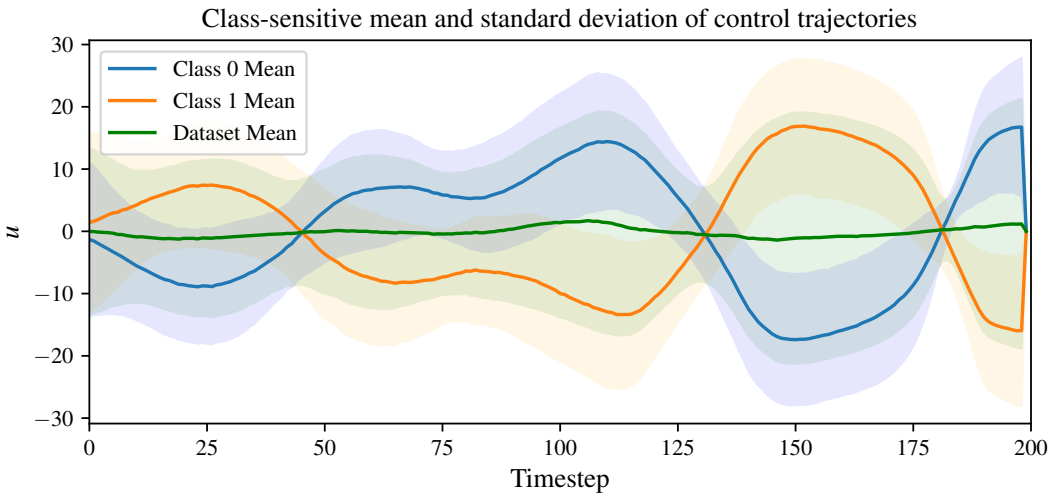


Figure 52: Mean and standard deviation of the control trajectories for the cart-pole swing-up dataset (from $(0, 0, 0, 0)$ to $(0, \pi, 0, 0)$): for the entire dataset (green) and for the labelled classes (orange, blue).

5.3.2 *Experiments on a quadrotor flying in a maze*

In the previous experiment, we tested the methodology on a low-dimensional toy problem using a cart-pole without considering cluttered environments. We now consider a set of experiments to investigate both the scalability in terms of problem dimensionality as well as to include multi-modality in the solutions introduced by collision avoidance in an environment. We also show the impact of the data separation on the success of warm-starting the original optimal control problem.

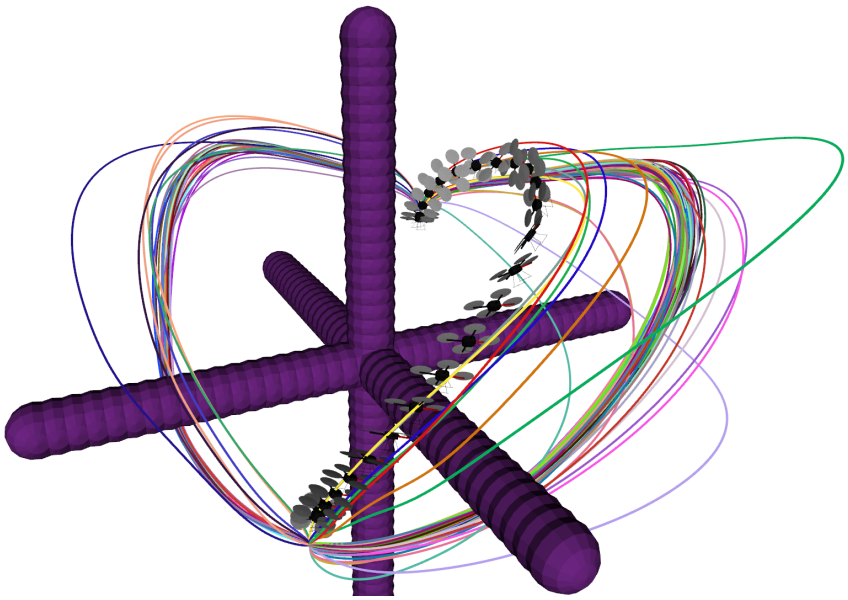


Figure 53: Unlabelled trajectories of a quadcopter flying from $(-1.75, -1.75, -1.75)$ to $(1.75, 1.75, 1.75)$ while avoiding obstacles. All trajectories obtained from zero-motion initialisation with Differential Dynamic Programming. We can identify several "bundles" of solutions and apply computational homology to extract the number of clusters.

We sample a dataset of trajectories for a quadrotor flying from a start to a goal position while avoiding a collision object represented as a set of spheres. We initialise the optimal control solver with a control trajectory for hovering with added random, uniform noise and solve until convergence. The resulting trajectories are visualised in Figure 53.

We apply the same preprocessing for time-series data to the state trajectories as for the cart-pole example in the previous section and run the computational homology routines. We depict the persistent homology graphs in Figure 54 and extract the number of present clusters as five (four holes).

Analog to the cart-pole example, we perform clustering using the obtained information. Here, we apply k-Means with $k = 5$ as extracted from the persistent homology diagram. The labelled state trajectories are shown in Figure 55. We also depict the mean for each of the clusters as a thicker line as well as a naïve mean over all samples without considering multi-modality.

We store the mean of the control trajectories of each of the clusters as well as across all samples as possible initialisation seeds (see Figure 56). Note, we are using the mean of the clusters here as a proxy for a generic function approximator as the start and goal states of the samples in question are the same.

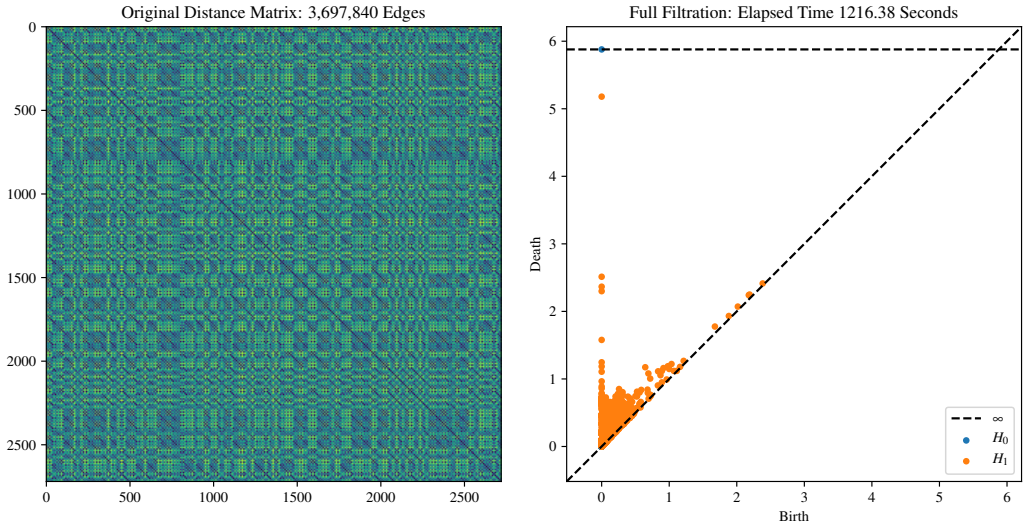


Figure 54: Persistent homology for the quadrotor example: There are one connected component (as all trajectory samples start and end at the same points, respectively) and four holes (i.e., five classes).

We now use these warm-start trajectories in the original optimal control problem and evaluate their impact on convergence. We show the results compared with zero-motion initialisation (grey) in Figure 57. Note, that the naïve and direct mean of all solutions without considering the multi-modality performs worse than a cold-start with a zero-motion initialisation. All warm-starts from within the extracted clusters start with a lower cost, however, some diverge (e.g., the red warm-start).

5.4 DISCUSSION

In this chapter, we have explored applying tools from computational topology to cluster solutions in a dataset before using machine learning for compression and generalisation. In particular, we used persistent homology to identify how many classes are present along with a separating distance to inform our data separation. While, in this chapter, we explored this concept on low-dimensional problems involving dynamics, we focused on establishing whether relevant topological information can be extracted to assist the encoding and learning stages of storing a memory-of-motion.

The results confirm that considering the underlying topological features of the dataset is essential and that tools from algebraic homology can be used to guide clustering. Indeed, exploring multiple modes can be important for warm-starting (see Figure 57) and further allows the development of ensemble methods that

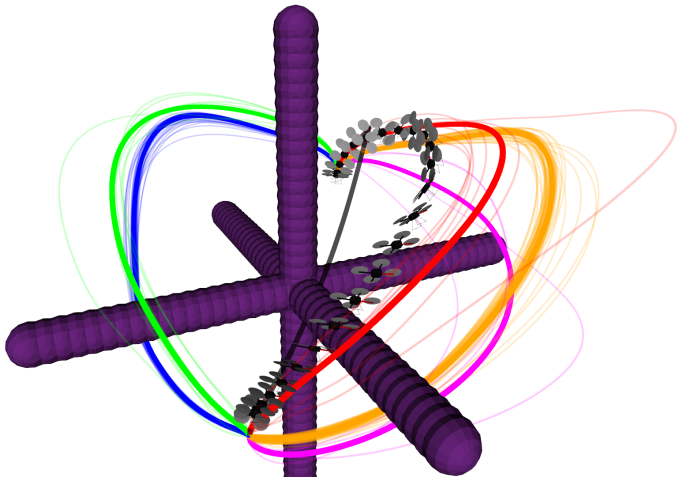


Figure 55: Labelled state trajectories for the quadrotor example along with means within each cluster and across all samples (black).

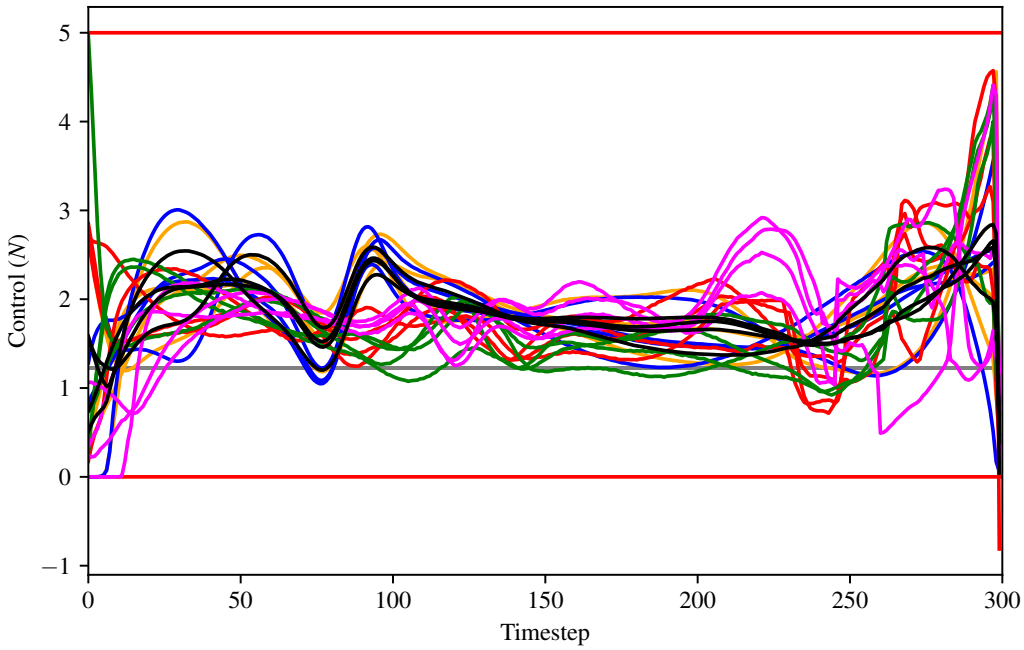


Figure 56: Extracted mean control trajectories which can be used to warm-start the optimal control solver. We also plot the actuation limits and note that warm-start extracted from the red cluster, for instance, violates the actuation limits. The grey, constant line is the required actuation to maintain the initial position (zero-motion initialisation).

explore multiple warm-start guesses in parallel [Lembono et al., 2020]. While these

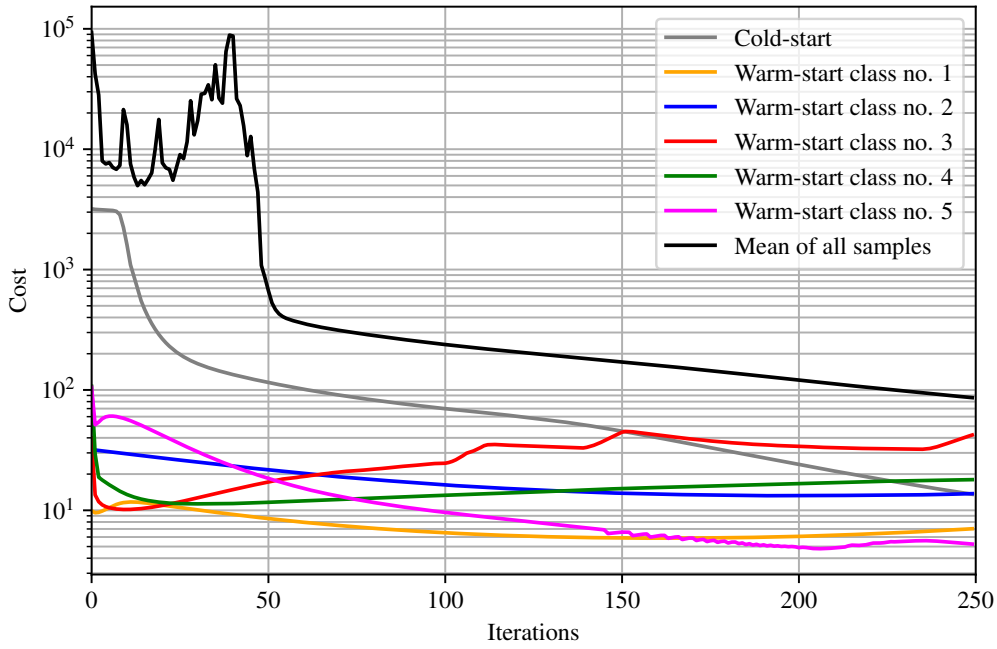


Figure 57: Cost evolution for solving the quadrotor task using *Differential Dynamic Programming* and different control trajectory seeds.

simultaneously explore multiple warm-starts, another opportunity is to test ranked warm-start modes/hypotheses akin to sequence or list prediction.

One limitation of our approach is that the tools for filtration are susceptible to the amount of data. In particular, the algorithms do not take advantage of parallelism, and available implementations are memory-intensive (e.g., performing the cart-pole analysis on 500 data samples required ≈6000 s and ≈60 GB of memory even after downsampling both the number of trajectories and the data within). While using sparse distance matrices for filtration is a promising way to reduce this process by one to two orders of magnitude, we have not been able to extract similar results despite high connectivity. Consequently, this is currently limiting the dataset sizes we can investigate. One way to address this issue could be to apply dimensionality reduction techniques such as Principal Component Analysis (PCA) before clustering to reduce the dimensionality of the data. However, this still does not address the number of samples or their length. Alternately, Perea and Harer [2015] previously applied sliding windows to discover periodicity in time series data using persistent homology. It is worthwhile to explore whether a similar approach can be applied in our case instead of the filtration of the full dataset.

Another challenge, and similar to most literature, is that we did not consider changing environments. Some authors [Mansard et al., 2018, Tang and Hauser, 2019] outlined the possibility to enhance the problem parametrisation with information on the size and location of primitive shaped objects. We consider this inflexible as fixed-size, basic representations always limit expressiveness and require vast numbers of samples to explore the increased dimensionality of the problem space. Jetchev and Toussaint [2013] investigated and compared learnt situation descriptors. More recent work focused on learning latent space representations directly from sensor data such as point clouds [Qureshi et al., 2019]. Along with large, labelled datasets on 3D objects and shapes such as ShapeNet [Chang et al., 2015], these are promising avenues for further investigation.

Finally, we explored systems with continuous dynamics and straight-forward start and goal situations. It would be interesting to explore the use of algebraic tools as in this chapter on tasks with discontinuous dynamics and periodicity, e.g. locomotion on legged platforms.

5.5 CONCLUSIONS

In this part of this thesis, we investigated (a) problem parametrisations and data structures for storing offline computed experience to bootstrap trajectory optimisation (Chapter 4) and (b) the underlying structure of the solution space and its impact on optimal control initialisation (Chapter 5). In both cases, we assumed that environment information can be sensed and is available in a suitable representation for the planning algorithms. Additionally, we assumed in both Part II and III that control systems on integrated platforms are able to execute motion as they are planned.

In the next part, we focus on complete system architectures for deploying algorithms to real-world applications. In Chapter 6, we describe a system that provides interactive perception and input interfaces for operators as well as a continuous scene monitoring for change detection. We detail and discuss semantic representations extracted from live sensing data that can be used in motion planning and scene monitoring. In Chapter 7, we address and evaluate the assumption of a synchronised control system to close the gap between visualisation/simulation and hardware deployment. We demonstrate a complete mobile manipulation system in a case study for whole-body loco-manipulation with synchronised whole-body control.

Part IV

DEPLOYMENT

6

A SYSTEMS APPROACH TO ROBUST SHARED AUTONOMY FOR MOBILE MANIPULATION

Industrial robots have been developed and deployed widely to automate manufacturing by precisely executing repeatable, meticulously designed motions in fully controlled environments. In order to allow maximum execution speed in mass production, the robots are cordoned off with security systems such as cages or fences. This separation is both to control the robot's working envelope fully as well as to keep people out of harm's way. The design of these production and assembly lines is a time-consuming and expensive process that goes along with the later stages of product design, namely Design for Manufacturability (DfM) and Design for Assembly (DfA). In sectors with long product life cycles (e.g., automotive), it used to be common to replace the production lines, including robots, with each product generation. Consequently, traditional automation was limited to industries with large volume and limited product variety (within the product or across generations). It further acted as a barrier to adoption significantly limiting Small and Medium Enterprises (SMEs) or those producing high-variety products from enjoying the benefits of automation. More recently, the broad adoption of (re-deployable) collaborative robots, which are safe to operate without safety barriers, as well as programming by demonstration have allowed industries with frequent batch changes and short product life cycles as well as Original Equipment Manufacturer (OEM) suppliers to adopt automation. Nevertheless, the ability to customise and dexterous manipulation often continue to pose a challenge for the broader deployment of robotic systems.

On the other end of the spectrum, when moving from predictable and engineered environments to unstructured settings, human input becomes indispensable. Here, robotic machines are deployed using remote teleoperation by a trained operator. While this removes people from the area of imminent danger or injury when carrying out hazardous or high-risk tasks, such as in nuclear decommissioning and Explosive Ordnance Disposal (EOD), the processes do not benefit from added

The work presented in this chapter has been awarded the *First Prize for Greatest Potential for Positive Impact* at the *International Robotics for Resilient Infrastructure Challenge* as part of the UK Robotics Week at the University of Leeds in June 2017.

productivity. As the operators bear the burden of high-level reasoning, planning, and responding to changes, they experience strain, stress, and fatigue. These factors apply in particular to jobs where operators command machinery to achieve complex multi-objective task requirements under high performance pressure. One example is spraying concrete linings in recently excavated tunnels (shotcrete application). Motivated by this example, Mower et al. [2019] conducted an extensive study on the impact of different control modes on execution speed and perceived fatigue. Their study found reduced completion times and increased execution quality when the operator used task-relevant and assisted, instead of joint-level, command spaces.

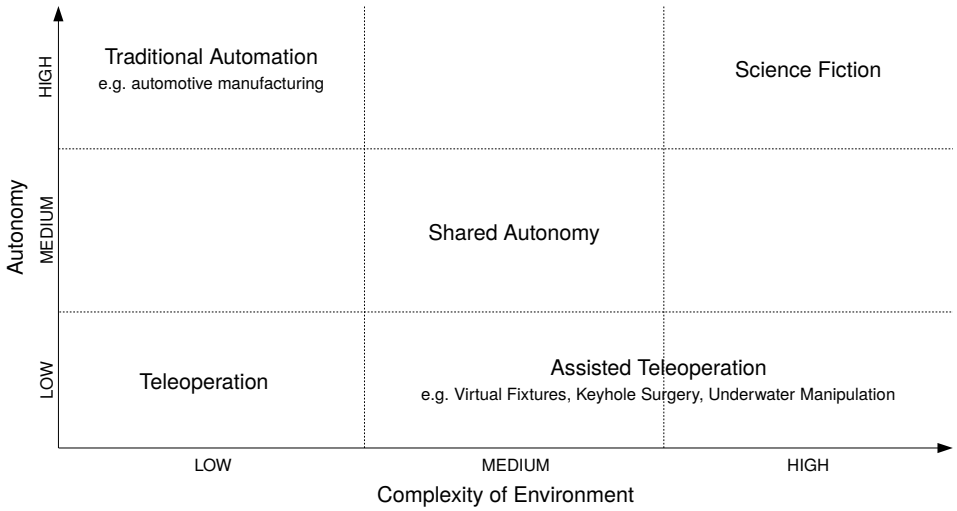


Figure 58: Different operation modes categorised by the dimensions of autonomy—from full teleoperation to full autonomy—and environment complexity—from engineered and controlled to cluttered, unpredictable and with dynamic changes. Fully autonomous systems are deployed where repeatability (*volume*) dominates, while teleoperation is employed when tasks come with high *variety*.

Overall, this motivates work into increasing autonomy in complex and unstructured environments to reduce strain and fatigue for the operator. In the following, we distinguish three operation modes (see Figure 58):

TELEOPERATION Here, operators directly or through a mapping control the degrees of freedoms (DoFs)/joints of a robot. Commonly used in fully unstructured or unmodelled areas that have high risk sensitivity (nuclear, surgery) or that have high task variety (e.g., Urban Search and Rescue).

ASSISTED TELEOPERATION Here, user input is post-processed to satisfy (safety) constraints, e.g., enforcing joint limits, limiting motion for self- and environ-

ment collision or to automatically synthesise constraint planes. Applications include for instance surgery and underwater manipulation (e.g., automatic stabilisation of the vehicle or its end-effector against disturbance induced by hydrodynamic effects such as sea currents).

FULL AUTONOMY Here, the robot executes motion and potentially makes decisions without the involvement of a human operator. At present, limited to structured environments such as factory floors which control working envelope and user access through barriers (fences), and are based on repeatability for efficiency.

In this chapter, we introduce a system for reliable grasping and manipulation that sits between assisted teleoperation and full autonomy following an operation paradigm referred to as *Shared Autonomy*. Shared autonomy can be applied in areas where there is no imminent danger from hand-over of control to a human operator or the time for human decision-making is sufficient in order to respond adequately. Our fully integrated system combines dense visual mapping, collision-free motion planning, and shared autonomy. Motion sequences are composed automatically based on high-level objectives provided by an operator. Continuous monitoring of the scene ensures safe execution by reasoning about the impact of changes in the environment on planned behaviours. All these components are combined in a state machine monitoring execution and providing direct user input for failure recovery. The system can automatically recover from a variety of disturbances and fall back to the operator if stuck or if it cannot otherwise guarantee safety. The operator can also take control at any time and then resume autonomous operation. A key assistive feature is to support the user in scene understanding (segmentation and affordance fitting) and allow high-level access and input to optimal goal state and motion planners, such as the ones described in Part II of this thesis. Our system allows a blend of remote, assisted teleoperation as well as autonomous execution of synthesised behaviours with the ability to include an operator for high-level commands. It is flexible to be adapted to new robotic systems, and we demonstrate our work on two real-world platforms—fixed and free-floating base—in shared workspace scenarios.

6.1 BACKGROUND

Many applications in industry are repetitive and require high levels of concentration and manual dexterity, often coupled with the human operator solely performing

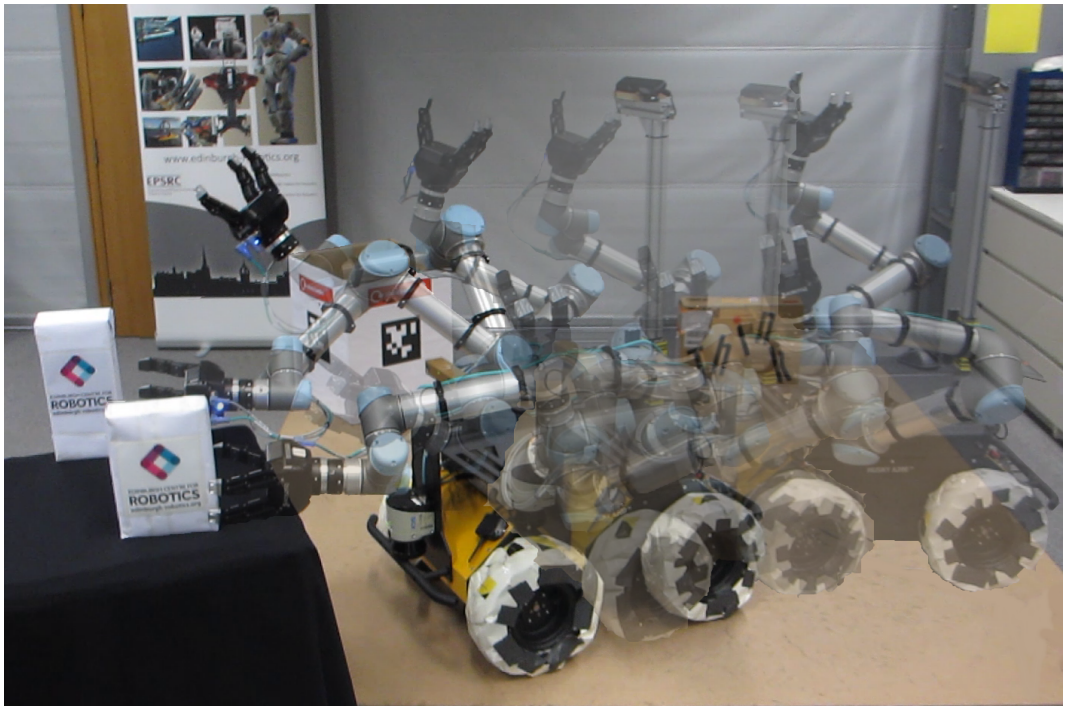


Figure 59: The bimanual mobile manipulator is executing a shared workspace task: It combines model-free segmentation of target objects with automatic mobile base placement selection and navigation to place them in a bin. The system automatically adapts to various dynamic changes, including changing object and target locations during execution as well as scene monitoring for safe avoidance and replanning of motion to accommodate human operators.

scene monitoring as well as hazard detection and prevention. As such, the operator is prone to fatigue, and this has been linked to severe accidents in the past. As a result, research has investigated guided teleoperation with on-the-fly synthesised constraints to provide assistance or resistance via force feedback—for instance, in surgical robotics via virtual fixtures. This guidance and protection of sensitive areas lead to a reduction in the concentration devoted to the task as it relaxes the mental criteria for task success and failure [Park et al., 2001].

Different state-of-the-art teleoperation and shared autonomy approaches were demonstrated at the DARPA Robotics Challenge (DRC) Finals in July 2015 for controlled, mostly static environments (*"high-level repeatable and low-level adaptable"*, Karumanchi et al. [2017]). However, more or fully autonomous systems are challenging in dynamic and unpredictable environments with clutter due to the sheer complexity of dealing with rare events. Shared autonomy is often perceived as a middle ground, combining autonomous sequences by the robot with input from

a human operator for high-level decision making reducing cognitive load and leading to more reliable systems.

These systems are especially crucial for manipulation and exploration in hazardous applications, such as for space exploration or disaster recovery with high-latency, low bandwidth communication and intermittent transmission cut-outs making teleoperation impractical. Furthermore, we argue that shared autonomy is vital to complement high-level human direction with high-frequency, closed-loop dexterity of a robotic system.

We refer to *teleoperation* as a control method of a robot that directly maps operator input to robot actions. An *autonomous* robot is an agent that perceives its environment, forms decisions based on its perception of the world, and realises actions according to these decisions. We define *shared autonomy*, similar to Sheridan [1992], to be a blend of teleoperation and autonomy that realises robot actions.

Several applications for teleoperation using immersive virtual reality head-mounted displays and remote controllers with haptic feedback have been demonstrated in recent years [Rodehutskors et al., 2015, Britton et al., 2015], e.g. with the Willow Garage PR2[1] and Rethink Robotics Baxter[2] robots. While the former used low-cost, consumer-grade hardware such as the Oculus Rift, the SRI Taurus Dexterous Robot[3] is an example of military-grade, immersive teleoperation equipment for EOD tasks. Fok et al. [2016] demonstrated the feasibility of web-based teleoperation by having inexperienced human operators interface with a humanoid robot through a web browser on a smartphone to complete a bimanual telemanipulation task.

Extensive studies have been performed to deduce the critical aspects of robot control methods that must be developed towards successful shared autonomy systems. The DRC has been the most significant demonstration of the state-of-the-art in disaster response robotics, where most systems implemented some form of shared autonomy. Yanco et al. [2015] identified the desired attributes of a shared system and highlighted specific design guidelines for being successful at the DRC, arguing that the design of a user interface is of high importance in regards to the overall success of a task.

The DIRECTOR interface and system architecture used to control the Atlas robot, developed by Team Massachusetts Institute of Technology (MIT), are described by Marion et al. [2017] and Fallon et al. [2015]. It features a shared autonomy system backed by interactive assisted perception and trajectory optimisation-based

1 http://ros.org/news/2013/09/pr2-surrogate.html
2 https://youtu.be/JHIz-Y5qCmY
3 https://archive.sri.com/engage/products-solutions/taurus-dexterous-robot

motion planning. Task sequences are composed of high-level motion primitives and constraints. The operator maintains a supervisory role, pausing execution and adjusting affordances and constraints in response to changes.

Team ViGIR took a similar approach and fitted affordance templates with semantic actions [Romay et al., 2014, Kohlbrecher et al., 2015]. They also used a virtual robot model for planning and review in a supervised semi-autonomous approach.

Karumanchi et al. [2017] and Hebert et al. [2015] detail the hardware design, algorithms, and system developed by the NASA Jet Propulsion Laboratory for RoboSimian. Their semi-autonomous system uses a behaviour planner and stored motion primitives along with non-linear trajectory optimisation. A human supervisor assists with object fitting, and reviews and approves candidate plans.

Similar semi- and shared autonomy systems have been developed to carry out related tasks. Stückler et al. [2016] use motion keyframes generated by an operator based on segmented perception data with the autonomous system interpolating between keyframes given constraints. Walter et al. [2015] describe an autonomous forklift able to operate safely in a shared workspace with humans accepting task-level commands through natural language and pen-based gestures. A high-level control method implementing shared autonomy for debris clearance is presented by Kaiser et al. [2016] where the system proposes affordances and actions for the operator to select and command.

Based on the surveyed literature, key challenges for autonomous operation in unstructured environments include:

PERCEPTION Segmentation of objects/affordances from single-view 2.5D data often fails in real-world settings due to misalignments and the limited area observed. This problem is exacerbated when the target affordance is occluded, or part of the robot geometry further obstructs the view.

ROBUST AND FAST MOTION PLANNING Synthesis of collision-free motion which handles the robot's redundancy (or lack thereof) to solve a task based on live sensor data.

DYNAMIC CHANGES Causing synthesised plans or motion under execution to be in a collision, requiring adaption or soft stopping and replanning.

ENVIRONMENT REPRESENTATION/COLLISION CHECKING Sensing artefacts can lead to spurious planning failures. As an alternative, semi-autonomous approaches often resort to having an operator perform the final verification.

COMPUTATION, MEMORY, AND COMMUNICATION Limited onboard computation capability, memory, battery runtime, and communication constraints require trade-offs between solution quality and execution speeds to achieve an online planning capability.

6.2 METHODOLOGY

We now detail the components of our fully integrated shared autonomy system. It consists of environment mapping, autonomous stance selection and navigation, model-free object segmentation, automatic grasp affordance selection, collision-free motion planning and execution, and continuous, dynamic scene monitoring and failure recovery (see Figure 60). We expand on prior work on shared autonomy [Marion et al., 2017] by

1. Incorporating *dense visual mapping* to capture and fuse multiple views and sensors into a dense 3D representation of the workspace to increase the robustness of model-free affordance segmentation algorithms.
2. Integrating scalable and robust *collision-free motion planning* using our proposed efficient hybrid scene representation.
3. Adding environment awareness and adaptation to dynamic changes through *continuous scene monitoring* with failure recovery enabling safe operation in shared workspaces.
4. Selecting an *optimised base position* for floating-base robots. We employ the inverse Dynamic Reachability Map (iDRM) [Yang et al., 2016a] to maximise workspace manipulability. Optimising the base placement reduces cycle time and increases planning and autonomy robustness.

6.2.1 *Perception*

At the core of our perception is a continuously updated map encoding free, occupied, and unknown space in an accurate, dense yet memory-efficient representation of the environment [Hornung et al., 2013].

FILTERING In order to reduce artefacts which hinder the reliability of motion planning and object segmentation algorithms, raw RGB-D and block-matched

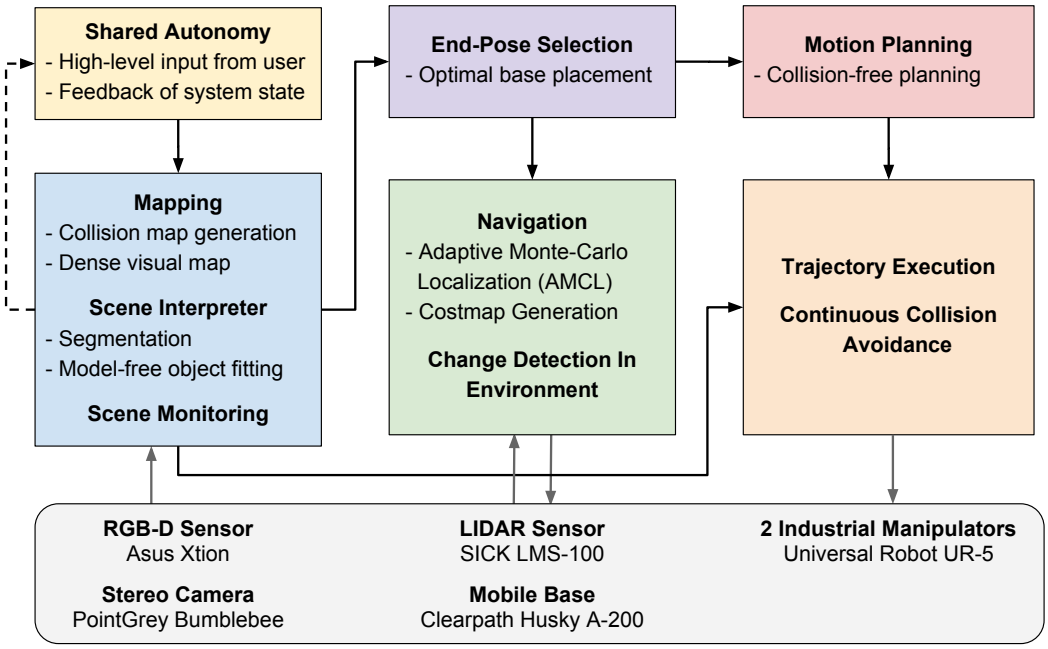


Figure 60: Overview of the robust shared autonomy system.

stereo sensor data are filtered in a preprocessing step to remove self-observations, shadow points, and noise. Multiple point cloud sources can be adjoined and fused. The preprocessing pipeline is shown in Figure 61. In a first step, pass-through and downsampling filters are applied in the sensor z-direction to cut out noisy areas which are far away (beyond the area of interest) or too close for the sensor to provide reliable readings as well as to regularise samples onto a voxel grid. It also removes invalid points and mixed pixels. In a second step, we project the point cloud into the robot base frame and filter measurements outside the work area. Subsequently, we use the geometric model of the robot and its proprioceptive sensing to remove self-observations. Finally, we eliminate unconnected patches using a statistical outlier removal. We combine measurements from multiple point cloud sensors, here a stereo camera and an RGB-D sensor, and apply the above filter chain before integrating their measurements into the map.

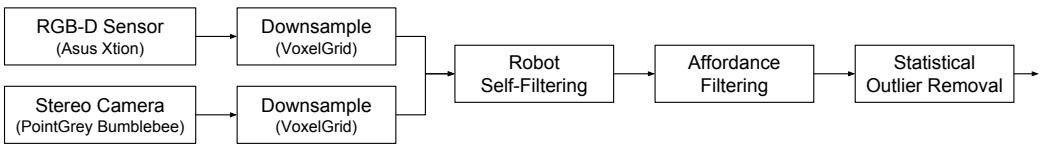
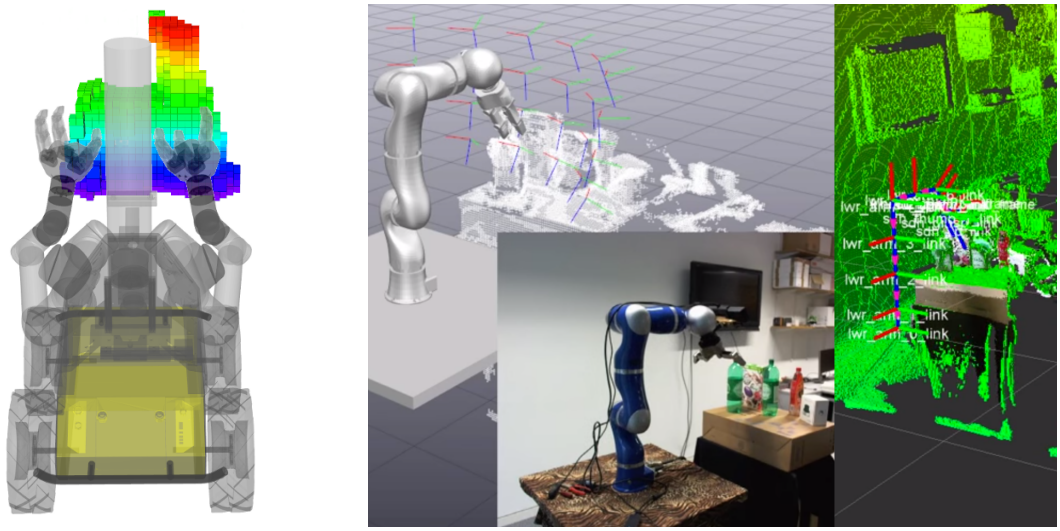


Figure 61: Filter chain from raw RGB-D and block-matched stereo data to data ready for integration into the map.



(a) (b)

Figure 62: Scene representations during the experiments: (a) Bimanual mobile manipulator with OctoMap used for planning, (b) Dual representation for fixed-base manipulator: Dense, visual map for segmentation shown on the left, and OctoMap for planning to the right.

HYBRID ENVIRONMENT REPRESENTATION We fuse preprocessed and adjoined point cloud into a memory-efficient, probabilistic octree-based scene representation that allows for change detection [Hornung et al., 2013]. Where available, we additionally fuse information from force/torque or fiducial sensing into the same environment map for collision detection. However, while this representation is efficient for collision and change detection, it does not contain the resolution and detail required to assist in manipulation.

We address this by creating a dense, high-resolution map based on the *Truncated Signed Distance Function* using the method described by Whelan et al. [2012]. This high-resolution map, an example of which is shown in Figure 68a, enables us to make use of multiple views to increase object segmentation and grasp affordance robustness. When we use this method with a point cloud sensor mounted the end-effector of a highly accurate fixed-base manipulator, we replace visual tracking with forward kinematics to increase mapping speed and reduce cycle times. In other cases, we use the forward kinematics to seed the tracking and mapping.

This dual environment representation (Figure 62) allows efficient collision avoidance, change detection, and tracking while providing a high fidelity fused map for accurate model-free affordance segmentation.

SEGMENTATION Crucially, the perception system needs to be able to extract objects of interest based on high-level user input and without prior models. These affordances are segmented from the higher resolution representation resulting in more accurate modelling of the areas of interest. After segmentation, this results in an efficient hybrid representation of segmented affordances and an environment occupancy map for planning and autonomy.

In order to segment objects of interest without a prior model, we use a combination of geometric insights and task-informed assumptions in an algorithm similar to Rusu et al. [2009]. First, we assume that objects of interests are placed on an approximately horizontal surface (e.g., a table or shelf) and have sufficient clearance from one another for a gripper to pass between to pick them up. These assumptions enable us to use segmentation by normals to extract a large, continuous plane. We then apply Euclidean clustering to extract distinct clusters of individual objects. Finally, we reconstruct a mesh through triangulation and fit an approximate bounding box shape primitive affordance annotated with candidate grasps to the mesh. The whole process is shown in Figure 68.

6.2.2 *Continuous scene monitoring*

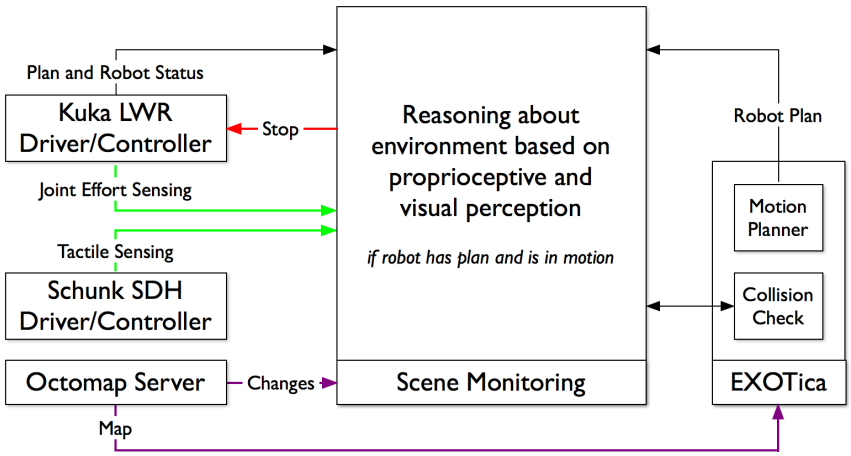


Figure 63: The implemented scene monitoring using different perception modalities: visual RGB-D data, tactile, and joint effort sensing. Diagram showing modules active during Experiment 1 (Kuka LWR).

Key to safe operation in shared workspaces is the ability to identify whether dynamic changes affect the intended robot motion. We distinguish between changes

that alter targets and affordances triggering replanning as well as updates which affect the safe and collision-free execution of the motion.

The continuously integrated discretised occupancy map is the basis for tracking changes. Our scene monitoring and reasoning (shown in Figure 63) works similar to the one described by Hermann et al. [2015]. Instead of analysing a sequence of swept volumes, we check the waypoints of the trajectory currently being executed directly against the collision map and its changed areas upon every map update. This choice enables us to efficiently run scene monitoring collision queries on the onboard CPU at sensor frame rate, with the GPU free to be used, for instance, for dense visual mapping or scene interpretation. When the scene monitoring detects that a future waypoint might result in a collision, it halts the execution and falls back to the human operator for decision-making. During the meantime, the scene monitoring continuously observes changes in the map and resumes motion execution when the remaining trajectory becomes collision-free again. Alternatively, a replanning is triggered on expiration of a countdown.

6.2.3 *Shared autonomy*

A shared autonomy user interface serves as an abstraction layer above task complexity. It enables the user to provide high-level objectives, gives feedback as well as involves the operator in decision-making if necessary.

Our shared autonomy builds on the task execution system described by Marion et al. [2017] which fills a sequence of task primitives with details acquired through operator-assisted perception. The operator can review, pause, and amend the execution at any time, and the autonomous mode can be resumed immediately after a phase of manual operation.

We expand on the work by Marion et al. [2017] to extend the task sequence system with automatic review and approval of planned motions through continuous scene monitoring and reasoning to reduce the amount of required human intervention. As a result, the modified system only falls back to the operator when unavoidable. In order to achieve this, a task tree with task dependencies is defined and automatically expanded to synthesise sequences comprised of high-level action primitives in response to perception input and changes in the environment.

Each high-level task primitive automatically expands into a series of verifiable low-level tasks, with the success criteria associated with each action monitored. For instance, grasping is executed as a power grasp with force, tactile, or visual

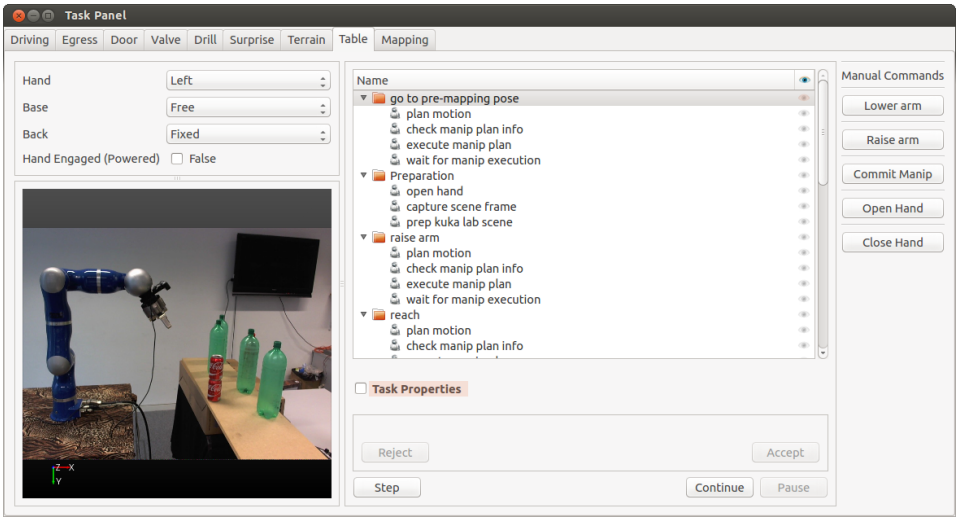


Figure 64: The shared autonomy task panel view for the table clearing task. A third-person surveillance camera view is shown alongside the expanded task tree. Manual intervention actions are accessible via buttons on the right panel. The progress through the task tree is visualised at every stage, and the operator can pause and manually step through execution, as well as resume autonomous operation. A confirmation dialogue is presented when the system falls back to the operator for approval. This is the case, for instance, when the system is not confident that it can proceed safely.

feedback serving as a success/completion criterion with a visual inspection for recovery. Automated reasoning immediately starts execution of valid trajectories to reduce cycle time.

6.2.4 *Collision-free motion planning*

Our system uses a combination of sampling- and optimisation-based planning algorithms included in Extensible Optimisation Toolset (EXOTica) [Ivan et al., 2019] to synthesise collision-free manipulation trajectories. Using affordances and occupancy grids from our perception module (Section 6.2.1, Figure 62), motion planning problems are centred around constraint or task sets [Fallon et al., 2015] that can be composed to represent high-level objectives. We build a constraint set based on the reachability and manipulability given a selected affordance and compute a goal configuration $\mathbf{q}_{goal}$ as outlined in Chapter 2. Given start and goal configurations $\mathbf{q}_{start}, \mathbf{q}_{goal}$, RRT-Connect is used to find a trajectory [Kuffner and LaValle, 2000]. Additionally, we can further apply the method introduced in Chapter 3.

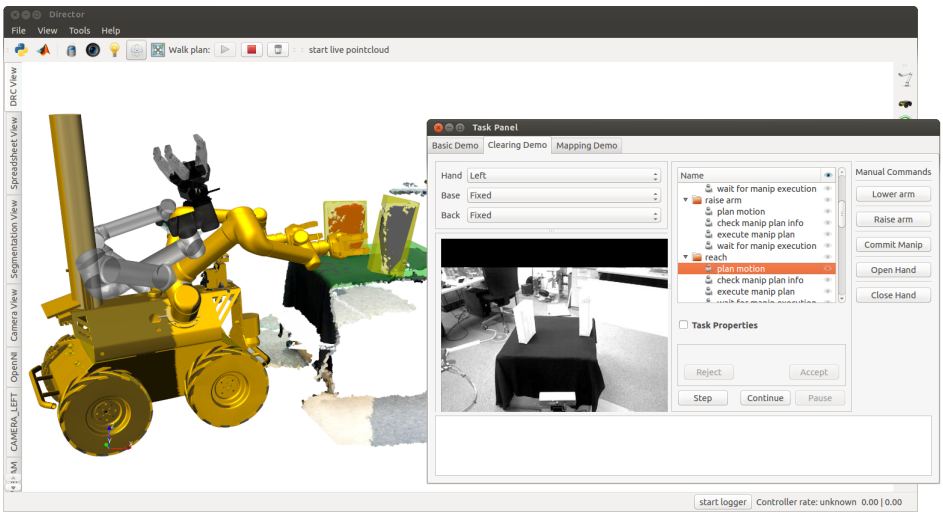


Figure 65: The user interface for the bimanual mobile manipulator showing live perception data, segmented objects and fitted affordances as well as candidate plans (gold). Situational awareness camera data is displayed in the shared autonomy panel. The system automatically executes the candidate trajectory if deemed safe by the continuous scene monitoring module. The operator can manually pause and step as well as adjust affordances and plans in the interactive user interface.

6.2.5 *Extension to free-floating base systems*

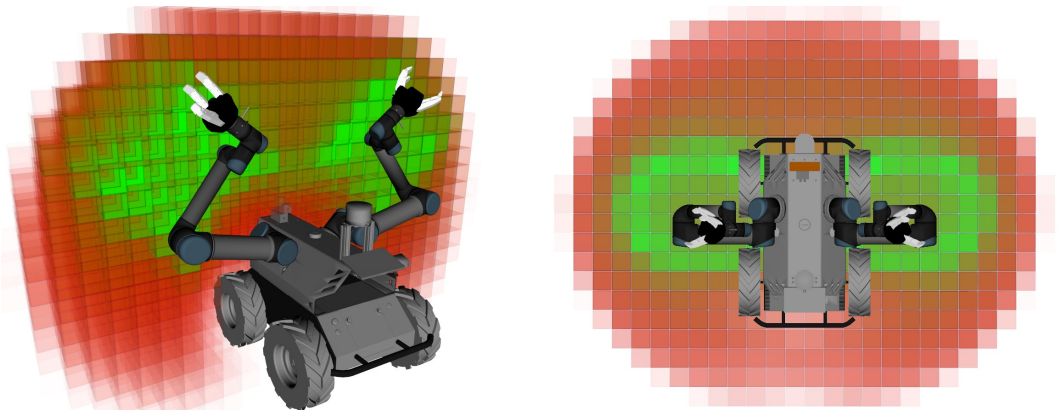
The system discussed so far is generic and was experimentally applied to and validated with a fixed-based manipulator. In the following paragraphs, we highlight additional components required to extend this system to mobile robots.

The kinematic reachability of non-redundant mobile platforms is often limited by potential self-collision, mounting points, and complex environments. This is demonstrated, for instance, by the reachability map of our mobile manipulator platform shown in Figure 66: Areas of high manipulability are limited and not necessarily overlapping with the field of view of installed sensors.

OPTIMISED FREE-FLOATING BASE POSITIONING In order to increase manipulability, we leverage the mobile base to reposition the manipulator based on intended motion plans. We select an appropriate standing pose to maximise manipulability using iDRM. This is the first time it is applied to improve autonomous operation, reduce cycle time, and increase robustness. Maximising manipulability is essential to cope with dynamic change and sensing uncertainty as it maximises

the likelihood that nearby task-space areas can be accessed from the same base placement.

We recap the core principles of the iDRM algorithm, as described in Yang et al. [2016a]. An offline preprocessing step generates a large number (millions) of self-collision-free robot configurations and transforms them such that the end-effector is at the origin of a voxel grid. Configurations which can reach a voxel with the end-effector at the origin are stored in the reach list of the corresponding voxel. Unlike previous algorithms for floating-base placement, iDRM stores voxels occupied by configurations in an occupation list during this offline step such that only a single collision query is required to find the voxels which are occupied in the environment representation and filter associated configurations. The remaining subset of the iDRM is collision-free. We can now select the most suitable goal state, i.e., a collision-free base location from where the robot can achieve the desired end-effector pose with the highest manipulability, based on a pre-calculated manipulability score. Yang et al. [2016a] demonstrated that iDRM can find valid end-poses in real-time in complex environments, which can then bootstrap bidirectional motion planning algorithms. Automatically deducing the goal state information, which is typically provided by a human operator, using our proposed process is crucial for robot autonomy by reducing planning failures and increasing the overall success rate.



(a) Backview (b) Top-down view

Figure 66: Reachability map for the dual-arm mobile manipulator used during our experiments. The mounting point and non-redundancy of the individual arms severely restricts workspace manipulability, while the compact construction means that sensor placement and highly manipulable workspace are not always aligned. This highlights the need for and importance of intelligent mobile base positioning to maximise task success.

NAVIGATION During mobile manipulator experiments, we used a front-mounted horizontal laser rangefinder for navigation and localisation against a static map (Adaptive Monte-Carlo Localisation [Fox et al., 1999]).

Goal base positions are computed using iDRM and passed to the Robot Operating System (ROS) navigation stack which provides cost map generation and path planning out-of-the-box.

6.3 EVALUATION

We validated the flexibility and adaptability of our system with hardware experiments on two different platforms. First, we use a 7-DoF Kuka LWR3+ manipulator with a Schunk SDH dexterous hand to clear objects from a table. For perception, an Asus Xtion Pro Live RGB-D sensor is mounted on the end-effector for dense visual mapping and continuous scene monitoring. Second, we use a bimanual Clearpath Husky with two 6-DoF Universal Robot UR5 manipulators fitted with Robotiq 3-finger grippers to clear a scene. For this system, the perception is based on an identical Asus Xtion, a PointGrey Bumblebee2 stereo camera, and a Sick LMS-100 LIDAR sensor. All sensors are mounted onboard the respective platform; no external sensing/perception is used.

For the industrial manipulator, computation is carried out on a personal computer with an Intel Core i7-4770 CPU with 3.4 GHz, 16 GB 1600 MHz DDR3 memory and an Nvidia GeForce GT645 GPU. For the experiments with the bimanual mobile manipulator, most computation is performed onboard the robot (Intel Core i5-4570TE with 2.7 GHz, 8 GB 1600 MHz DDR3 RAM), with mapping, planning, and shared autonomy offloaded to the operator workstation (same as above). Interprocess communication is maintained using ROS and LCM [Huang et al., 2010]. Onboard compression enables the operation via a wireless link without a line of sight. As a result, peak bandwidth use is approximately 3 MB s^{-1} for streaming of two depth cameras, three situational awareness cameras, as well as telemetry, and lidar. This bandwidth requirement permits experiments with a blend of teleoperated and autonomous sequences with no line of sight as depicted in Figure 67.

In our experiments, we illustrate that by using the same system architecture, we can generalise the manipulation and continuous scene monitoring system of a fixed-base robot to a floating-base system with two arms. As is evident, the mobile robot requires additional components compared with the fixed-base manipulator, which allow us to take full advantage of its advanced capabilities.

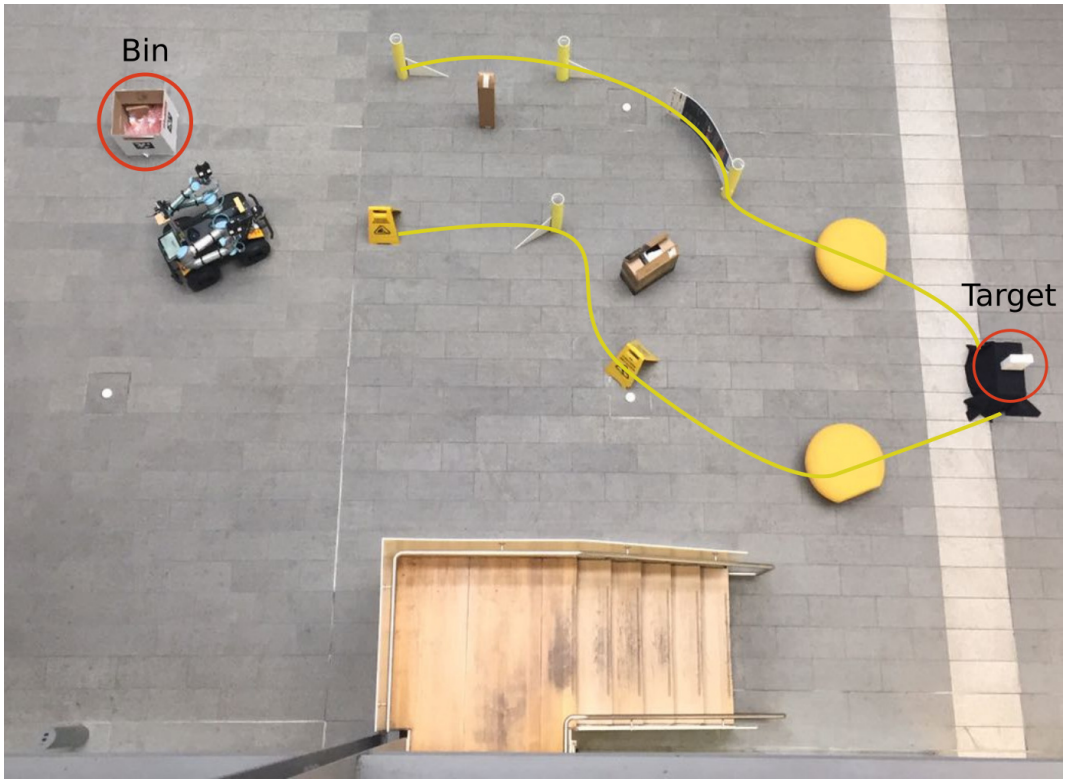


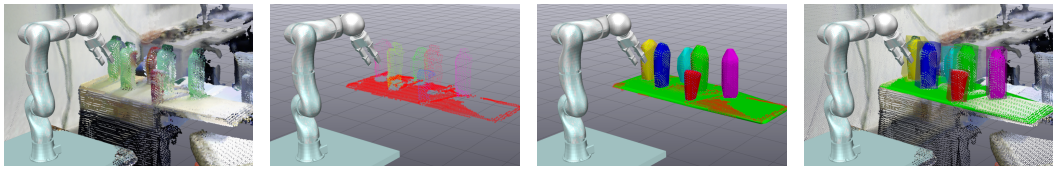
Figure 67: Navigation and recovery task with no line of sight and restricted communication with teleoperated as well as autonomous sequences.

6.3.1 *Experiment 1: Table clearance with a Kuka LWR arm*

The first experiment is an autonomous table clearance task. Here, the robot proceeds to identify objects on a surface given a single point on that surface and synthesises a plan for clearing the table. A dense visual map is created through volumetric fusion. Our system segments objects and affordances (Figure 68), with the corresponding occupancy grid for collision-free motion planning displayed in Figure 62b. The shared autonomy interface is shown in Figure 64. In this set of experiments, tactile information from the gripper is used during grasping, and joint effort sensing used as part of the continuous scene monitoring, see Figure 63.

6.3.2 *Experiment 2: Autonomous scene clearance with a bimanual Clearpath Husky*

In the second experiment, a mobile manipulator picks up objects in a shared workspace with humans and places them into a garbage bin (Figure 59). Operator input is limited to providing a single point in the scene denoting the surface for



(a) Map (b) Clusters (c) Meshes (d) Affordances

Figure 68: Steps of our model-free affordance segmentation pipeline: Starting with a dense visual map, a table surface and distinct point clusters are extracted, and meshes fit via triangulation. Finally, primitive shape affordance are fit to the meshes and annotated with possible grasps.

the robot to clear. This experiment is repeated with two different scenarios, where the robot uses either one or both arms depending on the number of objects placed on the designated surface. Each of the two scenarios has been tested with more than 20 different variations of the environment, including different objects, tables, and configurations of the furniture. Samples of these executions are highlighted in the video available at https://youtu.be/5jFU7oCP4vk. A striking feature of the proposed shared autonomy system is its capability to automatically determine the number of end-effectors required to pick the objects from the surface efficiently. Furthermore, in the case of multiple objects, it automatically determines whether both are reachable simultaneously through optimised base positioning using iDRM, and it leverages the floating-base to reduce cycle time.

In the following, we detail the tasks during operation and how they are divided between the robot and the operator. It is worth noting that operator tasks only convey high-level goals and confirmations to the robot.

1. *Operator* provides a point on the surface to be cleared.
2. *Perception* segments the scene and identifies the number of objects on the surface.
3. *iDRM* computes an goal state to grasp the objects.
4. *Robot* navigates to the base pose provided by iDRM.
5. *Operator* verifies that the robot navigated to within the vicinity of the target surface. They also confirm the surface by providing a further point.
6. *Perception* re-segments the scene in order to ensure that its plans remain compatible with any changes.

7. *Robot* plans and executes arm motions to grasp the object in three substeps, first to reach pre-grasp frame, then to grasp frame and finally grasps the object.

8. *Operator* verifies that the object(s) has been grasped.

9. *Robot* moves its arms in driving configuration while searching for the bin and navigates to it.

10. *Robot* drops the objects in the bin.

SCENARIO 1: SINGLE OBJECT ON SURFACE In this scenario, we show the robot removing an object from the indicated surface. Once the operator has provided the target surface, the robot automatically segments the scene and identifies that only one object is placed on the surface. Thus, the iDRM module provides a whole-body goal state solution using only the left arm, as shown in Figure 70a.

In the video available at `https://youtu.be/5jFU7oCP4vk`, we further demonstrate that the robot can cope with dynamic changes of the scene, such as relocation of the target object and bin.

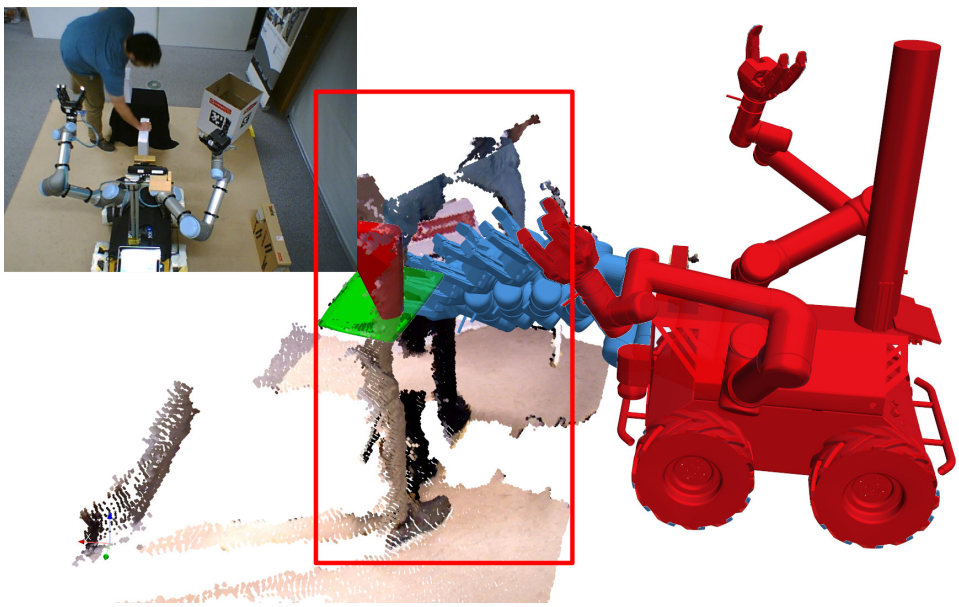
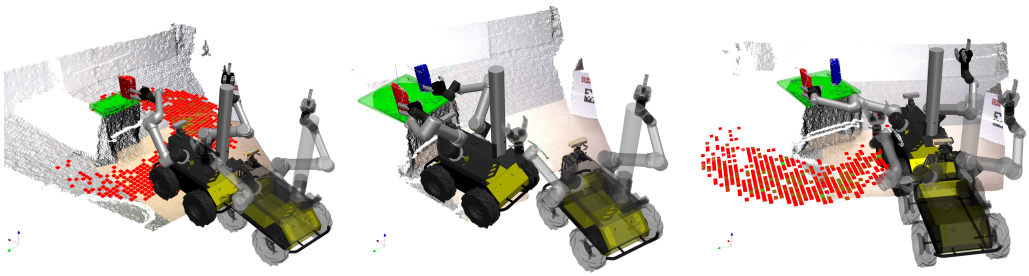


Figure 69: The scene monitoring continuously integrates fused and filtered sensor data in an OctoMap and reasons about changes: Here, a human reaches into the robot workspace crossing the planned trajectory (shown in blue), and the robot halts execution (robot state shown in red).

Our scene monitoring is demonstrated by Figure 69. During execution, the workspace is monitored, and motions are paused as soon as future trajectory waypoints would result in a collision. The map updates at 13 Hz at a 3 cm resolution (i.e., at frame rate—the sensor data is being captured at 15 Hz) with the verification whether the future trajectory waypoints are collision-free taking approximately 50 ms. Note, that this speed and interactivity can be scaled with the operating velocity of the manipulators by decreasing workspace voxel grid resolution and an increased safety margin/padding.

SCENARIO 2: TWO OBJECTS ON SURFACE In the second scenario, the robot removes two objects from the indicated surface. The robot identifies the number of objects on the surface autonomously. It utilises the iDRM module to obtain a solution which satisfies a whole-body goal state, reaching both objects simultaneously, as shown in Figure 70b. On the other hand, as shown in Figure 70c, if one were first to obtain a solution for the red object and subsequently for the blue, the robot would have to re-navigate to a new base-pose to reach the blue object with the right arm.



(a) (b) (c)

Figure 70: (a) Single arm goal state computed by iDRM. The red and green squares illustrate the variety of different base poses that satisfy the six-DoF constraint to reach the pre-grasp end-effector frame. (b) Dual arm goal state computed by iDRM. In this scenario there are limited feasible base poses, hidden under the robot, that satisfy the two six-DoF constraints, one for each arm. (c) Single arm goal state computed by iDRM reaching only the red object. Note that given this pose the blue object is not reachable by the right arm.

6.3.2.1 *Analysis of task-wise timing*

In Figure 71a and Figure 71b, we provide a detailed timing cumulatively for each task described above during the single object and the bimanual scenario,

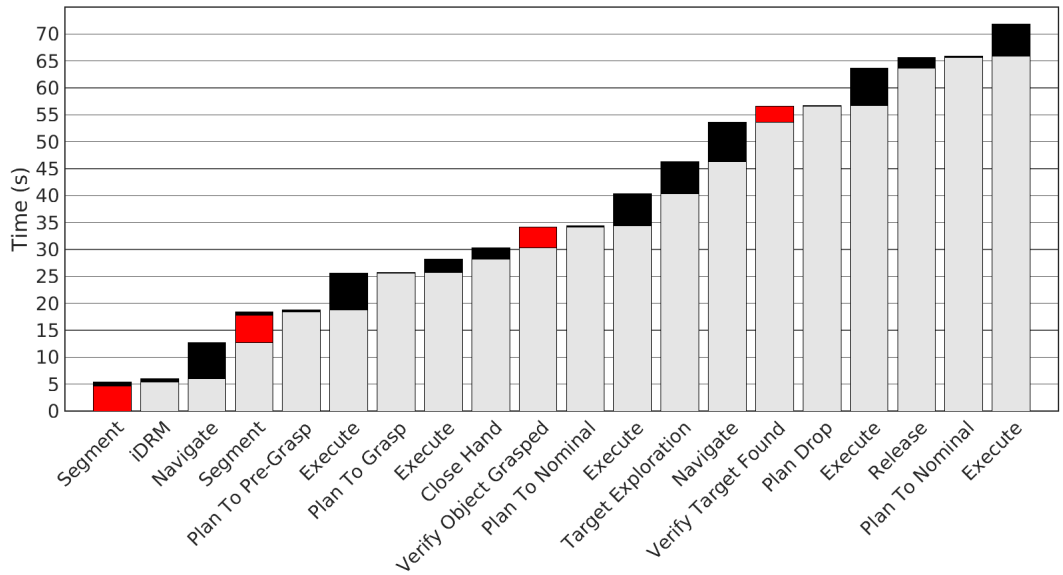
respectively. As it is evident from both figures, the interaction time of the operator, coloured in red, is minimal during the successful completion of the experiments. Also, it is worth noting that the duration of the planning steps is negligible, and most of the time is spent on execution. Execution times are significant due to the very restrictive velocity limits we are using to align with safety standards when the workspace is shared between a robot and a human.

6.4 DISCUSSION

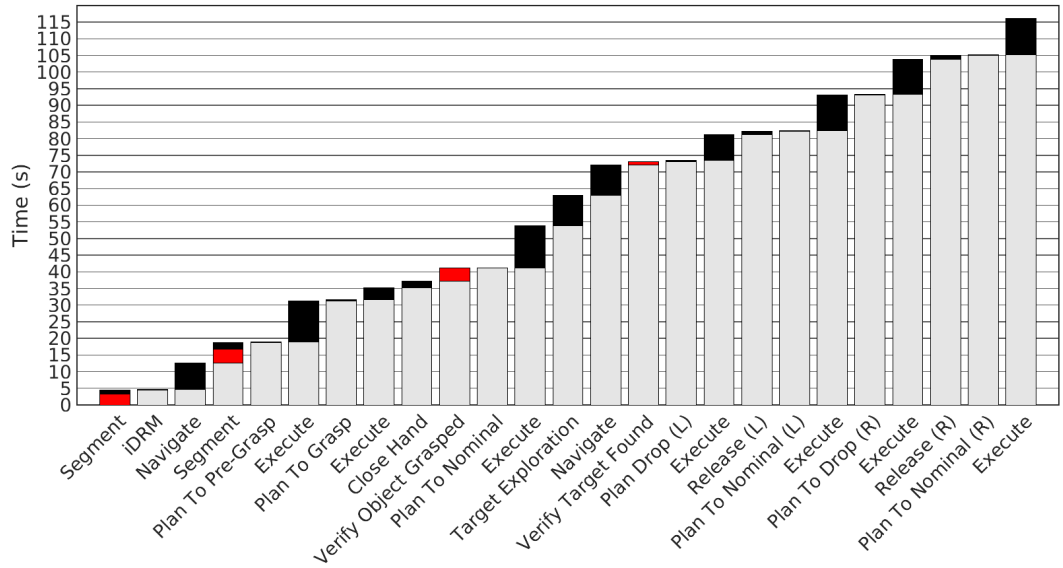
We demonstrated our proposed shared autonomy system to complete tasks with minimal high-level user input operating autonomously for the majority of the execution. It is able to deal with dynamic changes, such as updates of the target affordance or bin location, as well as an obstacle or human entering its shared workspace by safely pausing, replanning, and resuming motion execution.

Our scene monitoring can be added on top of many motion planning and execution pipelines and runs at 20 Hz on CPU, which is responsive enough for operation on a moving platform. Work by Hermann et al. [2015] checks swept volumes of trajectories on the GPU with additional predictive tracking of obstacles at 6 Hz to 8 Hz and with replanning using a library of motion primitives. As a result, they interactively adapt the execution speed or interrupt motion if in a collision. For our applications on mobile platforms with limited battery capacity and often without a top-of-the-line GPU, our proposed scene monitoring is suitable and efficient. However, pausing and replanning may result in short interruptions.

Selecting a suitable mobile base position improved autonomy robustness and adaptability to changes by maximising manipulability. Especially in the second scenario, due to the appropriate placement of the base, both objects can be picked at once optimising the execution time for any single reaching task. Inverse reachability maps have been previously used for instance during the DRC Trials [Kohlbrecher et al., 2015]. Using the iDRM algorithm, which moves collision checking to the offline preprocessing phase, enabled real-time interactive goal state queries during both teleoperation and increased robustness for our autonomous runs. However, the physical size of the workspace and robot model required large amounts of runtime memory to cover only a part of the workspace of the robot. For our single-arm experiments, we used 230k samples which translated to 520 MB memory, and the 1.3M samples of the bimanual dataset consumed 3.1 GB.



(a) Scenario 1: Single object



(b) Scenario 2: Two objects

Figure 71: Time taken by individual steps during the experiments. Operator input is shown in red and the autonomous behaviour is shown in black. Grey indicates the cumulative experiment time up to the start of the current step.

6.5 CONCLUSIONS

We have presented a shared autonomy system with continuous scene monitoring incorporating dense visual mapping, collision-free motion planning, and scene awareness. We furthermore showed the first employment of iDRM Yang et al. [2016a] to improve the robustness of autonomous motion planning and execution by selecting a mobile base position which maximises manipulability. The implemented continuous scene monitoring ensures safe execution of planned motion. It also paves the way for continuous adjustment of behaviour according to changes in the environment. It enables fast and accurate manipulation, allowing recovery when potential conflicts are detected during motion execution.

Future work includes continuous adaptation and local replanning of motion trajectories in response to environmental changes captured by our scene monitoring. To make optimal motion synthesis tractable in this case, we plan to leverage the concepts proposed in Part III. Additionally, to improve scene monitoring prediction, it is recommendable to incorporate motion flow and predictive tracking similar to Hermann et al. [2015].

One of the critical limitations observed in this chapter when moving from fixed-base industrial manipulators to mobile manipulation platforms and humanoids was the assumption that the system can execute planned whole-body trajectories reliably. Most commercially available mobile manipulation platforms, however, consist of unsynchronised subsystems (individual manipulators and the mobile base) operating at varying control frequencies and with unknown, changing delays. This makes successful execution of planned behaviours challenging and limits execution speed. In practice, it further to treat locomotion and manipulation as separate, sequential sub-problems (i.e., to first navigate/locomote to reposition to a fixed location, then carry out fixed-base manipulation). In fact, in this work, we employed an optimised base placement to address these limitations for practical applications. For a humanoid robot, this assumption is additionally only applicable for slow, quasi-static, and kinematic trajectories due to the corrective action of higher-level tasks within the balancing controller.

For the case of a mobile manipulator, we address these challenges in the following chapter through the use of synchronised, whole-body control inspired by frameworks deployed on legged robots allowing fast and flexible deployment of coordinated whole-body motion execution. We further demonstrate an application of tracking and continuous adaptation in response to changes in the target location.

7

COORDINATED WHOLE-BODY CONTROL FOR CONTINUOUS MOBILE MANIPULATION

Traditional industrial automation achieves productivity gains through fast and precisely repeated pre-programmed motion in controlled environments. Here, the robots are firmly mounted to the ground allowing high-speed movement, enclosed with security fencing, and attached to virtually unlimited shore power. These systems are custom-designed at the beginning of a product life cycle and amortise costs over a large production volume with low individual variability (mass manufacturing). The recent trend for customisation, however, is dominated by small batch sizes and short cycle times with frequent reconfiguration of work cells. This requirement demands flexibility and agility to respond quickly to changes in demand and is a particular challenge for the competitiveness of Small and Medium Enterprises (SMEs).

Recently, Autonomous Mobile Robots (AMRs) have started to replace conveyor belt systems as companies shift from line to matrix production. In product line manufacturing, a work cell strictly follows another with parts moving on fixed conveyor systems from one cell to another. In contrast, matrix production features work cells that specialise in carrying out a processing step, with AMRs carrying the parts between stations. The use of AMRs in the matrix production system enables to reconfigure workflows without changes to cells.

In order to achieve higher flexibility and address increasing labour shortage, companies have turned to integrate light-weight collaborative robots (cobots) widely into shared human-robot-assembly lines. cobots are safe to operate in the vicinity of people without extensive safety systems and can be programmed or taught by demonstration [Wang et al., 2018]. However, as they are operating from a fixed-base, their reachability is limited as we have shown and analysed in the previous chapter. In practice, this often requires a large number of cobots per plant and poses challenges for re-deployability and versatility.

In order to overcome this limitation, mobile collaborative robots—combinations of AMRs and cobots—have been proposed. Mobile collaborative robots open up flexible deployment opportunities, yet, require new considerations. Some manufacturers propose mobile manipulation as the repositioning of flexible workstations

that automatically plug themselves to shore power (e.g., FANUC Robotics[1]). Many system integrators, on the other hand, have combined AMRs with differential or omnidirectional drive systems and off-the-shelf collaborative robots into commercially available mobile manipulators. These developments are of interest to both traditional workshops looking to automate tasks, e.g., machine tending in existing factory floors, as well as in the development of flexible manufacturing concepts in existing infrastructure.

However, most mobile manipulation solutions decouple the navigation from the manipulation problem effectively performing fixed-base manipulation using a repositionable manipulator. As we have demonstrated and discussed in Chapter 6, this approach is not only inefficient but moreover limits the range of applications: By disregarding the inherent redundancy of a mobile manipulation system, tasks such as the surface finish of extended parts are more challenging to achieve. Furthermore, the split in control systems harbours practical limitations for translating novel algorithms that use the full potential and redundancy of these systems to real-world deployment. This restriction became particularly evident in the previous chapter: Methods that synthesise whole-body loco-manipulation plans as we have introduced in Part II cannot readily be demonstrated using existing solutions.

In this chapter, we introduce a high-performance omnidirectional mobile manipulation platform with integrated whole-body control, real-time collision-free whole-body motion planning, and perception. We build on concepts and formulations for operational space and whole-body control widely used in legged robots and humanoid control and leverage it for efficient, continuous mobile manipulation, which also allows whole-body visual servoing. We evaluate our proposed system by demoing a chicken-head task showcasing the decoupling of operational space manipulator motion from base motion. We highlight planning and loco-manipulation capabilities in a simulated automotive fitting task with moving manipulation targets and demonstrate a sensor placement task for certification of assets on an outdoor test site. Finally, we deploy and evaluate our solution in field trials on an industrial oil and gas training facility on a sensor placement and manipulation task.

Videos accompanying the results in this chapter are available at

http://www.wolfgangmerkt.com/continuous-manipulation.

1 https://youtu.be/rQBnZuby05s.

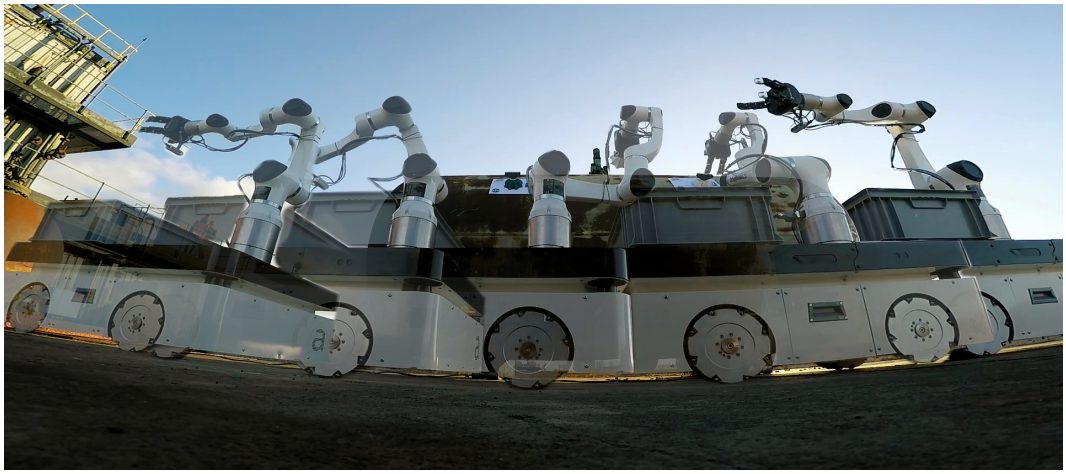


Figure 72: Continuous manipulation using whole-body control: an industrial Internet-of-Things (IoT) monitoring device is placed using live sensor feedback.

7.1 BACKGROUND

The recent proliferation of integrated solutions has been accelerated by the confluence of the maturity of the open-source robotics ecosystem (Robot Operating System (ROS) [Quigley et al., 2009]) and the availability of mature, standardised hardware platforms and software systems (e.g., ROS-Control [Chitta et al., 2017] for hardware abstraction and MoveIt [Chitta, 2016] for motion planning). This development has been further driven by increasing interest in ready-to-deploy industrial applications. At present, Autonomous Ground Vehicles (AGVs) are widely deployed for logistics and warehousing tasks such as moving shelves (e.g., Amazon/Kiva Robotics) and in-facility logistics (e.g., in hospitals and on factory floors). Manipulation tasks, on the other hand, are often limited to demonstrations of pick-and-place and material transport and are generally only applied to highly-specialised scenarios due to limitations of workspace interoperability, autonomy, battery runtime, and cost.

Autonomous mobile manipulation recently received a renewed focus in research with several high-profile international competitions centred on service robotics (e.g., World Robot Challenge, RoboCup@Home), disaster recovery/Urban Search and Rescue (e.g., DARPA Robotics Challenge, RoboCupRescue), or coordinated manipulation tasks (e.g., Mohammed Bin Zayed International Robotics Competition).

However, due to the challenge of planning loco-manipulation in real-time, locomotion/navigation and manipulation are often treated as separate problems. They are then joined and coordinated by a high-level state machine [Carius et al., 2018], sequence planner, or shared autonomy control interface as in Chapter 6. As optimal

base placement [Yang et al., 2017], navigation to the base placement, and fixed-base manipulation [Yang et al., 2016b, 2018a] are treated separately, systems are limited to applications using static targets and obstacles, and thus effectively disregarding the inherent redundancy of high-degrees of freedom (DoF) systems. However, to achieve time- and energy-optimal solutions, locomotion and manipulation need to be considered together.

Work in motion planning by Shin et al. [2003] considered both manipulation and locomotion in a joint optimisation problem maximising a directional manipulability metric. However, while the authors planned motion jointly, they enforced discrete repositioning and fixed-base manipulation in order to avoid errors from a lower-precision mobile base. Dang et al. [2011] studied humanoid loco-manipulation by planning in task-space introducing virtual joints for footsteps and an adaptive procedure to vary the number of foot placements. Together, these methods introduced important planning concepts and predominantly focused on manipulating static targets. Recent work addressed this by planning semi- and fully-constrained collision-free whole-body trajectories in time-configuration space using sampling- [Yang et al., 2018b] and optimisation-based [Merkt et al., 2019a] approaches. Using these advancements, the authors showcased highly complex motion manipulating moving targets in dynamic environments.

Early work in whole-body control of mobile manipulation systems by Hootsmans [1992] introduced the *Mobile Manipulator Jacobian Transpose* (MMJT) and demonstrated the ability to compensate vehicle motion from passive suspension to stabilise end-effector motions. Similarly, Yamamoto and Yun [1992] considered simultaneous motion to maximise manipulability of the end-effector while following a desired or guided operational space trajectory. More recently, Kim et al. [2019] applied Hierarchical Quadratic Programming (HQP) in a Stack-of-Tasks (SoT)—an approach traditionally used for the control of humanoid robots—on a holonomic mobile manipulator with continuous task transition.

In summary, Hootsmans [1992], Yamamoto and Yun [1992], and Kim et al. [2019] focused on instantaneous whole-body control to coordinate and compensate end-effector motion, while Shin et al. [2003], Dang et al. [2011], Yang et al. [2018b], and Merkt et al. [2019a] focused on loco-manipulation planning over longer horizons as an input into the former. In this chapter, we combine loco-manipulation planning as described in Part II with coordinated whole-body control for continuous manipulation in complex environments.

7.2 METHODOLOGY

7.2.1 *System overview*

Our system consists of a high-performance ($1.5\,\mathrm{m\,s^{-1}}$ maximum velocity) and high-payload (500 kg to 800 kg) omnidirectional mobile platform with a 6-DoF collaborative robot for a total of 9 DoF (Adabotics Ada500). The system features a built-in 1 kHz whole-body control layer based on ROS-Control, with the individual system components shown in Figure 73. The platform uses two horizontal laser rangefinders as well as an Intel RealSense D-435 RGB-D sensor mounted on the wrist. It further contains two onboard computers with one dedicated to control, motion planning, and safety features and the other performing perception tasks such as mapping and object identification and tracking. The system further comes with a remote control user interface available from any phone or tablet computer and is equipped with battery capacity for a full-shift autonomous operation (8 h to 10 h). In order to maximise operation with limited onboard power, we consider energy efficiency in our motion optimisation and target continuous, non-stop manipulation through execution of whole-body trajectories using coordinated, whole-body control. We continuously monitor the environment for conflicting changes and respond using a combination of real-time planning and reverting to operator input via shared autonomy using the method described in the previous chapter.

7.2.2 *Problem formulation*

Following the notation introduced in Section 2.1, the configuration for a robot manipulator is described by $\mathbf{q}_j \in \mathbb{R}^{n_{q,j}}$, where $n_{q,j}$ is the size of its configuration vector and $n_{v,j}$ the size of its tangent vector, or DoF. Its state at time instance t is $\mathbf{x}_{j,t} = (\mathbf{q}_{j,t}, \boldsymbol{\nu}_{j,t}) \in \mathbb{R}^{n_{q,j}+n_{v,j}}$ and directly and accurately measured. It can be directly controlled via position control, velocity control, admittance control, impedance control, or torque control. Furthermore, the state and the controls $\mathbf{u}_{j,t} \in \mathbb{R}^{n_{v,j}}$ are usually bounded (e.g., by joint position, velocity, acceleration, current, or torque limits) which limits the scope of motion planning and the working envelope in which we may want to avoid collisions or seek contacts. On the other hand, the state of a mobile platform moving on a surface is defined as $\mathbf{x}_{base}$. Its configuration space has the topology of the *Special Euclidean group* and

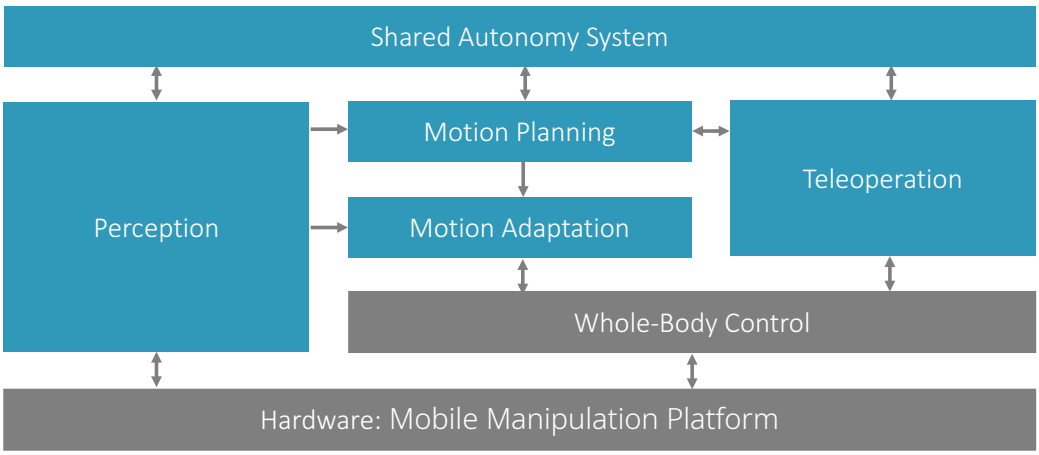


Figure 73: Overview of the deployed system: All components are modular and can be replaced due to the specification of commonly adopted interfaces, e.g., ROS-Control. We deploy the method described in Part II for motion planning and motion adaptation. For teleoperation, we use different input devices such as gamepads and mobile apps to interactively provide input to the goal state planning described in Chapter 2. The perception module and shared autonomy system are introduced in Chapter 6. We developed, designed, and built the mobile manipulation platform in-house.

it is unbounded, i.e., it has no translation and rotation limits. Additionally, some mobile platform designs are non-holonomic, which means that we cannot control the state in all directions directly. The constraints can be accounted for in the state transition and solved, for instance, using an optimal control approach as described in Section 5.1.

Despite these differences, our goal is to express the state of the whole robot as $x = [x_{base}; x_j] \in \mathbb{SE}(2) \times \mathbb{R}^{n_{q,j}}$, where x describes the state of a system with $3 + n_{v,j}$ DoF. This choice has a significant impact on the design of the controller. We will now formulate the combined loco-manipulation problem as a whole-body control problem.

7.2.3 *Whole-body control*

We formulate the whole-body control problem as the one-step look-ahead minimisation of an optimisation problem subject to all bound, linear and non-linear inequality and equality constraints and account for modelled actuation delays:

$$\arg\min_{\mathbf{x},\mathbf{u}} \ell(\mathbf{x},\mathbf{u}) \tag{47}$$

$$\text{s.t.: } h(\mathbf{x},\mathbf{u}) = 0 \tag{48}$$

$$g(\mathbf{x},\mathbf{u}) \leqslant 0 \tag{49}$$

$$c_{lb} \leqslant A\mathbf{x} + B\mathbf{u} \leqslant c_{ub} \tag{50}$$

Here, the upper and lower bounds of the decision variables are updated based on the current state, timestep, and proximity to higher-order limits through integration similar to the work by Del Prete [2018]. The equality constraints $h(\mathbf{x},\mathbf{u}) = 0$ and inequality constraints $g(\mathbf{x},\mathbf{u}) \leqslant 0$ are a set of non-linear functions of state and control following the concept of task-maps introduced in Section 2.1. Linear equality, inequality, and bound constraints (e.g., joint position and velocity limits) are encoded using A and B. We use a different set of constraints for each control mode. For example, we use general equality constraints on the end-effector position to trace a path with the tool, and we use general inequality constraints for limiting the tool speed. We can formalise a large variety of control modes using the same generic framework by combining different sets of constraints. We do the same with the optimality criteria $\ell(\mathbf{x},\mathbf{u})$, which often takes the form a weighted sum of squared error terms.

As we are solving a limited size problem initialised from the present state, we obtain fast convergence for control using a Non-linear Programming (NLP) formulation. While traditionally Quadratic Programming (QP)-based methods are chosen for whole-body control in legged platforms, e.g. in [Bolotnikova et al., 2017], the comparatively small size of a mobile manipulation problem (9–17 DoF) permits the use of non-linear optimisation to include much more expressive cost and constraint terms without linearisation. An overview of how this solver integrates into the full framework is shown in Figure 74.

7.2.3.1 *Low-level control*

For low-level control, we interface all components using EtherCAT and synchronise them with a common EtherCAT master. We provide an abstraction to our low-level

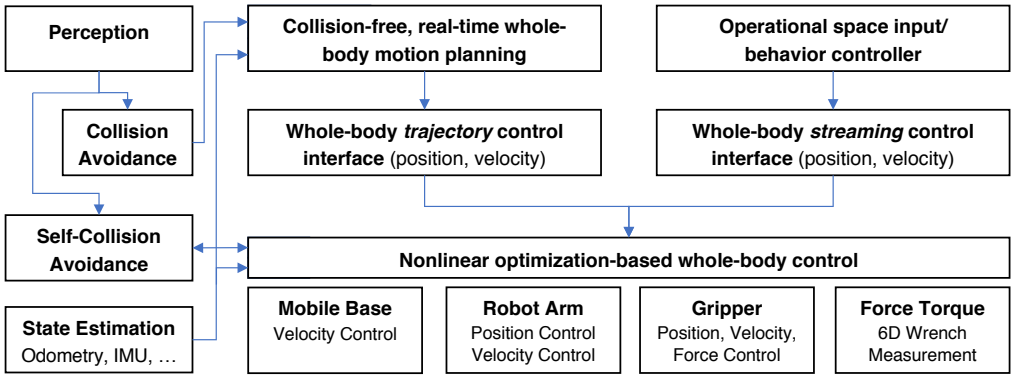


Figure 74: Overview of the non-linear optimisation-based whole-body control framework: Commands can be issued either as whole-body trajectories or from operational space targets in a streaming mode, where individual commands/targets are sent to the controller at every timestep. The controller satisfies all applicable bound constraints as well as general non-linear safety constraints (e.g., self-collision avoidance).

control system using as a ROS-Control interface [Chitta et al., 2017]. This interface enables users to load and apply controllers developed in other contexts flexibly. While we chose to connect all components using EtherCAT in this system, the ROS-Control framework enables us to provide a synchronised control loop for systems with different communication protocols and control frequencies. We send all joint commands to the motor drivers at 1 kHz. However, we have also experimented with varying lower control frequencies.

In this chapter, we use a holonomic mobile base. In this case, the velocity commands of the individual wheels for low-level control can be computed analytically from the desired twist. We obtain the twist from two subsequent base configurations as $\boldsymbol{\nu}_B^* = \mathbf{x}_{base,t+1} \boxminus \mathbf{x}_{base,t}$. As our manipulator can be controlled directly in position and velocity control mode, we enforce safe velocity and acceleration limits at the driver level before passing the commands to the motor drivers. This adds a level of safety in case a user-provided controller generates unsafe commands.

7.2.4 *State estimation*

The state estimation module is based on sensor fusion of the wheel odometry, the onboard Inertial Measurement Unit (IMU), and exteroceptive sensors (e.g., visual odometry or Simultaneous Localisation and Mapping (SLAM) from monocular, stereo vision, or RGB-D sensors, laser localisation, and Global Positioning System

(GPS)). We use the Unscented Kalman Filter (UKF) with the sensing modalities configured in two stages: local odometry frame and global frame. The odometry frame is a smooth signal updated at a high frequency and with high accuracy over short periods. However, this state estimate accumulates error over time (drift). The global frame stays consistent over long periods, but its updates are less frequent, often more costly, they are less accurate, and the updates to the global frame state may not be smooth over time.

We use the odometry frame estimate for control due to its smoothness and local consistency. We then use the global frame estimate for planning and for slow corrections of trajectories over time.

7.2.5 *Whole-body loco-manipulation planning*

In order to achieve fast motion synthesis for longer horizon planning which includes locomotion and manipulation in the presence of moving targets and obstacles, we formulate a trajectory optimisation problem in time-configuration space. Each timestep preserves the same formulation and expressiveness in cost and constraints as described in Section 7.2.3 for the one timestep look-ahead control, with further constraints introduced for dynamic consistency and smooth transition between states. This follows the method introduced in Chapter 3 However, due to local minima present in non-linear optimisation problems, solvers are not guaranteed to converge to a valid solution in a given time budget—or at all—unless provided with a suitable initialisation seed. This is especially the case when considering collision avoidance in complex, unknown environments. In known environments, suitable warm-start solutions can be encoded in a trajectory library (see Chapter 4). In order to operate in unseen environments, we employ fast, global sampling-based planners to provide a feasible initialisation to the trajectory optimisation. Random sampling-based planners, however, are not suited to satisfy general constraints.[2] As thus, we use constraint relaxation as well as a framework to solve constrained time-configuration space problems by decomposition [Yang et al., 2018b] for initialisation. We formulate both the real-time control optimisation as well as the non-linear loco-manipulation problem using the Extensible Optimisation Toolset (EXOTica) [Ivan et al., 2019].

2 Readers are referred to [Kingston et al., 2018] for a review of approaches for sampling-based planning in the presence of constraints.

7.3 EVALUATION

We use the Adabotics Ada500 omnidirectional mobile platform in all our experiments. This platform features a holonomic base, a 6-DoF manipulator with a Robotiq 3-finger gripper. It is equipped with an IMU, an RGB-D camera, two planar laser scanners, and two onboard computers with Intel i7 CPUs.

7.3.1 *Whole-body control evaluation on chicken-head task*

We evaluate the performance of the implemented whole-body control scheme by maintaining an operational space target for the end-effector while commanding the desired target for the base controller (commonly referred to as the *chicken-head problem*). In a laboratory setting, we increase the velocity of the base command (to track a circle on the ground) while tracking ground truth using a VICON camera system. We have formulated the tracking problem as unconstrained minimisation of the end-effector position in the global frame over the base position and the arm configuration and used the Levenberg-Marquardt [Moré, 1978] algorithm to solve this problem. Note, this is a relaxation of Equations (47)–(50) as the manipulator may pass through singular configurations resulting in a violation of real-time requirements. The results are shown in Figure 75 validating the relaxation to be suitable, and snapshots of an applicable real-world task experiment depicted in Figure 77. We evaluate the end-effector error against ground truth from a VICON motion capture system in Figure 76. This task is straightforward but demonstrates that our formulation enables us to track moving targets in arbitrary frames of reference, which opens our framework to a multitude of practical applications.

7.3.2 *Automotive assembly fitting simulation*

A frequent task for the deployment of collaborative robots is the fitting of insulation, adhesives, and sub-assemblies in automotive manufacturing (e.g., sealants on doors). These tasks are correlated with a high risk of Repetitive Strain Injury (RSI).

In this scenario, a mobile collaborative robot carries out manipulation tasks on a moving assembly part by coordinating whole-body motion. We formulate a whole-body constrained non-linear optimisation problem to minimise control effort in the presence of moving targets and solve it using the commercial solver SNOPT [Gill et al., 2002]. In particular, the manipulation motions (e.g., drilling and fitting

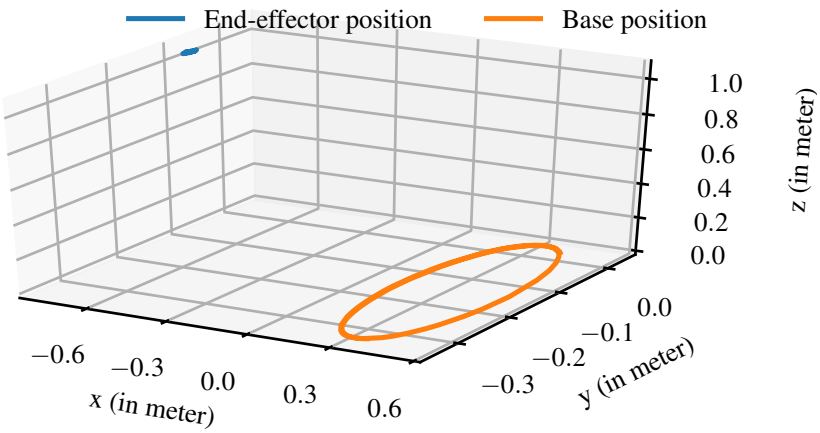


Figure 75: Visualisation of the internal robot state and ground truth for both the end-effector and base frames while carrying out the chicken-head task: This experiment demonstrates the ability of the whole-body control scheme to decouple the end-effector from the base motion and coordinate both at the same time.

trajectories) are encoded as semi-constrained end-effector paths (3-DoF position, 2-DoF axis alignment) with further constraints on continuous collision avoidance using the approach detailed in Chapter 3. Note, we use a constrained NLP solver here as the reference motion is planned using trajectory optimisation for a long horizon and then executed using our whole-body control scheme.

We have used the whole-body controller in the trajectory mode for executing the motion. This is sufficient in simulation (see Figure 78). However, a real-world deployment requires active sensing, tracking of the assembly, and other steps correcting the synchronisation of the robot motion with the environment. We address these issues in our next experiment.

7.3.3 *Sensor placement*

Off-shore assets such as oil and gas platforms are structures designed to operate for decades in harsh environments. Seawater, wind, and temperature changes cause material deterioration and failure that can be prevented by regular integrity monitoring and maintenance. Our industrial partners[3] have identified the need for automating these tasks. They are currently executed by humans which is both costly and potentially dangerous for the workers. A large amount of monitoring can be performed remotely, as long as the monitored structure can be equipped with

3 Through the UKRI Robotics and AI *Offshore Robotics for Certification of Assets* Hub, we engage with a variety of industrial partners, see https://www.orcahub.org.

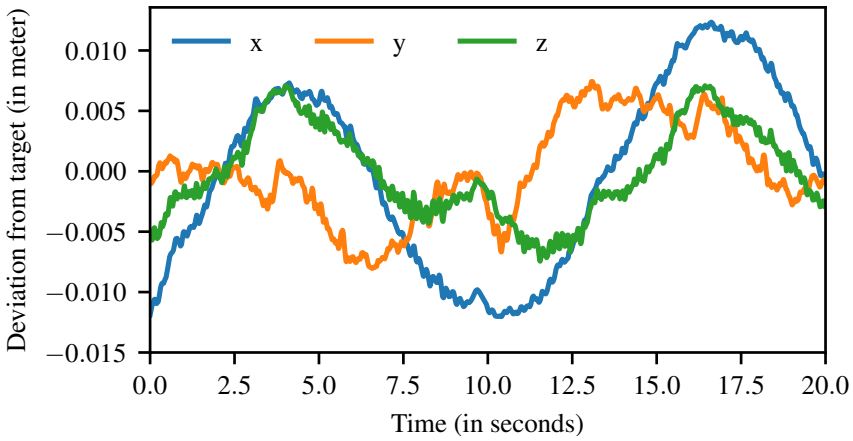


Figure 76: Visualisation of the task-space error of the end-effector while the base is following a circular motion using solely odometry (no sensor fusion with IMU): both x and y drift, where the error in z is due to stiff suspension on an uneven floor.

sensors. The Limpet sensor [Sayed et al., 2018] was designed for these applications in harsh environments, limited power, remote operation with long-distance communication, and real-time monitoring capability. We have applied our whole-body control framework to place these sensors semi-autonomously using the framework described in Chapter 6. The user remotely specifies the sensor location while the planning and control framework ensures accurate placement of the sensor. This process requires minimal data bandwidth, and it is, therefore, suitable for applications where teleoperation is not possible due to communication quality.

In our experiment, we placed a container with the Limpet sensors on top of the robot. The user then specifies the target location. For repeatability and visual confirmation, we mark and track the target locations with AprilTag fiducial markers [Wang and Olson, 2016]. However, the target detection and tracking can be performed using any combination of visual and depth features such as in Pauwels and Kragic [2015]. The execution then used a finite state machine to switch between sensor pick-up, sensor placement using visual servoing, and arm parking motion. The sensor placement was triggered by the sensing module detecting the target marker.

In the first phase, we have constructed a motion planning problem for computing a collision-free pick-up trajectory for the robot arm using the RRT-Connect sampling-based motion planner [Kuffner and LaValle, 2000]. The trajectory was executed using our controller in the trajectory operation mode. Once the target was detected, the tracker provides updates of the target position. These were fed into the controller in the streaming operation mode. The controller solves the inverse

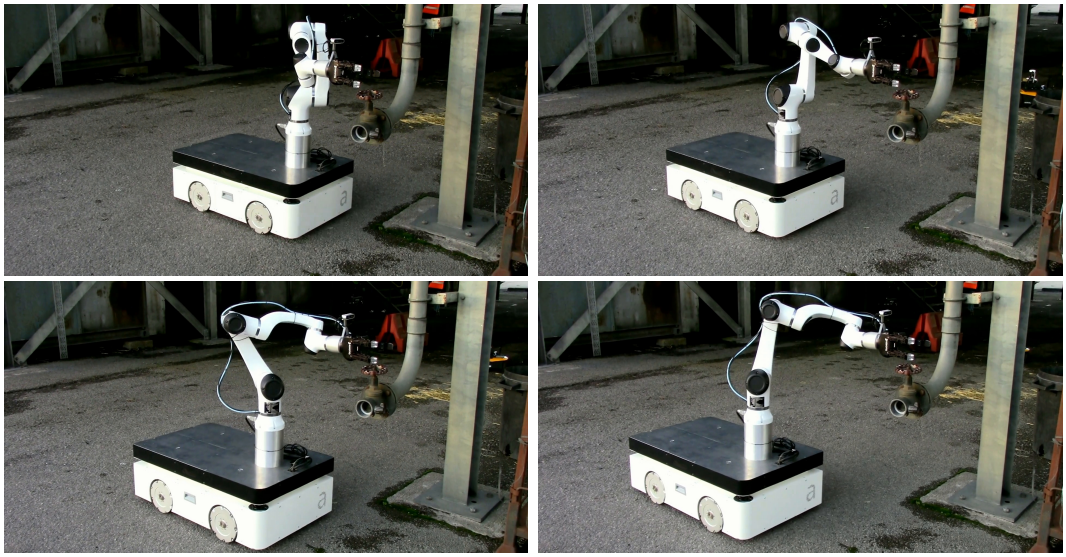


Figure 77: The chicken-head task on a rough outdoor surface carrying out a proposed scenario: manipulation of a static end-effector affordance while responding to disturbance with the redundancy of the omnidirectional base following a high velocity figure-eight target.

kinematics formulated as an unconstrained NLP problem (see Section 7.2.3) using the Levenberg-Marquardt algorithm [Moré, 1978]. The solution was then used to command the arm position in real-time to compensate for the relative target motion. We have also superimposed a short place, hold, and release trajectory over the target position. Using such a placement trajectory ensures that the gripper has enough time to open. The parking motion is then planned using RRT-Connect and executed in the trajectory operation mode, the same as the sensor pick-up motion in the first phase.

This experiment was executed both in a laboratory environment (see Figure 79) and in an outdoor mock oil rig designed for firefighter training (see Figure 72). The task can be easily adapted for similar scenarios by modifying the state machine or changing any of the sub-problems to fit the needs. The advantage of using the whole-body controller in this scenario is that the framework can handle all the different operation modes. This flexibility enabled us to execute the whole task continuously without stopping the base motion. This continuous manipulation is made possible due to the inherent synchronisation of the base and arm movement.

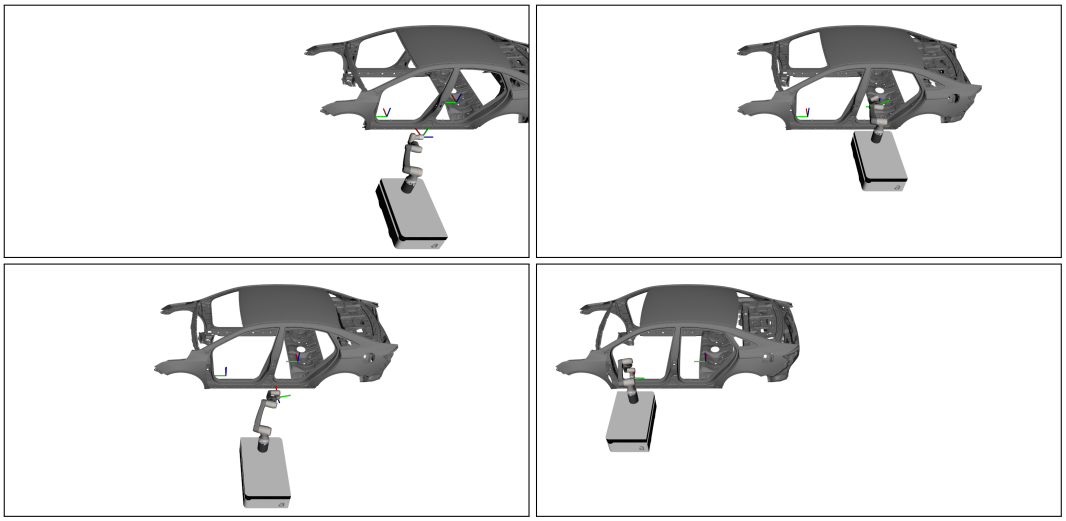


Figure 78: Assembly tasks on a car body structure using a mobile manipulator: While the assembly line is moving at $0.5\,\mathrm{m\,s^{-1}}$, the manipulator carries out two collision-free manipulation actions for 4 s each while following the moving target.

7.4 DISCUSSION

We have presented an architecture for whole-body control and planning of collaborative mobile manipulators. This system exploits a generic formulation of the task as a constrained NLP, and it integrates inputs from state estimation, perception, and the user to generate complex collision-free motion plans. The control architecture then minimises the tracking error while satisfying the task-specific constraints. The formulation of the problem allows us to formulate a wide variety of motion planning tasks and match them with customised controllers.

Our evaluation using the chicken-head task validates the architecture. The tracking results then show the overall performance of the system on our hardware platform. The results demonstrate the performance of the controller and of the platform itself in a controlled environment as well as in an outdoor trial. Using the controller implementation in EXOTica, we achieve a 100 Hz control rate on an Intel i7-7567U CPU with peak performance at 500 Hz. The bottleneck of the controller is state estimation. The accuracy of the end-effector tracking depends mainly on the quality of the state estimate that is used for closing the control loop. Drift, delays, and position error all contribute to this issue. Delays can be computed and accounted for, e.g., using Model-Predictive Control (MPC), as in work by Koenemann et al. [2015]. Drift can be eliminated by exploiting exteroceptive sensors and computing the global reference as described in Section 7.2.4.

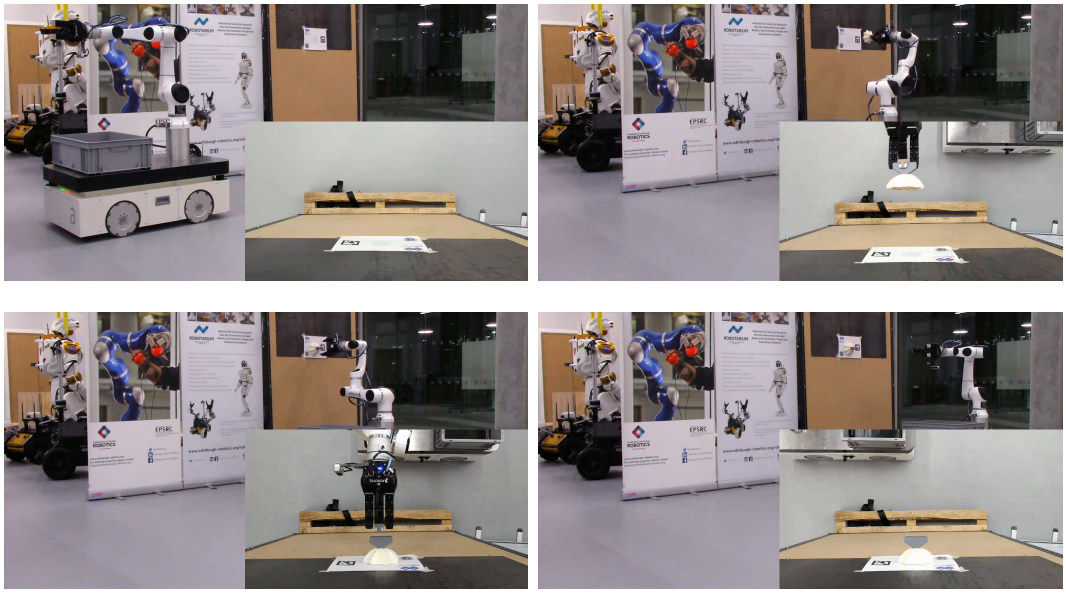


Figure 79: Sensor deployment trials in a laboratory: The robot proceeds to the next placement task in the stack and after acquiring the target carries out a whole-body visual servoing task without stopping the navigation/base motion. The accuracy of the sensor placement task with respect to the target can be seen from the over-head camera.

We have also used the generic problem formulation for solving a motion planning problem in the automotive industry. The versatility of this formulation allowed us to do trajectory optimisation with collision avoidance. This is a notoriously tricky problem due to the non-linearity of the collision constraint. We have exploited the state-of-the-art collision term formulation presented in Chapter 3. The whole-body paradigm then allowed us to use a relatively small robot and extend its range without sacrificing the optimality of the task. This experiment did not consider any control errors nor any control delays since a simulator was used.

In our last experiment, we deployed our system on the real robot to perform a non-stop pick-and-place task. While we used a visual marker to track the target location, the perception method can be easily swapped for a more advanced technique that does not require any fiducial information. We have also relied on the soft housing of the sensor when making contact during the placement. If the sensor did not provide a soft interface between the robot and the solid wall structure, we would employ compliant control using a force-torque sensor at the end-effector, e.g. as described by Moura et al. [2018].

Each industrial application requires a specific set of sensing, planning, and control solutions. The architecture we proposed opens possibilities for designing

these techniques using well-defined building blocks. Such an approach can rapidly accelerate the development and deployment of robotic systems in automotive manufacturing, off-shore asset maintenance, and many other fields. Beyond applications in industry, it is a key factor in translating novel formulations to be tested on real robot hardware.

Part V

FINAL REMARKS

8

CONCLUSIONS

In this thesis, we explored methods for bootstrapping trajectory optimisation from prior experience to make interactive real-world deployment in complex and shared environments practical. To this end, we first explored optimisation-based approaches for collision-free goal state and motion planning in Part II. Here, we compared several general non-linear optimisation-based formulations for discrete collision avoidance (Chapter 2). We then introduced a novel penalty for continuous-time collision avoidance in discrete-time trajectory optimisation based on harmonic potential fields and conservative advancement (Chapter 3).

In the second part, we focused on using offline computation to warm-start optimisation algorithms during run-time. First, we followed a trajectory library approach focusing on high-dimensional problems in complex environments. In particular, we applied a problem encoding capturing the relation between the robot and the environment. We then introduced an efficient indexing scheme which, together with an exploration strategy, allows the application of fast nearest neighbour look-up (Chapter 4). However, trajectory library approaches neither compress the original dataset nor generalise beyond it—aims that can be achieved using machine learning methods. Whether these approaches can be successfully applied and provide generalisation is influenced by discontinuity and multi-modality in the input data. In order to address this, we applied tools from computational topology to extract information on the underlying structure of the trajectory samples. We show that approximating within each homology-invariant class increases the success of warm-starting and further permits the exploration of multiple solutions simultaneously in an ensemble (Chapter 5).

In the third part, we focus on hardware embodiment and deployment in real-world scenarios. We first describe a full system for mobile manipulation and extend *shared autonomy*, an operating paradigm allowing the sliding control between teleoperation and full autonomy, with continuous scene monitoring to detect, track, and assess changes (Chapter 6). Subsequently, inspired by humanoid whole-body control, we focus on coordinated applications that consider locomotion and manipulation jointly and evaluate it on industrial sensing tasks (Chapter 7). Here,

our synchronised control scheme enabled the seamless transfer of algorithms from visualisation and simulation to real-world deployment.

Throughout this thesis, we focused on an integrated approach. In this purview, we tested our methods on a range of settings in simulation and experiments in the laboratory as well as in field applications: From low-dimensional systems with a focus on dynamics (e.g., cart-pole and quadrotor) to high-dimensional kinematic motion planning on humanoid robots with a focus on complex, changing, and shared environments. We selected these systems to explore both the fundamental concepts of established as well as newly proposed methods and to show the versatility and scalability of our approaches. In particular, we focused our experiment design on clearly demonstrating a particular challenge in an application and how our methods can address them. Furthermore, our experiments highlight the flexibility of our developed framework allowing a glimpse of its straight-forward reconfigurability between different problem formulations, cost and constraint combinations, algorithm implementations, and solver choices. This versatility was particularly influential in comprehensive comparison studies where we can easily control the confounding variables, and we hope it will be of use to the robotics community at large.

9

FUTURE DIRECTIONS

During the research conducted for this thesis, a range of interesting areas for further work became evident. We detail several of these below.

9.1 TRANSFERABLE ENVIRONMENT REPRESENTATION

While we focused on the structure of solution spaces in complex environments, we did not consider generalisation to new environments. A key challenge for this is the question of a transferable representation space of environments. In search-based methods, Dynamic Roadmaps (DRMs) store work-space occupancy information for vertices and edges and can quickly filter and reconnect the graph based on sensed obstacle occupancy information. These are often high-dimensional and thus prohibitive for large numbers of samples due to memory exhaustion. Yang et al. [2018a] proposed a hierarchical version which does not explicitly store vertex or edge information, yet, is unable to scale to problems involving (kino-)dynamics.

One possible avenue is to learn an environment descriptor representation from a large number of samples. In seminal work, Jetchev and Toussaint [2013] applied sparse feature selection through Principal Component Analysis (PCA) to compress a voxel grid-based environment occupancy representation.

Learnt trajectories (trajectory prediction) do not contain connectivity information as in a connected graph. As a hybrid, Mansard et al. [2018] combine a kino-dynamic Probabilistic Roadmap (PRM) with a learnt trajectory prediction during an iterative exploration scheme in obstacle-free environments. Their iterative training process converges when the memory performs as well as or better than the PRM. Intending to compress trajectories and directly use the resulting policy network as an online predictive controller, they focus on the learnt memory exclusively. A combination of their algorithm with a DRM and search methods could be a viable path for an environment-adaptive memory.

Another avenue is to learn either a general or task-specific latent space description of the environment. To generalise for similar task across a semantic environment class, e.g., a shelf or kitchen environment, one can make use of large-scale labelled

datasets of 3D objects. These have been applied to latent modelling of shapes, e.g., with Chang et al. [2015] introducing the popular ShapeNet dataset. Similar datasets exist and are widely used for grasp planning.

More recently, several authors have looked into combining learnt latent representations of the environment with Sampling-based Planning (SBP) to leverage the best of both worlds. Ichter et al. [2018] embedded learnt latent space information conditioned on obstacles, start, and goal states to bias exploration, with Ichter and Pavone [2019] extending the concept to a fully learnt motion planning framework. Similarly, Qureshi et al. [2019] introduced a recursive neural network planner which embeds a latent space from point-cloud information.

Finally, instead of directly extracting a latent space from environment information (e.g., voxel grids), the planning and prediction problem could be shifted from the commonly used task-space, configuration-space, or control-space to an alternate planning space. Here, topologically-inspired spaces which allow for the incorporation of relation and distance information such as interaction mesh [Ivan et al., 2013] and distance mesh [Yang et al., 2015] could be applied. However, these still require expert-guided specification of points of interest as well as tuning of weight matrices.

9.2 APPLICATION TO SCENARIOS WITH CONTACT OR DYNAMICS CHANGES

In this thesis, we did not consider scenarios which involved contact or dynamics changes, for instance, from interactions with the environment, during trajectory prediction. These applications are core to the European Union Horizon 2020 project Memory of Motion (MEMMO) which aims to solve locomotion in complex terrain with optimal control motion warm-started from a memory-of-motion. In practice, these scenarios often rely on decomposed pipeline approaches as we have discussed in Chapter 1. A key challenge in bootstrapping these algorithms is the formulation of the whole-body motion optimisation. Problem formulations that rely on hard constraints (e.g., for rigid contacts or target frames) often suffer if the prediction is infeasible (i.e., if constraints are violated). In these cases, trajectory library approaches feature a high success rate because constraints are satisfied even when the solution is far from the optimum. In combination with a recursive search, an indexing scheme similar to the one presented in this thesis may be a viable pathway. Similarly, ongoing work aims to identify the level at which prediction can occur, and if it requires recurrent approaches for generalisation.

9.3 CONSTRAINT-AWARE LEARNING

When considering general equality and inequality tasks as hard constraints, the feasibility of a prediction has a large influence on the success of the initial guess. Frequently, either a reconstruction loss from similarity to the original motion (e.g., Mean Squared Error (MSE) or Mean Absolute Error (MAE)) or a loss based on the initial cost of the prediction is used during learning (thus, requiring a roll-out). However, as a potential solution, a loss function using a roll-out of the prediction with the sum of constraint violations can be used for constraint-aware learning. Such an approach could, however, prove to be prohibitively slow as roll-outs are several orders of magnitude more expensive than a simple metric such as the MSE to the training dataset. Instead of a performing a full roll-out, Carius et al. [2020] recently proposed the use of the control Hamiltonian as the loss function for guided policy search to learn quadrupedal gaits from Model-Predictive Control (MPC).

By using insight into the task for the design of the regressors, Armesto et al. [2018] proposed an efficient constraint-aware policy learning method which first learns the constraint and then learns the null-space policy separately. This approach is efficient and promising as it guarantees feasibility. Work to show its applicability or automatic discovery of suitable features or regressors, possibly learnt, are exciting areas for further investigation.

9.4 MODEL-PREDICTIVE CONTROL ON FORCE CONTROLLED ROBOTS

In this thesis, we primarily focused on replanning and did not consider predictive control deployment. Extending the presented approaches in a MPC framework for deployment is an obvious direction of future work. Achieving a real-time MPC system largely depends on more efficient formulations (e.g., using spline- or kernel-based embeddings and parametrisations to reduce the optimisation problem size), approximations, and performance-guided implementation of algorithms. Grandia et al. [2019] recently demonstrated the use of the computed feedback gains from a Differential Dynamic Programming (DDP) policy—including with respect to desired contact forces—in hardware experiments using forward simulation to extract desired accelerations and inverse dynamics for torque computation. However, directly applying obtained feed-forward torques along with the feedback gains on legged systems is an open challenge.

Part VI

APPENDIX

A

COLLISION-FREE GOAL STATE PLANNING: BENCHMARK RESULTS

	Solver	% success	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{B}_A$	IPOPT	48.42%	100.0%	0.0%	0.0%	0.04 ± 0.23	1082.86 ± 1163.14	488.05 ± 418.95
$\mathcal{B}_A$	IPOPT-RR	70.9%	100.0%	0.0%	0.0%	0.05 ± 0.27	2118.14 ± 3403.83	553.77 ± 462.01
$\mathcal{B}_A$	KNITRO	34.71%	100.0%	0.0%	0.0%	0.02 ± 0.17	99.44 ± 61.72	59.79 ± 25.75
$\mathcal{B}_A$	KNITRO-RR	53.87%	100.0%	0.0%	0.0%	0.04 ± 0.25	278.57 ± 464.21	63.34 ± 27.75
$\mathcal{B}_A$	SNOPT	31.96%	100.0%	0.0%	0.0%	0.04 ± 0.22	89.71 ± 47.15	61.15 ± 28.55
$\mathcal{B}_A$	SNOPT-RR	52.14%	100.0%	0.0%	0.0%	0.04 ± 0.23	211.07 ± 206.28	135.47 ± 128.68
$\mathcal{B}_B$	IPOPT	50.71%	76.73%	23.27%	0.0%	0.03 ± 0.17	3344.77 ± 3391.93	351.43 ± 248.27
$\mathcal{B}_B$	IPOPT-RR	77.47%	97.06%	2.94%	0.0%	0.03 ± 0.19	6406.73 ± 9022.26	359.11 ± 257.03
$\mathcal{B}_B$	KNITRO	41.79%	84.5%	15.5%	0.0%	0.01 ± 0.13	420.95 ± 191.3	50.26 ± 17.38
$\mathcal{B}_B$	KNITRO-RR	70.64%	98.26%	1.74%	0.0%	0.02 ± 0.15	905.92 ± 886.63	52.09 ± 17.96
$\mathcal{B}_B$	SNOPT	54.38%	72.07%	27.93%	0.0%	0.03 ± 0.2	395.89 ± 132.38	44.62 ± 15.14
$\mathcal{B}_B$	SNOPT-RR	79.97%	97.2%	2.8%	0.0%	0.03 ± 0.2	719.02 ± 675.05	73.79 ± 63.71
$\mathcal{B}_C$	IPOPT	51.33%	78.85%	21.15%	0.0%	0.03 ± 0.21	165.57 ± 150.01	365.2 ± 278.56

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{B}_C$	IPOPT-RR	77.06%	98.44%	1.56%	0.0%	0.04 ± 0.24	280.73 ± 313.59	371.19 ± 293.55
$\mathcal{B}_C$	KNITRO	41.59%	85.95%	14.05%	0.0%	0.02 ± 0.14	18.12 ± 12.17	51.11 ± 21.09
$\mathcal{B}_C$	KNITRO-RR	67.89%	98.89%	1.11%	0.0%	0.03 ± 0.18	42.09 ± 39.6	53.68 ± 25.09
$\mathcal{B}_C$	SNOPT	53.67%	76.24%	23.76%	0.0%	0.03 ± 0.22	17.17 ± 6.49	45.09 ± 16.21
$\mathcal{B}_C$	SNOPT-RR	77.37%	98.2%	1.8%	0.0%	0.04 ± 0.23	28.75 ± 24.6	71.11 ± 56.86
$\mathcal{B}_D$	IPOPT	50.51%	76.52%	23.48%	0.0%	0.02 ± 0.15	35.63 ± 18.65	349.61 ± 246.92
$\mathcal{B}_D$	IPOPT-RR	78.03%	96.29%	3.71%	0.0%	0.03 ± 0.18	63.25 ± 60.85	356.58 ± 250.17
$\mathcal{B}_D$	KNITRO	41.9%	84.39%	15.61%	0.0%	0.01 ± 0.13	1.49 ± 0.58	50.31 ± 17.47
$\mathcal{B}_D$	KNITRO-RR	69.98%	97.96%	2.04%	0.0%	0.02 ± 0.16	11.68 ± 10.33	52.08 ± 18.04
$\mathcal{B}_D$	SNOPT	54.38%	71.96%	28.04%	0.0%	0.03 ± 0.2	1.5 ± 0.47	44.65 ± 15.17
$\mathcal{B}_D$	SNOPT-RR	80.68%	98.42%	1.58%	0.0%	0.03 ± 0.21	2.63 ± 2.2	72.65 ± 56.19
$\mathcal{H}_A$	SNOPT-RR	93.7%	94.31%	5.69%	0.0%	0.0 ± 0.0	503.14 ± 773.56	327.13 ± 501.46
$\mathcal{H}_A$	IPOPT	53.94%	93.44%	6.56%	0.0%	0.0 ± 0.0	705.84 ± 426.35	437.85 ± 275.61

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{H}_A$	IPOPT-RR	95.08%	92.71%	7.29%	0.0%	0.0 ± 0.0	2287.98 ± 2604.32	448.34 ± 266.19
$\mathcal{H}_A$	KNITRO	23.21%	99.8%	0.2%	0.0%	0.0 ± 0.0	117.76 ± 67.55	27.03 ± 37.71
$\mathcal{H}_A$	KNITRO-RR	69.21%	99.5%	0.5%	0.0%	0.0 ± 0.0	733.55 ± 589.78	24.34 ± 28.66
$\mathcal{H}_A$	SNOPT	48.1%	93.88%	6.12%	0.0%	0.0 ± 0.0	143.51 ± 249.87	63.81 ± 140.07
$\mathcal{H}_A$	SNOPT-CMAES-R	93.95%	96.61%	3.39%	0.0%	0.0 ± 0.0	777.59 ± 946.0	270.04 ± 468.3
$\mathcal{H}_B$	SNOPT-RR	67.33%	99.84%	0.16%	0.0%	0.0 ± 0.0	1058.37 ± 1190.63	514.74 ± 702.9
$\mathcal{H}_B$	IPOPT	47.6%	90.56%	9.44%	0.0%	0.01 ± 0.12	1212.24 ± 1707.1	432.4 ± 649.01
$\mathcal{H}_B$	IPOPT-RR	93.68%	86.29%	13.71%	0.0%	0.0 ± 0.0	3310.57 ± 4921.37	411.9 ± 523.2
$\mathcal{H}_B$	KNITRO	35.58%	97.15%	2.85%	0.0%	0.0 ± 0.04	136.1 ± 92.14	68.25 ± 57.01
$\mathcal{H}_B$	KNITRO-RR	88.33%	92.86%	7.14%	0.0%	0.0 ± 0.01	573.87 ± 577.37	74.2 ± 52.47
$\mathcal{H}_B$	SNOPT	27.57%	100.0%	0.0%	0.0%	0.0 ± 0.05	269.11 ± 394.32	61.41 ± 111.36
$\mathcal{H}_B$	SNOPT-CMAES-R	87.82%	87.45%	12.55%	0.0%	0.0 ± 0.0	826.11 ± 928.75	234.98 ± 435.64
$\mathcal{H}_C$	SNOPT-RR	14.04%	100.0%	0.0%	0.0%	0.0 ± 0.0	582.13 ± 712.3	324.53 ± 453.4

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{H}_C$	IPOPT	32.84%	99.69%	0.31%	0.0%	0.0 ± 0.0	757.84 ± 277.4	439.21 ± 171.46
$\mathcal{H}_C$	IPOPT-RR	52.61%	100.0%	0.0%	0.0%	0.0 ± 0.0	1642.29 ± 1716.19	455.08 ± 175.47
$\mathcal{H}_C$	KNITRO	46.57%	99.33%	0.67%	0.0%	0.0 ± 0.0	79.09 ± 61.04	46.59 ± 27.18
$\mathcal{H}_C$	KNITRO-RR	70.44%	100.0%	0.0%	0.0%	0.0 ± 0.0	163.34 ± 221.52	47.24 ± 21.48
$\mathcal{H}_C$	SNOPT	7.74%	100.0%	0.0%	0.0%	0.0 ± 0.0	219.98 ± 284.76	89.8 ± 101.2
$\mathcal{H}_C$	SNOPT-CMAES-R	29.82%	100.0%	0.0%	0.0%	0.0 ± 0.0	219.21 ± 249.41	37.83 ± 119.23
$\mathcal{H}_D$	SNOPT-RR	92.51%	81.63%	18.37%	0.0%	0.0 ± 0.0	5.59 ± 6.92	177.07 ± 275.92
$\mathcal{H}_D$	IPOPT	47.96%	81.49%	18.51%	0.0%	0.0 ± 0.0	18.55 ± 11.7	188.38 ± 128.33
$\mathcal{H}_D$	IPOPT-RR	92.25%	76.97%	23.03%	0.0%	0.0 ± 0.0	61.52 ± 70.24	187.38 ± 122.01
$\mathcal{H}_D$	KNITRO	21.15%	93.99%	6.01%	0.0%	0.0 ± 0.0	3.63 ± 1.99	26.2 ± 31.16
$\mathcal{H}_D$	KNITRO-RR	69.22%	95.03%	4.97%	0.0%	0.0 ± 0.0	44.72 ± 341.67	31.59 ± 35.92
$\mathcal{H}_D$	SNOPT	45.62%	83.04%	16.96%	0.0%	0.0 ± 0.0	1.72 ± 2.44	41.39 ± 58.98
$\mathcal{H}_D$	SNOPT-CMAES-R	78.95%	23.97%	76.03%	0.0%	0.0 ± 0.0	345.58 ± 315.43	157.64 ± 278.89

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{U}_A$	AICO	25.43%	92.82%	0.0%	7.18%	0.22 ± 4.04	235.96 ± 156.24	151.95 ± 101.15
$\mathcal{U}_A$	AICO-RR	34.91%	91.23%	0.0%	8.77%	0.13 ± 0.35	336.78 ± 296.68	239.95 ± 201.42
$\mathcal{U}_A$	IPOPT-RR	43.17%	94.26%	0.0%	5.74%	0.02 ± 0.17	3066.55 ± 4242.76	450.59 ± 244.93
$\mathcal{U}_A$	IPOPT	29.36%	96.39%	0.0%	3.61%	0.02 ± 0.17	1343.44 ± 1631.27	447.98 ± 243.22
$\mathcal{U}_A$	KNITRO-RR	36.19%	97.28%	0.0%	2.72%	0.02 ± 0.15	281.72 ± 250.3	90.16 ± 33.65
$\mathcal{U}_A$	KNITRO	24.21%	98.32%	0.0%	1.68%	0.01 ± 0.09	124.78 ± 54.34	86.94 ± 31.15
$\mathcal{U}_A$	LM	19.52%	83.09%	1.39%	15.52%	0.26 ± 5.1	114.38 ± 2.93	103.0 ± 0.0
$\mathcal{U}_A$	LM-RR	40.01%	73.75%	0.42%	25.83%	0.0 ± 0.07	365.93 ± 331.79	247.2 ± 205.03
$\mathcal{U}_A$	GN	19.88%	81.36%	2.1%	16.54%	0.26 ± 5.06	116.86 ± 8.93	103.0 ± 0.0
$\mathcal{U}_A$	TNewton	32.36%	97.74%	0.0%	2.26%	0.01 ± 0.12	600.91 ± 699.38	297.04 ± 465.52
$\mathcal{U}_A$	PInv-RR	47.86%	58.55%	0.29%	41.15%	0.01 ± 0.11	306.36 ± 314.37	191.08 ± 201.85
$\mathcal{U}_A$	PInv	25.08%	71.63%	2.93%	25.44%	0.22 ± 4.51	103.8 ± 32.45	58.16 ± 36.83
$\mathcal{U}_A$	SNOPT-RR	39.35%	95.38%	0.0%	4.62%	0.01 ± 0.13	217.73 ± 203.75	135.48 ± 134.12

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{U}_A$	SNOPT	24.41%	97.17%	0.0%	2.83%	0.01 ± 0.11	97.99 ± 50.41	62.92 ± 28.04
$\mathcal{U}_B$	AICO	38.28%	70.27%	14.86%	14.86%	0.04 ± 0.2	1579.1 ± 1024.33	158.49 ± 100.22
$\mathcal{U}_B$	AICO-RR	52.09%	79.57%	0.96%	19.47%	0.12 ± 0.33	2439.68 ± 2284.37	216.96 ± 166.97
$\mathcal{U}_B$	IPOPT-RR	55.66%	88.97%	1.26%	9.77%	0.01 ± 0.14	10057.58 ± 11558.34	415.16 ± 207.88
$\mathcal{U}_B$	IPOPT	29.46%	86.27%	9.39%	4.34%	0.01 ± 0.13	4723.09 ± 3513.58	413.93 ± 210.28
$\mathcal{U}_B$	KNITRO-RR	58.1%	92.34%	0.73%	6.93%	0.0 ± 0.06	1549.87 ± 1461.14	76.89 ± 28.87
$\mathcal{U}_B$	KNITRO	32.98%	85.78%	10.42%	3.8%	0.0 ± 0.03	742.19 ± 255.76	74.14 ± 29.24
$\mathcal{U}_B$	LM	24.46%	68.89%	6.88%	24.22%	0.0 ± 0.05	917.64 ± 17.04	103.0 ± 0.0
$\mathcal{U}_B$	LM-RR	41.44%	69.54%	0.17%	30.29%	0.0 ± 0.06	2144.93 ± 1845.75	198.66 ± 157.69
$\mathcal{U}_B$	GN	24.46%	68.83%	6.88%	24.29%	0.0 ± 0.05	950.57 ± 18.97	103.0 ± 0.0
$\mathcal{U}_B$	TNewton	48.93%	75.35%	20.16%	4.49%	0.0 ± 0.08	1536.66 ± 2942.25	179.34 ± 318.29
$\mathcal{U}_B$	PInv-RR	49.75%	52.84%	0.3%	46.86%	0.01 ± 0.13	1689.31 ± 1636.88	138.77 ± 143.26
$\mathcal{U}_B$	PInv	30.73%	54.75%	9.71%	35.54%	0.01 ± 0.13	824.04 ± 272.5	59.64 ± 36.65

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{U}_B$	SNOPT-RR	70.08%	87.05%	0.85%	12.1%	0.0 ± 0.07	837.83 ± 815.11	86.8 ± 71.03
$\mathcal{U}_B$	SNOPT	47.25%	74.69%	18.74%	6.57%	0.0 ± 0.08	489.34 ± 183.79	52.63 ± 19.72
$\mathcal{U}_C$	AICO	38.12%	73.15%	12.36%	14.5%	0.17 ± 3.66	58.65 ± 39.63	158.34 ± 100.29
$\mathcal{U}_C$	AICO-RR	49.18%	81.54%	0.0%	18.46%	0.12 ± 0.31	86.03 ± 73.54	207.35 ± 176.51
$\mathcal{U}_C$	IPOPT-RR	54.49%	89.47%	0.78%	9.74%	0.01 ± 0.11	402.58 ± 371.27	426.55 ± 236.17
$\mathcal{U}_C$	IPOPT	29.82%	87.87%	7.84%	4.28%	0.02 ± 0.13	200.63 ± 139.75	418.52 ± 225.0
$\mathcal{U}_C$	KNITRO-RR	56.98%	91.59%	0.47%	7.94%	0.01 ± 0.11	67.86 ± 58.93	78.23 ± 30.15
$\mathcal{U}_C$	KNITRO	32.87%	87.17%	9.04%	3.8%	0.01 ± 0.08	30.85 ± 15.21	74.75 ± 29.28
$\mathcal{U}_C$	LM	24.46%	68.89%	6.88%	24.22%	0.0 ± 0.05	36.48 ± 2.09	103.0 ± 0.0
$\mathcal{U}_C$	LM-RR	40.62%	67.12%	0.52%	32.36%	0.0 ± 0.04	79.67 ± 67.1	190.05 ± 149.26
$\mathcal{U}_C$	GN	24.46%	68.83%	6.88%	24.29%	0.0 ± 0.05	37.06 ± 2.53	103.0 ± 0.0
$\mathcal{U}_C$	TNewton	48.52%	77.23%	18.22%	4.55%	0.0 ± 0.08	65.19 ± 102.57	177.24 ± 320.18
$\mathcal{U}_C$	PInv-RR	49.29%	55.88%	0.7%	43.42%	0.01 ± 0.14	64.78 ± 62.95	139.15 ± 144.83

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{U}_C$	PInv	30.73%	54.75%	9.71%	35.54%	0.01 ± 0.13	31.83 ± 10.64	59.64 ± 36.65
$\mathcal{U}_C$	SNOPT-RR	66.82%	86.79%	1.23%	11.98%	0.01 ± 0.13	33.83 ± 26.77	80.91 ± 57.88
$\mathcal{U}_C$	SNOPT	46.69%	78.59%	14.91%	6.5%	0.01 ± 0.09	21.21 ± 8.38	53.21 ± 20.44
$\mathcal{U}_D$	AICO	38.28%	70.27%	14.86%	14.86%	0.04 ± 0.2	3.1 ± 2.01	158.49 ± 100.22
$\mathcal{U}_D$	AICO-RR	52.55%	80.02%	0.21%	19.76%	0.11 ± 0.3	5.12 ± 4.72	224.07 ± 201.6
$\mathcal{U}_D$	IPOPT-RR	54.94%	89.82%	0.79%	9.39%	0.01 ± 0.09	73.81 ± 64.06	416.2 ± 208.72
$\mathcal{U}_D$	IPOPT	29.66%	86.67%	9.42%	3.91%	0.01 ± 0.11	34.8 ± 14.51	417.63 ± 214.24
$\mathcal{U}_D$	KNITRO-RR	58.66%	91.25%	1.11%	7.64%	0.0 ± 0.03	13.8 ± 12.19	76.94 ± 28.08
$\mathcal{U}_D$	KNITRO	32.98%	85.93%	10.34%	3.73%	0.0 ± 0.01	1.87 ± 0.61	73.92 ± 28.36
$\mathcal{U}_D$	LM	24.46%	68.89%	6.88%	24.22%	0.0 ± 0.05	1.89 ± 0.03	103.0 ± 0.0
$\mathcal{U}_D$	LM-RR	41.69%	69.32%	0.09%	30.59%	0.0 ± 0.04	4.38 ± 3.8	204.26 ± 169.73
$\mathcal{U}_D$	GN	24.46%	68.83%	6.88%	24.29%	0.0 ± 0.05	1.91 ± 0.13	103.0 ± 0.0
$\mathcal{U}_D$	TNewton	48.93%	75.55%	19.96%	4.49%	0.0 ± 0.08	2.75 ± 4.85	187.24 ± 406.57

	Solver	% rate	Failure type			Final cost	Planning time	Problem updates
			Divergence	Collision	Joint limits			
$\mathcal{U}_D$	PInv-RR	48.62%	55.75%	0.2%	44.05%	0.01 ± 0.11	3.2 ± 3.02	125.0 ± 127.78
$\mathcal{U}_D$	PInv	30.73%	54.75%	9.71%	35.54%	0.01 ± 0.13	1.58 ± 0.6	59.64 ± 36.65
$\mathcal{U}_D$	SNOPT-RR	69.83%	84.97%	1.18%	13.85%	0.0 ± 0.07	3.22 ± 2.79	84.43 ± 67.67
$\mathcal{U}_D$	SNOPT	47.2%	74.61%	18.92%	6.47%	0.0 ± 0.08	1.82 ± 0.62	52.61 ± 19.64

Table 6: Goal state planning benchmark on Kuka LWR comparing different formulations and solvers. These are the full results and accompany the figures in Chapter 2.

BIBLIOGRAPHY

Shailen Agrawal and Michiel van de Panne. Pareto optimal control for natural and supernatural motions. In *Proceedings of Motion on Games*, pages 29–38. ACM, 2013. doi: 10.1145/2522628.2522902. (Cited on page 35.)

Leopoldo Armesto, João Moura, Vladimir Ivan, Mustafa Suphi Erden, Antonio Sala, and Sethu Vijayakumar. Constraint-aware learning of policies by demonstration. *The International Journal of Robotics Research*, 37(13-14):1673–1689, 2018. doi: 10.1177/0278364918784354. (Cited on page 159.)

Christopher G. Atkeson and Jun Morimoto. Nonparametric Representation of Policies and Value Functions: A Trajectory-Based Approach. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 1643–1650. MIT Press, 2003. (Cited on page 76.)

Ulrich Bauer. Ripser: a lean C++ code for the computation of Vietoris–Rips persistence barcodes. `https://github.com/Ripser/ripser`, 2017. (Cited on page 102.)

John T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. Society for Industrial and Applied Mathematics, second edition, 2010. doi: 10.1137/1.9780898718577. (Cited on page 6.)

Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. Robot Programming by Demonstration. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, pages 1371–1394. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_60. (Cited on page 11.)

Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. (Cited on page 94.)

Michael Bloesch, Hannes Sommer, Tristan Laidlow, Michael Burri, Gabriel Nuetzi, Péter Fankhauser, Dario Bellicoso, Christian Gehring, Stefan Leutenegger, Marco Hutter, and Roland Siegwart. A Primer on the Differential Calculus of 3D Orientations, 2016. (Cited on page 24.)

Anastasia Bolotnikova, Kévin Chappellet, Antonio Paolillo, Adrien Escande, Gholamreza Anbarjafari, Adolfo Suarez-Roos, Patrice Rabate, and Abderrah-

mane Kheddar. A circuit-breaker use-case operated by a humanoid in aircraft manufacturing. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 15–22, Aug 2017. doi: 10.1109/COASE.2017.8256069. (Cited on page 143.)

Nathan Britton, Kazuya Yoshida, John Walker, Keiji Nagatani, Graeme Taylor, and Loïc Dauphin. Lunar Micro Rover Design for Exploration through Virtual Reality Tele-operation. In Luis Mejias, Peter Corke, and Jonathan Roberts, editors, *Field and Service Robotics: Results of the 9th International Conference*, pages 259–272. Springer International Publishing, Cham, 2015. ISBN 978-3-319-07488-7. doi: 10.1007/978-3-319-07488-7_18. (Cited on page 119.)

Richard H. Byrd, Jorge Nocedal, and Richard A. Waltz. Knitro: An Integrated Package for Nonlinear Optimization. In G. Di Pillo and M. Roma, editors, *Large-Scale Nonlinear Optimization*, pages 35–59. Springer US, Boston, MA, 2006. ISBN 978-0-387-30065-8. doi: 10.1007/0-387-30065-1_4. (Cited on pages 36 and 59.)

Mylène Campana, Florent Lamiraux, and Jean-Paul Laumond. A gradient-based path optimization method for motion planning. *Advanced Robotics*, 30(17-18): 1126–1144, 2016. doi: 10.1080/01691864.2016.1168317. (Cited on page 68.)

Zhe Cao, Gines Hidalgo Martinez, Tomas Simon, Shih-En Wei, and Yaser A. Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. ISSN 1939-3539. doi: 10.1109/TPAMI.2019.2929257. (Cited on page 11.)

Jan Carius, Martin Wermelinger, Balasubramanian Rajasekaran, Kai Holtmann, and Marco Hutter. Autonomous Mission with a Mobile Manipulator—A Solution to the MBZIRC. In Marco Hutter and Roland Siegwart, editors, *Field and Service Robotics*, pages 559–573, Cham, 2018. Springer International Publishing. ISBN 978-3-319-67361-5. (Cited on page 139.)

Jan Carius, Farbod Farshidian, and Marco Hutter. MPC-Net: A First Principles Guided Policy Search. *IEEE Robotics and Automation Letters*, 5(2):2897–2904, April 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2974653. (Cited on pages 12 and 159.)

Gunnar Carlsson. Topology and data. *Bulletin of the American Mathematical Society*, 46(2):255–308, 2009. doi: 10.1090/S0273-0979-09-01249-X. (Cited on page 95.)

Justin Carpentier, Steve Tonneau, Maximilien Naveau, Olivier Stasse, and Nicolas Mansard. A versatile and efficient pattern generator for generalized legged locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3555–3561, May 2016. doi: 10.1109/ICRA.2016.7487538. (Cited on page 8.)

Justin Carpentier, Andrea Del Prete, Steve Tonneau, Thomas Flayols, Florent Forget, Alexis Mifsud, Kevin Giraud, Dinesh Atchuthan, Pierre Fernbach, Rohan Budhiraja, Mathieu Geisert, Joan Solà, Olivier Stasse, and Nicolas Mansard. Multi-contact Locomotion of Legged Robots in Complex Environments – The Loco3D project. In *RSS Workshop on Challenges in Dynamic Legged Locomotion*, page 3p., Boston, United States, July 2017. (Cited on page 7.)

Nicholas J. Cavanna, Mahmoodreza Jahanseir, and Donald R. Sheehy. A Geometric Perspective on Sparse Filtrations. In *Proceedings of the Canadian Conference on Computational Geometry (CCCG)*, 06 2015. (Cited on page 95.)

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. arXiv preprint arXiv:1512.03012, 2015. (Cited on pages 111 and 158.)

Chao Chen and Michael Kerber. Persistent Homology Computation with a Twist. In *Proceedings of the European Workshop on Computational Geometry*, pages 197–200, 2011. (Cited on page 95.)

Sachin Chitta. MoveIt!: An Introduction. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, pages 3–27. Springer International Publishing, Cham, 2016. ISBN 978-3-319-26054-9. doi: 10.1007/978-3-319-26054-9_1. (Cited on page 139.)

Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, and Enrique Fernández Perdomo. ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, 2017. doi: 10.21105/joss.00456. (Cited on pages 139 and 144.)

Erwin Coumans. Bullet physics engine. Open Source Software, `http://bulletphysics.org`, 2003–2019. (Cited on page 59.)

Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 295–302, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041375. (Cited on page 8.)

Robert Daily and David M. Bevly. Harmonic potential field path planning for high speed vehicles. In *American Control Conference*, pages 4609–4614. IEEE, Jun 2008. doi: 10.1109/ACC.2008.4587222. (Cited on page 57.)

Duong Dang, Florent Lamiraux, and Jean-Paul Laumond. A framework for manipulation and locomotion with realtime footstep replanning. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 676–681, Oct 2011. doi: 10.1109/Humanoids.2011.6100889. (Cited on page 140.)

Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 279–286, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041373. (Cited on page 8.)

Robin Deits and Russ Tedrake. Efficient mixed-integer planning for UAVs in cluttered environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49, May 2015. doi: 10.1109/ICRA.2015.7138978. (Cited on pages 52 and 68.)

Robin Deits, Twan Koolen, and Russ Tedrake. LVIS: Learning from Value Function Intervals for Contact-Aware Robot Controllers. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7762–7768, May 2019. doi: 10.1109/ICRA.2019.8794352. (Cited on page 12.)

Andrea Del Prete. Joint Position and Velocity Bounds in Discrete-Time Acceleration/Torque Control of Robot Manipulators. *IEEE Robotics and Automation Letters*, 3(1):281–288, Jan 2018. ISSN 2377-3766. doi: 10.1109/LRA.2017.2738321. (Cited on page 143.)

Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Prioritized motion–force control of constrained fully-actuated robots: "Task Space Inverse Dynamics". *Robotics and Autonomous Systems*, 63:150 – 157, 2015. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2014.08.016. (Cited on page 8.)

Ron S. Dembo and Trond Steihaug. Truncated-Newton algorithms for large-scale uncon strained optimization. *Mathematical Programming*, 26(2):190–212, Jun 1983. ISSN 1436-4646. doi: 10.1007/BF02592055. (Cited on page 36.)

Debadeepta Dey, Tian Yu Liu, Martial Hebert, and J. Andrew Bagnell. Contextual sequence prediction with application to control library optimization. In N. Roy, P. Newman, and S. Srinivasa, editors, *Robotics: Science and Systems VIII*. MIT Press, 2013. ISBN 9780262315722. (Cited on pages 10 and 90.)

Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion Planning as Probabilistic Inference using Gaussian Processes and Factor Graphs. In David Hsu, Nancy M. Amato, Spring Berman, and Sam Ade Jacobs, editors, *Robotics: Science and Systems XII*, volume 12. The Robotics: Science and Systems Foundation, 2016. ISBN 978-0-9923747-2-3. (Cited on page 91.)

Herbert Edelsbrunner and John Harer. Persistent homology—a survey. *Discrete & Computational Geometry*, 453, 01 2008. doi: 10.1090/conm/453/08802. (Cited on page 95.)

Mohamed Elbanhawi and Milan Simic. Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, 2:56–77, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2302442. (Cited on pages 4 and 76.)

Adrien Escande, Sylvain Miossec, Mehdi Benallegue, and Abderrahmane Kheddar. A Strictly Convex Hull for Computing Proximity Distances With Continuous Gradients. *IEEE Transactions on Robotics*, 30(3):666–678, Jun 2014. ISSN 1552-3098. doi: 10.1109/TRO.2013.2296332. (Cited on pages 50 and 51.)

Maurice Fallon, Scott Kuindersma, Sisir Karumanchi, Matthew Antone, Toby Schneider, Hongkai Dai, Claudia Pérez D'Arpino, Robin Deits, Matt DiCicco, Dehann Fourie, Twan Koolen, Pat Marion, Michael Posa, Andrés Valenzuela, Kuan-Ting Yu, Julie Shah, Karl Iagnemma, Russ Tedrake, and Seth Teller. An Architecture for Online Affordance-based Perception and Whole-body Planning. *Journal of Field Robotics*, 32(2):229–254, 2015. doi: 10.1002/rob.21546. (Cited on pages 119 and 126.)

Farbod Farshidian, Edo Jelavić, Alexander W. Winkler, and Jonas Buchli. Robust whole-body motion control of legged robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4589–4596, Sep. 2017. doi: 10.1109/IROS.2017.8206328. (Cited on page 8.)

Christian Feller and Christian Ebenbauer. Relaxed Logarithmic Barrier Function Based Model Predictive Control of Linear Systems. *IEEE Transactions on Automatic Control*, 62(3):1223–1238, March 2017. ISSN 0018-9286. doi: 10.1109/TAC.2016.2582040. (Cited on page 24.)

Henrique Ferrolho, Wolfgang Merkt, Yiming Yang, Vladimir Ivan, and Sethu Vijayakumar. Whole-Body End-Pose Planning for Legged Robots on Inclined Support Surfaces in Complex Environments. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 944–951, Nov 2018. doi: 10.1109/HUMANOIDS.2018.8625026. (Cited on pages 44 and 46.)

Chien Liang Fok, Fei Sun, Matt Mangum, Al Mok, Binghan He, and Luis Sentis. Web Based Teleoperation of a Humanoid Robot. arXiv preprint arXiv:1607.05402, 2016. (Cited on page 119.)

Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, Jul 1999. (Cited on page 129.)

Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch Informed Trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, May 2015. doi: 10.1109/ICRA.2015.7139620. (Cited on page 5.)

Christian Gehring, Stelian Coros, Marco Hutter, C. Dario Bellicoso, Huub Heijnen, Remo Diethelm, Michael Bloesch, Péter Fankhauser, Jemin Hwangbo, Markus A. Hoepflinger, and Roland Siegwart. Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot. volume 23, pages 34–43, March 2016. doi: 10.1109/MRA.2015.2505910. (Cited on page 35.)

Robert Ghrist. Barcodes: the persistent topology of data. *Bulletin of the American Mathematical Society*, 45(1):61–75, 2008. doi: 10.1090/S0273-0979-07-01191-3. (Cited on page 99.)

Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal of Robotics and Automation*, 4(2):193–203, Apr 1988. ISSN 0882-4967. doi: 10.1109/56.2083. (Cited on pages 32 and 50.)

Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 12(4): 979–1006, 2002. doi: 10.1137/S1052623499350013. (Cited on pages 36, 59, 85, and 146.)

E. Goto, Y. Shi, and N. Yoshida. Extrapolated surface charge method for capacity calculation of polygons and polyhedra. *Journal of Computational Physics*, 100(1): 105–115, 1992. ISSN 0021-9991. doi: 10.1016/0021-9991(92)90313-N. (Cited on pages 57 and 58.)

Ruben Grandia, Farbod Farshidian, René Ranftl, and Marco Hutter. Feedback MPC for Torque-Controlled Legged Robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4730–4737, Nov 2019. doi: 10.1109/IROS40897.2019.8968251. (Cited on page 159.)

Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. In Jose A. Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation: Advances in the Estimation of Distribution Algorithms*, pages 75–102. Springer, Berlin, Heidelberg, 2006. ISBN 978-3-540-32494-2. doi: 10.1007/3-540-32494-1_4. (Cited on page 35.)

Kris Hauser. Semi-Infinite Programming for Trajectory Optimization with Non-convex Obstacles. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2018. (Cited on page 52.)

Paul Hebert, Max Bajracharya, Jeremy Ma, Nicolas Hudson, Alper Aydemir, Jason Reid, Charles Bergh, James Borders, Matthew Frost, Michael Hagman, John Leichty, Paul Backes, Brett Kennedy, Paul Karplus, Brian Satzinger, Katie Byl, Krishna Shankar, and Joel Burdick. Mobile Manipulation and Mobility as Manipulation—Design and Algorithms of RoboSimian. *Journal of Field Robotics*, 32(2): 255–274, 2015. doi: 10.1002/rob.21566. (Cited on page 120.)

Andreas Hermann, Felix Mauch, Klaus Fischnaller, Sebastian Klemm, Arne Roennau, and Ruediger Dillmann. Anticipate your surroundings: Predictive collision detection between dynamic obstacles and planned robot trajectories on the GPU. In *European Conference on Mobile Robots (ECMR)*, pages 1–8, Sep. 2015. doi: 10.1109/ECMR.2015.7324047. (Cited on pages 125, 134, and 136.)

Geoffrey E. Hinton. Products of experts. *Proceedings of the 9th International Conference on Artificial Neural Networks*, pages 1–6(5), 1999. (Cited on page 94.)

Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned Neural Networks for Character Control. *ACM Transactions on Graphics (TOG)*, 36(4):42:1–42:13, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073663. (Cited on page 11.)

Norbert Antony Murray Hootsmans. *The motion control manipulators on mobile vehicles*. PhD thesis, Massachusetts Institute of Tech., 1992. (Cited on page 140.)

Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: an efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, Apr 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9321-0. (Cited on pages 121 and 123.)

Albert S. Huang, Edwin Olson, and David C. Moore. LCM: Lightweight Communications and Marshalling. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4057–4062, Oct 2010. doi: 10.1109/IROS.2010.5649358. (Cited on page 129.)

Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi: 10.1126/scirobotics.aau5872. (Cited on page 12.)

Brian Ichter and Marco Pavone. Robot Motion Planning in Learned Latent Spaces. *IEEE Robotics and Automation Letters*, 4(3):2407–2414, July 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2901898. (Cited on page 158.)

Brian Ichter, James Harrison, and Marco Pavone. Learning Sampling Distributions for Robot Motion Planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094, May 2018. doi: 10.1109/ICRA.2018.8460730. (Cited on page 158.)

Vladimir Ivan, Dmitry Zarubin, Marc Toussaint, Taku Komura, and Sethu Vijayakumar. Topology-based representations for motion planning and generalization in dynamic environments with interactions. *The International Journal of Robotics Research*, 32(9-10):1151–1163, 2013. doi: 10.1177/0278364913482017. (Cited on pages 22, 31, 78, 80, and 158.)

Vladimir Ivan, Yiming Yang, Wolfgang Merkt, Michael P. Camilleri, and Sethu Vijayakumar. EXOTica: An Extensible Optimization Toolset for Prototyping and Benchmarking Motion Planning and Control. In Anis Koubaa, editor, *Robot Operating System (ROS): The Complete Reference (Volume 3)*, pages 211–240. Springer

International Publishing, Cham, 2019. ISBN 978-3-319-91590-6. doi: 10.1007/978-3-319-91590-6_7. (Cited on pages 16, 59, 84, 102, 126, and 145.)

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79. PMID: 31141872. (Cited on page 94.)

David H. Jacobson. Differential dynamic programming methods for solving bang-bang control problems. *IEEE Transactions on Automatic Control*, 13(6):661–675, December 1968. ISSN 0018-9286. doi: 10.1109/TAC.1968.1099026. (Cited on page 97.)

Nikolay Jetchev and Marc Toussaint. Fast motion planning from experience: trajectory prediction for speeding up movement generation. *Autonomous Robots*, 34(1):111–127, Jan 2013. ISSN 1573-7527. doi: 10.1007/s10514-012-9315-y. (Cited on pages 13, 32, 76, 80, 111, and 157.)

Steven G. Johnson. The NLopt nonlinear-optimization package. `https://github.com/stevengj/nlopt`, 2014. (Cited on page 36.)

Tomasz Kaczynski, Konstantin Mischaikow, and Marian Mrozek. *Computational homology*, volume 157. Springer Science & Business Media, 2006. doi: 10.1007/b97315. (Cited on page 99.)

Peter Kaiser, Dimitrios Kanoulas, Markus Grotz, Luca Muratore, Alessio Rocchi, Enrico M. Hoffman, Nikos G. Tsagarakis, and Tamim Asfour. An affordance-based pilot interface for high-level control of humanoid robots in supervised autonomy. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 621–628, Nov 2016. doi: 10.1109/HUMANOIDS.2016.7803339. (Cited on page 120.)

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, May 2011. doi: 10.1109/ICRA.2011.5980280. (Cited on pages 5, 21, 50, 75, and 91.)

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. doi: 10.1177/0278364911406761. (Cited on page 5.)

Sisir Karumanchi, Kyle Edelberg, Ian Baldwin, Jeremy Nash, Jason Reid, Charles Bergh, John Leichty, Kalind Carpenter, Matthew Shekels, Matthew Gildner, David Newill-Smith, Jason Carlton, John Koehler, Tatyana Dobreva, Matthew Frost, Paul Hebert, James Borders, Jeremy Ma, Bertrand Douillard, Paul Backes, Brett Kennedy, Brian Satzinger, Chelsea Lau, Katie Byl, Krishna Shankar, and Joel Burdick. Team RoboSimian: Semi-autonomous Mobile Manipulation at the 2015 DARPA Robotics Challenge Finals. *Journal of Field Robotics*, 34(2):305–332, 2017. doi: 10.1002/rob.21676. (Cited on pages 118 and 120.)

Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics*, 12(4):566–580, Aug 1996. ISSN 1042-296X. doi: 10.1109/70.508439. (Cited on page 4.)

S. Mohammad Khansari-Zadeh and Aude Billard. Learning Stable Nonlinear Dynamical Systems With Gaussian Mixture Models. *IEEE Transactions on Robotics*, 27(5):943–957, Oct 2011. ISSN 1552-3098. doi: 10.1109/TRO.2011.2159412. (Cited on page 11.)

Jin-Oh Kim and Pradeep K. Khosla. Real-time obstacle avoidance using harmonic potential functions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 790–796, Apr 1991. doi: 10.1109/ROBOT.1991.131683. (Cited on page 57.)

Sanghyun Kim, Keunwoo Jang, Suhan Park, Yisoo Lee, Sang Yup Lee, and Jaeheung Park. Whole-body Control of Non-holonomic Mobile Manipulator Based on Hierarchical Quadratic Programming and Continuous Task Transition. In *Proceedings of the International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 414–419, July 2019. doi: 10.1109/ICARM.2019.8834269. (Cited on page 140.)

Zachary Kingston, Mark Moll, and Lydia E. Kavraki. Sampling-Based Methods for Motion Planning with Constraints. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):159–185, 2018. doi: 10.1146/annurev-control-060117-105226. (Cited on pages 4 and 145.)

Jonas Koenemann, Andrea Del Prete, Yuval Tassa, Emo Todorov, Olivier Stasse, Maren Bennewitz, and Nicolas Mansard. Whole-body model-predictive control applied to the HRP-2 humanoid. In *Proceedings of the IEEE/RSJ International*

Conference on Intelligent Robots and Systems (IROS), pages 3346–3351, Sep. 2015. doi: 10.1109/IROS.2015.7353843. (Cited on pages 9 and 150.)

Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf, Anant Gupta, Oskar von Stryk, Felipe Bacim, Doug A. Bowman, Alex Goins, Ravi Balasubramanian, and David C. Conner. Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials. *Journal of Field Robotics*, 32(3):352–377, 2015. doi: 10.1002/rob.21558. (Cited on pages 120 and 134.)

James J. Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001, April 2000. doi: 10.1109/ROBOT.2000.844730. (Cited on pages 4, 21, 86, 126, and 148.)

Steven M. LaValle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. Technical report, Iowa State University, 1998. (Cited on page 4.)

Teguh Santoso Lembono, Antonio Paolillo, Emmanuel Pignat, and Sylvain Calinon. Memory of Motion for Warm-Starting Trajectory Optimization. *IEEE Robotics and Automation Letters*, 5(2):2594–2601, April 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2972893. (Cited on page 109.)

Peter Leven and Seth Hutchinson. A Framework for Real-time Path Planning in Changing Environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002. doi: 10.1177/027836490202101200l. (Cited on page 13.)

Sergey Levine and Vladlen Koltun. Continuous Inverse Optimal Control with Locally Optimal Examples. In *Proceedings of the 29th International Conference on Machine Learning*, ICML '12, pages 475–482, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. (Cited on page 11.)

Sergey Levine and Vladlen Koltun. Guided Policy Search. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, page III–1–III–9. JMLR.org, 2013. (Cited on page 12.)

Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989. ISSN 1436-4646. doi: 10.1007/BF01589116. (Cited on page 5.)

Christian Mandery, Ömer Terlemez, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. The KIT whole-body human motion database. In *Proceedings of the*

International Conference on Advanced Robotics (ICAR), pages 329–336, July 2015. doi: 10.1109/ICAR.2015.7251476. (Cited on page 11.)

Nicolas Mansard, Andrea Del Prete, Matthieu Geisert, Steve Tonneau, and Olivier Stasse. Using a Memory of Motion to Efficiently Warm-Start a Nonlinear Predictive Controller. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2986–2993, May 2018. doi: 10.1109/ICRA.2018.8463154. (Cited on pages 12, 76, 90, 93, 111, and 157.)

Zita Marinho, Anca Dragan, Arun Byravan, Byron Boots, Siddhartha Srinivasa, and Geoffrey Gordon. Functional Gradient Motion Planning in Reproducing Kernel Hilbert Spaces. In David Hsu, Nancy M. Amato, Spring Berman, and Sam Ade Jacobs, editors, *Robotics: Science and Systems XII*, volume 12. The Robotics: Science and Systems Foundation, 2016. ISBN 978-0-9923747-2-3. (Cited on page 91.)

Pat Marion, Maurice Fallon, Robin Deits, Andrés Valenzuela, Claudia Pérez D'Arpino, Greg Izatt, Lucas Manuelli, Matt Antone, Hongkai Dai, Twan Koolen, John Carter, Scott Kuindersma, and Russ Tedrake. Director: A User Interface Designed for Robot Operation with Shared Autonomy. *Journal of Field Robotics*, 34(2):262–280, 2017. doi: 10.1002/rob.21681. (Cited on pages 16, 119, 121, and 125.)

David Q. Mayne. A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems. *International Journal of Control*, 3 (1):85–95, 1966. doi: 10.1080/00207176608921369. (Cited on page 97.)

Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Srinath Sridhar, Gerard Pons-Moll, and Christian Theobalt. Single-Shot Multi-person 3D Pose Estimation from Monocular RGB. In *International Conference on 3D Vision (3DV)*, pages 120–130, Sep. 2018. doi: 10.1109/3DV.2018.00024. (Cited on page 11.)

Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2520–2525, May 2011. doi: 10.1109/ICRA.2011.5980409. (Cited on page 97.)

Wolfgang Merkt, Yiming Yang, Theodoros Stouraitis, Christopher Edwin Mower, Maurice Fallon, and Sethu Vijayakumar. Robust Shared Autonomy for Mobile Manipulation with Continuous Scene Monitoring. In *Proceedings of the IEEE*

Conference on Automation Science and Engineering (CASE), pages 130–137, Aug 2017. doi: 10.1109/COASE.2017.8256092. (Cited on page 16.)

Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar. Leveraging Precomputation with Problem Encoding for Warm-Starting Trajectory Optimization in Complex Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5877–5884, Oct 2018. doi: 10.1109/IROS.2018.8593977. (Cited on pages 16 and 80.)

Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar. Continuous-Time Collision Avoidance for Trajectory Optimization in Dynamic Environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7248–7255, Nov 2019a. doi: 10.1109/IROS40897.2019.8967641. (Cited on pages 16 and 140.)

Wolfgang Merkt, Vladimir Ivan, Yiming Yang, and Sethu Vijayakumar. Towards Shared Autonomy Applications using Whole-body Control Formulations of Locomanipulation. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 1206–1211, Aug 2019b. doi: 10.1109/COASE.2019.8843153. (Cited on page 16.)

Juan C. Meza, Ricardo A. Oliva, Patricia D. Hough, and Pamela J. Williams. OPT++: An Object-oriented Toolkit for Nonlinear Optimization. *ACM Transactions on Mathematical Software (TOMS)*, 33(2), Jun 2007. ISSN 0098-3500. doi: 10.1145/1236463.1236467. (Cited on page 85.)

Brian Vincent Mirtich. *Impulse-based dynamic simulation of rigid body systems*. PhD thesis, University of California, Berkeley, 1996. (Cited on pages 52 and 56.)

Djordje Mitrovic, Stefan Klanke, and Sethu Vijayakumar. Adaptive Optimal Feedback Control with Learned Internal Dynamics Models. In Olivier Sigaud and Jan Peters, editors, *From Motor Learning to Interaction Learning in Robots*, pages 65–84. Springer, Berlin, Heidelberg, 2010. ISBN 978-3-642-05181-4. doi: 10.1007/978-3-642-05181-4_4. (Cited on page 91.)

Katja Mombaur, Anh Truong, and Jean-Paul Laumond. From human to humanoid locomotion—an inverse optimal control approach. *Autonomous Robots*, 28(3): 369–383, Apr 2010. ISSN 1573-7527. doi: 10.1007/s10514-009-9170-7. (Cited on page 11.)

Igor Mordatch, Emanuel Todorov, and Zoran Popović. Discovery of Complex Behaviors Through Contact-invariant Optimization. *ACM Transactions on Graphics (TOG)*, 31(4):43:1–43:8, July 2012. ISSN 0730-0301. doi: 10.1145/2185520.2185539. (Cited on page 6.)

Jorge J. Moré. The Levenberg-Marquardt algorithm: Implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, pages 105–116. Springer, Berlin, Heidelberg, 1978. ISBN 978-3-540-35972-2. doi: 10.1007/BFb0067700. (Cited on pages 5, 36, 146, and 149.)

Takeshi Mori, Matthew Howard, and Sethu Vijayakumar. Model-free apprenticeship learning for transfer of human impedance behaviour. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 239–246, Oct 2011. doi: 10.1109/Humanoids.2011.6100830. (Cited on page 11.)

João Moura, William McColl, Gerard Taykaldiranian, Tetsuo Tomiyama, and Mustafa S. Erden. Automation of Train Cab Front Cleaning With a Robot Manipulator. *IEEE Robotics and Automation Letters*, 3(4):3058–3065, Oct 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2849591. (Cited on page 151.)

Christopher Edwin Mower, Wolfgang Merkt, and Sethu Vijayakumar. Comparing Alternate Modes of Teleoperation for Constrained Tasks. In *Proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 1497–1504, Aug 2019. doi: 10.1109/COASE.2019.8843265. (Cited on page 116.)

Mustafa Mukadam, Xinyan Yan, and Byron Boots. Gaussian Process Motion Planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15, May 2016. doi: 10.1109/ICRA.2016.7487091. (Cited on page 91.)

Jia Pan, Sachin Chitta, and Dinesh Manocha. FCL: A general purpose library for collision and proximity queries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3859–3866, May 2012. doi: 10.1109/ICRA.2012.6225337. (Cited on page 59.)

Chonhyon Park, Jia Pan, and Dinesh Manocha. ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2012. (Cited on page 76.)

Shinsuk Park, Robert D. Howe, and David F. Torchiana. Virtual Fixtures for Robotic Cardiac Surgery. In Wiro J. Niessen and Max A. Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2001: 4th International Conference Utrecht, The Netherlands, October 14–17, 2001 Proceedings*, pages 1419–1420. Springer, Berlin, Heidelberg, 2001. ISBN 978-3-540-45468-7. doi: 10.1007/3-540-45468-3_252. (Cited on page 118.)

Rajnikant V Patel, Farshid Shadpey, Farzam Ranjbaran, and Jorge Angeles. A collision-avoidance scheme for redundant manipulators: Theory and experiments. *Journal of Robotic Systems*, 22(12):737–757, 2005. doi: 10.1002/rob.20096. (Cited on page 51.)

Karl Pauwels and Danica Kragic. SimTrack: A simulation-based framework for scalable real-time object pose detection and tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1300–1307, Sep. 2015. doi: 10.1109/IROS.2015.7353536. (Cited on page 148.)

Dmytro Pavlichenko and Sven Behnke. Efficient stochastic multicriteria arm trajectory optimization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4018–4025, Sep 2017. doi: 10.1109/IROS.2017.8206256. (Cited on pages 51, 75, and 76.)

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided Deep Reinforcement Learning of Physics-based Character Skills. *ACM Transactions on Graphics (TOG)*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. (Cited on page 11.)

Jose A. Perea and John Harer. Sliding Windows and Persistence: An Application of Topological Methods to Signal Analysis. *Foundations of Computational Mathematics*, 15(3):799–838, Jun 2015. ISSN 1615-3383. doi: 10.1007/s10208-014-9206-z. (Cited on page 110.)

Michael Phillips, Benjamin Cohen, Sachin Chitta, and Maxim Likhachev. E-Graphs: Bootstrapping Planning with Experience Graphs. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012. doi: 10.15607/RSS.2012.VIII.043. (Cited on page 90.)

Emmanuel Pignat and Sylvain Calinon. Bayesian Gaussian Mixture Model for Robotic Policy Imitation. *IEEE Robotics and Automation Letters*, 4(4):4452–4458, Oct 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2932610. (Cited on pages 11 and 94.)

Florian T. Pokorny, Majd Hawasly, and Subramanian Ramamoorthy. Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, 35(1-3):204–223, 2016. doi: 10.1177/0278364915586713. (Cited on page 95.)

Brahayam Ponton, Alexander Herzog, Andrea Del Prete, Stefan Schaal, and Ludovic Righetti. On Time Optimization of Centroidal Momentum Dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, May 2018. doi: 10.1109/ICRA.2018.8460537. (Cited on page 8.)

Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014. doi: 10.1177/0278364913506757. (Cited on page 7.)

Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an Open-Source Robot Operating System. In *Proceedings of the ICRA Workshop on Open Source Software*, volume 3, page 5, 2009. (Cited on page 139.)

Ahmed H Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion Planning Networks: Bridging the Gap Between Learning-based and Classical Motion Planners. arXiv preprint arXiv:1907.06013, 2019. (Cited on pages 111 and 158.)

Andreea Radulescu, Ioannis Havoutis, Darwin G. Caldwell, and Claudio Semini. Whole-body trajectory optimization for non-periodic dynamic motions on quadrupedal systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5302–5307, May 2017. doi: 10.1109/ICRA.2017.7989623. (Cited on page 35.)

Joose Rajamäki, Kourosh Naderi, Ville Kyrki, and Perttu Hämäläinen. Sampled differential dynamic programming. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1402–1409, Oct 2016. doi: 10.1109/IROS.2016.7759229. (Cited on page 35.)

Daniel Rakita, Bilge Mutlu, and Michael Gleicher. RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion. In *Robotics: Science and Systems*, 2018. (Cited on page 44.)

Nathan Ratliff, Matt Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Proceedings of*

the IEEE International Conference on Robotics and Automation (ICRA), pages 489–494, May 2009. doi: 10.1109/ROBOT.2009.5152817. (Cited on pages 5, 50, 75, and 91.)

Nathan Ratliff, Marc Toussaint, and Stefan Schaal. Understanding the geometry of workspace obstacles in Motion Optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4202–4209, May 2015. doi: 10.1109/ICRA.2015.7139778. (Cited on pages 51 and 75.)

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. Path Integral Control by Reproducing Kernel Hilbert Space Embedding. In *International Joint Conference on Artificial Intelligence*, pages 1628–1634, 2013. (Cited on page 91.)

Stephane Redon, Ming C. Lin, Dinesh Manocha, and Young J. Kim. Fast continuous collision detection for articulated models. *Journal of Computing and Information Science in Engineering*, 5(2):126–137, 02 2005. ISSN 1530-9827. doi: 10.1115/1.1884133. (Cited on page 52.)

Tobias Rodehutskors, Max Schwarz, and Sven Behnke. Intuitive bimanual telemanipulation under communication restrictions by immersive 3D visualization and motion tracking. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 276–283, Nov 2015. doi: 10.1109/HUMANOIDS.2015.7363547. (Cited on page 119.)

Alberto Romay, Stefan Kohlbrecher, David C. Conner, Alexander Stumpf, and Oskar von Stryk. Template-based manipulation in unstructured environments for supervised semi-autonomous humanoid robots. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 979–986, Nov 2014. doi: 10.1109/HUMANOIDS.2014.7041482. (Cited on page 120.)

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–6, Oct 2009. doi: 10.1109/IROS.2009.5354683. (Cited on page 124.)

Mohammed E. Sayed, Markus P. Nemitz, Simona Aracri, Alistair C. McConnell, Ross M. McKenzie, and Adam A. Stokes. The Limpet: A ROS-Enabled Multi-Sensing Platform for the ORCA Hub. *Sensors*, 18(10), 2018. ISSN 1424-8220. doi: 10.3390/s18103487. (Cited on page 148.)

Stefan Schaal. Dynamic Movement Primitives: A Framework for Motor Control in Humans and Humanoid Robotics. In Hiroshi Kimura, Kazuo Tsuchiya, Akio Ishiguro, and Hartmut Witte, editors, *Adaptive Motion of Animals and Machines*, pages 261–280. Springer Tokyo, Tokyo, 2006. ISBN 978-4-431-31381-6. doi: 10.1007/4-431-31381-8_23. (Cited on page 11.)

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014. doi: 10.1177/0278364914528132. (Cited on pages 7, 21, 52, 55, 56, 59, 66, 67, 68, 75, and 91.)

Thomas B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT Press, Cambridge, MA, USA, 1992. ISBN 0-262-19316-7. (Cited on page 119.)

Dong Hun Shin, Bradley S. Hamner, Sanjiv Singh, and Myung Hwangbo. Motion planning for a mobile manipulator with imprecise locomotion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 847–853, Oct 2003. doi: 10.1109/IROS.2003.1250735. (Cited on page 140.)

Sebastian Starke, Norman Hendrich, Dennis Krupke, and Jianwei Zhang. Evolutionary multi-objective inverse kinematics on highly articulated and humanoid robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6959–6966, Sep. 2017. doi: 10.1109/IROS.2017.8206620. (Cited on page 46.)

Oliver Stasse, Adrien Escande, Nicolas Mansard, Sylvain Miossec, Paul Evrard, and Abderrahmane Kheddar. Real-time (self)-collision avoidance task on a HRP-2 humanoid robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3200–3205, May 2008. doi: 10.1109/ROBOT.2008.4543698. (Cited on page 50.)

Martin Stolle and Christopher G. Atkeson. Policies based on trajectory libraries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3344–3349, May 2006. doi: 10.1109/ROBOT.2006.1642212. (Cited on pages 10 and 76.)

Martin Stolle, Hanns Tappeiner, Joel Chestnutt, and Christopher G. Atkeson. Transfer of policies based on trajectory libraries. In *Proceedings of the IEEE/RSJ Interna-*

tional Conference on Intelligent Robots and Systems (IROS), pages 2981–2986, Oct 2007. doi: 10.1109/IROS.2007.4399364. (Cited on page 10.)

Theodoros Stouraitis, Iordanis Chatzinikolaidis, Michael Gienger, and Sethu Vijayakumar. Dyadic collaborative Manipulation through Hybrid Trajectory Optimization. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 869–878. PMLR, 29–31 Oct 2018. (Cited on page 7.)

Jörg Stückler, Max Schwarz, Mark Schadler, Angeliki Topalidou-Kyniazopoulou, and Sven Behnke. NimbRo Explorer: Semiautonomous Exploration and Mobile Manipulation in Rough Terrain. *Journal of Field Robotics*, 33(4):411–430, 2016. doi: 10.1002/rob.21592. (Cited on page 120.)

Hisashi Sugiura, Michael Gienger, Herbert Janssen, and Christian Goerick. Real-Time Self Collision Avoidance for Humanoids by means of Nullspace Criteria and Task Intervals. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 575–580, Dec 2006. doi: 10.1109/ICHR.2006.321331. (Cited on page 51.)

Masashi Sugiyama, Hirotaka Hachiya, Christopher Towell, and Sethu Vijayakumar. Geodesic Gaussian kernels for value function approximation. *Autonomous Robots*, 25(3):287–304, Oct 2008. ISSN 1573-7527. doi: 10.1007/s10514-008-9095-6. (Cited on pages 80 and 91.)

Gao Tang and Kris Hauser. Discontinuity-Sensitive Optimal Control Learning by Mixture of Experts. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7892–7898, May 2019. doi: 10.1109/ICRA.2019.8793909. (Cited on pages 12, 94, and 111.)

Yuval Tassa. *Theory and Implementation of Biomimetic Motor Controllers*. PhD thesis, Hebrew University of Jerusalem, 2011. (Cited on page 5.)

Yuval Tassa, Tom Erez, and William D. Smart. Receding Horizon Differential Dynamic Programming. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1465–1472. Curran Associates, Inc., 2008. (Cited on pages 10 and 76.)

Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *Proceedings of the IEEE International Conference on*

Robotics and Automation (ICRA), pages 1168–1175, May 2014. doi: 10.1109/ICRA.2014.6907001. (Cited on pages 6 and 98.)

Russ Tedrake. Underactuated robotics: Algorithms for walking, running, swimming, flying, and manipulation (course notes for MIT 6.832). http://underactuated.mit.edu, 2014–2019. (Cited on page 96.)

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, 11(Nov):3137–3181, 2010. (Cited on pages 5 and 75.)

Steve Tonneau, Andrea Del Prete, Julien Pettré, Chonhyon Park, Dinesh Manocha, and Nicolas Mansard. An Efficient Acyclic Contact Planner for Multiped Robots. *IEEE Transactions on Robotics*, pages 1–16, 2018a. ISSN 1552-3098. doi: 10.1109/TRO.2018.2819658. (Cited on page 8.)

Steve Tonneau, Pierre Fernbach, Andrea Del Prete, Julien Pettré, and Nicolas Mansard. 2PAC: Two-Point Attractors for Center Of Mass Trajectories in Multi-Contact Scenarios. *ACM Transactions on Graphics (TOG)*, 37(5):176:1–176:14, October 2018b. ISSN 0730-0301. doi: 10.1145/3213773. (Cited on page 8.)

Marc Toussaint. Robot Trajectory Optimization Using Approximate Inference. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1049–1056, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553508. (Cited on pages 5, 21, 36, and 75.)

Marc Toussaint. Newton methods for k-order Markov constrained motion problems. arXiv preprint arXiv:1407.0414, 2014. (Cited on pages 5, 75, and 77.)

Marc Toussaint. A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal Control, and Probabilistic Inference. In Jean-Paul Laumond, Nicolas Mansard, and Jean-Bernard Lasserre, editors, *Geometric and Numerical Foundations of Movements*, pages 361–392. Springer International Publishing, Cham, 2017. ISBN 978-3-319-51547-2. doi: 10.1007/978-3-319-51547-2_15. (Cited on page 77.)

Christopher Tralie, Nathaniel Saul, and Rann Bar-On. Ripser.py: A Lean Persistent Homology Library for Python. *The Journal of Open Source Software*, 3(29):925, Sep 2018. doi: 10.21105/joss.00925. (Cited on page 102.)

Nikolaus Vahrenkamp, Dominik Muth, Peter Kaiser, and Tamin Asfour. IK-Map: An enhanced workspace representation to support inverse kinematics solvers.

In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 785–790, Nov 2015. doi: 10.1109/HUMANOIDS.2015.7363443. (Cited on page 46.)

Gino Van Den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game Developers Conference*, pages 821–837, Mar 2001. (Cited on pages 32 and 50.)

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, ICML '00, pages 1079–1086, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1-55860-707-2. (Cited on page 93.)

Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006. ISSN 1436-4646. doi: 10.1007/s10107-004-0559-y. (Cited on page 36.)

Matthew R. Walter, Matthew Antone, Ekapol Chuangsuwanich, Andrew Correa, Randall Davis, Luke Fletcher, Emilio Frazzoli, Yuli Friedman, James Glass, Jonathan P. How, Jeong hwan Jeon, Sertac Karaman, Brandon Luders, Nicholas Roy, Stefanie Tellex, and Seth Teller. A Situationally Aware Voice-commandable Robotic Forklift Working Alongside People in Unstructured Outdoor Environments. *Journal of Field Robotics*, 32(4):590–628, 2015. doi: 10.1002/rob.21539. (Cited on page 120.)

He Wang, Kirill A. Sidorov, Peter Sandilands, and Taku Komura. Harmonic Parameterization by Electrostatics. *ACM Transactions on Graphics (TOG)*, 32(5): 155:1–155:12, Sep 2013. ISSN 0730-0301. doi: 10.1145/2503177. (Cited on pages 57 and 58.)

John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198, Oct 2016. doi: 10.1109/IROS.2016.7759617. (Cited on page 148.)

Yue Wang, Yanmei Jiao, Rong Xiong, Hongsheng Yu, Jiafan Zhang, and Yong Liu. MASD: A Multimodal Assembly Skill Decoding System for Robot Programming by Demonstration. *IEEE Transactions on Automation Science and Engineering*, 15(4):

1722–1734, Oct 2018. ISSN 1545-5955. doi: 10.1109/TASE.2017.2783342. (Cited on page 137.)

Eric W. Weisstein. Homotopy. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Homotopy.html, 1999–2019. (Cited on page 99.)

Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John J. Leonard, and John McDonald. Kintinuous: Spatially Extended KinectFusion. In *R:SS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, pages 1–8, 2012. (Cited on page 123.)

Pierre-Brice Wieber. Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 137–142, Dec 2006. doi: 10.1109/ICHR.2006.321375. (Cited on page 9.)

Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian Processes for Regression. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press, 1996. (Cited on page 93.)

Alexander W. Winkler, C. Dario Bellicoso, Marco Hutter, and Jonas Buchli. Gait and Trajectory Optimization for Legged Systems Through Phase-Based End-Effector Parameterization. *IEEE Robotics and Automation Letters*, 3(3):1560–1567, July 2018. ISSN 2377-3766. doi: 10.1109/LRA.2018.2798285. (Cited on page 7.)

Matthew Wright. Introduction to persistent homology, 2015. URL https://www.youtube.com/watch?v=h0bnG1Wavag. (Cited on page 99.)

Yoshio Yamamoto and Xiaoping Yun. Coordinating locomotion and manipulation of a mobile manipulator. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2643–2648 vol.3, Dec 1992. doi: 10.1109/CDC.1992.371337. (Cited on page 140.)

Holly A. Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. Analysis of Human-robot Interaction at the DARPA Robotics Challenge Trials. *Journal of Field Robotics*, 32(3):420–444, 2015. doi: 10.1002/rob.21568. (Cited on page 119.)

Yiming Yang, Vladimir Ivan, and Sethu Vijayakumar. Real-time motion adaptation using relative distance space representation. In *Proceedings of the International*

Conference on Advanced Robotics (ICAR), pages 21–27, July 2015. doi: 10.1109/ICAR.2015.7251428. (Cited on pages 22, 78, 80, and 158.)

Yiming Yang, Vladimir Ivan, Zhibin Li, Maurice Fallon, and Sethu Vijayakumar. iDRM: Humanoid motion planning with realtime end-pose selection in complex environments. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 271–278, Nov 2016a. doi: 10.1109/HUMANOIDS.2016.7803288. (Cited on pages 44, 46, 121, 128, and 136.)

Yiming Yang, Vladimir Ivan, Wolfgang Merkt, and Sethu Vijayakumar. Scaling sampling-based motion planning to humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1448–1454, Dec 2016b. doi: 10.1109/ROBIO.2016.7866531. (Cited on pages 4, 76, and 140.)

Yiming Yang, Wolfgang Merkt, Henrique Ferrolho, Vladimir Ivan, and Sethu Vijayakumar. Efficient Humanoid Motion Planning on Uneven Terrain Using Paired Forward-Inverse Dynamic Reachability Maps. *IEEE Robotics and Automation Letters*, 2(4):2279–2286, Oct 2017. ISSN 2377-3766. doi: 10.1109/LRA.2017.2727538. (Cited on pages 44, 46, and 140.)

Yiming Yang, Wolfgang Merkt, Vladimir Ivan, Zhibin Li, and Sethu Vijayakumar. HDRM: A Resolution Complete Dynamic Roadmap for Real-Time Motion Planning in Complex Scenes. *IEEE Robotics and Automation Letters*, 3(1):551–558, Jan 2018a. ISSN 2377-3766. doi: 10.1109/LRA.2017.2773669. (Cited on pages 13, 21, 33, 51, 54, 76, 140, and 157.)

Yiming Yang, Wolfgang Merkt, Vladimir Ivan, and Sethu Vijayakumar. Planning in Time-Configuration Space for Efficient Pick-and-Place in Non-Static Environments with Temporal Constraints. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1–9, Nov 2018b. doi: 10.1109/HUMANOIDS.2018.8624989. (Cited on pages 61, 62, 140, and 145.)

Qin-Zhong Ye. The signed Euclidean distance transform and its applications. In *International Conference on Pattern Recognition*, pages 495–499 vol.1, May 1988. doi: 10.1109/ICPR.1988.28276. (Cited on page 7.)

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. Mode-adaptive Neural Networks for Quadruped Motion Control. *ACM Transactions on Graphics (TOG)*, 37(4):145:1–145:11, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201366. (Cited on page 11.)

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, Jul 2008. (Cited on page 11.)

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. doi: 10.1177/0278364913488805. (Cited on page 21.)