EPFL

Thèse n° 10 702

A Stochastic Approach to Contact-rich Manipulation

Présentée le 28 février 2025

Faculté des sciences et techniques de l'ingénieur
Laboratoire de traitement des signaux 5
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Julius Maximilian JANKOWSKI

Acceptée sur proposition du jury

Prof. C. N. Jones, président du jury
Prof. J.-Ph. Thiran, Dr S. Calinon, directeurs de thèse
Prof. M. Toussaint, rapporteur
Prof. G. Chalvatzaki, rapporteuse
Prof. S. Coros, rapporteur

École polytechnique fédérale de Lausanne

2025

All models are wrong,
but some are useful.
— George Box

To my mother and my father,
for their unconditional love and support.

Acknowledgements

After four years of research in robotics and even more years of studies leading up to this thesis, I am deeply grateful to all the people who have made this journey possible in one way or another.

First, I would like to thank my advisors, Prof. Jean-Philippe Thiran and Dr. Sylvain Calinon, for giving me the opportunity to pursue this journey and for their guidance and support throughout. I would also like to thank my colleagues in the RLI group for all the fun and fruitful collaborations. A special thanks goes to Hakan and Mattia, who helped me with my first steps during my doctoral studies. I would also like to especially thank Teguh, Teng, and Ante for great collaborations and discussions that contributed to this thesis.

During my undergraduate time, I was lucky enough to work with much more experienced researchers, who taught me about the art of controlling robots. A special thanks goes to my early mentors, Johannes Kühn, Nico Mansfeld, Ahmed Wafik, Jonas Wittmann and Daniel Wahrmann.

Next, I would like to thank my family for their continuous support. My father, who has nurtured my curiosity for physics, electronics, and programming from an early age, and my mother, who has always believed in me, even when I did not.

This journey would not have been possible without the most amazing people that I am lucky enough to call friends and family since my childhood. They have always been a reminder that life is about special moments with the people you love.

Last, I would like to thank Lara, my partner in every imaginable way. We share the same passion for robotics, work together on our projects, and we even write *love papers* (cf. Chapter 4). But most importantly, we always manage to stick together even over long distances. I am looking forward to everything that is yet to come.

Lausanne, October 2, 2024 J. J.

Abstract

For robots to operate in unstructured environments, they are required to interact with objects through contact. Those contacts may be used to push objects to the side, deform objects, or manipulate objects in-hand. This thesis addresses the problem of controlling robots to exploit contacts to manipulate objects. Being able to anticipate the outcome of such physical interactions is essential for robots to gain true autonomy. However, contact interactions are particularly challenging to reason over in model-based control approaches due to the discontinuous nature of contacts. Moreover, interacting with objects the robot has not interacted with before will naturally lead to uncertainty in the prediction of contact dynamics. For instance, the robot can not anticipate the mass distribution of an object before making contact, which requires the robot to reason over possible outcomes before touching the object in a potentially unfavorable or unsafe way.

Throughout this thesis, we formulate the problem of contact-rich manipulation with a robot manipulator as a model-predictive control problem. We explore stochastic optimization to plan for robot control trajectories in realtime. We show that the stochasticity in the optimization process enables the algorithm to explore the space of contacts without relying on local gradients or discretization of the contact space. We furthermore study how uncertainties in the physical properties of the object propagate through the contact dynamics and how the robot can actively reduce such uncertainty by exploiting favorable contact modes and sequences. We integrate the above contributions into a planning and control framework for robots to manipulate objects through contacts in realtime. The framework is evaluated in a series of robot experiments, demonstrating robots autonomously performing dynamic hand-overs, push objects to a moving target, play air hockey, and manipulate objects robustly using two arms in the presence of uncertainty in the dynamics of the object.

Keywords: Trajectory Optimization, Underactuated Systems, Contact-Rich Manipulation, Stochastic Optimization, Model Predictive Control, Stochastic Dynamics, Robust Control.

Zusammenfassung

Damit Roboter in unstrukturierten Umgebungen arbeiten können, müssen sie mit Objekten durch Kontakt interagieren. Diese Kontakte können genutzt werden, um Objekte zur Seite zu schieben, Objekte zu verformen oder Objekte in der Hand zu manipulieren. Diese Arbeit befasst sich mit dem Problem der Regelung von Robotern zur Manipulation von Objekten durch Kontakte. Um echte Autonomie zu erlangen, müssen Roboter in der Lage sein, das Ergebnis solcher physischen Interaktionen vorherzusehen. Kontaktinteraktionen sind jedoch aufgrund der diskontinuierlichen Natur von Kontakten besonders schwierig in modellbasierten Regelungsansätzen zu berücksichtigen. Darüber hinaus führt die Interaktion mit Objekten, mit denen der Roboter zuvor noch nicht interagiert hat, zu Unsicherheiten bei der Vorhersage der Kontaktdynamik. Zum Beispiel kann der Roboter die Massenverteilung eines Objekts nicht vorhersehen, bevor er es berührt. Dies erfordert, dass der Roboter über mehrere mögliche Ergebnisse schlussfolgert, bevor er das Objekt auf eine potenziell ungünstige oder unsichere Weise berührt.
In dieser Arbeit formulieren wir das Problem der kontaktreichen Manipulation mit einem Robotermanipulator als ein *Model-predictive Control* Problem. Wir untersuchen die stochastische Optimierung zur Planung von Robotertrajektorien in Echtzeit. Wir zeigen, dass die Stochastik im Optimierungsprozess es dem Algorithmus ermöglicht, den Raum der Kontakte zu erkunden, ohne sich auf lokale Gradienten oder eine Diskretisierung des Kontaktraums zu verlassen. Darüber hinaus untersuchen wir, wie sich Unsicherheiten in den physikalischen Eigenschaften des Objekts durch die Kontaktdynamik ausbreiten und wie der Roboter diese Unsicherheiten aktiv reduzieren kann, indem er günstige Kontaktmodi und -sequenzen nutzt. Wir integrieren die oben genannten Beiträge in Planungs- und Regelungsalgorithmen für Roboter, die Objekte durch Kontakte in Echtzeit manipulieren. Die Algorithmen werden in einer Reihe von Roboterexperimenten evaluiert, die zeigen, dass Roboter autonom dynamische Übergaben durchführen, Objekte zu einem sich bewegenden Ziel schieben, Airhockey spielen und Objekte robust mit zwei Armen manipulieren, wenn Unsicherheiten in der Dynamik des Objekts vorhanden sind.

Schlüsselwörter: Trajektorienoptimierung, unterbetätigte Systeme, kontaktreiche Manipulation, stochastische Optimierung, Model Predictive Control, stochastische Dynamik, robuste Steuerung.

Contents

Acknowledgements i

Abstract (English/Deutsch) iii

1 Introduction 1
- 1.1 Main Challenges . . . 2
  - 1.1.1 Discontinuity of Contact Dynamics . . . 2
  - 1.1.2 Uncertainty in Contact Dynamics . . . 3
- 1.2 State-of-the-Art in Contact-Rich Manipulation . . . 4
  - 1.2.1 Contact-Implicit Trajectory Optimization . . . 4
  - 1.2.2 Mixed Discrete/Continuous Optimization . . . 5
  - 1.2.3 Reinforcement Learning . . . 6
  - 1.2.4 Sampling-based Planning and Control . . . 7
- 1.3 Core Contributions & Thesis Outline . . . 8
  - 1.3.1 Optimal Basis Functions for Efficient Trajectory Synthesis . . . 8
  - 1.3.2 VP-STO for Making-and-Breaking Contacts in Realtime . . . 8
  - 1.3.3 Belief Prediction through Contacts for Robust Pushing . . . 9
- 1.4 Thesis Statement . . . 9

2 Background 11
- 2.1 Low-level Robot Control . . . 11
  - 2.1.1 Direct Force Control . . . 12
  - 2.1.2 Stiffness Control . . . 12
- 2.2 Modeling Contact Dynamics . . . 13
  - 2.2.1 Second-Order Models . . . 14
  - 2.2.2 Quasi-Static & Quasi-Dynamic Models . . . 14
  - 2.2.3 Collision Models . . . 15
- 2.3 Trajectory Optimization . . . 15
  - 2.3.1 Direct Transcription . . . 16
  - 2.3.2 Direct Shooting . . . 16
- 2.4 Trajectory Representation . . . 17
  - 2.4.1 Discretization in Time . . . 17
  - 2.4.2 Superposition of Basis Functions . . . 18
- 2.5 Zero-order Optimization . . . 18

2.5.1 Cross-Entropy Method (CEM) . . . 19
2.5.2 Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) . . 19
2.6 Belief-Space Control . . . 20
2.6.1 Covariance Steering Approach . . . 20
2.6.2 Trajectory Optimization for Non-Gaussian Belief Spaces . . . 20

3 From Optimal Control to Time-Parameterized Basis Functions 23
3.1 Optimal Basis Functions . . . 24
3.1.1 Minimal-Effort Trajectories with Linear Constraints . . . 25
3.1.2 Optimal Trajectory as a Linear Mapping of Constraints . . . 27
3.2 Optimal Time Parameterization . . . 29
3.2.1 Direct Minimization of the Trajectory Duration . . . 30
3.2.2 Iterative Optimization of the Timing of Key Positions . . . 33
3.3 Skill Learning from Sparse Key Positions . . . 35
3.3.1 Learning a Distribution of Optimal Trajectories . . . 38
3.3.2 Control Phase . . . 40
3.3.3 Experimental Evaluation in a User Study . . . 41

4 Via-point-based Stochastic Trajectory Optimization 47
4.1 Related Work . . . 49
4.2 Preliminaries: Trajectory Representation . . . 50
4.3 Via-Point-based Stochastic Trajectory Optimization . . . 51
4.3.1 Informed Sampling with a Gaussian Prior . . . 52
4.3.2 Synthesis of Kinodynamically Admissible Trajectories . . . 54
4.3.3 Cost Evaluation . . . 54
4.4 Online VP-STO (MPC) . . . 55
4.4.1 No-Via-Point Trajectory for Stopping Behavior . . . 56
4.4.2 Initialization: Exploration vs. Warm-Starting . . . 56
4.4.3 Impedance Control . . . 57
4.5 Experiments . . . 57
4.5.1 Simulation . . . 58
4.5.2 Real-World Experiments . . . 59
4.5.3 Ablation Studies . . . 61

5 Stochastic Impact Control in Real-Time 65
5.1 Related Work . . . 67
5.2 Learning Stochastic Contact Models . . . 68
5.2.1 Mixture of linear-Gaussian Contact Dynamics . . . 69
5.2.2 Learning Model Parameters from Data . . . 70
5.2.3 Piecewise-linear Kalman Filtering . . . 71
5.2.4 Probability of Hitting the Goal . . . 71
5.3 Fast Contact Planning under Uncertainty . . . 73
5.3.1 Stochastic Optimal Control for Shooting . . . 74

5.3.2 Shooting Angle as Reduced Action Space . . . 74
5.3.3 Training an Energy-based Shooting Policy . . . 75
5.3.4 Online Inference with Warm-Starting . . . 76
5.4 Experiments . . . 77
5.4.1 Implementation Details . . . 77
5.4.2 Experimental Setup . . . 77
5.4.3 Experimental Results . . . 78

6 Belief-space Planning through Contacts 79
6.1 Related Work . . . 81
6.1.1 Contact-rich Manipulation . . . 81
6.1.2 Robust Manipulation . . . 82
6.1.3 Modeling Uncertainty in Contact Dynamics . . . 83
6.2 Problem Formulation & Approach . . . 84
6.3 Belief Dynamics through Contacts . . . 85
6.3.1 Stochastic Quasi-Static Dynamics for Pushing . . . 85
6.3.2 Object Belief Dynamics . . . 88
6.3.3 Variance Prediction . . . 90
6.4 Stochastic Optimization for Robust Manipulation . . . 93
6.4.1 Variance Gain Control . . . 94
6.4.2 Trajectory Sampling with a Contact Prior . . . 96
6.5 Receding-horizon BS-VP-STO . . . 103
6.6 Experiments . . . 105
6.6.1 Implementation Details . . . 105
6.6.2 Open-Loop Single-Hand Pushing . . . 105
6.6.3 Open-Loop Bimanual Pushing . . . 107
6.6.4 Closed-Loop Bimanual Pushing . . . 112
6.7 Discussion . . . 113

7 Conclusion 115
7.1 Limitations . . . 116
7.1.1 Limitations of Current Models of Contact Dynamics . . . 116
7.1.2 Limitations of Gaussian Trajectory Priors . . . 117
7.1.3 Limitations of Local Optimization . . . 117
7.2 Future Work . . . 118
7.2.1 Physics-based Learning of Stochastic Contact Dynamics . . . 118
7.2.2 Learning Generative Trajectory Priors . . . 118
7.2.3 Reasoning over Tactile Feedback . . . 119

A Appendix to Chapter 6 121

Bibliography 133

1 Introduction

Enabling robots to interact with the world physically is a key challenge in robotics. In particular, the ability to move, reorient, or localize objects through contact is a fundamental capability for robots to perform tasks in unstructured environments. *Making-and-breaking* contact is therefore a fundamental aspect of manipulation. Research in robotic manipulation has traditionally focused on prehensile manipulation, where robots grasp objects with their end-effectors and manipulate them by moving the end-effector. Robots do not require physical models of contact interactions to perform prehensile manipulation tasks, as the object may be assumed to be rigidly attached to the robot once the grasp is achieved. Consequently, the control task becomes a problem of controlling the robot's end-effector towards the grasp configuration without touching anything, closing the gripper to grasp the object, and then moving the end-effector with the object to the desired configuration again without touching anything. While this is a capability that is sufficient in many applications situated in structured environments where objects are known and the environment is predictable, prehensile manipulation fails if, for instance, the target object is too large for the gripper or if other objects block the path of the gripper to enclose the object.

In this thesis, we study control approaches that not only allow robots to safely establish contacts with their environment, but to exploit contacts to manipulate objects in more versatile ways. This problem has been approached in the literature as non-prehensile manipulation, where robots interact with objects through contact. One class of approaches uses *contact-implicit* constraints on the state of the underactuated system to enforce physically feasible trajectories when optimizing jointly over robot and object states. Such an approach is prone to local minima and thus requires good initial solutions to find the desired contact interactions. Another class of approaches is based on a discretization of the contact space and a subsequent *mixed-integer programming* formulation to discover promising contact interactions. Such an approach is known to not scale well with the number of contact modes and the number of potential switching times. *Learning-based* approaches on the other hand have shown successful synthesis of contact-rich manipulation actions for complex tasks such as

in-hand manipulation. One of the keys to the success of learning-based approaches is the stochastic process of sampling control actions during policy search. Another important aspect is the deployed stochasticity of the dynamics, known as domain randomization, which enables robust simulation-to-real transfer. However, in contrast to (explicit) model-based approaches, learning-based approaches typically require large amounts of data and cannot thus synthesize new skills ad-hoc. Towards scalable dexterous robotic manipulation, the contributions of this thesis aim to bring together the ad-hoc generalizability of model-based approaches and the stochastic and robust aspects of learning-based approaches.

In the following, we discuss the main challenges that arise when reasoning over contacts is required. Then, the core contributions of this thesis are presented, and an outline of the thesis is given.

1.1 Main Challenges

The gap between what we want robots to do – operating autonomously in an unseen, unstructured environment – and what robots are currently capable of – operating for a few minutes in a lab environment – is still very large. In this section, we discuss the main challenges that make current approaches to controlling robots fail when reasoning through contacts is required.

1.1.1 Discontinuity of Contact Dynamics

In non-prehensile manipulation, the discontinuity of contact dynamics is a major challenge. The tools that have been developed for controlling fully-actuated and even underactuated systems typically rely on gradients that inform the robot how to correct its sequence of actions in order to improve the outcome of the task. However, if the objective of the task is to change the state of the object, the gradient with respect to the object state is zero for any sequence of actions that do not involve contact. Figure 1.1 illustrates this problem in a simple example of a robot aiming to push an object. Even in such a simple scenario, gradient-based optimization methods fail to generate a plan for moving the object to the right if the initial guess for the solution does not involve contact. Moreover, making contact with an object is still not enough for the robot to arbitrarily move the object (assuming the contact is not adhesive). For non-prehensile actions, such contacts can only be used to push into the object and not to pull the object. Thus, the applicable range of forces is defined by friction cones (see Lynch and Park (2017) for more details). In the example of Figure 1.1, the robot would have to first move around the object to be able to push it to the left. This limited control authority makes planning and control for non-prehensile manipulation prone to local minima.

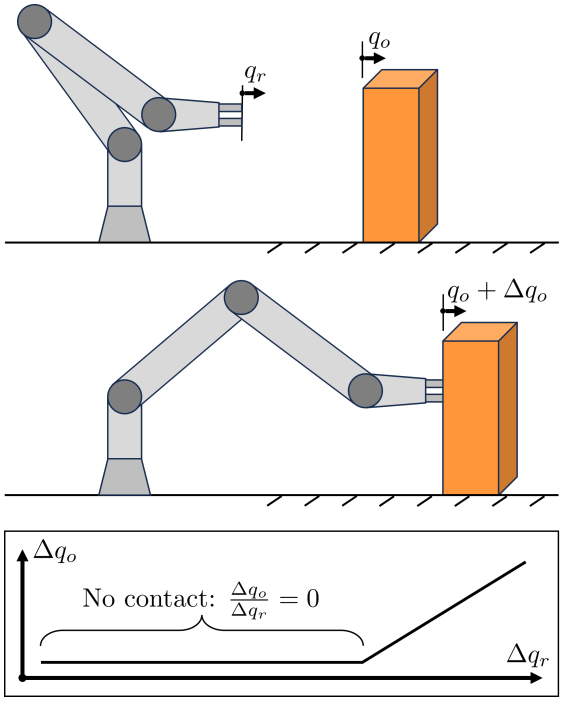


Figure 1.1: A 1D toy example showcasing the fundamental challenge of discontinuous contact dynamics. In robotic manipulation, we are interested in changing the state of the object. However, this requires a robot to first make contact with the object. Since there is no gradient guiding the robot towards a meaningful contact mode, algorithms either rely on heuristics or on a systematic exploration of the contact space.

As a result, the fundamental challenge of reasoning over discontinuous contact dynamics requires exploration of the contact space without fully relying on gradients. But how can this exploration be done efficiently? Exploring the space of contacts involves not only the single point in time where the robot is in contact with an object. It also involves a longer time horizon for exploring the ways the robot is able to move through space before those contacts are made and for exploring how an object can be moved after contact is made. This not only requires exploration in space but also in variable horizon lengths. A short horizon may not allow the robot to reach the object due to the dynamic limits of the robot, while a long horizon increases the complexity of the problem.

1.1.2 Uncertainty in Contact Dynamics

Robots that should operate autonomously in uncertain, unstructured environments are required to manipulate a large range of different, possibly previously unseen objects. In other words, there will always be objects the robot is uncertain about and consequently, it is not sufficient to train or plan a robot's skill for a specific set of object parameters. This is a natural source of uncertainty that robots have to deal with in realtime. Even with a high-fidelity physics engine, it is impossible to accurately predict the outcome of physical interactions if the object's properties such as mass, friction, or shape are uncertain. The highly non-linear nature of contact dynamics may even amplify the effect of small uncertainties if the contact mechanics are unfavorable, *e.g.* when the contact surfaces are convex. Figure 1.2 illustrates this problem for a simple

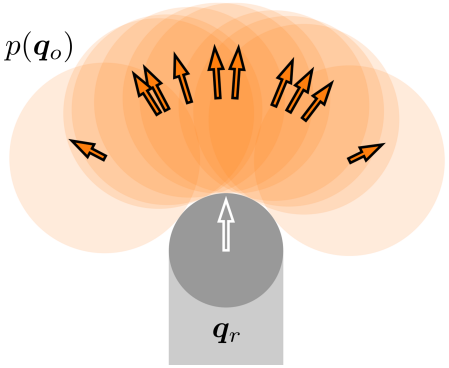


Figure 1.2: Top view of a robotic finger • pushing an object open-loop after a couple of time steps. The object particles • illustrate possible outcomes of the interaction due to uncertain dynamics. The convex contact geometries lead to a build-up of uncertainty in the object's position over time.

example of a robot with a convex geometry pushing an object with a convex geometry. Even if the initial object location is known, the object may move tangentially to the contact surface due to imperfect mass distribution or sliding conditions. Thus, the uncertainty in the object's position may build up over time, leading to a failure of the manipulation task. When generating open-loop plans by assuming an accurate model of the dynamics, those plans are likely to fail as they do not take into account the underlying uncertainty. Rodriguez (2021) describes this effect with an experiment of repeatedly picking and placing a queen chess piece, which fails if the queen is picked from the top. In contrast, picking the queen from the side stabilizes the repeated pick-and-place process. This raises the question of how to let robots deal with this uncertainty autonomously. Deploying fast feedback loops is an implicit approach to compensate for uncertain dynamics, such as done in model-predictive control schemes. Yet, modeling uncertainties and reasoning over an anticipated distribution of outcomes, *i.e.* a belief, is a more explicit way of coping with uncertain dynamics.

1.2 State-of-the-Art in Contact-Rich Manipulation

This section reflects on different approaches to contact-rich manipulation categorized into four groups: contact-implicit optimization, mixed discrete/continuous optimization, reinforcement learning, and sampling-based planning and control. Based on the main challenges described in the previous section, we will discuss advantages and limitations of each approach.

1.2.1 Contact-Implicit Trajectory Optimization

Contact-implicit optimization is a framework that allows for gradient-based optimization of the motion of a robot while considering the discontinuous dynamics of contacts (Posa et al. (2014)). The idea is to encode the contact dynamics as constraints in the optimization problem and to optimize over the system state as described in

Section 2.3.1. Contact-implicit trajectory optimization has shown to be successful in non-manipulation applications, *i.e.* controlling robots by exploiting contacts (Manchester et al. (2019); Cleac'h et al. (2021); Aydinoglu et al. (2022)). Such optimization algorithms are typically combined with techniques for smoothing the dynamics to enable the discovery of new contact modes in the vicinity of the current solution.

Recently, Aydinoglu et al. (2024) have shown that contact-implicit optimization can be used to manipulate objects in a closed loop. However, as contact-implicit trajectory optimization remains local and gradient-based, the capability of discovering contact modes heavily depends on the initialization of the optimization problem and thus on task-specific heuristics.

Last, contact-implicit optimization approaches to date depend on accurate and differentiable models of the contact dynamics. Especially if used in an open-loop fashion, deviations from the modeled dynamics can lead to failure of the generated plan. This limits the use of such approaches to manipulating objects with known properties or to tasks where the robot has time to learn the dynamics of the object before executing the plan.

1.2.2 Mixed Discrete/Continuous Optimization

An explicit way of dealing with discontinuous contact dynamics is to discretize the space of making contact. The problem of making and breaking contact is thus turned into a discrete decision-making problem. Mixed-integer programming approaches combine the search for sequences of discrete contact modes with continuous optimization of the system within the selected modes. The problem is thus reduced to multiple continuous optimization problems that are connected through the discrete decisions, rather than attempting to optimize over the entire discontinuous system. Marcucci et al. (2017) use this approach for stabilizing a humanoid robot in a push recovery task. Contacts between the hands of the robot and the surrounding walls are encoded as contact modes such that the robot decides which wall to use for stabilization and how to generate control actions for that particular wall. Chen et al. (2021) combine tree search with trajectory optimization for planning multi-fingered manipulation of a box-shaped object. The search is performed not only over combinatorial sequences of contacts, but also over combinations of multiple contact points, *i.e.* one potential contact point per finger. The subsequent trajectory optimization generates plans for the robot to realize the discrete plan of multi-contact sequences. Toussaint et al. (2022a) present a general framework for combining sequences of discrete decisions with continuous optimization. Here, the discrete decisions are encoded as constraints for the underlying continuous optimization problem. The method is showcased on a pushing task, *e.g.* by encoding the contact between the robot and the object as a constraint that can be activated or deactivated. Most recently, Graesdal et al. (2024) propose to

translate the problem of planning through contacts into a graph of convex sets, where each convex set represents a contact mode or a convex set of the contact-free space. The approach is demonstrated on a 2D pushing task with the robot abstracted as a circle in the task space.

Ha et al. (2020) have shown that mixed-integer programming can be extended to include uncertainty in the contact dynamics in individual continuous optimization problems. This enables discrete reasoning about contact modes and sequences that are robust against uncertainties in the contact dynamics. In an experiment, they show that their approach successfully reasons that using two fingers to push an object is more robust than using one finger.

The great benefit of mixed discrete and continuous optimization is that it decomposes the planning for making and breaking contacts into combinations of tractable subproblems. While there are applications that naturally provide a finite set of contact modes, such as footstep planning for legged robots or manipulation of objects with a finite number of flat surfaces, the approach is limited by the combinatorial explosion of the number of contact modes that can be combined in time, *i.e.* sequence of contacts, and in space, *i.e.* number and location of contact points.

1.2.3 Reinforcement Learning

Reinforcement learning has shown great success in learning manipulation skills for robots. Examples include learning to push objects (Shetty et al. (2024)) and learning for in-hand manipulation (Andrychowicz et al. (2020); Handa et al. (2023)). Reinforcement learning techniques benefit from stochastic rollouts in simulated environments, allowing those techniques to explore the space of possible interactions with the environment without explicitly relying on gradients of the reward function with respect to the parameters of the policy. This explorative aspect of the learning process is crucial for learning manipulation skills that entail making-and-breaking contacts without relying on close-to-optimal initializations of the policy.

Moreover, in many reinforcement learning approaches (Haarnoja et al. (2019); Schulman et al. (2015, 2017)), domain randomization is a natural way of informing the skill learning process about all the possibilities that may be encountered when moving from simulation to reality. Domain randomization may include probability distributions over parameters of the dynamics model such as friction coefficients, object mass, or object geometry (Andrychowicz et al. (2020); Muratore et al. (2022)). By simulating a large number of combinations of policy samples and domain samples, the policy ideally converges to a behavior that is robust against the uncertainty modeled through domain randomization. Domain randomization as an interface for modeling uncertainties in physical interactions is a key aspect for successful sim-to-real transfer even

if objects are encountered for the first time.

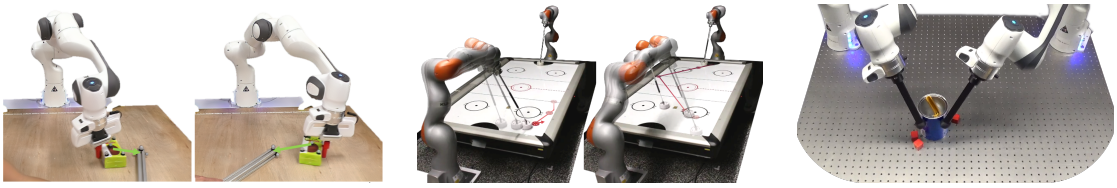
However, exploration and domain randomization in reinforcement learning approaches come at the cost of long training times involving many rollouts in the simulator. While resulting policies are often usable in slightly different scenarios or objects if those changes are within the training domain, skills or plans can not be generated ad-hoc for new out-of-domain tasks without retraining. This is in contrast to model-based planning and control techniques, which are limited by the underlying model rather than the training setup and the reward function used at training time.

1.2.4 Sampling-based Planning and Control

Sampling-based planning and control techniques aim to overcome the limitations of gradient-based optimization by sampling the space of possible actions and states. In the spirit of shooting methods for trajectory optimization, sampling-based methods draw candidate control actions, simulate the system forward using a model of contact dynamics, and evaluate the resulting state. Howell et al. (2022) show that a simple algorithm that rolls out many sampled candidates and selects the best one can be successfully used to control contact-rich manipulation in a model-predictive control fashion. Such an approach is purely based on rapidly exploring contact modes, without refining plans. Thus, resulting plans are suboptimal and jerky, and thus have not yet been shown to transfer from simulation to real systems. However, it highlights that controlling through contacts does not require plans to be optimal with respect to a designed objective function, but rather requires the system to rapidly find a feasible plan.

More recently, Pang et al. (2023) propose to combine traditional sampling-based planning (RRT) with a locally smoothed model of contact dynamics. While the sampling-based planner provides a way to explore the space of possible contact modes by sampling set points of a stiffness controller, the locally smoothed model guides the planner towards contact modes that let the robot move the object towards the desired state. However, the approach is an offline planning method and thus relies on accurate models of the contact dynamics for successful execution of the open-loop plan.

To summarize, sampling-based planning and control techniques are a promising approach to efficiently explore the space of possible contact interactions. The main shortcoming of discussed state-of-the-art methods is the lack of efficient, informative sampling strategies to generate feasible manipulation plans in realtime. This thesis aims to address this shortcoming to achieve closed-loop manipulation of objects.



(a) Pushing a box towards a moving target.

(b) Controlling collisions with a highly dynamic puck.

(c) Bi-manual pushing under uncertainty.

Figure 1.3: A selection of experiments on contact-rich manipulation conducted in the context of this thesis.

1.3 Core Contributions & Thesis Outline

Towards robots that decide to robustly *make and break* contacts in realtime, this thesis investigates and develops algorithms for controlling robots through contacts. All algorithms presented in this thesis are deployed on real robots, showcasing the applicability of the developed methods to real-world manipulation tasks. Figure 1.3 illustrates three example experiments on controlling objects by exploiting contacts. The following provides an overview of the core contributions of this thesis.

1.3.1 Optimal Basis Functions for Efficient Trajectory Synthesis

To search over the space of possible contact interactions in realtime, efficient synthesis of feasible robot trajectories is required. In Chapter 3, we introduce a novel set of basis functions that are derived from the analytical solution to a constrained linear optimal control problem. We furthermore derive an algorithm for directly[1] optimizing the duration of the trajectory. The derived online algorithm generates smooth and timing-optimal trajectories for a fully actuated robot given a set of via-points, *i.e.* a small number of positions that represent the trajectory. Since the basis functions can be pre-computed offline, computing smooth and timing-optimal trajectories can be performed online at high replanning frequencies.

1.3.2 VP-STO for Making-and-Breaking Contacts in Realtime

In Chapter 4, we further exploit our proposed low-dimensional representation of trajectories as a superposition of optimal basis functions inside a zero-oder shooting approach, which we call Via-point-based Stochastic Trajectory Optimization (VP-STO). We optimize for a low-dimensional set of via-points, which are used to synthesize trajectories from the optimal basis functions, via stochastic optimization. This setup enables highly efficient optimization loops, as well as probabilistic warm-starting. The

[1] Direct optimization refers to a non-iterative optimization algorithm.

key advantage of this approach is its ability to handle discontinuous cost landscapes, such as optimizing for contact-making trajectories. In Chapter 5, VP-STO is combined with stochastic optimal impact control to control a robot to shoot a puck for playing air hockey.

1.3.3 Belief Prediction through Contacts for Robust Pushing

Beyond the ability to efficiently optimize for timing-optimal contact trajectories, the robot will also require mechanisms to anticipate contact dynamics and their stochasticity when touching an object, which is influenced by typically unknown object properties, such as mass distribution, friction, or shape. Hence, in Chapter 6, we study how uncertainties propagate through contacts using quasi-static models for contact dynamics. We derive a new metric measuring the variance in the predicted object belief dynamics given a candidate control sequence, which we use to generate robust pushing plans.

1.4 Thesis Statement

This thesis aims to push the boundaries of contact-rich manipulation. The contributions of this thesis enable robots to achieve reactive, robust, and autonomous manipulation of objects in structured lab environments. Moreover, the developed theoretical insights and algorithms should also serve as a basis for further research toward robots that can operate in naturally unstructured environments. In summary, this thesis aims to show that:

Reactive, robust, and autonomous contact-rich manipulation in unstructured environments requires efficient sampling-based control algorithms to reason through stochastic models of contact dynamics.

While achieving this level of autonomy will require a strong interplay between advancements in robotic hardware, perception, state estimation, and control, the insights gathered within our robot experiments should help us anticipate what control algorithms are suited for the challenges that arise from operating *in the wild*.

2 Background

This chapter provides a brief overview of the main concepts and tools that are used as building blocks throughout this thesis. The main focus is on fundamental concepts for low-level robot control, modeling contact dynamics, optimal control, zero-order optimization and belief-space control. The goal of this chapter is to discuss different concepts within the sub-categories in the context of contact-rich manipulation.

2.1 Low-level Robot Control

In order to develop algorithms for manipulating objects, it is essential to establish low-level controllers that allow us to control the robot manipulator in a stable way even in the presence of contacts. The dynamics of the robot manipulator govern the motion of the robot and are defined as

$$\boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{u} + \boldsymbol{\tau}_{\text{ext}}, \tag{2.1}$$

where $\boldsymbol{q}$ are the generalized coordinates of the robot, $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ are the Coriolis and centrifugal forces, and $\boldsymbol{g}(\boldsymbol{q})$ are the gravitational forces. The generalized forces generated by the robot's actuators are denoted as a control input $\boldsymbol{u}$. Note that this assumes that the actuators of the robot can accurately output a desired generalized force. This is typically a fair assumption for direct drive actuators or actuators equipped with a torque sensor and a motor-level control loop tracking the desired torque. In addition, the robot manipulator is also subject to external forces $\boldsymbol{\tau}_{\text{ext}}$. In non-prehensile manipulation, these external forces are used to move objects.

For the goal of object manipulation, we are interested in controlling the motion and interaction forces of the robot. In the following, we will discuss two different control strategies that are widely used in manipulation: direct force control and stiffness control.

2.1.1 Direct Force Control

An intuitive way of abstracting the robot dynamics is to compensate for the Coriolis, centrifugal and gravitational forces. Consequently, the control law becomes

$$\boldsymbol{u} = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{\text{des}}, \tag{2.2}$$

with $\boldsymbol{\tau}_{\text{des}}$ being the desired generalized forces acting on the compensated system. The direct force controller is particularly useful if the robot is already in contact with its environment and we are interested in exerting a desired force on the environment. In this case, we may assume that the robot does not accelerate as a result of a non-zero desired force, *i.e.* $\ddot{\boldsymbol{q}} = \boldsymbol{0}$, which corresponds to a quasi-dynamic equilibrium of the system with

$$\boldsymbol{\tau}_{\text{ext}} = -\boldsymbol{\tau}_{\text{des}}. \tag{2.3}$$

As a result, the auxiliary control variable $\boldsymbol{\tau}_{\text{des}}$ can be used to control the force $\boldsymbol{\tau}_{\text{ext}}$ that the robot exerts on the environment. However, if the robot is not in contact with its environment, *i.e.* $\boldsymbol{\tau}_{\text{ext}} = \boldsymbol{0}$, the direct force command results in accelerating the robot in the direction of the desired force with

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{\tau}_{\text{des}}. \tag{2.4}$$

This second-order relationship between the auxiliary control command and the motion of the robot entails practical challenges when directly optimizing for $\boldsymbol{\tau}_{\text{des}}$ in downstream algorithms. First, good initial solutions for the force trajectory are difficult to obtain for contact-rich manipulation. Second, deviations from the model of the robot dynamics in (2.1) can lead to large deviations in the state trajectory of the robot, even if the deviation in the acceleration of the robot is small. This sensitivity to model errors, such as friction within the robot joints, makes direct force control challenging for tasks that require precise control. Thus, deploying a closed-loop controller that handles these low-level model errors and abstracts the motion and force response of the robot facilitates planning and control for manipulation tasks.

2.1.2 Stiffness Control

Impedance control is a widely used control strategy for robots interacting with the environment (Hogan (1984)). It is based on the idea of imposing mass-spring-damper dynamics on the robot while compensating for other dynamic effects of the robot. However, shaping the inertia of the robot manipulator is practically challenging as it requires feeding back the joint accelerations. Since joints are typically only equipped with a position encoder, the estimate of the joint acceleration is noisy. Impedance control is therefore usually reduced to *stiffness control* by adopting the actual robot inertia $\boldsymbol{M}$ for the imposed dynamics. In practice, the terms impedance control and

stiffness control are typically used interchangebly while usually referring to stiffness control. The stiffness control law is given by

$$\boldsymbol{u} = \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{q}_{\text{des}}) - \boldsymbol{D}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\text{des}}) + \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}}_{\text{des}}, \tag{2.5}$$

where $\boldsymbol{K}$ and $\boldsymbol{D}$ are the stiffness and damping matrices, respectively, and $\boldsymbol{q}_{\text{des}}$, $\dot{\boldsymbol{q}}_{\text{des}}$, and $\ddot{\boldsymbol{q}}_{\text{des}}$ are the desired position, velocity, and acceleration of the robot, respectively. By tuning the parameter of the virtual spring, *i.e.* the resulting stiffness of the robot, the robot becomes more or less compliant while having Lyapunov-stability guarantees even when facing unmodeled contact interactions (Albu-Schäffer et al. (2007)). As a result, the closed-loop dynamics of a stiffness-controlled robot are given by

$$\boldsymbol{M}(\boldsymbol{q})(\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_{\text{des}}) + \boldsymbol{D}(\dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}_{\text{des}}) + \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{q}_{\text{des}}) = \boldsymbol{\tau}_{\text{ext}}. \tag{2.6}$$

Using stiffness control as a low-level control layer, the robot manipulator itself is stable and robust against errors in the robot dynamics while exposing the time-varying set point of the controller in (2.5) as an auxiliary control command. Virtually moving the set point $\boldsymbol{q}_{\text{des}}$ into the penetration of the environment or an object will result in a force that is linear with respect to the penetration depth $\boldsymbol{q} - \boldsymbol{q}_{\text{des}}$. Let's consider the steady-state case where the robot is in contact with the environment, *i.e.* $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} = \boldsymbol{0}$ and the commanded velocity is zero. We can see that the stiffness of the robot linearly maps the penetration depth to the force the robot exerts on the environment with

$$\boldsymbol{\tau}_{\text{ext}} = \boldsymbol{K}(\boldsymbol{q} - \boldsymbol{q}_{\text{des}}). \tag{2.7}$$

Moreover, we can see that the robot converges to the desired position with $\boldsymbol{q} = \boldsymbol{q}_{\text{des}}$ if the robot is not in contact with the environment. In both steady-state cases, the robot position and the interaction force $\boldsymbol{\tau}_{\text{ext}}$ are affine in the auxiliary control command $\boldsymbol{q}_{\text{des}}$, which is beneficial for optimization. This relationship is exploited in the remainder of this thesis for simultaneously planning for the robot's motion and interaction forces by using optimal control approaches that optimize the set point of the stiffness controller.

2.2 Modeling Contact Dynamics

In Section 2.1, we have discussed the dynamics of a robot manipulator (cf. (2.1)), *i.e.* how the robot responds to contact, and how stiffness controllers help to tune the response of the controlled robot to contact. However, to manipulate an object, we are interested in the dynamics of this object as a response to such contact interactions. Thus, we are interested in models of contact dynamics that describe the interaction of two bodies in contact in a mathematical form. Those models can be used to predict the state trajectory of objects manipulated by a robot. In non-prehensile manipulation, the full system describing the robot and the manipulated object is often described as

an *underactuated* system. It consists of an actuated part of the system, *i.e.* the robot, and an unactuated part of the system, *i.e.* one or multiple objects. Systems governed by contact dynamics can thus be described by a state vector

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}^r \\ \boldsymbol{x}^o \end{pmatrix}, \tag{2.8}$$

that contains the state of the robot $\boldsymbol{x}^r$ and the state of the object $\boldsymbol{x}_{\text{object}}$. Assuming that the robot is fully actuated with a control input $\boldsymbol{u}^r$, the discretized dynamics of the underactuated system can generally be described by

$$\begin{pmatrix} \boldsymbol{x}^r_{k+1} \\ \boldsymbol{x}^o_{k+1} \end{pmatrix} = \boldsymbol{f}\left(\begin{pmatrix} \boldsymbol{x}^r_k \\ \boldsymbol{x}^o_k \end{pmatrix}, \boldsymbol{u}^r_k \right), \tag{2.9}$$

where k denotes the discretized time step. The function $\boldsymbol{f}$ encapsulates the model of the contact dynamics. In robotics, different models of contact dynamics are used depending on the downstream algorithms and the level of detail required. In the following, we will briefly discuss different types of models of contact dynamics focused on rigid bodies. Note that the dynamics of deformable objects are out of the scope of this thesis.

2.2.1 Second-Order Models

Second-order models of contact dynamics resolve contacts at force level, resulting in a second-order differential equation. The state of the system consequently contains the generalized coordinates as well as the generalized velocities. Second-order models typically combine Newtonian mechanics of rigid bodies with soft contact models such that forces are assumed to be the result of small penetrations/deformations of the bodies without explicitly computing deformations. Physics engines such as Bullet (Coumans (2015)), MuJoCo (Todorov et al. (2012)), or Drake (Tedrake and the Drake Development Team (2019)) are typically based on such models with the goal of providing as accurate and as generic models as possible. Due to the modeling of contacts between rigid bodies at force level, the differential equations are *stiff* and require small time steps for numerical integration. Thus, computing rollouts using second-order models can be computationally expensive. However, second-order models are particularly popular for applications that do not demand fast rollouts, such as for training policies or testing control algorithms.

2.2.2 Quasi-Static & Quasi-Dynamic Models

Quasi-static and quasi-dynamic models have been used to simplify the prediction of slow physical interactions between robots and objects (Mason (2001); Koval et al.

(2016); Hogan and Rodriguez (2020); Cheng et al. (2021); Pang (2021); Pang et al. (2023)). Both classes of models assume that effects that are related to velocities and accelerations can be neglected as they do not affect the outcome of the prediction. As a result, the system state only contains the generalized coordinates of the robot and the object(s). This limits the range of applications to slow interactions such as pushing tasks, insertion tasks, or in-hand manipulation. Yet, the benefits of quasi-static and quasi-dynamic models are the lower dimensionality of the system state, i.e. half the number of states compared to second-order models, and the lower temporal resolution required to compute stable predictions of the system state. Both aspects effectively allow for faster simulated rollouts of robot plans and thus for more efficient downstream algorithms such as model-based planning and control.

2.2.3 Collision Models

In contrast to models that specifically capture slow contact interactions as described in Section 2.2.2, collision models are simplified models that capture the interaction of two bodies only at the moment of contact. The dynamics of high-impact contacts are typically governed by a short period of time (in the range of milliseconds) in which the bodies in contact exchange energy, deform, and change their states. Thus, collision models aim to capture this fast and complex interaction by predicting the state of the system after the collision based on the state of the system before the collision. In some robotic applications such as legged locomotion (Wieber et al. (2016)), catching objects (Yan et al. (2024)), or playing table tennis (Koç et al. (2018)), the dynamics of the system can be described by a combination of decoupled robot and object dynamics and a collision model for a single point in time. Such a simplification allows for fast and efficient rollouts as the contact dynamics only have to be evaluated at the moment of contact.

2.3 Trajectory Optimization

In this thesis, we encode the desired behavior of the robot for contact-rich manipulation as an optimal control problem. Let's consider a simple optimal control objective J that only depends on the state of the object after K time steps with respect to a desired state $\boldsymbol{x}^o_{\text{des}}$, *i.e.*

$$J = \left\| \boldsymbol{x}^o_K - \boldsymbol{x}^o_{\text{des}} \right\|^2 . \tag{2.10}$$

As we do not specify any desired behavior for the robot (for the sake of simplicity), the state and the controls of the robot should be optimized such that it makes contact with the object to subsequently move the object toward the desired state. The solution may be constrained by actuation limits of the robot, for instance $\boldsymbol{u}^r_k \in \mathcal{U}^r$. The discontinuous, and thus non-linear, nature of contact dynamics make the optimal control

problem non-linear and non-convex. Since Dynamic Programming, *i.e.* computing the value of every possible state, for non-linear optimal control problems does not scale well with the dimensionality of the state space, such optimal control problems are typically transcribed into trajectory optimization problems. In contrast to Dynamic Programming, trajectory optimization yields a single open-loop trajectory of the system. However, we can design closed-loop controllers by continuously replanning the trajectory based on the current state of the system, *i.e.* model-predictive control. In the following, we discuss two classes of approaches to solve trajectory optimization problems that mainly differ in the definition of the optimization variable and how the system dynamics are enforced. For a more rigorous review, please refer to Betts (2010).

2.3.1 Direct Transcription

In direct transcription techniques, we search over the space of state and control trajectories jointly. Thus, we jointly optimize for the robot and the object trajectory as well. As a result, the system dynamics are enforced through explicit constraints on the decision variables. The trajectory optimization problem can then be formulated as

$$\begin{aligned} \min_{\boldsymbol{u}^r_{0:K},\, \boldsymbol{x}^r_{0:K},\, \boldsymbol{x}^o_{0:K}} \quad & J \\ \text{s.t.} \quad & \begin{pmatrix} \boldsymbol{x}^r_{k+1} \\ \boldsymbol{x}^o_{k+1} \end{pmatrix} = \boldsymbol{f}\left(\begin{pmatrix} \boldsymbol{x}^r_{k} \\ \boldsymbol{x}^o_{k} \end{pmatrix}, \boldsymbol{u}^r_k \right), \\ & \boldsymbol{u}^r_k \in \mathcal{U}^r. \end{aligned} \tag{2.11}$$

Including the object trajectory in the set of decision variables has the advantage that the gradient of the objective with respect to the optimization variable is continuous and informative. However, in the case of contact-rich manipulation, the non-convexity of constraints from the contact dynamics make the optimization problem prone to local minima. In addition, the decision variables are subject to equality constraints. This is particularly challenging for stochastic optimization techniques that rely on sampling from the decision space, as the equality constraints have to be satisfied for each sample. As a result, direct transcription techniques for contact-rich manipulation require gradient-based optimization techniques that rely on good initial solutions to evade undesirable local minima.

2.3.2 Direct Shooting

In contrast to direct transcription techniques, direct shooting techniques only optimize for the control trajectory of the system. Instead of enforcing the contact dynamics through constraints on the decision variables, they are now part of the evaluation of the objective function. A candidate solution is evaluated by simulating the system dynamics forward in time using the control trajectory. In contact-rich manipulation with

fully actuated robots, this leads to decision variables that are not subject to equality constraints, which allows us to exploit stochastic optimization techniques. This is a key aspect to overcome the challenges associated with the discontinuous nature of contact dynamics and is exploited throughout this thesis to plan for non-prehensile manipulation actions. A drawback compared to direct transcription techniques is that the gradient of the objective with respect to the decision variables is not informative if the robot does not make contact with the object as discussed in Section 1.1.1. This thesis addresses this by combining zero-order optimization with probabilistic priors that facilitate efficient exploration of the contact space.

2.4 Trajectory Representation

In Section 2.3, we have introduced trajectory optimization as a tool for controlling robots to manipulate objects. In general, a trajectory is a continuous function that describes a variable, typically a spatial variable, over time, *e.g.* the generalized coordinates of the system over time $\boldsymbol{q}(t)$. Optimizing over the space of continuous functions is mathematically challenging. Therefore, trajectory optimization is oftentimes made tractable by representing trajectories using a finite set of variables. This enables kinematic planning of trajectories, *e.g.* for planning collision-free movements, or for planning dynamically constrained trajectories using direct transcription or shooting methods as described in Section 2.3, *e.g.* for contact-rich manipulation. In the following, we will discuss two different ways of representing trajectories that are commonly used in robotics.

2.4.1 Discretization in Time

The most common way of encoding a trajectory is to discretize it along a finite number of time steps, *i.e.* $\{\boldsymbol{q}_k\}_{k=0}^{K}$. The discretization is typically tightly linked to the discretized model of the system dynamics, as shown in (2.11). The underlying assumption is that the state of the system does not change significantly within a time step. For this assumption to be valid, the time step has to be chosen small enough such that the dynamics of the system are well approximated by the discretized model. However, a higher temporal resolution results in a higher number of discretized points and thus in a higher-dimensional optimization problem. This trade-off between temporal resolution and dimensionality of the decision variable is a key aspect of setting up direct transcription formulations for trajectory optimization. Especially if the computation time for the optimization is limited, this becomes an unfortunate trade-off, which oftentimes requires compromise on the temporal resolution or the horizon length.

2.4.2 Superposition of Basis Functions

Discretizing the trajectory in time provides a generic tool for making trajectory optimization problems tractable. However, given the trade-off mentioned above, it might be desirable to decouple the number of decision variables from the temporal resolution with which the dynamics of the system are enforced. One way of achieving this is to represent the trajectory as a superposition of basis functions, *i.e.*

$$\boldsymbol{q}(t) = \sum_{n=1}^{N} \boldsymbol{\phi}_n(t) w_n, \tag{2.12}$$

where $\boldsymbol{\phi}_n(t)$ is the n-th basis function and w_n is the weight of the corresponding basis function. The weights are used as the decision variables for optimizing the trajectory, such that the number of basis functions N directly corresponds to the dimensionality of the optimization problem. Due to the superposition of continuous functions in time, the trajectory is continuous in time as well. This enables evaluating constraints on the trajectory at arbitrary temporal resolutions. However, this decoupling entails that we limit the expressiveness of the trajectory to the span of the basis functions. Thus, for underactuated systems, such as robots manipulating objects, we cannot use a superposition of basis functions to represent the trajectory of the full system.

However, we may still use the benefits of decoupling the number of decision variables from the temporal resolution by using shooting methods to optimize for the system's control trajectory as described in Section 2.3.2. For this, we can use the superposition of basis functions to represent the control trajectory, *i.e.* the control input of the system over time. The dynamics of an underactuated system are then used to compute the state trajectory, which enforces the dynamics without posing additional constraints on the control trajectory. In Section 2.1, we discussed the role of stiffness control as a low-level control layer that abstracts the motion and force response of the closed-loop robot into an auxiliary control variable, the set point of the stiffness controller. Representing the set point of the closed-loop robot using basis functions, *i.e.*

$$\boldsymbol{q}_{\text{des}}(t) = \sum_{n=1}^{N} \boldsymbol{\phi}_n(t) w_n, \tag{2.13}$$

corresponds to a shooting method that can be used to optimize through contacts. This concept is used throughout this thesis to plan for non-prehensile manipulation.

2.5 Zero-order Optimization

In Section 1.1, we have discussed the challenges of optimizing through contacts due to the discontinuous nature of contact dynamics. Zero-order optimization techniques provide an aspect of exploration that can help to overcome the pitfalls associated with

discontinuous contact dynamics and in general with local minima. The general idea of zero-order optimization techniques is to generate candidate solutions by sampling from a probabilistic approximation of the local cost landscape. By evaluating the objective for each candidate solution, a new approximation of the local landscape can be computed. Apart from the exploration aspect, sampling-based optimization techniques can benefit heavily from parallelizing the evaluation of sampled candidate solutions. This make zero-order optimization techniques suitable for model-predictive control loops. The following provides a non-exhaustive discussion on zero-order optimization techniques that are commonly used in robotics.

2.5.1 Cross-Entropy Method (CEM)

Different variations of the cross-entropy method have been used for offline trajectory optimization for robotics (Kalakrishnan et al. (2011); Kobilarov (2012)). In addition, these ideas have been extended to online optimization in model-predictive control loops, *e.g.* model-predictive path integral control (MPPI) introduced by Williams et al. (2017). CEM approaches have in common that the cost function - defining the desired behavior of the system - is translated into a probability distribution using an exponential transformation, *e.g.*

$$p(\boldsymbol{u}^r_{0:K}) \propto \exp\left(-J(\boldsymbol{u}^r_{0:K})\right), \tag{2.14}$$

Evaluated samples are then used to update the parameters of the approximated probability distribution with the goal of minimizing the cross-entropy between the cost-induced distribution and the approximated distribution. Most approaches based on cross-entropy methods do not update the covariance of the approximated distribution but introduce a temperature parameter for tuning the exploration of the search space. This is because the number of samples is typically too low compared to the dimensionality of the optimization variable, which makes it challenging to robustly estimate the covariance of the evaluated samples. This leads to optimization approaches with a constant exploration rate, which may be too low for discovering vicinities of desired local minima, or too high for exploiting the local minima.

2.5.2 Covariance Matrix Adaptation - Evolution Strategy (CMA-ES)

A zero-order optimization approach that does take covariance updates into account is the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES, Hansen (2016)). Compared to CEM, CMA-ES is not widely adopted for trajectory optimization. While CMA-ES is based on the same principle as CEM of updating a probability distribution to approximate the cost-induced distribution, CMA-ES introduces an improved update step that also includes the covariance of the distribution taking information from previous candidate populations into account. In Chapter 4, we will see that

the combination of low-dimensional trajectory representations (cf. Section 2.4) and CMA-ES can be used for optimizing trajectories in realtime.

2.6 Belief-Space Control

As discussed in Section 1.1.2, contact-rich manipulation is prone to be subject to uncertainty in the dynamics of the underactuated system which can be taken into account by using a stochastic model of the contact dynamics. Rolling out a control trajectory using a stochastic model yields a probability distribution over the system's state, *i.e.* the belief. Belief-space control aims to not only control the mean trajectory of the system, but also to control higher-order moments of the system state distribution, for instance the covariance. Since the predicted state of the system becomes a random variable, control objectives are thus defined by stochastic measures of the belief, *e.g.* the expected value of the control error at the end of the trajectory:

$$J = \mathrm{E}\left[\left\|\boldsymbol{x}_K^o - \boldsymbol{x}_{\mathrm{des}}^o\right\|^2\right]. \tag{2.15}$$

For systems with linear-Gaussian dynamics and observation models, the solution of an unconstrained, quadratic optimal problem can be computed analytically, known as the Linear Quadratic Gaussian (LQG) controller (Atrom (1971)). In the following, we discuss concepts for belief-space control problems with different objectives/constraints and for non-linear systems.

2.6.1 Covariance Steering Approach

Goldshtein and Tsiotras (2017) have shown that the control of the belief can be separated into a mean-steering and a covariance-steering control problem if the belief is Gaussian. This formulation generalizes the LQG controller to handle costs and constraints on the covariance of the state distribution beyond minimizing the expectation of the control error (Okamoto et al. (2018); Zheng et al. (2022)). However, analytical solutions still only exist for linear-Gaussian systems.

2.6.2 Trajectory Optimization for Non-Gaussian Belief Spaces

For non-Gaussian belief spaces, belief-space control can be framed as a trajectory optimization problem (Platt et al. (2010)) in order to find locally optimal solutions. However, the tractablity of the optimization problem depends on the complexity of the belief model and the respective belief representation. Instead of constraining the belief space to Gaussian distributions, Platt et al. (2010); Nikandrova et al. (2014) use a non-parametric particle-based representation to track the belief, which is more suitable for the discontinuous dynamics of contact-rich manipulation but has been limited

to 2D uncertainty. Platt et al. (2017) project the belief space onto a set of competing hypotheses in the underlying state space. The control problem is then formulated by finding an action sequence that will generate observations distinguishing between the competing hypotheses. This can be solved via standard solvers for trajectory optimization, such as sequential quadratic programming (SQP) or sampling-based methods, like probabilistic roadmap planners (PRM).

While the general assumption in belief-space control is that observations gathered upon execution of the control trajectory reduce the uncertainty in the system belief, in the case of non-linear dynamics, the system dynamics can also reduce the uncertainty in the system belief. More specifically, the non-linear dynamics allow us to consider particular modes of the dynamics that result in a decrease in uncertainty even without any observations. This is particularly useful for contact-rich manipulation where the dynamics of the system are discontinuous and non-linear. In Chapter 6, we will see how belief-space control can be used to generate robust plans for contact-rich manipulation by exploiting this insight.

3 From Optimal Control to Time-Parameterized Basis Functions

Publication Note

The material presented in this chapter is adapted from the following publication:

- Jankowski, J., Racca, M., and Calinon, S. (2022). From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control. *IEEE Robotics and Automation Letters*, 7(2):3242–3249

M. Racca helped in the conception and execution of the user study and writing the paper.

Supplementary Material

Video related to this chapter is available at: https://youtu.be/bqWNzkSiFl0.

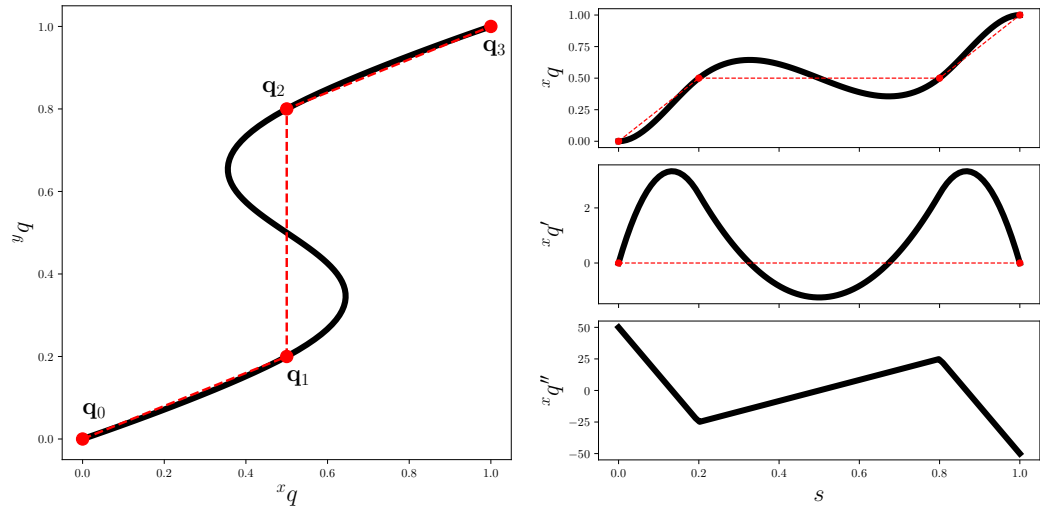
Optimal control can be used as an implicit way of designing and parameterizing trajectories with desirable properties such as smoothness and satisfying boundary constraints. However, the tractability of the optimal control problem depends on the dimensionality of the state space, as well as the temporal resolution and horizon of the trajectory. Therefore, in this chapter, we derive how to leverage optimal control in order to compute minimal-effort trajectories that are parameterized by a low-dimensional set of key positions acting as constraints on the trajectory for fully actuated systems. The key positions, which will be the solution to the optimal control problem, can be interpreted as parameters of the trajectory. We show that the optimal trajectory, synthesized from the key positions, and its derivatives are linear in the key positions. This allows us to reformulate the optimal trajectory as a linear combination of optimal basis functions that are computed once and that can be reused when changing the key positions. This not only enables highly efficient process of synthesizing trajectories, but also allows us to map Gaussian distributions over single key positions to Gaussian distributions over optimal trajectories. While in later chapters we investigate how these properties can be used for efficient exploration of making and breaking contacts, this chapter validates the effectiveness of optimal basis functions in a user study. We show that demonstrating a few key points to a dynamic pick-and-place task is sufficient for a non-expert to teach the robot within less than a minute.

In robotics, planning and control is oftentimes concerned with optimizing or generating robot state trajectories that achieve a desired robot behavior. As discussed in Section 2.4, trajectories can be represented in various ways depending on the technique used to compute them. When planning for more complex robot behavior such as contact-rich manipulation, the optimization of robot trajectories typically becomes a non-convex problem. At the same time, robot trajectories are still subject to constraints that are related to the kinodynamic limits of the robot, such as velocity limits, and acceleration limits. In addition, trajectories are typically subject to boundary constraints acting on the first and last time step, *e.g.* zero velocities. To facilitate planning or learning for more complex behavior, it is beneficial to search and learn in low-dimensional spaces. The low-dimensional solutions can then be mapped to continuous timing-optimal trajectories while ensuring that the above task-agnostic constraints are satisfied.

In the following, we first derive how optimal control can be used as a tool for parameterizing trajectories with only a few key positions as constraints on the trajectory, and show that the optimal trajectory is a superposition of optimal basis functions that pass that pass through the key positions in Section 3.1. In Section 3.2, we then develop algorithms for directly optimizing the duration of the trajectory and the timing of key positions to enforce dynamic constraints such as joint velocity and acceleration limits. Last, in Section 3.3, we show that the efficiency of representing trajectories with just a few key positions enables learning from demonstrations of dynamic robot behavior with less than ten demonstrations.

3.1 Optimal Basis Functions

In this section, we derive optimal basis functions from the analytical solution to a minimal-effort trajectory problem. The key positions are imposed as linear constraints on the trajectory and thus parameterize the optimal trajectory. Optimal basis functions are closely related to B-splines that are parameterized by so called control points. Both sets of basis functions are composed of piecewise polynomial functions that are superposed to form a continuous trajectory. In contrast to B-splines, optimal basis functions explicitly expose the boundary velocities of the trajectory as parameters. Furthermore, the optimal basis functions are derived from the analytical solution to an optimal control problem and are thus optimal in the sense of minimizing the effort of the trajectory. The main difference to B-splines, however, is that the key positions are not control points that pull the trajectory towards them, but are constraints that the trajectory has to pass through. Figure 3.1 illustrates the optimal trajectory for a two-degrees-of-freedom system with three key positions. Optimal basis functions allow us to compute this trajectory efficiently as a linear mapping.



(a) Optimal path.

(b) Optimal position, velocity and acceleration profiles in ${}^{x}q$-direction.

Figure 3.1: Example of a minimal-effort trajectory that passes through key positions. The trajectory is furthermore constrained in position and velocity at the first and last time step. The acceleration is constrained to be continuous on the interval $s \in [0, 1]$.

3.1.1 Minimal-Effort Trajectories with Linear Constraints

We first consider the problem of generating a trajectory that passes through a set of key positions with minimal control effort. The trajectory is denoted as a function that maps a continuous phase variable $s \in [0, 1]$ to a position $\boldsymbol{q} \in \mathbb{R}^{n_{\text{dof}}}$ of a robot that has n_{dof} degrees of freedom. The phase variable is a linear function of time t with $s = {}^{t}/_{T}$, where T is the total duration of the trajectory, which will be discussed in more detail in the subsequent section. In the following, the control effort of a respective trajectory is defined as a function of its second derivative with respect to the phase, *i.e.* acceleration. The trajectory is constrained to pass through a set of N key positions $\boldsymbol{q}_n$ at given phase values s_n, *i.e.* $\boldsymbol{q}_n = \boldsymbol{q}(s_n)$. The problem of generating a trajectory of smoothness class $\mathcal{C}^2$ that passes through a set of key positions with minimal effort can be formulated as an optimal control problem, *i.e.*

$$\boldsymbol{q}^*(s) = \underset{\boldsymbol{q}(s)}{\arg\min} \quad \int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds, \tag{3.1}$$

$$\text{s.t.} \quad \boldsymbol{q}(0) = \boldsymbol{q}_0, \quad \boldsymbol{q}(s_n) = \boldsymbol{q}_n, \tag{3.2}$$

$$\boldsymbol{q}'(0) = \boldsymbol{q}'_0, \quad \boldsymbol{q}'(1) = \boldsymbol{q}'_N, \tag{3.3}$$

$$0 < s_n < s_{n+1}, \quad n = 1, ..., N-1, \quad s_N = 1. \tag{3.4}$$

The solution to the optimization problem in (3.1)-(3.4) is a continuous trajectory $\boldsymbol{q}^*(s)$. Zhang et al. (1997) have shown that the resulting trajectory is a polynomial spline in

the phase space. The spline is computed by decomposing the optimization problem in (3.1)-(3.4) into multiple more simple optimization problems that are linked through variable boundary conditions. Note that the trajectories of the individual degrees of freedom can be optimized independently since the objective, as well as the constraints, are separable. Thus, in the following, we derive the analytical solution for a trajectory with a single degree of freedom.

Zhang et al. (1997) show that for a linear system xy, with system matrices

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{3.5}$$

the acceleration

$$\ddot{q}(t) = \boldsymbol{b}^\top e^{-\boldsymbol{A}^\top t} \left(\int_0^T e^{-\boldsymbol{A}\tau} \boldsymbol{b}\boldsymbol{b}^\top e^{-\boldsymbol{A}^\top \tau} d\tau \right)^{-1} \left(e^{-\boldsymbol{A}T} \begin{pmatrix} q_T \\ \dot{q}_T \end{pmatrix} - \begin{pmatrix} q_0 \\ \dot{q}_0 \end{pmatrix} \right) \tag{3.6}$$

minimizes the functional $J = \int_0^T \ddot{q}^2(\tau) d\tau$ among all controls that move a linear system***, *i.e.*

$$\begin{pmatrix} \dot{q} \\ \ddot{q} \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} q \\ \dot{q} \end{pmatrix} + \boldsymbol{b}\ddot{q}, \tag{3.7}$$

with system matrices

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{3.8}$$

from $q_0, \dot{q}_0$ to $q_T, \dot{q}_T$ within T seconds. The acceleration in (3.6) can be used to drive the system from key point q_n to the subsequent key point q_{n+1}, defined as a segment n with $s \in [s_n, s_{n+1}]$, with minimal effort. As the acceleration is a function of the boundary conditions for each segment n, the optimal trajectory can only be computed if the velocities at the intermediate key positions, *i.e.* $\boldsymbol{v} = \left(q'_1, q'_2, \ldots, q'_{N-1}\right)^\top$, are known. With the optimal acceleration in (3.6) being linear in $\boldsymbol{v}$, the velocities can be computed by solving a linear system of equations that enforces continuity of the acceleration at the individual key positions. This means that the resulting trajectory is of smoothness class $\mathcal{C}^2$. We concatenate the given parameters, comprising the boundary variables, as well as the key positions, into a single vector $\boldsymbol{w} = (q_0, q_1, \ldots, q_N, q'_0, q'_N)^\top \in \mathbb{R}^{N+3}$, where N denotes the number of key positions. Imposing the $\mathcal{C}^2$ constraint at the key positions, we obtain a linear system of equations, *i.e.* $\boldsymbol{P_v v} = \boldsymbol{P_w w}$, with $\boldsymbol{P_v} \in \mathbb{R}^{N-1 \times N-1}$ and $\boldsymbol{P_w} \in \mathbb{R}^{N-1 \times N+3}$, that can be solved for $\boldsymbol{v} = \boldsymbol{P_v}^{-1} \boldsymbol{P_w w}$. Given the solution for $\boldsymbol{v}$, the optimal acceleration for each segment n can be computed via (3.6). Note that the optimal acceleration in (3.6) is linear in both phase and time space since

$$e^{-\boldsymbol{A}^\top t} = \sum_{k=0}^{\infty} \frac{(-t)^k}{k!} \boldsymbol{A}^k = \begin{pmatrix} 1 & 0 \\ -t & 1 \end{pmatrix}. \tag{3.9}$$

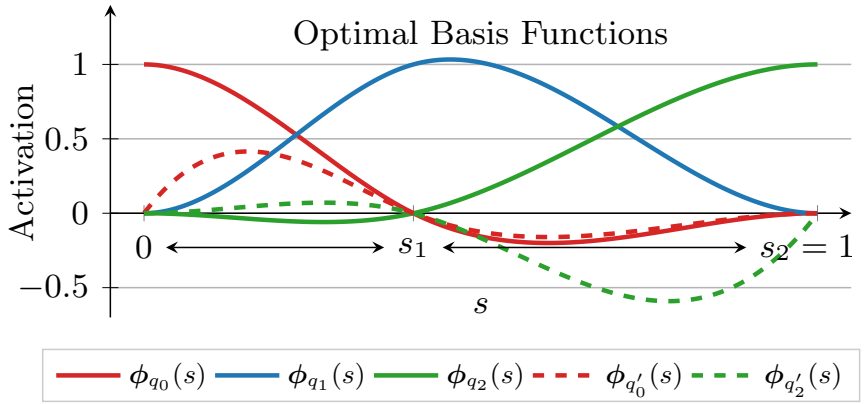


Figure 3.2: Basis functions constructed from the optimal control problem in (3.1) for a single DoF and $N = 2$ key positions.

Accordingly, the optimal velocity and position can be computed by integrating the acceleration, *i.e.* $q'(s) = \int_0^s q''(\tau)d\tau$ and $q(s) = \int_0^s q'(\tau)d\tau$. Figure 3.1 illustrates the optimization problem for a system with two degrees of freedom and three key positions. The resulting trajectory can be interpreted as a function that is not only parameterized by the phase variable but also by the constraints, *i.e.* the set of N key positions $\boldsymbol{q}_n$ and boundary velocities $\boldsymbol{q}'_0$ and $\boldsymbol{q}'_N$, that are encapsulated in the vector $\boldsymbol{w}$.

In robotics, optimal control problems are typically subject to non-linear dynamics, such as contact dynamics, as well as non-linear constraints, such as avoiding collisions. Nevertheless, even these more complex problems encapsulate the minimal-effort trajectory problem with boundary conditions as a sub-problem. Thus, these problems can be formulated as an optimization problem of the constraint parameters that parameterize the optimal trajectory. This entails the following questions: What if we change the key positions which are used as constraints in the optimization problem in (3.1)-(3.4)? Can we exploit the mathematical structure of the analytical solution to this problem to efficiently compute a new optimal trajectory from a new set of key positions? In the following, we show that the optimal trajectory is a linear function of the key positions, which allows us to compute optimal basis functions that can be reused when changing the key positions.

3.1.2 Optimal Trajectory as a Linear Mapping of Constraints

Inserting the optimal velocities $\boldsymbol{v}$, derived in the above, back into (3.6), we obtain the optimal acceleration for segment n as a linear function of the given parameter vector

$\boldsymbol{w}$ with

$$\begin{aligned} q_n''(s) &= \boldsymbol{b}^\top e^{-\boldsymbol{A}^\top(s-s_n)} \left(\int_{s_n}^{s_{n+1}} e^{-\boldsymbol{A}\tau} \boldsymbol{b}\boldsymbol{b}^\top e^{-\boldsymbol{A}^\top\tau} d\tau \right)^{-1} \left(e^{-\boldsymbol{A}(s_{n+1}-s_n)} \begin{pmatrix} q_{n+1} \\ q'_{n+1} \end{pmatrix} - \begin{pmatrix} q_n \\ q'_n \end{pmatrix} \right) \\ &= \boldsymbol{b}^\top e^{-\boldsymbol{A}^\top(s-s_n)} \left({}^{\boldsymbol{w}}\boldsymbol{L}_n \boldsymbol{w} + {}^{\boldsymbol{v}}\boldsymbol{L}_n \boldsymbol{v} \right) \\ &= \boldsymbol{b}^\top e^{-\boldsymbol{A}^\top(s-s_n)} \left({}^{\boldsymbol{w}}\boldsymbol{L}_n + {}^{\boldsymbol{v}}\boldsymbol{L}_n \boldsymbol{P}_{\boldsymbol{v}}^{-1} \boldsymbol{P}_{\boldsymbol{w}} \right) \boldsymbol{w}, \end{aligned} \tag{3.10}$$

with ${}^{\boldsymbol{w}}\boldsymbol{L}_n \in \mathbb{R}^{2\times N+3}$ and ${}^{\boldsymbol{v}}\boldsymbol{L}_n \in \mathbb{R}^{2\times N-1}$. As a result, the optimal acceleration for the entire interval $s \in [0,1]$ can be expressed as a scalar product of two vectors, *i.e.*

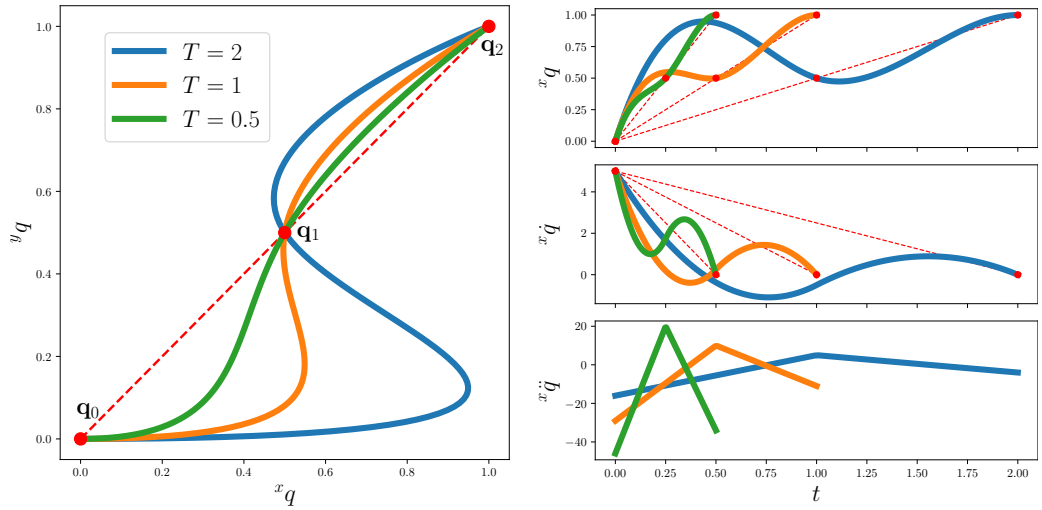
$$q''(s) = \boldsymbol{\phi}''(s)\boldsymbol{w}, \tag{3.11}$$

where only the vector $\boldsymbol{\phi}''(s)$ depends on the phase variable s, while the parameter vector $\boldsymbol{w}$ is independent of s. The optimal trajectory is thus also a linear function of $\boldsymbol{w}$ with

$$q(s) = \boldsymbol{\phi}(s)\boldsymbol{w}. \tag{3.12}$$

This mathematical structure allows us to interpret the optimal trajectory in (3.12) as a linear combination of optimal basis functions $\boldsymbol{\phi}(s)$ that are weighted by the given parameter vector $\boldsymbol{w}$, *i.e.* the key positions and the boundary velocities. When changing a key position in (3.1)-(3.4), the optimal trajectory can be computed without recomputing its basis functions $\boldsymbol{\phi}(s)$. Figure 3.2 illustrates the optimal basis functions for a single degree of freedom system with two key positions at $s_1 = 0.4$ and $s_2 = 1$.

For systems with n_{dof} degrees of freedom such that $\boldsymbol{w} = \left(\boldsymbol{q}_0^\top, \ldots, \boldsymbol{q}_N^\top, \boldsymbol{q}_0'^\top, \boldsymbol{q}_N'^\top \right)^\top \in \mathbb{R}^{(N+3)n_{\text{dof}}}$, the optimal trajectory is given by $\boldsymbol{q}(s) = \boldsymbol{\Phi}(s)\boldsymbol{w}$, where the basis function matrix is given via Kronecker product, *i.e.* $\boldsymbol{\Phi}(s) = \boldsymbol{\phi}(s) \otimes \boldsymbol{I}_{n_{\text{dof}}}$, with $\boldsymbol{I}_{n_{\text{dof}}} \in \mathbb{R}^{n_{\text{dof}} \times n_{\text{dof}}}$ being the identity matrix.



(a) Optimal path.

(b) Optimal position, velocity and acceleration profiles in ${}^{x}q$-direction.

Figure 3.3: Illustration of the impact of scaling the trajectory duration on the optimal path (a) and the trajectory over time (b). The system has an initial velocity of $\dot{\boldsymbol{q}} = (5, 0)^\top$.

3.2 Optimal Time Parameterization

In Section 3.1, we derived the optimal trajectory in (3.12) as a function of the phase variable $s \in [0, 1]$. As introduced above, we define the phase variable as a linear function of time t, *i.e.* $s = {}^{t}/_{T}$, where T is the total duration of the trajectory. Assuming that the duration is given, the optimal trajectory can be parameterized in time as

$$\boldsymbol{q}(t) = \boldsymbol{\Phi}\left(\frac{t}{T}\right)\boldsymbol{w}. \tag{3.13}$$

Accordingly, the optimal velocity and acceleration with respect to time are given by

$$\dot{\boldsymbol{q}}(t) = \frac{\partial \boldsymbol{q}(t)}{\partial t} = \frac{1}{T}\boldsymbol{\Phi}'\left(\frac{t}{T}\right)\boldsymbol{w}, \tag{3.14}$$

$$\ddot{\boldsymbol{q}}(t) = \frac{\partial^2 \boldsymbol{q}(t)}{\partial t^2} = \frac{1}{T^2}\boldsymbol{\Phi}''\left(\frac{t}{T}\right)\boldsymbol{w}. \tag{3.15}$$

Note that the optimal trajectory in (3.13) suggests that the optimal path does not change when the duration of the trajectory is changed. However, the boundary velocities in $\boldsymbol{w}$, *i.e.* $\dot{\boldsymbol{q}}_0, \dot{\boldsymbol{q}}_N$, scale with the duration of the trajectory T and not with respect to the phase variable s, since we are interested in constraining the velocity with respect to time. Thus, the parameters of the optimal trajectory are given as $\boldsymbol{w} = (\boldsymbol{q}_0^\top, \ldots, \boldsymbol{q}_N^\top, T\dot{\boldsymbol{q}}_0{}^\top, T\dot{\boldsymbol{q}}_N{}^\top)^\top$.

3.2.1 Direct Minimization of the Trajectory Duration

Given the ability to efficiently recompute the optimal trajectory for a new set of key positions and a trajectory duration parameter T, the solution to the optimization problem in (3.1)-(3.4) is a unique mapping from inputs describing spatial and temporal aspects to an optimal trajectory. The optimal velocity profile scales linearly with the inverse of the duration T, while the optimal acceleration profile scales quadratically with the inverse of the duration T. Hence, in this section, we aim to find the minimal duration T such that the resulting velocity and acceleration profiles do not exceed user-defined limits. Note that this section assumes the key positions, as well as the boundary velocities $\dot{\boldsymbol{q}}_0, \dot{\boldsymbol{q}}_N$ to be fixed and known. The optimization problem is thus given by:

$$\begin{aligned} T_{\text{opt}} = \min \quad & T, \\ \text{s.t.} \quad & T > 0, \\ & \dot{\boldsymbol{q}}(t) = \frac{1}{T}\boldsymbol{\Phi}'\left(\frac{t}{T}\right)\boldsymbol{w} \in [-\dot{\boldsymbol{q}}_{\max}, \dot{\boldsymbol{q}}_{\max}], \quad \forall t \in [0, T], \\ & \ddot{\boldsymbol{q}}(t) = \frac{1}{T^2}\boldsymbol{\Phi}''\left(\frac{t}{T}\right)\boldsymbol{w} \in [-\ddot{\boldsymbol{q}}_{\max}, \ddot{\boldsymbol{q}}_{\max}], \quad \forall t \in [0, T]. \end{aligned} \tag{3.16}$$

The optimization problem in (3.16) is a non-convex optimization problem due to the non-linear constraints on the velocity and acceleration profiles. However, we can exploit the mathematical properties of the optimal velocity and acceleration profiles to derive an algorithm for computing the minimal trajectory duration in a finite amount of operations (direct optimization). For this, we make use of the observation that either the velocity or the acceleration reaches the maximum value at least one point in time if the trajectory duration is minimal. If the velocity and the acceleration do not reach their maximum values, the system could move *faster*, resulting in a shorter trajectory duration.

Pointwise Maximization of Velocity

The optimal velocity at phase s is given by (3.14) with $t = Ts$. However, note that the trajectory parameter $\boldsymbol{w}$ depends on the duration of the trajectory, since

$$\boldsymbol{w} = \begin{pmatrix} \boldsymbol{q}_0 \\ \vdots \\ \boldsymbol{q}_N \\ T\dot{\boldsymbol{q}}_0 \\ T\dot{\boldsymbol{q}}_N. \end{pmatrix} \tag{3.17}$$

Thus, the velocity at phase s can be rewritten by separating the basis function matrix $\boldsymbol{\Phi}(s)$ into a part for the key positions, *i.e.* $\boldsymbol{\Phi}_q(s)$, and a part for the boundary velocities,

i.e. $\boldsymbol{\Phi}_{\dot{q}}(s)$, such that

$$\begin{aligned}\dot{\boldsymbol{q}}(s) &= \frac{1}{T}\boldsymbol{\Phi}'(s)\,\boldsymbol{w} = \frac{1}{T}\left(\boldsymbol{\Phi}'_q(s)\,\boldsymbol{w_q} + \boldsymbol{\Phi}'_{\dot{q}}(s)\,T\boldsymbol{w}_{\dot{\boldsymbol{q}}}\right)\\ &= \frac{1}{T}\boldsymbol{\Phi}'_q(s)\,\boldsymbol{w_q} + \boldsymbol{\Phi}'_{\dot{q}}(s)\,\boldsymbol{w}_{\dot{\boldsymbol{q}}},\end{aligned} \tag{3.18}$$

with $\boldsymbol{w_q} = (\boldsymbol{q}_0^\top, ..., \boldsymbol{q}_N^\top)^\top$ and $\boldsymbol{w}_{\dot{\boldsymbol{q}}} = (\dot{\boldsymbol{q}}_0^\top, \dot{\boldsymbol{q}}_N^\top)^\top$. In the limit case of $T \to \infty$, the velocity profile is governed by the transition between the boundary velocities, *i.e.*

$$\lim_{T\to\infty} \dot{\boldsymbol{q}}(s) = \boldsymbol{\Phi}'_{\dot{q}}(s)\,\boldsymbol{w}_{\dot{\boldsymbol{q}}}. \tag{3.19}$$

The transition velocities between the boundary velocities are bound by the boundary velocities themselves, such that the following element-wise inequality holds:

$$\min(\dot{\boldsymbol{q}}_0, \dot{\boldsymbol{q}}_N) \leq \lim_{T\to\infty} \dot{\boldsymbol{q}}(s) \leq \max(\dot{\boldsymbol{q}}_0, \dot{\boldsymbol{q}}_N), \quad \forall s \in [0,1]. \tag{3.20}$$

Thus, we can conclude that there exists a trajectory duration T such that the velocity limits are satisfied *iff* the boundary velocities are within the velocity limits. Next, we determine the duration such that the velocity limits are not only satisfied but also exploited such that there is at least one degree of freedom i with ${}^i\dot{q}(s) = {}^i\dot{q}_{\max}$ for a given $s \in [0,1]$. The corresponding duration $T_{\text{opt},{}^i\dot{q}}(s)$ for the i-th degree of freedom is given by

$$T_{\text{opt},{}^i\dot{q}}(s) = \frac{\boldsymbol{\phi}'_q(s)\,{}^i\boldsymbol{w_q}}{\boldsymbol{\phi}'_{\dot{q}}(s)\,{}^i\boldsymbol{w}_{\dot{\boldsymbol{q}}} \pm {}^i\dot{q}_{\max}}. \tag{3.21}$$

Note that the computation in (3.21) has two results, one for the upper and one for the lower limit of the velocity. Since one of the two results is negative, and the other result is positive, we discard the negative result and use the positive duration. Due to the monotonic relationship between the velocity and the duration, the pointwise optimal duration in (3.21) serves as a lower bound for the optimal duration T_{opt} for the entire interval in (3.16).

Given the pointwise optimal duration with respect to velocity limits, the optimal duration $T_{\text{opt},\dot{\boldsymbol{q}}}$ can be computed by selecting the maximum of the pointwise optimal durations for all phases $s \in [0,1]$ and among all degrees of freedom. This involves solving a non-linear optimization problem in s to determine the maximum of (3.21). However, since the phase space is one-dimensional and bounded between zero and one, the optimal duration $T_{\text{opt},\dot{\boldsymbol{q}}}$ can be approximated efficiently by discretizing the phase space and evaluating the pointwise optimal duration for each phase.

Pointwise Maximization of Acceleration

Analog to maximizing the velocity for a given phase, acceleration limits can be used to determine the optimal duration as well. For this, the acceleration at phase s is given by

$$\begin{aligned}\ddot{\boldsymbol{q}}(s) &= \frac{1}{T^2}\boldsymbol{\Phi}''(s)\,\boldsymbol{w} = \frac{1}{T^2}\left(\boldsymbol{\Phi}''_q(s)\,\boldsymbol{w_q} + \boldsymbol{\Phi}''_{\dot{q}}(s)\,T\boldsymbol{w_{\dot{q}}}\right)\\ &= \frac{1}{T^2}\boldsymbol{\Phi}''_q(s)\,\boldsymbol{w_q} + \frac{1}{T}\boldsymbol{\Phi}''_{\dot{q}}(s)\,\boldsymbol{w_{\dot{q}}}.\end{aligned} \tag{3.22}$$

In the limit case of $T \to \infty$, the acceleration profile is equal to zero:

$$\lim_{T\to\infty} \ddot{\boldsymbol{q}}(s) = \boldsymbol{0}. \tag{3.23}$$

Thus, there exists a trajectory duration T such that the acceleration limits are satisfied. Similarly to determining the pointwise optimal duration with respect to the velocity limits in (3.21), the pointwise optimal duration with respect to the acceleration limits can be computed by solving for T such that the resulting acceleration is equal to either the lower or the upper acceleration limit. For this, the following quadratic function has to be solved for the corresponding duration $T_{\text{opt},^i\ddot{q}}(s)$ for the i-th degree of freedom:

$$\pm {}^i\ddot{q}_{\max} T_{\text{opt},^i\ddot{q}}(s)^2 - \boldsymbol{\phi}''_{\dot{q}}(s)\,{}^i\boldsymbol{w_{\dot{q}}} T_{\text{opt},^i\ddot{q}}(s) - \boldsymbol{\phi}''_q(s)\,{}^i\boldsymbol{w_q} = 0. \tag{3.24}$$

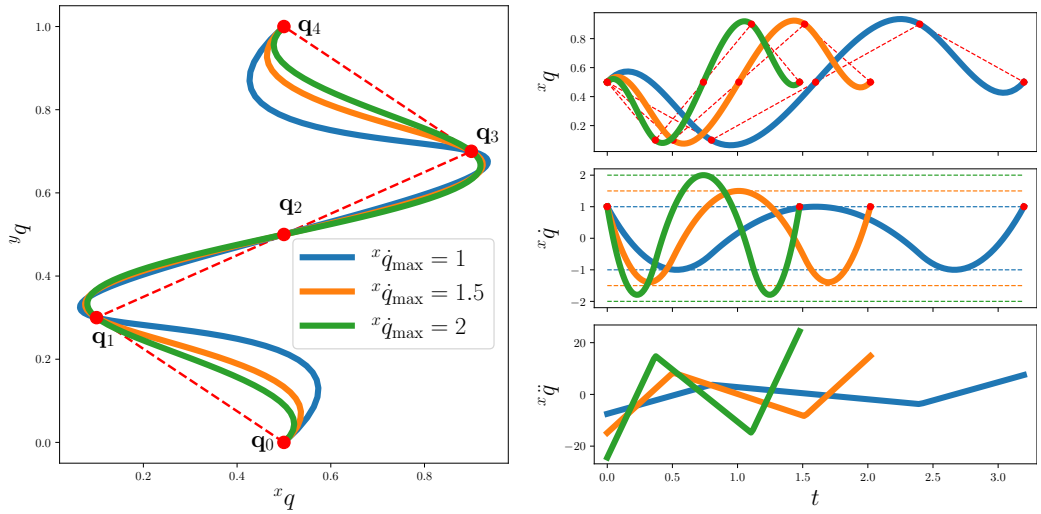
The lower and the upper acceleration limit yield two results each, such that

$${}^u T_{\text{opt},^i\ddot{q}}(s) = \frac{1}{2\,{}^i\ddot{q}_{\max}}\left(-\boldsymbol{\phi}''_{\dot{q}}(s)\,{}^i\boldsymbol{w_{\dot{q}}} \pm \sqrt{\left(\boldsymbol{\phi}''_{\dot{q}}(s)\,{}^i\boldsymbol{w_{\dot{q}}}\right)^2 - 4\boldsymbol{\phi}''_q(s)\,{}^i\boldsymbol{w_q}\,{}^i\ddot{q}_{\max}}\right), \tag{3.25}$$

$${}^l T_{\text{opt},^i\ddot{q}}(s) = \frac{1}{2\,{}^i\ddot{q}_{\max}}\left(\boldsymbol{\phi}''_{\dot{q}}(s)\,{}^i\boldsymbol{w_{\dot{q}}} \pm \sqrt{\left(\boldsymbol{\phi}''_{\dot{q}}(s)\,{}^i\boldsymbol{w_{\dot{q}}}\right)^2 + 4\boldsymbol{\phi}''_q(s)\,{}^i\boldsymbol{w_q}\,{}^i\ddot{q}_{\max}}\right). \tag{3.26}$$

Each result may be negative, positive, or complex. We discard the negative and complex results and use the highest positive duration, which serves as a lower bound for the optimal duration T_{opt} in (3.16).

Given the pointwise optimal duration with respect to acceleration limits, the optimal duration $T_{\text{opt},\ddot{\boldsymbol{q}}}$ can be computed by selecting the maximum of the pointwise optimal durations for all phases $s \in [0,1]$ and among all degrees of freedom. However, since the acceleration profile is a linear spline in phase, the highest acceleration can only appear at the boundary of two phases, *i.e.* $s = s_n$. Thus, it is sufficient to evaluate the pointwise optimal duration at the $N+1$ phase boundaries to exactly determine the optimal duration $T_{\text{opt},\ddot{\boldsymbol{q}}} = \max_{n,i} T_{\text{opt},^i\ddot{q}}(s_n)$.



(a) Optimal path.

(b) Optimal position, velocity and acceleration profiles in ${}^x q$-direction.

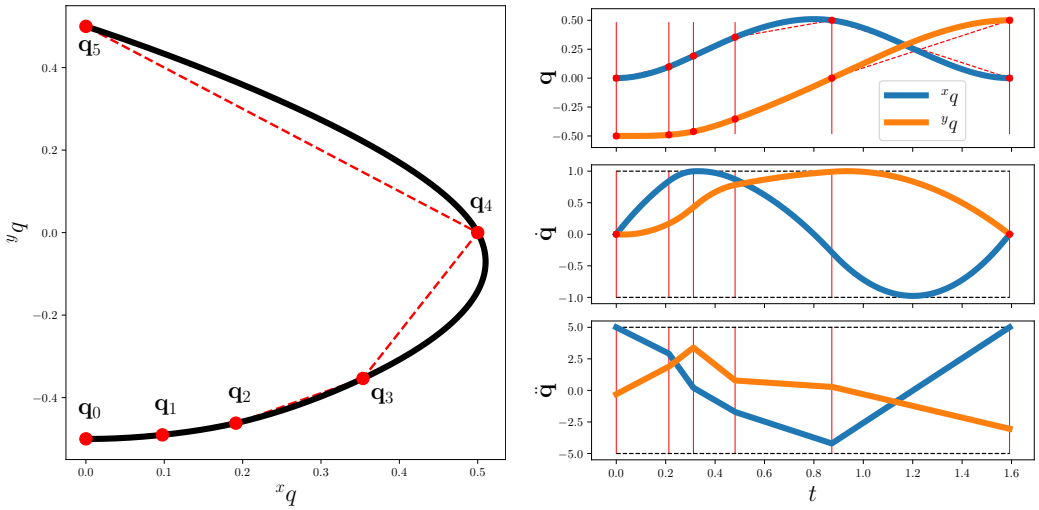
Figure 3.4: Illustration of the timing-optimal path (a) and the timing-optimal trajectory (b) for different velocity limits in ${}^x q$-direction. The trajectory duration is directly optimized for and used to scale the basis functions accordingly.

Maximum of Lower Bounds

The optimal duration T_{opt} is given by the maximum of the lower bounds for the optimal durations with respect to the velocity and acceleration limits, *i.e.* $T_{\text{opt}} = \max(T_{\text{opt},\dot{\boldsymbol{q}}}, T_{\text{opt},\ddot{\boldsymbol{q}}})$. Assuming that errors introduced by discretizing the phase space for computing $T_{\text{opt},\dot{\boldsymbol{q}}}$ are negligible, the optimal duration T_{opt} is guaranteed to be the global minimizer of the optimization problem in (3.16). Figure 3.4 illustrates timing-optimal trajectories for different velocity limits. It can be seen that the velocity profiles are scaled such that velocity limits are maximally exploited at one or multiple points in time. Due to the analytical result, the implementation of the optimization problem is efficient and can be solved in batch form for multiple trajectory parameters. Furthermore, the complexity of the computation is $O(n_{\text{dof}})$, where n_{dof} is the number of degrees of freedom. In later chapters, this analytical computation of the optimal duration is used within model-predictive control problems to uniquely map a set of key positions not only to a trajectory but also to a minimal duration of the trajectory.

3.2.2 Iterative Optimization of the Timing of Key Positions

In the previous section, we derived an analytical solution to compute the optimal duration of a trajectory for a given set of key positions and a given relative timing of the key positions, *e.g.* equally spaced in time. However, a heuristic choice of the timing of the key positions may not lead to the minimal duration of the trajectory if the key



(a) Optimal path.

(b) Optimal position, velocity and acceleration profiles.

Figure 3.5: Example of the optimal path (a) and the optimal trajectory (b). The duration is indirectly optimized for by iteratively optimizing the timing of the key positions.

positions are not evenly distributed in space. Thus, in scenarios in which key positions are provided and are not subject to further optimization, the overall trajectory may be improved by optimizing the timing of the key positions. To denote the relative timing of key positions, we introduce a timing parameter $\boldsymbol{h} = (h_0, \dots, h_{N-1})^\top$, such that the timing of the n-th key position is given by $t_n = \sum_{i=0}^{n-1} h_i$ and the total duration of the trajectory is given by $T = \sum_{n=0}^{N-1} h_n$. The optimization problem for the timing parameter is formulated as follows:

$$
\begin{aligned}
T_{\text{opt}} = \min_{\boldsymbol{h}} \quad & \sum_{n=0}^{N-1} h_n, \\
\text{s.t.} \quad & h_n > 0, \quad n = 0, \dots, N-1, \\
& \dot{\boldsymbol{q}}(t) = \frac{1}{T} \boldsymbol{\Phi}' \left(\frac{t}{T} \right) \boldsymbol{w} \in [-\dot{\boldsymbol{q}}_{\max}, \dot{\boldsymbol{q}}_{\max}], \quad \forall t \in [0, T], \\
& \ddot{\boldsymbol{q}}(t) = \frac{1}{T^2} \boldsymbol{\Phi}'' \left(\frac{t}{T} \right) \boldsymbol{w} \in [-\ddot{\boldsymbol{q}}_{\max}, \ddot{\boldsymbol{q}}_{\max}], \quad \forall t \in [0, T].
\end{aligned}
\tag{3.27}
$$

The computation of the basis functions depends on the timing parameter $\boldsymbol{h}$, such that the optimization problem in (3.27) is non-convex. Therefore, an iterative optimization method such as sequential least squares quadratic programming (Kraft (1988)) is required to locally minimize the duration of the trajectory. The result depends on the initial guess provided and it is not guaranteed that the result is the global minimizer of the trajectory duration. In the following, we exploit the iterative optimization for finding the optimal timing of key positions provided through human demonstrations.

3.3 Skill Learning from Sparse Key Positions

For robots to be widely adopted across applications and environments, it is critical that a wide range of users can program robots with as little effort as possible. State-of-the-art Learning from Demonstration (LfD) techniques enable robots to learn from demonstrations, *i.e.* exemplary behaviors of a target task. Demonstrations often consist of the time series of relevant variables, *e.g.* the robot's end-effector position or its joint configuration. Movement primitive methods learn from such *full trajectory* demonstrations, providing mechanisms to adapt the robot's motion to situations unseen during teaching.

Typically, methods based on movement primitives provide little prior knowledge to the learning procedure. As a consequence, the demonstrations need to teach the robot not only the task at hand but also desirable characteristics of the robot's motion, such as the smoothness of the trajectory. Even with intuitive interfaces like kinesthetic teaching, it can be challenging for novice users to provide full trajectory demonstrations that successfully complete the task and, simultaneously, convey the desired characteristics (Weiss et al. (2009); Wrede et al. (2013); Ajaykumar et al. (2021)).

As an alternative to full trajectories, the literature has explored the use of *key position* demonstrations, requiring the user to demonstrate only a sparse set of consecutive poses. This alternative interface alleviates some issues of kinesthetic teaching, chiefly the need of teachers to provide smooth trajectories (Akgun et al. (2012b)), letting the user focus on the task of programming (Huang and Cakmak (2017); Ajaykumar et al. (2021)) or teaching a successful task execution (Akgun et al. (2012a)). Furthermore, the teacher is not required to demonstrate at the same pace as the desired robot executions, allowing for *e.g.* more accurate placements of key positions. We expect this aspect to facilitate in particular the kinesthetic teaching of high-precision robot tasks such as peg-in-hole.

We propose an approach that learns adaptive movement primitives from key position demonstrations, as illustrated in Figure 3.6. Given an ordered set of key positions, our method fills the gaps between them by recovering the temporal information missing from the demonstrations (*i.e.* when to reach the key positions) and generating a smooth-by-design distribution of trajectories (*i.e.* how to reach the key positions). Similarly to other motion primitive approaches, the learned trajectory distribution is encoded as a stochastic, linear combination of basis functions. To learn the timing for the robot to reach the key positions a cascaded time-optimal control problem is solved for each key position demonstration, and a common set of basis functions that capture a conservative timing is extracted. Furthermore, by including information about the task context and leveraging variability in the demonstrations, our method can adapt the generated robot trajectories to unseen scenarios. Paired with the robot controller presented in Jankowski et al. (2021), the generated trajectories adapt on the fly based

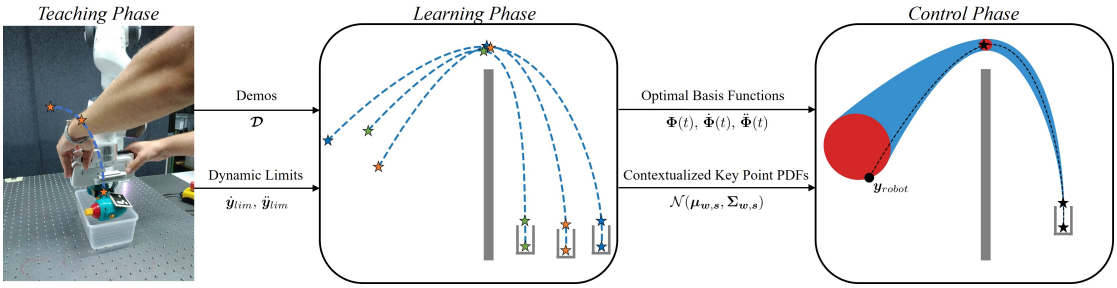


Figure 3.6: *Teaching Phase*: The user provides contextualized demonstrations as a set of ordered key positions (asterisks) and dynamic limits. *Learning Phase*: Time-optimal trajectories (blue-dashed curve) are computed in order to construct an optimal basis for the stochastic key position space. *Control Phase*: By computing the distribution of the key positions in the given context (red ellipses), a probability distribution of optimal trajectories (blue area) is computed and used to efficiently infer an optimal reference trajectory (black-dashed curve) based on the current robot state $\boldsymbol{y}_{\text{robot}}$.

on the current state of both the robot and the environment.

To evaluate the proposed LfD pipeline, we conducted a user study, where 16 participants provided either full trajectory or key position demonstrations to teach a 7 DoF manipulator a pick-and-place task. Results show that the proposed key position method can generate trajectories as successfully (in terms of task completion) as a Probabilistic Movement Primitive (ProMP) (Paraschos et al. (2018)) learned from full trajectories while being smoother. Furthermore, we showcase how our method can be used to solve a peg-in-hole task and a pushing task purely from key position demonstrations.

Related Work on Learning from Key Positions

Demonstrating a motion, either by showing the full trajectory or by recording a sequence of via-points, is a commonly adopted interface for programming robots (Lozano-Perez (1983); Billard et al. (2008); Calinon (2019a)). Both modalities provide the user with an intuitive programming interface while differing in terms of advantages and disadvantages. Full trajectory demonstrations, especially when paired with gravity-compensated robots and kinesthetic teaching, are an intuitive interface for novice users. While allowing expert users to demonstrate the dynamics of motions, demonstrating high-quality full trajectories can be challenging, especially as the number of degrees of freedom rises (Chernova and Thomaz (2014)). Programming robots with key positions can produce more readable and editable robot programs while removing the dynamic component from the demonstration and letting the user focus solely on the definition of key positions (Akgun et al. (2012b); Ajaykumar et al. (2021)). It has been shown however that users find more mentally taxing to specify key positions and

to form mental models of collision avoidance, compared to full trajectories (Akgun et al. (2012b)).

Via-Point Movement Primitives (VMPs) (Zhou et al. (2019)) are a hybrid approach that learns from full trajectories that can be later offset to pass through desired via-points. While allowing for skill extrapolation, VMPs still require full trajectory demonstrations to learn, therefore inheriting the aforementioned limitations. Another hybrid learning approach that accepts as input both full trajectory and key position demonstrations is proposed by Akgun et al. (2012a,b), converting full trajectory demonstrations into a key position representation in order to merge them.

While showing successful skill reproduction capabilities, these approaches can not learn solely from key position demonstrations a skill with variations. Furthermore, they lack the capability to generalize the skill to unseen situations (*e.g.* novel starting robot configurations or target object locations), and to adapt on the fly as the environment changes (lack of adaptability).

Related Work on Adaptive Movement Primitives

While both modalities (and their hybrids, like *e.g.* automatic extraction of via-points from demonstrations (Steinmetz et al. (2019); Ajaykumar et al. (2021))) have been thoroughly adopted in robot programming interfaces, the literature on learning adaptive movement primitives has been tightly linked to the concept of full trajectory demonstrations.

To represent a single trajectory or a distribution of trajectories in a compact form, Dynamical Movement Primitives (DMPs) (Ijspeert et al. (2002)) and ProMPs (Paraschos et al. (2018)) rely on a predefined set of basis functions (typically, radial basis functions with uniform spacing and constant bandwidth). Alternatively, approaches based on (Bayesian) Gaussian Mixture Regression (GMR) (Calinon (2019b)) first encode the trajectories as a joint distribution of space and time/phase variables, and then use the conditional property of Gaussians for regression, providing an approach to automatically estimate the number and placement of the basis functions. As shown by Calinon et al. (2014), probabilistic trajectory learning and adaptation can also be formulated as an optimal control problem, by considering a standard Linear Quadratic Regulator (LQR) with full precision matrices and a virtual system in the form of a simple or double integrator.

Jankowski et al. (2021) propose a controller for ProMPs that enables online adaptation to dynamically changing task contexts. In the following, we show how our approach learns, starting from key position demonstrations, a probabilistic movement primitive with optimally chosen basis functions. This allows us to reuse the controller from Jankowski et al. (2021) and obtain adaptable movement primitives from key posi-

tion demonstrations. In Section 3.3.3, we, therefore, compare the proposed approach against a baseline of ProMPs learned from full trajectories.

Related Work on Learning from Demonstrations using Product of Experts

Calinon (2016) proposes to learn the statistics of the demonstrations in multiple frames (e.g. in an object-centric frame) and to retrieve a trajectory as the mean of the product of the individual distributions. We reuse this approach, however, we fuse the distributions in the space of key positions. Using the product of expert distributions to model more complex distributions in robotics has been introduced more generally by Pignat et al. (2020).

3.3.1 Learning a Distribution of Optimal Trajectories

During the learning phase, the user input is processed into a distribution of optimal trajectories. The user provides a set of K demonstrations $\boldsymbol{\mathcal{D}} = \{\boldsymbol{\mathcal{D}}_k\}_{k=1}^{K}$, with N key positions per demonstration, *i.e.* $\boldsymbol{\mathcal{D}}_k = \{\{{}^k\boldsymbol{y}_n\}_{n=1}^{N}, \boldsymbol{s}_k\}$. The proposed approach can be applied in the robot joint space as well as in the robot's end-effector space. In the latter case, the presented approach applies to the end-effector position only and needs to be complemented by an additional orientation controller (see Section 3.3.2). $\boldsymbol{s}_k$ reflects the state of the environment during the k-th demonstration. While the K demonstrations show different solutions for one task, we assume that the number of key positions is constant among all demonstrations and that the order of the key positions is given by the user. In contrast to learning from full trajectories, there is no information about the timing of the demonstrated task (except for the order in which key positions are given). We recover this information by means of optimal control by exploiting the velocity and acceleration limits we want to impose. These limits are given (*e.g.* by the user) in the form of element-wise symmetric intervals, *e.g.* $\dot{y}^i \in [-\dot{y}^i_{\text{lim}}, \dot{y}^i_{\text{lim}}]$ and $\ddot{y}^i \in [-\ddot{y}^i_{\text{lim}}, \ddot{y}^i_{\text{lim}}]$ for the i-th DoF.

The learning phase consists of (1) constructing optimal basis functions as presented in Section 3.1 and (2) learning a distribution of optimal trajectories by learning a probability density function of the basis function weights plus correlations with the task context. In our approach, the basis function weights explicitly correspond to the N key positions and the start and end velocities, *i.e.* $\boldsymbol{w} = (\boldsymbol{y}_1^\top, \boldsymbol{y}_2^\top, \ldots, \boldsymbol{y}_N^\top, \dot{\boldsymbol{y}}_1^\top, \dot{\boldsymbol{y}}_T^\top)^\top$.

Probability Distribution of Optimal Trajectories

The result of the previously described cascaded optimization is a common basis $\boldsymbol{\Phi}(t)$ for all demonstrated key positions. Note that the mathematical structure of an optimal trajectory in (3.12) given a set of key positions resembles the structure of a ProMP.

Unlike the hand-crafted basis functions of ProMPs, the basis functions of our approach are implicitly learned from the demonstrations by means of optimal control. By demonstrating ordered key positions for a given task context, the user provides direct samples of an underlying joint distribution $p(\boldsymbol{w}, \boldsymbol{s})$ modeling the task through correlations between key positions and task contexts. We assume that the provided samples $\boldsymbol{\mathcal{D}}$ can be modeled by a Gaussian Mixture Model (GMM) with a finite number of components, *i.e.*

$$\boldsymbol{w}, \boldsymbol{s}|\boldsymbol{\mathcal{D}} \sim \sum_{c} \mathcal{N}({}^{c}\boldsymbol{\mu}_{\boldsymbol{w},\boldsymbol{s}}, {}^{c}\boldsymbol{\Sigma}_{\boldsymbol{w},\boldsymbol{s}}). \tag{3.28}$$

Each component of the GMM is translated into a controller that is activated if the system state and the task context can be explained through the corresponding demonstrations of the component (see Jankowski et al. (2021) for details). Thus without loss of generality, we consider a single Gaussian distribution in the remainder for the sake of readability.

The optimal, contextualized robot position at a given time t can be inferred as

$$\boldsymbol{y}(t)|\boldsymbol{s}, \boldsymbol{\mathcal{D}} \sim \mathcal{N}\Big(\boldsymbol{\Phi}(t)\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Phi}(t)\boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{s}}\boldsymbol{\Phi}^{\top}(t)\Big), \tag{3.29}$$

with $\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}_{\boldsymbol{w}|\boldsymbol{s}})$ being the basis function weight distribution conditioned on the state of the environment $\boldsymbol{s}$. The blue area in the right-hand block in Figure 3.6 depicts a time-dependent Gaussian distribution resulting from (3.29).

Learning in Multiple Frames

The task model in (3.28) captures piecewise linear relations between the key positions and the state of the environment. However, orientations of objects in the environment can impose nonlinear relations that would require multiple components in (3.28). In Calinon et al. (2014), poses of objects in the environment are specifically handled by learning the movement statistics in a coordinate system that is attached to the corresponding object. The same can be applied to our approach, resulting in multiple probability distributions of optimal trajectories, with $p_{\mathcal{M}}(\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{\mathcal{D}}) = \mathcal{N}(\boldsymbol{\mu}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}})$ being the distribution learned in a coordinate system $\mathcal{M}$. Since the timing of the task is equal in all coordinate systems, the optimal basis functions $\boldsymbol{\Phi}(t)$ remain the same in all coordinate systems. The online fusion of multiple distributions can thus be done in the basis function weights by computing a product of Gaussians, after transforming all basis function weight distributions into a world-fixed coordinate system $\mathcal{W}$, as

$$\begin{aligned} \boldsymbol{w}|\boldsymbol{s}, \boldsymbol{\mathcal{D}} &\sim \mathcal{N}(\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}, \boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) \\ &= \prod_{\mathcal{M}} \mathcal{N}(\bar{\boldsymbol{R}}_{\mathcal{M}}\boldsymbol{\mu}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}} + \bar{\boldsymbol{t}}_{\mathcal{M}}, \bar{\boldsymbol{R}}_{\mathcal{M}}\boldsymbol{\Sigma}^{\mathcal{M}}_{\boldsymbol{w}|\boldsymbol{s}}\bar{\boldsymbol{R}}^{\top}_{\mathcal{M}}), \end{aligned} \tag{3.30}$$

where $\bar{\boldsymbol{R}}_{\mathcal{M}}$ and $\bar{\boldsymbol{t}}_{\mathcal{M}}$ transform a $\boldsymbol{w}$ that is expressed in frame $\mathcal{M}$ into the world-fixed coordinate system $\mathcal{W}$.

3.3.2 Control Phase

During the control phase, a controller generates motor commands allowing the robot to reproduce the learned behavior by feeding back the robot state as well as the state of the environment. In Section 3.3.2, we recall the concept of probabilistic adaptive control and apply it to the previously derived movement primitive. Section 3.3.2 provides a controller design for the orientation to complete the control of the end-effector pose. The control commands derived in different spaces are composed in the robot's joint space in realtime as an information fusion problem (product of Gaussians).

Probabilistic Adaptive Control

In Jankowski et al. (2021), we designed the controller as a force/torque control action, which has the benefit that the time-varying feedback gains are interpretable as mechanical compliance. However, for composing multiple control policies as in Ratliff et al. (2018), acceleration control actions are better suited. Thus, we define a trajectory tracking controller, *i.e.*

$$\boldsymbol{a} = -\boldsymbol{K}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}) - \boldsymbol{D}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{\Phi}}\boldsymbol{w}) + \ddot{\boldsymbol{\Phi}}\boldsymbol{w}, \tag{3.31}$$

that forces the system to track an optimal trajectory that is given by $\boldsymbol{y}_{des}(t) = \boldsymbol{\Phi}(t)\boldsymbol{w}$. Here, the matrices $\boldsymbol{K}$ and $\boldsymbol{D}$ are design parameters. As the basis function weights $\boldsymbol{w}$ are Gaussian-distributed, the mean of the acceleration control action is inferred as

$$\boldsymbol{\mu}_{\boldsymbol{a}|\boldsymbol{x},\boldsymbol{s}} = -\tilde{\boldsymbol{K}}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) - \tilde{\boldsymbol{D}}(\dot{\boldsymbol{y}} - \dot{\boldsymbol{\Phi}}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}) + \ddot{\boldsymbol{\Phi}}\boldsymbol{\mu}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{s}}, \tag{3.32}$$

with $\boldsymbol{x}$ being the robot state composed of position and velocity, and

$$\begin{aligned} \tilde{\boldsymbol{K}} &= \boldsymbol{K} - (\boldsymbol{K}\boldsymbol{\Phi} + \boldsymbol{D}\dot{\boldsymbol{\Phi}} + \ddot{\boldsymbol{\Phi}})\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}\boldsymbol{\Phi}^{\top}\boldsymbol{\Sigma}_{\boldsymbol{y}}^{-1}, \\ \tilde{\boldsymbol{D}} &= \boldsymbol{D} - (\boldsymbol{K}\boldsymbol{\Phi} + \boldsymbol{D}\dot{\boldsymbol{\Phi}} + \ddot{\boldsymbol{\Phi}})\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}\dot{\boldsymbol{\Phi}}^{\top}\boldsymbol{\Sigma}_{\dot{\boldsymbol{y}}}^{-1}. \end{aligned} \tag{3.33}$$

The matrix $\boldsymbol{\Sigma}^{\mathcal{W}}_{\boldsymbol{w}|\boldsymbol{x},\boldsymbol{s}}$ is the covariance of the basis function weights conditioned on the current robot state and the state of the environment. The matrices $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ and $\boldsymbol{\Sigma}_{\dot{\boldsymbol{y}}}$ represent the expected tracking error, *e.g.* due to modeling errors of the system dynamics.

Orientation Control

For full pose control of the robot end-effector, we propose to combine the probabilistic adaptive controller learned from the key positions with a reactive orientation controller

learned from the corresponding key orientations. We adopt the idea of probabilistic adaptive control for orientations by defining an angular velocity control command as $\boldsymbol{\omega}_d = \boldsymbol{K}_{\text{ori}}(\boldsymbol{\Delta\theta} - \boldsymbol{\Delta\theta}_d)$, with $\boldsymbol{\Delta\theta} = \text{Log}_q(\mu_q)$ the logarithmic map at the robot's end-effector quaternion q and $\boldsymbol{\Delta\theta}_d \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}_d})$. In this formulation, the parameters μ_q and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}_d}$ are directly learned from the orientations provided during the demonstrations within the coordinate system of interest (*e.g.* the coordinate system of an object to grasp). Analogue to the tracking error matrices $\boldsymbol{\Sigma}_{\boldsymbol{y}}$ and $\boldsymbol{\Sigma}_{\dot{\boldsymbol{y}}}$, we introduce a control error matrix for the orientation controller by modeling $\boldsymbol{\Delta\theta} \sim \mathcal{N}(\boldsymbol{\Delta\theta}_d, \boldsymbol{\Sigma}_{\boldsymbol{\theta}})$. Consequently, the conditional mean of the desired angular velocity simplifies to

$$\boldsymbol{\mu}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}} = \boldsymbol{K}_{\text{ori}} \left(\boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1} + \boldsymbol{\Sigma}_{\boldsymbol{\theta}_d}^{-1}\right)^{-1} \boldsymbol{\Sigma}_{\boldsymbol{\theta}_d}^{-1} \boldsymbol{\Delta\theta}. \tag{3.34}$$

Finally, the angular acceleration control command is defined as $\boldsymbol{a}_{\boldsymbol{\theta}} = -\boldsymbol{D}_{\text{ori}}(\boldsymbol{\omega} - \bar{\boldsymbol{\mu}}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}})$, with $\boldsymbol{\omega}$ being the angular velocity of the robot end-effector and $\bar{\boldsymbol{\mu}}_{\boldsymbol{\omega}_d|\boldsymbol{\Delta\theta}}$ being the clipped result of (3.34) in order to limit the angular velocity on-the-fly.

3.3.3 Experimental Evaluation in a User Study

We here present the results of a user study comparing the effectiveness of teaching through key positions (Method **KP**) against full trajectories (Method **FT**). We conducted a study where 16 novice robot users (mean age 28.8, SD 3.2) provided demonstrations to solve a pick-and-place task with a 7 DoF FRANKA EMIKA Panda arm. This study was approved by Idiap Research Institute's Data and Research Ethics Committee.

Experimental Setup

The task consisted of picking a toy drill (from the YCB dataset by Calli et al. (2015)) and placing it inside one of two boxes, as shown in Figure 3.7. The participants provided 8 demonstrations by displacing the gravity-compensated robot arm. The number of demonstrations was selected as a trade-off between data collection and a sensible duration of the study (mean duration of 25 minutes). For each demonstration, a drill pose was randomly selected from a pool of predefined demonstration scenarios, along with a corresponding robot's starting configuration. Variability in the selection of scenarios was systematically enforced to prevent participants from providing uninformative demonstrations, *e.g.* multiple demonstrations with the same drill pose.

Conditions and Protocol

Participants first filled a brief questionnaire consisting of three 7-point Likert scale statements, aimed at assessing their robotics expertise. An average score of 1.95 was observed (Cronbach's $\alpha = 0.9$), indicating novice users. After a brief familiarization phase

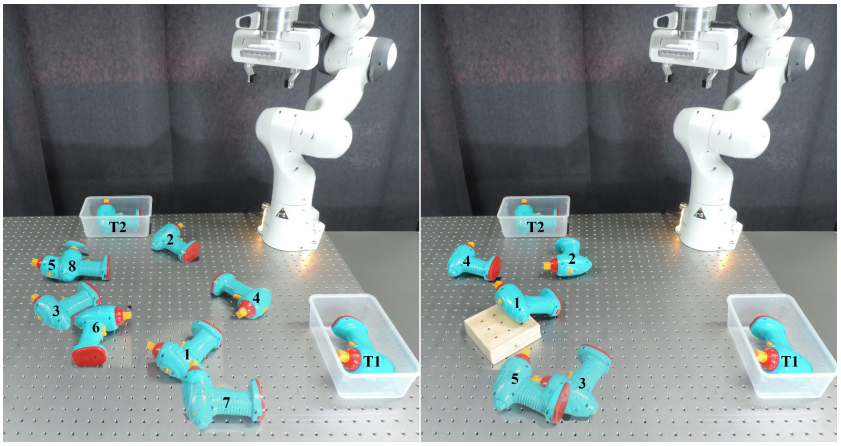


Figure 3.7: Experimental setup of the user study: drill pick-up (1-8, left figure) and drop locations (T1 and T2) used for the demonstration collection, and test locations used in the evaluation phase (1-5, right figure).

with the gravity-compensated robot, each participant provided 8 demonstrations for the aforementioned pick-and-place task, for one of two methods (between-subjects study design): the proposed via-points demonstrations (Method **KP**, presented in Section 3.3.1), and full trajectory demonstrations (Method **FT**, acting as baseline). Each method was therefore used by 8 participants.

For Method **KP**, the participants were instructed to perform the task by using 5 key positions, *i.e.* 3 for the pick action (including the starting pose), and 2 for the place action. Participants displaced the robot in its workspace (kinesthetic teaching) and added key positions by uttering a verbal command (*e.g. "Insert here!"*). For Method **FT**, the participants solely needed to verbally specify the beginning and the end of their demonstrations. For Method **FT**, the number of radial basis functions was set to 24, evenly split between the pick and the place actions. Since the basis functions of Method **FT** are equally distanced in time, a larger number of them compared to Method **KP** is required, in order to capture local details (*e.g.* the grasping pose). For both methods, the task context (*i.e.* the drill pose) is captured by learning in multiple frames (as presented in Section 3.3.1), thus no explicit task context variable s was used for the user study. The trajectory of the robot end-effector pose in the world frame (at the base of the robot) and in the object frame (located on the drill) was recorded, although Method **KP** used only the provided key positions for the learning. To facilitate the learning for Method **FT**, we pre-processed the recorded end-effector position trajectory by automatically finding the start- and end-time step of each demonstration and cutting off the idle parts of the trajectory. The pre-processed data is then used to learn a ProMP representation of the task for Method **FT**, following the algorithm in Gomez-Gonzalez et al. (2020).

The average computation time of the learning phase with 8 demonstrations for one

participant is 4.6 seconds for Method **KP** and 4.2 seconds for Method **FT**. For learning the key position timing for Method **KP**, we empirically set the velocity limit to $\dot{y}^i_{lim} = 0.15\ \frac{\mathrm{m}}{\mathrm{s}}$ and the acceleration limit to $\ddot{y}^i_{lim} = 2\ \frac{\mathrm{m}}{\mathrm{s}^2}$ (equal for all axes). Since both learning approaches result in the mathematical form given in (3.28), we apply the inference in (3.30) to both task representations (with slight modifications for the ProMP) and consequently use the control design in Section 3.3.2 for both approaches with the same hyper-parameters. A demonstration was considered failed if either the pick or the place actions were unsuccessful (*e.g.* the drill slipped from the robot gripper or was placed outside of the designated boxes). Failed demonstrations were excluded from the learning and no corrective demonstrations were collected.

Evaluation

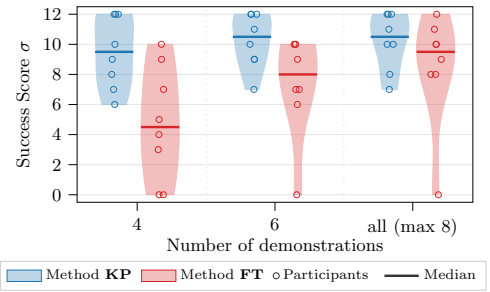
For each participant, we learned task-parameterized trajectory distributions, following the method in Section 3.3.1 for the participants providing demonstrations consisting of key positions (Method **KP**), and by learning a ProMP for Method **FT**. Three feedback controllers as in (3.32) were learned for each participant, with as input the first 4, 6, and all successful demonstrations to evaluate how this impacts the quality of the skill reproduction. Each controller was tested on 5 novel static scenarios, different from the ones used during the demonstration collection (see Figure 3.7). These static drill poses are prerecorded and provided to the controllers, avoiding computer vision inaccuracies to impact the results. We additionally tested the controllers in a mock-up handover task, where a second robot arm[1] pushes the drill on the table towards the controlled robot, equipped with a camera at the wrist to track the drill's pose. This scenario requires the trajectory to be adapted on the fly, therefore testing the online adaptation capabilities of the proposed approach. For each reproduction on a novel static scenario, we computed the acceleration effort e, defined as

$$e = \int_0^T \ddot{\boldsymbol{y}}(t)^\top \ddot{\boldsymbol{y}}(t) dt, \tag{3.35}$$

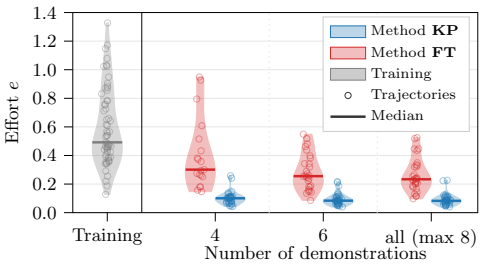
where $\ddot{\boldsymbol{y}}(t)$ is the recorded acceleration of the robot end-effector, therefore measuring the smoothness of the reproduced trajectory.

We further adopted a scoring system to operationalize the success of the task: 1 point for a successful pick (0 otherwise), and 1 point for a successful place (0 otherwise), for a maximum of 2 points per reproduction. The pick action was considered successful if the drill did not slip from the closed gripper during the transport. The place action was considered successful if the drill landed in either of the two boxes. We refer to the cumulative success obtained by a participant with *e.g.* 4 demonstrations and Method **KP** as σ^4_{KP}, ranging from 0 (failure on all 6 novel scenarios) to $2 \times 6 = 12$ (complete

[1]The second robot performs two fixed (although unknown to the first robot) pushing actions with a short intermediate pause, ensuring a consistent experimental scenario.



(a) Distribution of cumulative success score σ, separated by method and by number of demonstrations utilized for training.



(b) Distribution of acceleration effort e of trajectory reproductions, separated by method and by the number of demonstrations utilized for training. The leftmost violin instead shows e of the collected demonstrations.

Figure 3.8: Illustration of the optimization problem for different trajectory durations.

success).

Results

We compare the participants' effectiveness of teaching with Method **KP** and Method **FT** and with different numbers of demonstrations, in terms of the success of the task reproductions and of quality of the reproduced trajectory. Informed by the literature on teaching through key positions (Akgun et al. (2012b); Ajaykumar et al. (2021)), we expected

1. the two methods to achieve similar success scores (with the score improving with a larger number of demonstrations), and

2. Method **KP** to produce smoother trajectories than Method **FT**, thanks to the optimization procedure described in Section 3.1.

We furthermore expected both methods to improve, in terms of trajectory smoothness, over their input, *i.e.* the participants' demonstrations.

Figure 3.8a compares the distribution of cumulative score σ for each teaching method. We observe a statistically significant score difference between methods when trained with 4 and 6 demonstrations (Mann–Whitney U test[2], $p < .01^{**}$), with a median score of 9.5 for Method **KP** vs 4.5 for Method **FT** when trained with 4 demonstrations. This difference can be explained by the fact that Method **KP** needs to learn in a lower-dimensional space than Method **FT**; a relevant difference when learning from a few

[2]Data was tested for normality, always rejecting the null hypothesis (Shapiro-Wilk test, $p < .01^{**}$). We therefore used non-parametric tests.

demonstrations. As expected, σ improves for both methods with more than 6 demonstrations, with negligible differences between methods.

Figure 3.8b compares the distribution of acceleration effort e for each teaching method and for the participants' demonstrations. Unsurprisingly, we see how the reproduced trajectories of both methods require lower acceleration effort e compared to the participants' kinesthetic demonstrations. Method **KP** produces however trajectories with significantly lower e compared to Method **FT** (Mann–Whitney U test, $p<.01^{**}$ for any number of demonstrations used in the training). This is a direct consequence of the optimization process presented in Section 3.1 and Section 3.2.2.

The differences between Method **KP** and Method **FT** are further accentuated when looking at the hand-over task (see the accompanying video for qualitative results). With 4 demonstrations, Method **KP** succeeded at the task for 5 participants out of 8. The number of successful executions goes to 6 when trained with all the available demonstrations. In contrast, no successful execution was possible with Method **FT**, when trained with 4 and 6 demonstrations. Even when using all available demonstrations, Method **FT** obtains successful executions only for 2 of the 8 participants.

We observe that the main reason for failing to grasp the drill is that the robot, with a camera mounted on its wrist, loses track of the moving object while approaching it due to abrupt movements not directed toward the drill. The presented results, including the higher effort required for static scenarios (see Figure 3.8b), indicate that Method **FT** is more prone to produce those abrupt movements compared to Method **KP**.

As also observed in previous work (Akgun et al. (2012b)), participants are almost twice as fast when providing full trajectory demonstrations (median 18 s) compared to key position demonstrations (median 28 s). However, the success score achieved by Method **KP** trained with 4 demonstrations ($\sigma^{4}_{\text{KP, median}} = 9.5$) is comparable with the one of Method **FT** with twice the demonstrations ($\sigma^{8}_{\text{FT, median}} = 9.5$). We argue that, in a real scenario, operations other than the demonstration collection, such as setting the environment or the robot arm's initial configuration, would dominate the process, making the aforementioned time difference negligible.

4 Via-point-based Stochastic Trajectory Optimization

Publication Note

The material presented in this chapter is adapted from the following publication:

- Jankowski, J., Brudermüller, L., Hawes, N., and Calinon, S. (2023). VP-STO: Via-point-based stochastic trajectory optimization for reactive robot behavior. *International Conference on Robotics and Automation (ICRA)*

L. Brudermüller helped in co-developing the framework, the execution of the experiments and the writing of the paper.

Supplementary Material

Webpage with supplementary videos and code related to this chapter is available at: https://sites.google.com/oxfordrobotics.institute/vp-sto.

In Chapter 3, we introduced a low-dimensional representation of robot trajectories based on via-points that allow us to efficiently solve for an optimal duration. In this chapter, we embed this trajectory representation into a stochastic optimization framework, *Via-Point-based Stochastic Trajectory Optimization (VP-STO)*, that enables us to synthesize smooth and timing-optimal robot trajectories in joint space. Achieving reactive robot behavior in complex dynamic environments is still challenging as it relies on being able to solve trajectory optimization problems quickly enough, such that we can replan the future motion at frequencies that are sufficiently high for the task at hand. For contact-rich manipulation that requires online reasoning over making-and-breaking contacts such as pushing, gradient-based optimization fails due to discontinuous contact dynamics. In this chapter, we investigate sampling-based model-predictive control (MPC) as an approach that does not rely on closed-form gradients and is thus able to combine online exploration and exploitation. We argue that current limitations in sampling-based MPC for robot manipulators arise from inefficient, high-dimensional trajectory representations and the negligence of dynamic limits in the trajectory synthesis process. Therefore, we propose a motion optimization framework that optimizes *jointly* over space and time, generating smooth and timing-optimal robot trajectories in joint space. While being task-agnostic, our formulation

can incorporate additional task-specific objectives, such as pushing progress, and yet maintain real-time control rates, demonstrated in simulation and real-world robot experiments on closed-loop manipulation.

We consider the problem of generating continuous, *timing-optimal* and smooth trajectories for robots operating in dynamic environments. Such task settings require the robot to be *reactive* to unforeseen changes in the environment, *e.g.* due to dynamic obstacles, as well as to be *robust* and *compliant* when operating alongside or together with humans. However, generating this kind of reactive and yet efficient robot behavior within a high-dimensional configuration space is significantly challenging. This is especially the case in robot manipulation scenarios with many degrees of freedom (DoFs) as the resulting high-dimensional and multi-objective optimization problems are difficult to solve on-the-fly. A widespread approach in robotics is to formulate the task of motion generation as an optimization problem. Such *trajectory-optimization* based methods aim at finding a trajectory that minimizes a cost function, *e.g.* motion smoothness, subject to constraints, *e.g.* collision avoidance. Solution strategies can either be *gradient-based* or *sampling-based*. Approaches falling in the former category, *e.g.* CHOMP (Zucker et al. (2013)) and TrajOpt (Schulman et al. (2014)), typically employ second-order iterative methods to find locally optimal solutions. However, they require the cost function to be once or even twice-differentiable, which constitutes a major limitation for manipulation tasks as they usually involve many complex, discontinuous cost terms and constraints. In contrast, sampling-based methods (Kalakrishnan et al. (2011); Rubinstein (1999)) can operate on discontinuous costs by sampling candidate trajectories from a proposal distribution, evaluating them on the objective, and updating the proposal distribution according to their relative performance. Compared to gradient-based optimization, stochastic approaches typically also achieve higher robustness to difficult reward landscapes due to their exploratory properties (Hansen (2016)). Yet, achieving reactive robot behavior is challenging as it requires solving trajectory optimization problems at frequencies that are sufficiently high for the task at hand. This issue can be alleviated in *Model Predictive Control (MPC)* settings by optimizing over a shorter receding time horizon. Stochastic, gradient-free trajectory optimization, such as Model-Predictive Path Integral (MPPI) control (Williams et al. (2017)) and the Cross-Entropy-Method (CEM) (Rubinstein (1999)), combined with MPC, also known as *sampling-based MPC*, has proven state-of-the-art real-time performance on real robotic systems in challenging and dynamic environments (Williams et al. (2016); Bangura and Mahony (2014); Bhardwaj et al. (2022)). However, these works still suffer from limited long-term anticipation, *e.g.* getting stuck in front of obstacles, due to the optimization over a short receding horizon.

Motivated by the above, we propose *Via-Point-based Stochastic Trajectory Optimization (VP-STO)*, a framework that introduces the following contributions:

1. A low-dimensional, time-continuous representation of trajectories in joint space

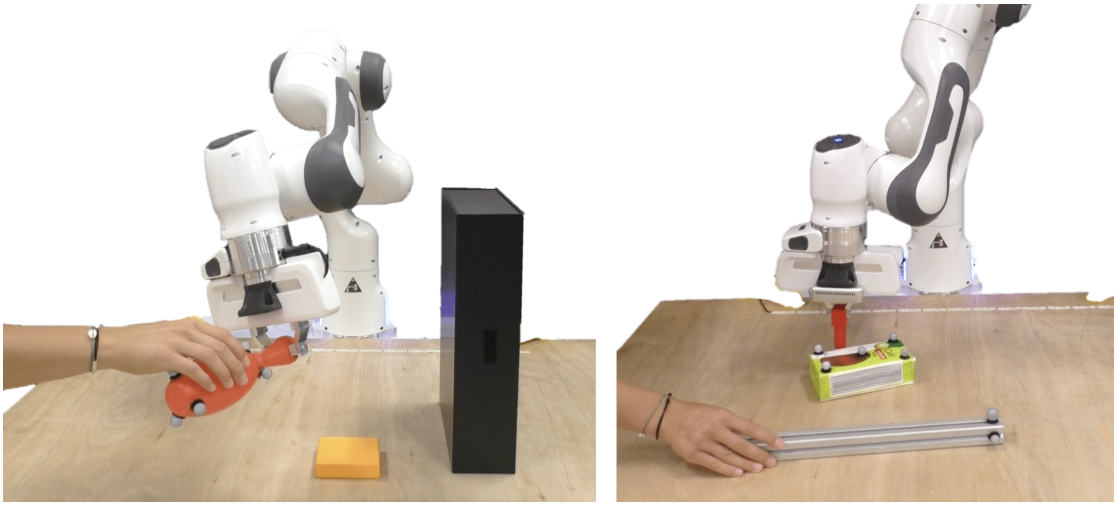


Figure 4.1: ***Experiment settings.*** *Left:* Pick-and-place scenario, where the task is to grasp a bowling pin that is arbitrarily handed over to the robot and to place it upright in the middle of the table. *Right:* Pushing scenario, where the robot has to push the center of the green coffee packet to a moving target location indicated by the tip of the metal stick.

based on via-points that by design respect the kinodynamic constraints of a robot.

2. Stochastic via-point optimization, based on an evolutionary strategy, aiming at minimizing movement duration and task-related cost terms.

3. An MPC algorithm optimizing over the full horizon for real-time application in complex high-dimensional task settings, such as closed-loop object manipulation.

4.1 Related Work

In the context of closed-loop object manipulation with MPC, successful approaches to producing reactive robot behavior typically optimize in joint space subject to kinodynamic constraints. While Fishman et al. (2020) use gradient-based MPC in order to find trajectories for human-robot handovers, Bhardwaj et al. (2022) employed STORM, a sampling-based MPC framework, on prehensile manipulation tasks. It is able to generate particularly smooth trajectories via low discrepancy action sampling, smooth interpolation, and careful cost function design. Moreover, the parallelizability of sampling-based MPC is exploited by deploying the stochastic tensor optimization framework on a GPU. However, in contrast to our work, the approach relies on optimizing over a short receding horizon.

In the realm of time-parametrization of trajectories, most existing approaches fix the

overall motion duration or do not specify it at all. For instance, the majority of MPC-based approaches only handle time implicitly via kinodynamic constraints. While the works of Van den Broeck et al. (2011); Rosmann et al. (2017) progress the state of the art in time-optimal MPC, their applicability to high-dimensional robotic systems yet is limited. In the context of motion planning, T-CHOMP (Byravan et al. (2014)) jointly optimizes a trajectory and the corresponding via-point timings. Yet, the total execution time is still fixed in advance. The way we approach the minimization of the movement duration is most similar to the work of Toussaint et al. (2022b). However, in contrast to our work, their approach optimizes via-points and their timing separately.

4.2 Preliminaries: Trajectory Representation

The way we represent trajectories is based on the optimal basis functions presented in Chapter 3. The basis function weights $\boldsymbol{w}$ include the trajectory constraints consisting of the boundary condition parameters $\boldsymbol{w}_{\text{bc}} = [\boldsymbol{q}_0^\top, \boldsymbol{q}_0'^\top, \boldsymbol{q}_T^\top, \boldsymbol{q}_T'^\top]^\top$ and N via-points the trajectory has to pass through $\boldsymbol{q}_{\text{via}} = [\boldsymbol{q}_1^\top, \dots, \boldsymbol{q}_N^\top]^\top \in \mathbb{R}^{n_{\text{dof}}N}$, such that $\boldsymbol{w} = \left[\boldsymbol{q}_{\text{via}}^\top, \boldsymbol{w}_{\text{bc}}^\top\right]^\top$. Throughout this chapter, the via-point timings s_n are assumed to be uniformly distributed in $s \in [0, 1]$. Note that boundary velocities map to boundary derivatives w.r.t. s by multiplying them with the total duration T, *i.e.* $\boldsymbol{q}_0' = T\dot{\boldsymbol{q}}_0$ and $\boldsymbol{q}_T' = T\dot{\boldsymbol{q}}_T$. Furthermore, the optimization problem in (3.1) minimizes not only the objective w.r.t. $\boldsymbol{q}''(s)$, but also the integral over accelerations, since $\boldsymbol{q}''(s) = T^2\ddot{\boldsymbol{q}}(s)$ and thus the objective is

$$\int_0^1 \boldsymbol{q}''(s)^\top \boldsymbol{q}''(s) ds = T^4 \int_0^1 \ddot{\boldsymbol{q}}(s)^\top \ddot{\boldsymbol{q}}(s) ds, \tag{4.1}$$

corresponding to the control effort. The control effort is minimal *iff* the objective in (3.1) is minimal. As a result, this trajectory representation provides a linear mapping from via-points, boundary conditions and the movement duration to a time-continuous and smooth trajectory.

In the remainder of the chapter, we exploit this explicit parameterization with via-points and boundary conditions by optimizing only the via-points while keeping the predefined boundary condition parameters fixed. Thus, we write the computation of the trajectory as a superposition of a via-point term and a boundary constraints term, *i.e.*

$$\boldsymbol{q}(s) = \boldsymbol{\Phi}_{\text{via}}(s)\boldsymbol{q}_{\text{via}} + \boldsymbol{\Phi}_{\text{bc}}(s)\boldsymbol{w}_{\text{bc}}. \tag{4.2}$$

The matrices $\boldsymbol{\Phi}_{\text{via}}(s)$ and $\boldsymbol{\Phi}_{\text{bc}}(s)$ are extracted from the basis function matrix $\boldsymbol{\Phi}(s)$.

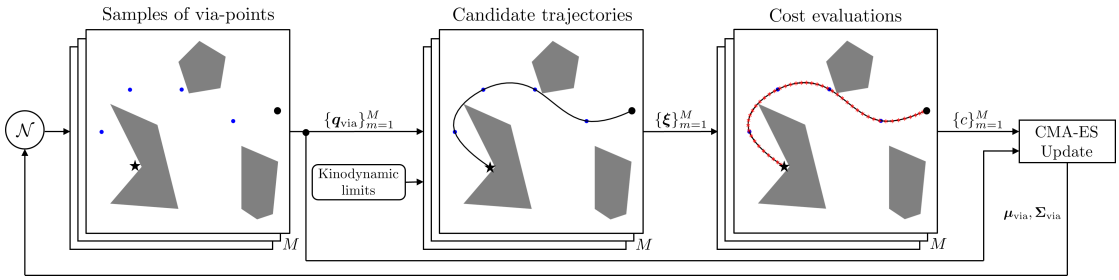


Figure 4.2: An illustration of the via-point-based stochastic trajectory optimization loop. First, a new population of M via-points $\boldsymbol{q}_{\text{via}}$ is sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}})$. Then, the sampled via-points are transformed into a population of candidate trajectories subject to kinodynamic limits. Next, the resulting trajectories are ranked according to their cost evaluations. Last, the parameters of the Gaussian sampling distribution are updated via CMA-ES using the cost rankings and the via-point sets themselves.

4.3 Via-Point-based Stochastic Trajectory Optimization

In the following, we introduce our stochastic trajectory optimization framework. The core idea is to find via-points $\boldsymbol{q}_{\text{via}}$ such that the synthesized trajectory minimizes a task-related objective, *i.e.*

$$\min_{\boldsymbol{q}_{\text{via}}} \quad c\left[\,\boldsymbol{q}(s), \dot{\boldsymbol{q}}(s), \ddot{\boldsymbol{q}}(s), T\,\right]. \tag{4.3}$$

Based on these via-points, we efficiently synthesize high-quality trajectories, *i.e.* $\boldsymbol{q}_{\text{via}} \rightarrow \boldsymbol{\xi}$ with $\boldsymbol{\xi} = \{\boldsymbol{q}(s), \dot{\boldsymbol{q}}(s), \ddot{\boldsymbol{q}}(s), T\}$. We aim at synthesizing trajectories that *by-design* minimize task-agnostic objectives, *i.e. minimum time* and *smoothness*, and satisfy task-agnostic constraints, *i.e.* equality constraints on the *initial* and *final state* and inequality constraints on *joint-space velocities* and *accelerations*. We employ stochastic black-box optimization, namely *Covariance Matrix Adaptation (CMA-ES)* Hansen (2016) to optimize for the via-points. As each trajectory constructed from the sampled via-points already provides the optimal solution to the optimization problem given in 3.1, the CMA-ES optimization in the low-dimensional via-point space is particularly fast, evaluating only high-quality trajectories. Moreover, with CMA-ES we are not only able to quickly converge to a local minimum but to also leverage the exploration aspect of the evolutionary strategy (ES). In more detail, this nested optimization process, which is also illustrated in Figure 4.2, comprises the following steps. First, a new population of M via-points $\boldsymbol{q}_{\text{via}}$ is sampled from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}})$. As $\boldsymbol{q}_{\text{via}}$ is a vector of the stacked via-points, note that $\boldsymbol{\mu}_{\text{via}} \in \mathbb{R}^{n_{dof}N}$ and $\boldsymbol{\Sigma}_{\text{via}} \in \mathbb{R}^{n_{dof}N \times n_{dof}N}$. By taking M samples in this higher-dimensional space, instead of MN samples for all via-points separately in the configuration space, we are able to sample M sets of correlated via-points. Then, as described in detail in Section 4.3.2, the sampled via-points are transformed into a population of candidate trajectories

that are evaluated according to cost terms as outlined in Section 4.3.3. Finally, we use CMA-ES in order to update the parameters $\boldsymbol{\mu}_{\text{via}}, \boldsymbol{\Sigma}_{\text{via}}$ of the Gaussian distribution of via-points. This optimization setup enables us to find a valid local minimum or even the global minimum at rates sufficient for reactive robot behavior in closed-loop manipulation tasks, as we demonstrate in our experiments outlined in Section 4.5.

4.3.1 Informed Sampling with a Gaussian Prior

Let's consider that we are given a Gaussian distribution on the via-points $p_{\text{prior}}(\boldsymbol{q}_{\text{via}})$, which is a prior on the via-points encoding a probabilistic initial guess. For this prior to be informative, it has a higher probability density at optimal via-points compared to a naive initialization, *e.g.* white noise with a scaled variance. We use the via-point prior distribution $p_{\text{prior}}(\boldsymbol{q}_{\text{via}})$ as a probabilistic initial guess for optimizing robot trajectories with CMA-ES. Instead of directly sampling via-point candidates $\boldsymbol{q}_{\text{via}}$ from an uninformed distribution, e.g. white noise, we sample the latent variable $\boldsymbol{\varepsilon} \in \mathbb{R}^{n_{\text{dof}}N}$. For a given $\boldsymbol{\varepsilon}$, we compute $\boldsymbol{q}_{\text{via}}$ through an affine transformation as follows:

$$\boldsymbol{q}_{\text{via}} = \bar{\boldsymbol{q}}_{\text{via}} + \boldsymbol{L}_{\text{prior}}\boldsymbol{\varepsilon}. \tag{4.4}$$

Here, $\boldsymbol{L}_{\text{prior}}$ is the Cholesky decomposition of the covariance matrix $\boldsymbol{\Sigma}_{\text{prior}}$ of the Gaussian prior. The parameter $\bar{\boldsymbol{q}}_{\text{via}}$ is the mean of the Gaussian prior. The idea of this additional transformation is to decouple the optimization variable $\boldsymbol{\varepsilon}$ from the particular prior. In each iteration of VP-STO, we obtain the new population of candidate solutions by sampling M robot trajectories via

$$\boldsymbol{q}_{\text{via}} \sim \mathcal{N}\left(\bar{\boldsymbol{q}}_{\text{via}} + \boldsymbol{L}_{\text{prior}}\,\bar{\boldsymbol{\varepsilon}}, \boldsymbol{L}_{\text{prior}}\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}\boldsymbol{L}_{\text{prior}}^{\top}\right). \tag{4.5}$$

Here, the mean $\bar{\boldsymbol{\varepsilon}}$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}}$ of the latent variable $\boldsymbol{\varepsilon}$ are given by the current solution of the CMA-ES algorithm. When initializing the CMA-ES distribution as white noise, i.e. $\bar{\boldsymbol{\varepsilon}} = \boldsymbol{0}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\varepsilon}} = \boldsymbol{I}$, we effectively sample the first population from the informed distribution $p_{\text{prior}}(\boldsymbol{q}_{\text{via}})$, as inserting the initial parameters into (4.5) yields

$$\boldsymbol{q}_{\text{via}} \sim \mathcal{N}\left(\bar{\boldsymbol{q}}_{\text{via}}, \boldsymbol{L}_{\text{prior}}\boldsymbol{L}_{\text{prior}}^{\top}\right) = p_{\text{prior}}(\boldsymbol{q}_{\text{via}}). \tag{4.6}$$

Next, we describe how the Gaussian prior can be computed to encode a prior on smooth trajectories.

Gaussian Smoothness Prior

The smoothness prior incorporates temporal correlations between via-points by computing a Gaussian distribution that expresses a high likelihood for low-acceleration profiles. A typical objective in trajectory optimization is to minimize the integral over

squared accelerations of the candidate trajectory, i.e.

$$J_{\mathrm{s}} = \frac{1}{2}\int_0^1 \ddot{\boldsymbol{q}}^\top(s)\boldsymbol{R}\ddot{\boldsymbol{q}}(s)ds. \tag{4.7}$$

The positive definite matrix $\boldsymbol{R}$ encodes the desired smoothing for the individual degrees of freedom. Using the parameterization in (4.2), this objective can be expressed using the via-point parameter $\boldsymbol{q}_{\mathrm{via}}$ and the boundary conditions for the trajectory $\boldsymbol{w}_{\mathrm{bc}}$. In fact, the acceleration is an affine function of the via-point parameter $\boldsymbol{q}_{\mathrm{via}}$, i.e.

$$\ddot{\boldsymbol{q}}(s) = \ddot{\boldsymbol{\Phi}}_{\mathrm{via}}(s)\boldsymbol{q}_{\mathrm{via}} + \ddot{\boldsymbol{\Phi}}_{\mathrm{bc}}(s)\boldsymbol{w}_{\mathrm{bc}}. \tag{4.8}$$

We exploit this affine dependency on the via-points and the boundary conditions by rewriting the acceleration as a linear function in a combined trajectory parameter $\boldsymbol{w}$, i.e.

$$\ddot{\boldsymbol{q}}(s) = \ddot{\boldsymbol{\Phi}}(s)\boldsymbol{w}, \tag{4.9}$$

where the new basis function matrix and the new trajectory parameter are given by

$$\ddot{\boldsymbol{\Phi}}(s) = \begin{pmatrix}\ddot{\boldsymbol{\Phi}}_{\mathrm{via}}(s) & \ddot{\boldsymbol{\Phi}}_{\mathrm{bc}}(s)\end{pmatrix}, \quad \boldsymbol{w} = \begin{pmatrix}\boldsymbol{q}_{\mathrm{via}} \\ \boldsymbol{w}_{\mathrm{bc}}\end{pmatrix}. \tag{4.10}$$

As a result, the smoothness objective is equivalent to

$$\begin{aligned} J_{\mathrm{s}} &= \frac{1}{2}\int_0^1 \boldsymbol{w}^\top\ddot{\boldsymbol{\Phi}}(s)^\top\boldsymbol{R}\ddot{\boldsymbol{\Phi}}(s)\boldsymbol{w}\,ds, \\ &= \frac{1}{2}\boldsymbol{w}\int_0^1 \ddot{\boldsymbol{\Phi}}(s)^\top\boldsymbol{R}\ddot{\boldsymbol{\Phi}}(s)\,ds\,\boldsymbol{w}, \\ &= \frac{1}{2}\boldsymbol{w}\boldsymbol{R}_{\boldsymbol{w}}\boldsymbol{w}. \end{aligned} \tag{4.11}$$

The solution of the integral yields a smoothness matrix $\boldsymbol{R}_{\boldsymbol{w}}$ that can be written in block form as

$$\boldsymbol{R}_{\boldsymbol{w}} = \int_0^1 \ddot{\boldsymbol{\Phi}}(s)^\top\boldsymbol{R}\ddot{\boldsymbol{\Phi}}(s)\,ds = \begin{pmatrix}\boldsymbol{R}_{\boldsymbol{q}_{\mathrm{via}}} & \boldsymbol{R}_{\boldsymbol{q}_{\mathrm{via}}|\boldsymbol{w}_{\mathrm{bc}}} \\ \boldsymbol{R}^\top_{\boldsymbol{q}_{\mathrm{via}}|\boldsymbol{w}_{\mathrm{bc}}} & \boldsymbol{R}_{\boldsymbol{w}_{\mathrm{bc}}}\end{pmatrix}. \tag{4.12}$$

With this result, the smoothness objective is in fact a quadratic function in $\boldsymbol{w}$.

Next, we use an exponential transformation to express the smoothness prior as a probability distribution parameterized with the negative objective in (4.7), i.e.

$$p_{\mathrm{prior}}(\boldsymbol{w}) \propto e^{-J_{\mathrm{s}}}. \tag{4.13}$$

Due to the quadratic form in (4.11), the prior is in fact a zero-mean Gaussian distribution in the trajectory parameter $\boldsymbol{w}$, i.e.

$$p_{\mathrm{prior}}(\boldsymbol{w}) = p_{\mathrm{prior}}(\boldsymbol{q}_{\mathrm{via}}, \boldsymbol{w}_{\mathrm{bc}}) = \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{R}_{\boldsymbol{w}}^{-1}\right). \tag{4.14}$$

This Gaussian distribution is a joint distribution over the via-point parameter $\boldsymbol{q}_{\text{via}}$ and the boundary conditions $\boldsymbol{w}_{\text{bc}}$. At the time of constructing the smoothness prior, the boundary conditions are given. Consequently, we obtain the smoothness prior for $\boldsymbol{q}_{\text{via}}$ by conditioning on the boundary conditions, i.e.

$$\begin{aligned} p_{\text{prior}}(\boldsymbol{q}_{\text{via}}|\boldsymbol{w}_{\text{bc}}) &= \mathcal{N}\left(\bar{\boldsymbol{q}}_{\text{via}}, \boldsymbol{R}_{\boldsymbol{q}_{\text{via}}}^{-1}\right), \\ \bar{\boldsymbol{q}}_{\text{via}} &= \boldsymbol{R}_{\boldsymbol{q}_{\text{via}}}^{-1} \boldsymbol{R}_{\boldsymbol{q}_{\text{via}}|\boldsymbol{w}_{\text{bc}}} \boldsymbol{w}_{\text{bc}}. \end{aligned} \tag{4.15}$$

This smoothness prior is used throughout the remainder of this chapter. However, note that the framework is not limited to the smoothness prior, but can be extended to any informative prior. In Chapter 6, a probabilistic prior is used to improve the chances of making contact with an object given a belief about the object's position.

4.3.2 Synthesis of Kinodynamically Admissible Trajectories

In this section, we show how we translate the sampled via-points $\boldsymbol{q}_{\text{via}}$ into kinodynamically admissible trajectories. So far, the trajectory is implicitly given in phase space as described in Section 4.2. Given the via-points and the boundary conditions $\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T$, the computation of the explicit continuous trajectory only depends on the total movement duration T. It is determined by the dynamic limits on velocity $\dot{\boldsymbol{q}}_{\text{min}}, \dot{\boldsymbol{q}}_{\text{max}}$ and acceleration $\ddot{\boldsymbol{q}}_{\text{min}}, \ddot{\boldsymbol{q}}_{\text{max}}$ and is thus given as the minimum positive duration such that the resulting velocity and acceleration profiles satisfy the limits. We compute the optimal trajectory duration T using the direct optimization algorithm presented in Section 3.2.1. This procedure will result in trajectories where either the velocity or the acceleration profile reaches the limit in at least one evaluation point. Finally, having determined T, we are able to explicitly compute the kinodynamically admissible trajectory $\boldsymbol{\xi}$.

4.3.3 Cost Evaluation

Given the sampled and synthesized population of trajectories, we evaluate the performance, *i.e.* the cost c of each trajectory, independently. The gradient-free optimizer allows for sharp cost function profiles, *e.g.* trajectory constraints expressed through discontinuous barrier functions (cf. Section 4.5 for examples). We approximate $c(\boldsymbol{\xi})$ by sampling the given trajectory with a predefined resolution Δs in the phase space $\mathcal{S}$ and accumulating the costs at these K evaluation points. In the time domain, this can still map to varying resolutions of individual trajectories, as $\Delta t = T\Delta s$. Note that the evaluation points at s_k are not equivalent to the via-points at s_n, as depicted in Figure 4.2. The resolution of s_k can be higher than that of s_n in order to have a better approximation of the trajectory cost while keeping the actual optimization variable $\boldsymbol{q}_{\text{via}}$ low-dimensional.

4.4 Online VP-STO (MPC)

In order to perform closed-loop control via continuous online re-optimization, we embed the *VP-STO* framework into an MPC algorithm. In this online setting, the main focus lies on rapidly finding valid movements connecting the current robot state $\boldsymbol{q}, \dot{\boldsymbol{q}}$ with a goal state $\boldsymbol{q}_T, \dot{\boldsymbol{q}}_T$ and re-optimizing them at a sufficient rate $f_{\text{mpc}} = \frac{1}{\Delta t_{\text{mpc}}}$. Algorithm 1 outlines a single MPC step that, given the current robot state, attempts to find an optimal *full-horizon* trajectory and to extract a short-horizon reference to be tracked by a lower-level impedance controller. The details of the algorithm will be outlined in the remainder of this section.

Algorithm 1: Online VP-STO: i-th MPC Step

Input: $\boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{q}_T, \dot{\boldsymbol{q}}_T, \dot{\boldsymbol{q}}_{\text{min}}, \dot{\boldsymbol{q}}_{\text{max}}, \ddot{\boldsymbol{q}}_{\text{min}}, \ddot{\boldsymbol{q}}_{\text{max}}, \Delta t_{\text{mpc}}, T_{\text{stop}}$
Output: Short-horizon reference $\boldsymbol{q}_{\text{d}}(t), \dot{\boldsymbol{q}}_{\text{d}}(t), \ddot{\boldsymbol{q}}_{\text{d}}(t)$
$t_{\text{optimize}} \leftarrow 0$
$\boldsymbol{q}_0, \dot{\boldsymbol{q}}_0 \leftarrow \boldsymbol{q}, \dot{\boldsymbol{q}}$
$\boldsymbol{\xi}_{\text{direct}} \leftarrow$ synthesize() // 4.4.1
if $\boldsymbol{\xi}_{direct}$ *is valid* **and** $\boldsymbol{\xi}_{direct}$ *is shorter than* T_{stop} **then**
  $\boldsymbol{\xi}_i^* \leftarrow \boldsymbol{\xi}_{\text{direct}}$
else
  if $\boldsymbol{\xi}_{i-1}^*$ *is valid* **then**
    ${}^0\boldsymbol{\mu}_{\text{via}}, {}^0\boldsymbol{\Sigma}_{\text{via}}, N \leftarrow$ warmStart$(\boldsymbol{\xi}_{i-1}^*)$ // 4.4.2
  else
    ${}^0\boldsymbol{\mu}_{\text{via}}, {}^0\boldsymbol{\Sigma}_{\text{via}}, N \leftarrow$ exploreInit() // 4.4.2
  end
  $j \leftarrow 0$
  while $t_{optimize} < \Delta t_{mpc}$ **do**
    $\{\boldsymbol{q}_{\text{via}}\}_{m=1}^M \leftarrow$ sample$({}^j\boldsymbol{\mu}_{\text{via}}, {}^j\boldsymbol{\Sigma}_{\text{via}})$ // 4.3.1
    $\{\boldsymbol{\xi}\}_{m=1}^M \leftarrow$ synthesize$\left(\{\boldsymbol{q}_{\text{via}}\}_{m=1}^M\right)$ // 4.3.2
    $\{c\}_{m=1}^M \leftarrow$ evaluate$\left(\{\boldsymbol{\xi}\}_{m=1}^M\right)$ // 4.3.3
    $\boldsymbol{\mu}_{\text{via}}^{j+1}, \boldsymbol{\Sigma}_{\text{via}}^{j+1} \leftarrow$ sep-CMA-ES$\left(\{\boldsymbol{q}_{\text{via}}, c\}_{m=1}^M\right)$
    $j \leftarrow j + 1$
  end
  $\boldsymbol{\xi}_i^* \leftarrow$ synthesize$\left(\boldsymbol{\mu}_{\text{via}}^j\right)$
end
$\boldsymbol{q}_{\text{d}}(t), \dot{\boldsymbol{q}}_{\text{d}}(t), \ddot{\boldsymbol{q}}_{\text{d}}(t) \leftarrow$ shortHorizon$(\boldsymbol{\xi}_i^*)$ // 4.4.3

In the online setting, the number of via-points N used to parameterize the trajectory plays an important role. A large number of via-points can capture highly complex movements and may find more optimal solutions. However, it also implies a higher-dimensional decision space which increases the computational complexity of the optimization loop. Consequently, a particular focus within the MPC algorithm lies in the selection of N.

4.4.1 No-Via-Point Trajectory for Stopping Behavior

VP-STO is based on optimizing the locations of a given number of via-points. However, the trajectory synthesis, described in Section 4.3.2, also works without any via-points, *i.e.* $N{=}0$. The resulting trajectory connects the current robot state and the desired state by a third-order polynomial that minimizes the smoothness objective in (3.1) and satisfies the kinodynamic limits. As this no-via-point trajectory is a unique solution, it can not account for any other movement objectives, *e.g.* to avoid collisions. Yet, the advantage is a cheap-to-construct trajectory that has no stochasticity, which is useful for driving the robot to the target configuration and stopping with zero velocity. Therefore, at the beginning of each optimization cycle, we first check if this simple direct trajectory is valid, *e.g.* collision-free, and if the corresponding duration of the movement is below the user-defined threshold T_{stop}. By setting the threshold rather small, we let the mechanism take over towards the final part of the total trajectory to achieve robust stopping behavior for reaching the goal. If the direct solution is not used, we perform a *VP-STO* optimization cycle.

4.4.2 Initialization: Exploration vs. Warm-Starting

The use of an evolutionary optimization strategy, such as CMA-ES, allows us to initialize the optimization not only with an initial guess of the via-points $\boldsymbol{\mu}_{\text{via}}$, but also to set the corresponding initial variance $\boldsymbol{\Sigma}_{\text{via}}$ as an estimate of how certain we are about the initial solution. The initial variance can thus be interpreted as an exploration parameter influencing how the very first population of candidate trajectories will be sampled. Therefore, in each MPC step we use two possible modes on how to initialize these parameters. The effects of each mode on the resulting candidate trajectories are shown in Figure 4.3.

Exploration. If a MPC step is not successful in finding a valid trajectory, the successive MPC step will be used to *explore* a larger area of the trajectory space to ideally discover a valid solution, as can be seen from the sampled trajectories in the left of Figure 4.3. We initialize the mean solution $\boldsymbol{\mu}_{\text{via}}$ with a naive straight-line guess with high uncertainty, *i.e.* large diagonal values of $\boldsymbol{\Sigma}_{\text{via}}$. The number of via-points used to parameterize the trajectory is set to $N = N_{\max}$. $N_{\max}$ needs to be specified by the user and depends on the complexity of the task, as well as on the available computational resources.

Warm-Starting. If a valid solution was found in an MPC step, we shift the solution forward in time and use it to *warm-start* the mean $\boldsymbol{\mu}_{\text{via}}$ in the successive MPC step, potentially further improving the current solution. In this case, we initialize the covariance matrix $\boldsymbol{\Sigma}_{\text{via}}$ with low values on the diagonal as we are more certain about the proximity of the current solution to a valid local minimum, as can be seen on the right of Figure 4.3. In order to determine the number of via-points N for the successive MPC

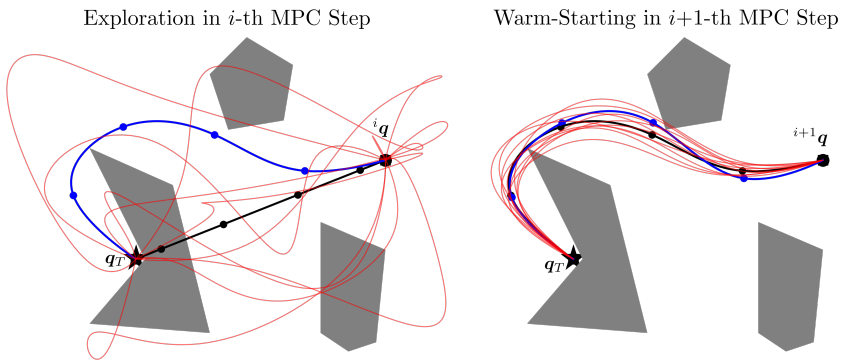


Figure 4.3: An illustration of the stochastic optimization process within the proposed MPC algorithm. ***Left:*** In the *exploration* mode, trajectories are sampled and synthesized with a large initial variance in order to discover valid solutions. ***Right:*** If a valid solution is available from the previous MPC step, we *warm-start* the optimization by shifting the solution and sampling from a lower-variance initial distribution. All sampled trajectories are shown in red. The initial guesses ${}^{0}\boldsymbol{\mu}_{\text{via}}$ of an MPC step are depicted by the black solid lines, while the blue trajectories illustrate the mean solution ${}^{20}\boldsymbol{\mu}_{\text{via}}$ after 20 optimization iterations.

step, we use the movement duration of the current solution as a proxy for how complex the remainder of the movement will be. We therefore set $N = \max(1, \min(\lceil \alpha T \rceil, N_{\max}))$, where T is the total duration of the current solution and α a user-defined scaling parameter.

4.4.3 Impedance Control

At a lower control level, we deploy an impedance controller that runs at a control rate of 1 kHz, which requires a finely sampled reference trajectory. Due to our time-continuous representation of the optimized trajectory, we can sample configurations from it with arbitrarily small temporal resolution. Each MPC step yields an optimized trajectory $\boldsymbol{\xi}_i^*$, from which we extract a position-, velocity- and acceleration-reference enabling the robot to track the current movement plan.

4.5 Experiments

We evaluate the effectiveness and performance of the *VP-STO* framework in simulation, as well as in real-world experiments with a Franka Emika robot arm.

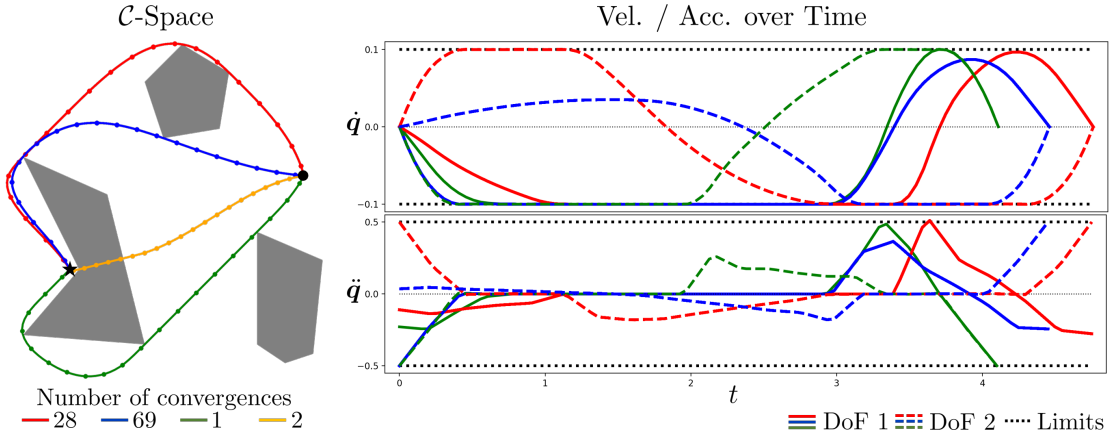


Figure 4.4: ***Offline VP-STO.*** *Left:* The resulting trajectories from 100 experiment runs when initializing with a straight-line guess between the start position (black circle) and the target position (black asterisk). The number of convergence indicates how often *VP-STO* converged to the corresponding color-coded solution. *Right:* The velocity and acceleration profiles for each degree of freedom correspond to the valid solutions on the left.

4.5.1 Simulation

We begin by evaluating our framework in an offline planning setting for a 2D point mass in a cluttered toy environment adopted from Bhardwaj et al. (2022). In this experiment, we run *VP-STO* (cf. Section 4.3) for 100 times with a straight-line initialization. The left plot in Figure 4.4 shows the resulting 100 trajectories after convergence. The majority of the found solutions converged to 3 valid local optima, *i.e.* 28 solutions to the red, 69 to the blue, and one to the green trajectory. Only 2 runs produce a non-valid solution, shown in yellow. We note here that gradient-based trajectory optimization methods given the straight-line initial guess in such a challenging environment would only converge to this non-valid local optimum. Moreover, this also shows that the choice of CMA-ES as a solver for our framework helps to converge to the present local optima with negligible error, despite the stochasticity in the sampling of the via-points. Last, the corresponding velocity and acceleration profiles (only shown for the valid solutions), depicted on the right of Figure 4.4, reflect the timing-optimal property of the generated trajectories. After applying maximum acceleration at the start of the movement, the robot moves at maximum speed within the limits before it again applies the maximum acceleration to stop at the goal. This implies that our framework generates trajectories that not only respect the given dynamic limits but also exploit them in the spirit of timing optimality.

For the online setting, as described in Section 4.4, we compare *VP-STO* to *STORM* Bhardwaj et al. (2022), which we consider as state-of-the-art in sampling-based MPC for producing reactive robot behavior. Again using the scenario from above, we run 5

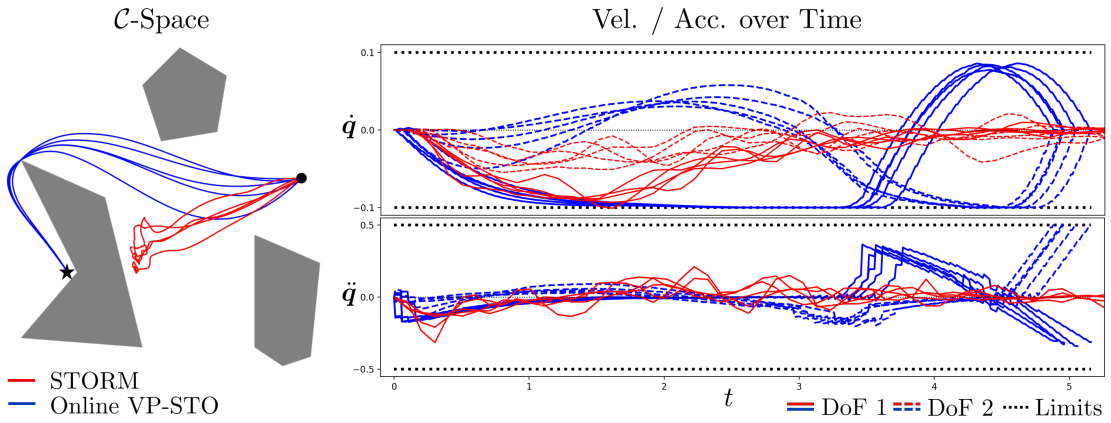


Figure 4.5: ***Online VP-STO (MPC).*** *Left:* The trajectories taken by the robot when deploying *VP-STO* in an MPC setting (blue), as opposed to using *STORM* Bhardwaj et al. (2022) (red). *Right:* The velocity and acceleration profiles for each degree of freedom correspond to the found solutions on the left.

experiments in which we deploy *VP-STO* within the MPC-algorithm (cf. Algorithm 1). The resulting trajectories are shown in blue in Figure 4.5 alongside the 5 solutions in red generated by *STORM*. It can be seen that *STORM* is not able to reach the goal. Especially, due to the short-horizon optimization scheme, the robot first follows the path with the shortest distance towards the goal while not being able to anticipate moving around the obstacle early enough. Therefore, it gets stuck in front of the obstacle. In contrast, *Online VP-STO* produces solutions that allow the robot to smoothly navigate to the goal while exploiting its velocity and acceleration limits. The given setting and experiment emphasize the advantage of our efficient formulation which allows us to always optimize over the full horizon.

4.5.2 Real-World Experiments

We demonstrate *VP-STO* on a real robot using the manipulation scenarios in Figure 4.1: a *pick-and-place* and a *box pushing* task. We increase the complexity of both scenarios by disturbing the robot and the target objects. This requires a fast feedback loop provided by *Online VP-STO*.

Setup. Both experiments are performed on a Franka Emika robot arm. The framework was run on Ubuntu 20.04 with an Intel Core i7-8700 CPU@3.2GHz and 16GB of RAM. The poses of the objects were tracked with a Vicon motion capture system and post-processed with an extended Kalman filter. The MPC steps are executed at a fixed control rate (specified below). In a single MPC step, we run optimization iterations until the next MPC step starts.

Pick-and-Place. First, we consider a *pick-and-place* scenario under human interven-

tion. The robot's task is to grasp a pin, *i.e.* the picking phase, and to place it in an upright position in a given target location in the workspace, *i.e.* the placing phase. In the picking phase, the pin can be either handed over to the robot in arbitrary poses or the robot needs to pick it up from the table. This phase requires real-time collision avoidance in narrow configuration passages, *i.e.* the robot has to avoid collisions between its hand, including the fingers, and the pin while reaching a configuration where the hand encloses the pin. For the grasp pose, we run a separate pose optimization process in parallel to *VP-STO*, providing the final robot configuration $\boldsymbol{q}_T$. After a successful grasp, the robot continues with the placing phase. The challenge here is that the pin might still move within the gripper due to its own weight or due to interference from a user. Consequently, feedback of the current pin pose is needed to avoid collisions between the pin and the environment and to correctly place the pin. We parameterize the sampled trajectories with a maximum number of via-points N_{max}=4 and α=2. *VP-STO* replans with a rate of 12.5 Hz.

Box Pushing. In the second scenario, we address the task of planning and control through physical contacts, *i.e.* the robot is supposed to push a box towards a moving target position. Such a task requires the robot to deliberately make and break contacts, which is subject to discontinuous cost-landscapes. Here, we exploit the presented trajectory parameterization by setting the final robot configuration $\boldsymbol{q}_T$ of each MPC step such that the end-effector moves towards the center of the box. This enforces all sampled candidate trajectories to make contact with the box. The point of contact and the resulting dynamics of the box depends on the location of the via-points which are subject to minimizing the distance between the box position and the target. For the sake of fast simulations of the contact dynamics, we use a quasi-dynamic model for the box dynamics parallel to the table surface. *VP-STO* is executed with a constant number of via-points N=3 at a control rate of 20 Hz.

Cost Terms. We begin with the task-agnostic terms and conclude with more task-specific terms.

Movement Duration. The movement duration is used explicitly as part of the cost function in order to minimize the time needed for the remaining robot movement.

Smoothness. In order to optimize not only for fast but also efficient movements, we use the same metric as in (3.1) as the smoothness cost term.

Joint Limit Avoidance. For keeping the robot configuration inside the joint angle limits, we deploy a discontinuous metric that accounts for joint limit violations, *i.e.*

$$c_{\text{jla}}(q) = \begin{cases} 1 + q - q_{\max}, & \text{if } q \geq q_{\max} \\ 1 + q_{\min} - q, & \text{if } q \leq q_{\min} \\ 0, & \text{otherwise} \end{cases} . \tag{4.16}$$

We consider a trajectory to be invalid if it results in a joint limit violation, *i.e.* $q \geq q_{\max}$ or $q \leq q_{\min}$.

Collision Avoidance. In order to efficiently evaluate the validity of a trajectory regarding collisions between the robot and the environment, we perform binary collision checks for each configuration evaluated along the trajectory, instead of computing a distance between two geometries. Thus, the collision cost for a single trajectory is equal to the number of evaluation points that are in collision. Similarly to the joint limit avoidance cost, we consider a trajectory to be invalid if it results in a collision.

Pushing Progress. In the case of a pushing task, we further require a cost term that rewards trajectories that let the robot move the box closer to the current desired target $\boldsymbol{x}_{\text{box}}^{\text{des}}$. We evaluate the pushing progress of a single trajectory by first simulating the contact dynamics that result in a trajectory of the box $\boldsymbol{x}_{\text{box}}(t)$; and then computing the box position error at the beginning $e_{\text{box},0} = ||\boldsymbol{x}_{\text{box}}(0) - \boldsymbol{x}_{\text{box}}^{\text{des}}||_2^2$ and at the end $e_{\text{box},T} = ||\boldsymbol{x}_{\text{box}}(T) - \boldsymbol{x}_{\text{box}}^{\text{des}}||_2^2$ of the robot movement. The final pushing progress cost is given by $c_{\text{push}}(\boldsymbol{\xi}) = \exp(e_{\text{box},T} - e_{\text{box},0})$. Additionally, we consider trajectories that move the box away from the target, *i.e.* $e_{\text{box},T} \geq e_{\text{box},0}$, to be invalid. In that case, the *exploration* mode in the next MPC step (cf. Section 4.4.2) is triggered.

Results. First, we note that throughout the experiments, the robot did not collide with any objects in the workspace and did not violate the joint limits. When the experimenter perturbs the robot, *i.e.* disturbing it through physical interaction or pulling the pin out of the gripper, the robot is compliant and adapts its motion. In the pick-and-place scenario, it robustly picked up the pin from various locations in the workspace, including handovers by the experimenter; and placed it at the desired target location in all runs. In the box-pushing scenario, the robot manages to find pushing motions from arbitrary configurations and box locations and to eventually push the box into the target. We note, however, that some changes in the target location resulted in the robot not finding a valid pushing motion quickly enough, which in turn made the robot push the box out of the workspace. This could only be recovered by the experimenter. Recordings of the experiments and additional material can be found in the supplementary material.

4.5.3 Ablation Studies

In this chapter, we present design choices that we want to further justify via ablation studies.

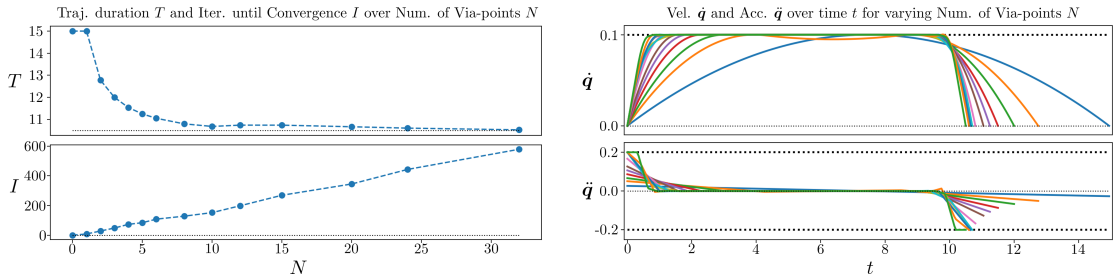


Figure 4.6: A study of the impact of the number of via-points in a 1D time-optimization problem. **Top-Left:** Impact on the resulting movement duration. The dotted black line illustrates the duration of the optimal *bang-bang* solution. **Bottom-Left:** Impact on the number of iterations required until convergence. **Right:** Velocity and acceleration profiles for evaluated numbers of via-points.

Impact of the Number of Via-points

In this ablation study, we investigate the impact of the number of via-points used to represent the robot movement. This hyper-parameter has a high impact on the overall framework performance. On the one hand, it directly sets the dimensionality of the optimization problem to solve; on the other hand, it directly spans the space of movements that can be synthesized. From an optimization perspective, tuning the number of via-points gives us an intuitive way of increasing/decreasing resources on an optimization result with a decreasing/increasing cost. We illustrate this relationship in Figure 4.6, where we let a 1D double-integrator move from $q_0 = 0.0$, $\dot{q}_0 = 0.0$ to $q_T = 1.0$, $\dot{q}_T = 0.0$ in minimal time, subject to a maximum velocity $|\dot{q}| < 0.1$ and an acceleration limit $|\ddot{q}| < 0.2$; with a varying number of via-points. This time-optimal control problem is known to be solved by a bang-bang acceleration profile, such that we know the analytic limit of the minimal time to be $c_{\text{bang-bang}} = T_{\text{bang-bang}} = 10.5$, which is depicted as a dashed black line in the upper-left plot. We observe that the solution cost exponentially converges to $c_{\text{bang-bang}}$ as we increase the number of via-points. The lower-left plot shows the number of CMA-ES iterations required to converge as a function of the number of via-points. Here, we detect convergence if $|c_k - c_{k-1}| < 10^{-6}$ in the k-th iteration. Interestingly, the number of iterations grows linearly with the number of via-points. Note that this does not mean that the computational cost grows linearly with the number of via-points, since the computational cost for a single iteration is either linear (sep-CMA-ES) or quadratic (CMA-ES) in the number of via-points. Nevertheless, those results motivate to use a low number of via-points as with a growing number of via-points, the benefit of adding a via-point is not worth the extra computational cost.

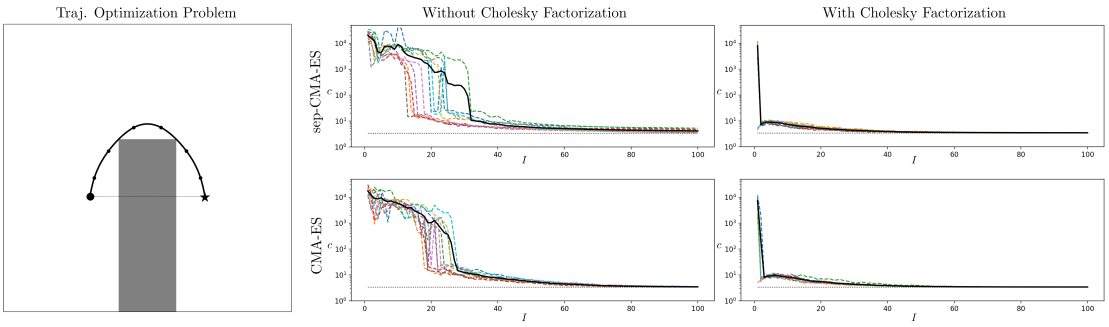


Figure 4.7: A study of the impact of the Cholesky factorization of the Covariance Matrix $\boldsymbol{\Sigma}_{\text{via}}$ in a 2D time-optimization problem with obstacle avoidance. **Left:** The configuration space including the obstacle in gray, the initial guess as dashed line, and the optimal solution around the obstacle as a solid line together with the corresponding via-points as circles. **Center:** The via-point covariance matrix is explicitly updated, *i.e.* $\boldsymbol{\Sigma}_{\text{via}} = \boldsymbol{\Sigma}_{\text{CMA}}$. **Right:** The via-point covariance matrix is updated through a Cholesky factorization, *i.e.* $\boldsymbol{\Sigma}_{\text{via}} = \boldsymbol{L}\boldsymbol{\Sigma}_{\text{CMA}}\boldsymbol{L}^{\top}$. **Top:** sep-CMA-ES iterates on diagonal covariance matrices only, *i.e.* $\boldsymbol{\Sigma}_{\text{CMA}} = \text{diag}(\boldsymbol{\sigma}_{\text{CMA}})$, with linear computational complexity $\mathcal{O}(ND)$. **Bottom:** CMA-ES iterates on full covariance matrices $\boldsymbol{\Sigma}_{\text{CMA}}$ with quadratic computational complexity $\mathcal{O}(N^2D^2)$.

Impact of the Cholesky Factorization of the Covariance Matrix

In this ablation study, we look at a 2D minimal-time planning problem including an obstacle that is to be avoided. We fix the number of via-points to $N = 6$ and set up four different optimization loops that are supposed to solve the same problem. Each setup uses either CMA-ES or sep-CMA-ES and runs with or without the Cholesky factorization of the covariance matrix as described in Section 4.3.1. For comparison, we look at the cost evolution over the number of iterations. The dashed black line in all plots (except for the left-hand plot) indicates the minimum cost measured in any experiment. Note also the jump in all the cost profiles from $\approx 10^3 - 10^4$ to $\approx 10^0 - 10^1$, which reflects if the updated solution is collision-free. We observe that the choice of CMA-ES vs. sep-CMA-ES does not have a substantial impact on the cost evolution for this particular problem, indicating that it is justified to use sep-CMA-ES with linear complexity. However, we observe a substantial impact when using the presented Cholesky factorization, imposing smoothness on the candidate trajectories. In all experiments using the Cholesky factorization, it converged to a collision-free solution after 3 iterations at maximum. This is an especially important result justifying the use of the Cholesky factorization inside the MPC loop, as the real-time requirements limit the number of iterations.

5 Stochastic Impact Control in Real-Time

Publication Note

The material presented in this chapter is adapted from the following publication:

- Jankowski, J., Marić, A., Liu, P., Tateo, D., Peters, J., and Calinon, S. (2024b). Energy-based contact planning under uncertainty for robot air hockey

A. Marić implemented the learning pipeline of the energy-based model and helped write the paper. P. Liu implemented the real-robot setup and conducted the experiments. D. Tateo provided guidance on the project and helped write the paper.

Supplementary Material

Video related to this chapter is available at: https://youtu.be/7L-iJgs87gM.

Planning robot contact often requires reasoning over a horizon to anticipate outcomes, making such planning problems computationally expensive. After developing a framework that enables realtime control through slow contact interactions such as pushing in Chapter 4, this chapter addresses the problem of controlling impacts, *i.e.* collisions between the robot and the object. We propose a learning framework for efficient contact planning in realtime subject to uncertain contact dynamics. We implement our approach for the example task of robot air hockey. Based on a learned stochastic model of puck dynamics, we formulate contact planning for shooting actions as a stochastic optimal control problem with a chance constraint on hitting the goal. To achieve online re-planning capabilities, we propose to train an energy-based model to generate optimal shooting plans in realtime. The performance of the trained policy is validated in simulation and on a real-robot setup. Furthermore, our approach was tested in a competitive setting as part of the *NeurIPS 2023 Robot Air Hockey Challenge*.

Planning and control through non-prehensile contacts is an essential skill for robots to interact with their environment. Model-based approaches enable robots to anticipate the outcome of contact interactions given a candidate action allowing them to find an action with the desired outcome. While model-based planning approaches are shown to be successful at generating contact-rich plans for slow tasks (Pang et al. (2023); Jankowski et al. (2024a)), highly dynamic tasks require the agent to regenerate contact plans at a sufficiently high rate for reacting to inherent perturbations. Highly

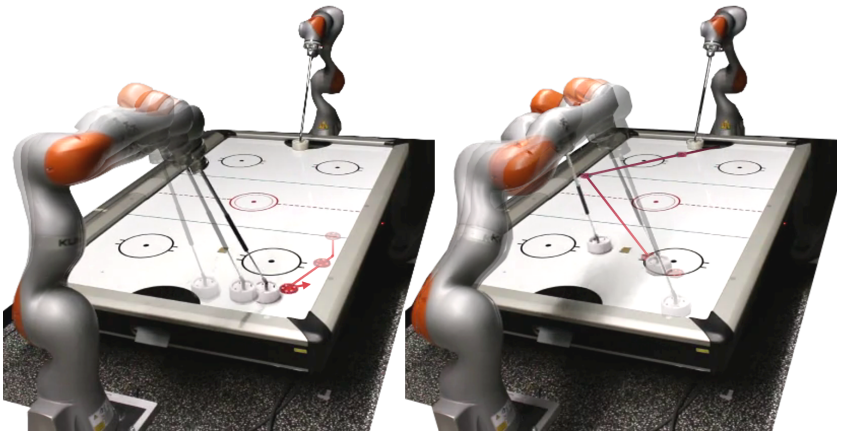


Figure 5.1: The proposed control framework enables a robot arm to autonomously play matches of air hockey. First, a learned stochastic model of contact dynamics is used to predict the trajectory of a puck. An energy-based contact planner is then trained to generate agile behavior in realtime.

dynamic tasks not requiring reasoning through contacts have historically been used as a testbed for hardware and algorithms in robotics. These tasks include different types of games and sports, such as ball-in-a-cup (Kawato et al. (1994); Kober and Peters (2008)), juggling (Ploeger et al. (2021); Ploeger and Peters (2022)), diabolo (von Drigalski et al. (2021)). Dynamic tasks involving contacts, such as soccer (Haarnoja et al. (2024)), tennis (Zaidi et al. (2023)), table tennis (Mülling et al. (2011); Büchler et al. (2022)), and air hockey (Liu et al. (2022, 2024)), are typically approached with reinforcement learning methods to off-load the computationally expensive reasoning through contacts to an offline exploration phase. Yet, these tasks have in common that contacts with the ball or puck are instantaneous, i.e. the contact happens in a short period of time, resulting in a jump in the state of the object. The reasoning over the contact between the robot and the object of interest can therefore be divided into three segments of the planning horizon: *i)* Moving the robot into contact, *ii)* the contact itself at a single time instance, and *iii)* the passive trajectory of the object after contact.

In this chapter, we exploit the separability in the planning horizon by combining a model-based control approach for moving the robot into contact with a learning-based approach for planning the next best contact that results in the desired trajectory of the object. Towards this end, we learn a mixture of linear-Gaussian modes for modeling the object dynamics from data, which allows us to extract a stochastic model for the contact between the robot and the object. Based on the learned model, we train an energy-based contact policy by generating example contacts that are optimal w.r.t. a stochastic optimal control objective offline. During the online phase, we retrieve optimal contact plans from the energy-based policy using derivative-free inference in realtime.

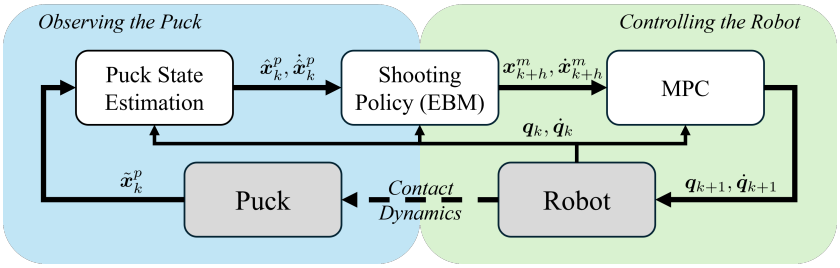


Figure 5.2: Overview of the interplay between puck state estimation • and robot control • in our framework for agile robot air hockey. The contact planner uses the estimated puck state to predict the puck trajectory based on the learned model. It subsequently plans a shooting angle that is used to construct an optimal control objective solved within a model-predictive controller. All modules are updated at a control rate of 50 Hz.

In the following, we present our approach in the context of the highly dynamic game *air hockey*. Figure 5.2 illustrates the online control framework that consists of state estimation, the proposed learning-based contact planner (shooting policy), and a subsequent model-based robot controller (MPC). For the robot controller, we use VP-STO (see Chapter 4) that enforces the execution of the contact plan while respecting safety constraints such as collision avoidance with the walls. We summarize our key scientific contributions as follows:

- We present an approach for learning the parameters of a stochastic model for discontinuous contact dynamics as a mixture of linear-Gaussian modes.
- We formulate the planning of contacts as a chance-constrained stochastic optimal control problem.
- We propose an approach for training an energy-based model to capture the optimal policy according to the chance-constrained stochastic optimal control problem.

Our approach is experimentally validated in a dynamic air hockey shooting task with comparisons to control-based and reinforcement learning baselines. We additionally deploy our framework in a competitive setting as part of the NeurIPS 2023 Robot Air Hockey Challenge (see Figure 5.1). Our framework outperforms all other approaches in real-robot matches, establishing a new state-of-the-art in robot air hockey.

5.1 Related Work

The air hockey task has been part of the robotics literature for a long time (Bishop and Spong (1999)). One of the first works using the air hockey task as a benchmark

focused on skill learning of a humanoid robot (Bentivegna et al. (2004a,b)). In more recent years, this benchmark has been used in combination with planar robots due to high-speed motion requirements (Namiki et al. (2013); Shimada et al. (2017); Igeta and Namiki (2017); Tadokoro et al. (2022)), and the possibility of adapting the playing style against the opponent (Igeta and Namiki (2015)). This benchmark has been recently extended to the cobot setting, where a 7-DoF robotic arm controls the mallet and maintains the table surface while striking (AlAttar et al. (2019); Liu et al. (2021); Chuck et al. (2024)).

Another use of the robot air hockey setting is as a testbed for learning algorithms. In Taitler and Shimkin (2017), deep reinforcement learning techniques are used to learn on planar robots, while in Liu et al. (2022), both the planar 3-DoF and the 7-DoF cobot air hockey tasks are used to learn control policies in simulation. More recent techniques directly use the real 7-DoF air hockey setting as a testbed for learning algorithms: in Kicki et al. (2023), the authors use learning-to-plan techniques to generate air hockey hitting trajectories in the real-world setting, while in Liu et al. (2024), this task is used to perform real-world reinforcement learning.

In general, existing solutions to the robot air hockey problem can be categorized in two main directions: learning-based approaches (Bentivegna et al. (2004b); Taitler and Shimkin (2017); Liu et al. (2024)), and control-based approaches (Tadokoro et al. (2022); AlAttar et al. (2019); Liu et al. (2021)). Generally speaking, pure control-based approaches lead to better and faster solutions than learning-based methods but require considerable efforts in engineering and model identification, and are particularly challenging to implement to run at realtime control rates. Instead, pure learning-based approaches obtain a worse-quality solution but make it possible to obtain more robust behaviors by relying on domain randomization and fine-tuning on the real platform. In this chapter, we aim to combine the advantages of learning-based and control-based approaches. We exploit the optimality of control-based approaches for controlling the robot without consideration of the puck and exploit the robustness and flexibility of learning-based approaches to efficiently generate plans for the contact between the robot and the puck to maximize the chance of scoring.

5.2 Learning Stochastic Contact Models

Planning and controlling the contacts of the robot with the puck requires the anticipation of puck trajectories. To enable this, we learn a simplified stochastic model of the puck dynamics for *i)* estimating the current state of the puck online, *ii)* predicting the trajectory of the puck online, and *iii)* solving a stochastic optimal control problem to plan the next best contact between the robot and the puck.

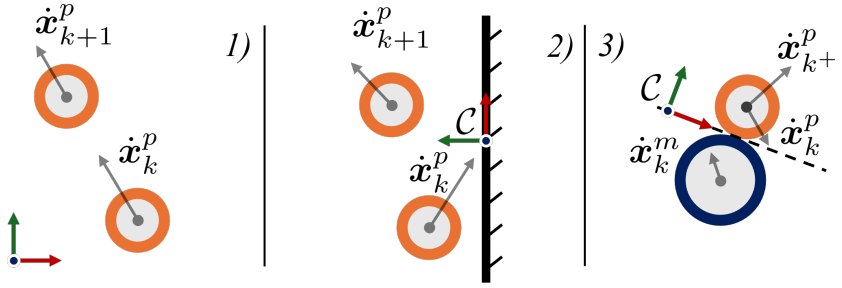


Figure 5.3: Illustration of three modes of the puck dynamics that are parameterized as linear-Gaussian models. Mode *1)* captures the dynamics of the puck • when floating on the surface of the table. Mode *2)* captures collisions between puck and walls. Mode *3)* models collisions between puck and the mallet • in a contact-aligned frame $\mathcal{C}$. The parameters for the nominal dynamics and the corresponding uncertainty are learned.

5.2.1 Mixture of linear-Gaussian Contact Dynamics

Suppose that $\boldsymbol{x}^p_k \in \mathbb{R}^2$ is the position of the puck w.r.t. the surface of the air hockey table at time step k. The robot interacts with the puck by making contact with its mallet, i.e. the circular part of the robot's end-effector. The position of the mallet is denoted with $\boldsymbol{x}^m_k \in \mathbb{R}^2$ w.r.t. the surface of the air hockey table. We assume that the robot arm is controlled such that the mallet maintains contact with the table at all times. In order to efficiently perform rollouts of the puck dynamics, we impose a piecewise-linear structure on the model. Figure 5.3 illustrates the three modes that we present in the following: *1) Floating*, *2) Puck-Wall Collision*, and *3) Puck-Mallet Collision*. To account for modeling errors introduced through the piecewise-linear structure, we model each mode as a conditional Gaussian distribution, resulting in a mixture of linear-Gaussian contact dynamics. In the following, we present the individual modes and their respective parameters that are learned subsequently. Note that the learnable parameters define the stochastic prediction of the puck velocity, while the one-step prediction of the position is derived from numerical integration and is deterministic.

Floating

The first mode captures the dynamics of the puck when it is freely floating on the table and is not in collision with the wall or mallet. The prediction of the puck velocity is modeled stochastically with

$$\Pr_1\left(\dot{\boldsymbol{x}}^p_{k+1} | \dot{\boldsymbol{x}}^p_k\right) = \mathcal{N}\left(\boldsymbol{\Theta}_1 \dot{\boldsymbol{x}}^p_k + \boldsymbol{\theta}_1, \boldsymbol{\Sigma}_1\right), \tag{5.1}$$

where $\boldsymbol{\Theta}_1, \boldsymbol{\theta}_1, \boldsymbol{\Sigma}_1$ are parameters of the conditional Gaussian distribution.

Puck-Wall Collision

The second mode models the dynamics of the puck reflecting against the wall. The prediction of the velocity is modeled in a coordinate system $\mathcal{C}$ that is aligned with the contact surface of the corresponding wall. The one-step prediction of the puck velocity is modeled with

$$\Pr_2\left({}^c\dot{\boldsymbol{x}}^p_{k+1}|{}^c\dot{\boldsymbol{x}}^p_k\right) = \mathcal{N}\left(\boldsymbol{\Theta}_2\,{}^c\dot{\boldsymbol{x}}^p_k + \boldsymbol{\theta}_2, \boldsymbol{\Sigma}_2\right), \tag{5.2}$$

where ${}^c\dot{\boldsymbol{x}}^p$ is the puck velocity in the contact-aligned coordinate system. $\boldsymbol{\Theta}_2, \boldsymbol{\theta}_2, \boldsymbol{\Sigma}_2$ are parameters of this mode.

Puck-Mallet Collision

As a third mode, we model the interaction between the puck and the mallet as a collision in which the velocity of the puck changes instantaneously at the time of contact. We also model this mode using a conditional Gaussian distribution

$$\Pr_3\left({}^c\dot{\boldsymbol{x}}^p_{k^+}|{}^c\dot{\boldsymbol{x}}^p_{k^-}, {}^c\dot{\boldsymbol{x}}^m_k\right) = \mathcal{N}\left(\boldsymbol{\Theta}^{pc}_3\,{}^c\dot{\boldsymbol{x}}^p_{k^-} + \boldsymbol{\Theta}^{mc}_3\,{}^c\dot{\boldsymbol{x}}^m_k + \boldsymbol{\theta}_3, \boldsymbol{\Sigma}_3\right). \tag{5.3}$$

The velocities of the puck ${}^c\dot{\boldsymbol{x}}^p$ and of the mallet ${}^c\dot{\boldsymbol{x}}^m$, respectively, are expressed in the contact-aligned coordinate system $\mathcal{C}$. The index k^+ corresponds to time step k after applying the collision model, while k^- describes the instant right before the collision. The model parameters for the third mode are $\boldsymbol{\Theta}^p_3, \boldsymbol{\Theta}^m_3, \boldsymbol{\theta}_3, \boldsymbol{\Sigma}_3$.

5.2.2 Learning Model Parameters from Data

Given recorded trajectories of the puck and the mallet, the data is fragmented into consecutive puck velocity pairs together with the mallet velocity, i.e. $\dot{\boldsymbol{x}}^p_k, \dot{\boldsymbol{x}}^p_{k+1}, \dot{\boldsymbol{x}}^m_k$, and the corresponding mode is assigned to each data sample. As a result, we assume to obtain a dataset $\{\boldsymbol{y}_{i,n}, \boldsymbol{\xi}_{i,n}\}_{n=0}^{N_i}$ for each mode i, where $\boldsymbol{y}_{i,n}$ is the n-th velocity prediction sample for mode i, e.g. $\boldsymbol{y}_1 = \dot{\boldsymbol{x}}^p_{k+1}$, and $\boldsymbol{\xi}_{i,n}$ is the n-th prediction condition sample for mode i, e.g. $\boldsymbol{\xi}_1 = \dot{\boldsymbol{x}}^p_k$. To learn the parameters of the model, we fit a Gaussian distribution to the dataset for each mode modeling the joint probability distribution of prediction and condition with

$$\Pr_i(\boldsymbol{y}_i, \boldsymbol{\xi}_i) = \mathcal{N}\left(\begin{pmatrix}\boldsymbol{\mu}_{\boldsymbol{y}_i}\\ \boldsymbol{\mu}_{\boldsymbol{\xi}_i}\end{pmatrix}, \begin{pmatrix}\boldsymbol{\Sigma}_{\boldsymbol{y}_i} & \boldsymbol{\Sigma}_{\boldsymbol{y}_i\boldsymbol{\xi}_i}\\ \boldsymbol{\Sigma}^\top_{\boldsymbol{y}_i\boldsymbol{\xi}_i} & \boldsymbol{\Sigma}_{\boldsymbol{\xi}_i}\end{pmatrix}\right). \tag{5.4}$$

Given the parameters of the joint probability distribution, the parameters of the linear-Gaussian models can be computed by conditioning the probability distribution on the

input $\boldsymbol{\xi}$. The parameters are thus given by

$$\begin{aligned}\boldsymbol{\Theta}_i &= \boldsymbol{\Sigma}_{\boldsymbol{y}_i\boldsymbol{\xi}_i}\boldsymbol{\Sigma}_{\boldsymbol{\xi}_i}^{-1},\\ \boldsymbol{\theta}_i &= \boldsymbol{\mu}_{\boldsymbol{y}_i} - \boldsymbol{\Sigma}_{\boldsymbol{y}_i\boldsymbol{\xi}_i}\boldsymbol{\Sigma}_{\boldsymbol{\xi}_i}^{-1}\boldsymbol{\mu}_{\boldsymbol{\xi}_i},\\ \boldsymbol{\Sigma}_i &= \boldsymbol{\Sigma}_{\boldsymbol{y}_i} - \boldsymbol{\Sigma}_{\boldsymbol{y}_i\boldsymbol{\xi}_i}\boldsymbol{\Sigma}_{\boldsymbol{\xi}_i}^{-1}\boldsymbol{\Sigma}_{\boldsymbol{y}_i\boldsymbol{\xi}_i}^{\top}.\end{aligned} \tag{5.5}$$

5.2.3 Piecewise-linear Kalman Filtering

The learned linear-Gaussian models allow us to update the estimated state of the puck using the Kalman filter. As a result, an estimate of the puck state at time step k, i.e. $\hat{\boldsymbol{s}}_k = \left(\hat{\boldsymbol{x}}_k^{p\top}, \hat{\dot{\boldsymbol{x}}}_k^{p\top}\right)^{\top}$, is obtained based on a noisy measurement of the puck position $\tilde{\boldsymbol{x}}_k^p$. For this, the mode of the dynamics is detected at each time step such that the corresponding parameters are used within the Kalman filter update. The parameters are translated into linear-Gaussian state-space dynamics, i.e.

$$\Pr_i\left(\boldsymbol{s}_{k+1}|\boldsymbol{s}_k\right) = \mathcal{N}\left(\boldsymbol{A}_i\boldsymbol{s}_k + \boldsymbol{b}_i, \boldsymbol{Q}_i\right), \tag{5.6}$$

with system parameters $\boldsymbol{A}_i, \boldsymbol{b}_i$ and process noise covariance matrix $\boldsymbol{Q}_i$ computed with

$$\boldsymbol{A}_i = \begin{pmatrix}\boldsymbol{A}_{i,\boldsymbol{x}\boldsymbol{x}} & \boldsymbol{A}_{i,\boldsymbol{x}\dot{\boldsymbol{x}}}\\ \boldsymbol{0} & \boldsymbol{\Theta}_i\end{pmatrix};\ \boldsymbol{b}_i = \begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{\theta}_i\end{pmatrix};\ \boldsymbol{Q}_i = \begin{pmatrix}\boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{\Sigma}_i\end{pmatrix}. \tag{5.7}$$

Here, the model parameters $\boldsymbol{A}_{i,\boldsymbol{x}\boldsymbol{x}}, \boldsymbol{A}_{i,\boldsymbol{x}\dot{\boldsymbol{x}}}$ determine the prediction of the puck position at the next time step given the current puck position and velocity. These parameters are derived using numerical integration and are constant.

5.2.4 Probability of Hitting the Goal

For the robot to anticipate whether a candidate shot may lead to scoring a goal, we predict the probability of hitting the goal based on the learned linear-Gaussian puck dynamics. Note that the probability of hitting the goal does not account for a defending opponent. Without loss of generality, suppose that the collision between mallet and puck happens at $k = 0$. Given the puck state at the time of collision $\hat{\boldsymbol{s}}_{0^-}$ and the corresponding mallet state $\boldsymbol{x}_0^m, \dot{\boldsymbol{x}}_0^m$, the expected puck velocity after the collision is computed as defined in (5.3), resulting in the expected puck state $\hat{\boldsymbol{s}}_{0^+}$. By rolling out the discretized stochastic model with

$$\begin{aligned}\hat{\boldsymbol{s}}_{k+1} &= \boldsymbol{A}_{i_k}\hat{\boldsymbol{s}}_k + \boldsymbol{b}_{i_k},\\ \boldsymbol{P}_{k+1} &= \boldsymbol{A}_{i_k}\boldsymbol{P}_k\boldsymbol{A}_{i_k}^{\top} + \boldsymbol{Q}_{i_k},\end{aligned} \tag{5.8}$$

a Gaussian distribution of puck states, i.e. $\boldsymbol{s}_k \sim \mathcal{N}(\hat{\boldsymbol{s}}_k, \boldsymbol{P}_k)$ is obtained for each time step $k > 0$. The rollout is initialized with $\hat{\boldsymbol{s}}_0 = \hat{\boldsymbol{s}}_{0^+}$ and $\boldsymbol{P}_0 = \boldsymbol{Q}_3$, exploiting the separated

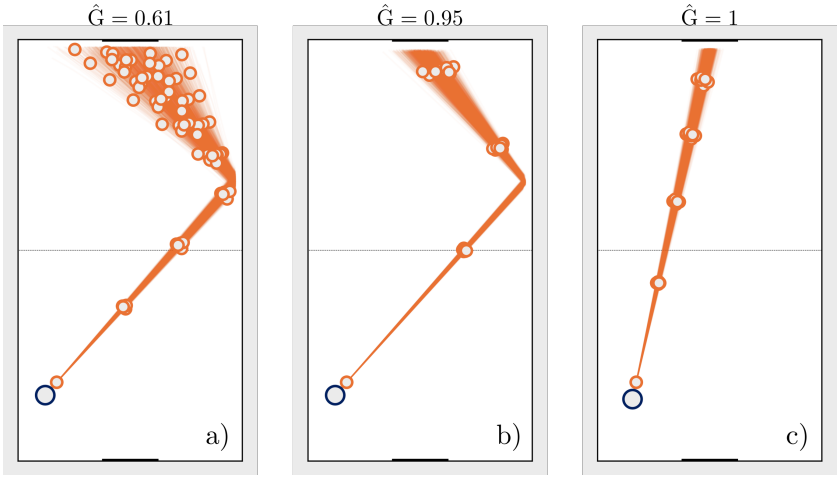


Figure 5.4: A qualitative comparison of the probability of hitting the goal $\hat{\mathrm{G}}$ for different shooting angles and shooting speeds. The shooting angles are indicated by the mallet position • w.r.t. the puck position • at the time of contact. The shooting speed, i.e. the speed of the mallet at the time of contact, is 1.2 $\frac{\mathrm{m}}{\mathrm{s}}$ for a) and c), while the shooting speed is 2 $\frac{\mathrm{m}}{\mathrm{s}}$ for b).

stochastic model of collisions between mallet and puck.

To evaluate the probability of scoring a goal, we perform the stochastic rollout as defined in (5.8) until the expected puck position $\hat{\boldsymbol{x}}_k^p$ crosses the goal line. We denote this time step with k_{goal}. In the following, we denote the probability of scoring a goal, i.e. $\mathrm{G} = 1$, given a puck position as a Bernoulli distribution with

$$\Pr\left(\mathrm{G} = 1 | \boldsymbol{x}_k^p\right) = \begin{cases} 1, & \text{if } \boldsymbol{x}_k^p \in \mathcal{X}_{\text{goal}} \\ 0, & \text{else.} \end{cases} \tag{5.9}$$

The subset in puck position space $\mathcal{X}_{\text{goal}}$ represents the goal region. Consequently, we can compute the probability of scoring a goal given the initial conditions of a shot by marginalizing over the puck position at time step k_{goal} with

$$\Pr\left(\mathrm{G} = 1 | \hat{\boldsymbol{s}}_{0^-}, \boldsymbol{x}_0^m, \dot{\boldsymbol{x}}_0^m\right) = \int_{\mathcal{X}_{\text{goal}}} \Pr\left(\boldsymbol{x}_{k_{\text{goal}}}^p\right) d\boldsymbol{x}_{k_{\text{goal}}}^p. \tag{5.10}$$

We compute the probability in (5.10) using Monte-Carlo approximation by sampling N_{G} puck positions from the Gaussian distribution at prediction time step k_{goal} and counting the number of samples that would hit the goal

$$\Pr\left(\mathrm{G} = 1 | \hat{\boldsymbol{s}}_{0^-}, \boldsymbol{x}_0^m, \dot{\boldsymbol{x}}_0^m\right) \approx \frac{1}{N_{\mathrm{G}}} \sum_{n=1}^{N_{\mathrm{G}}} \Pr\left(\mathrm{G} = 1 | \boldsymbol{x}_{k_{\text{goal}},n}^p\right), \tag{5.11}$$

with $\boldsymbol{x}_{k_{\text{goal}},n}^p \sim \Pr(\boldsymbol{x}_{k_{\text{goal}}}^p)$. In the following, we denote the approximated probability of

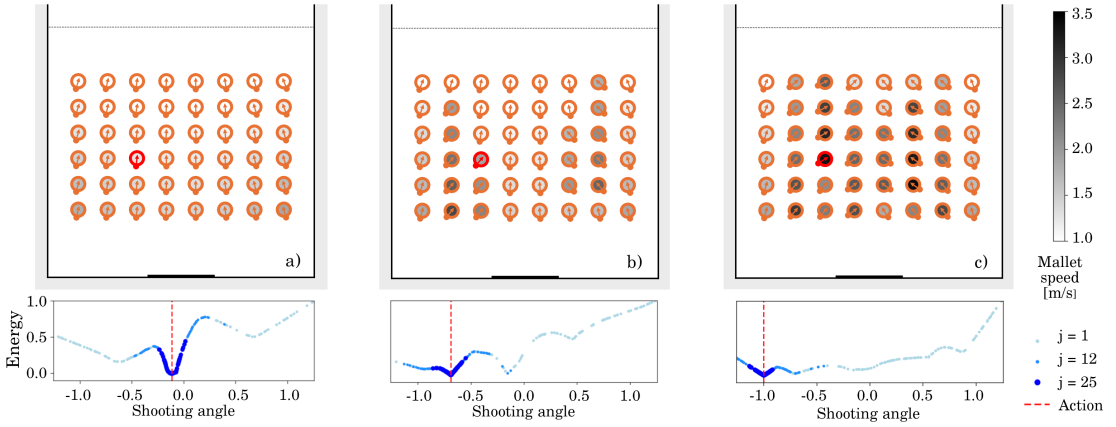


Figure 5.5: Examples of differently tuned shooting plans and corresponding energy landscapes. Shooting direction and mallet speed are displayed for varying initial puck positions. Instance a) evaluates only scoring probability ($\lambda_1 = 1,\ \lambda_2 = 0,\ \beta = 0.5$); b) adds additional weight on expected puck speed at the goal line ($\lambda_1 = 1,\ \lambda_2 = 0.2,\ \beta = 0.5$); c) evaluates only the expected puck speed at the goal line ($\lambda_1 = 0,\ \lambda_2 = 1,\ \beta = 0.5$). The energy landscape and sampling process at time steps $j \in \{1, 12, 25\}$ are visualized for an example shot denoted in red. All pucks are static at $j = 0$.

hitting the goal, corresponding to the right-hand side of (5.11), with $\hat{\mathrm{G}}$.

Figure 5.4 illustrates stochastic rollouts for various initial conditions of a shot. Evaluating $\hat{\mathrm{G}}$ as defined in (5.11), we observe that those initial conditions have a significant effect even if the expected puck trajectory hits the center of the goal for all conditions. Fast shots (Figure 5.4-b) accumulate less uncertainty compared to slow shots (Figure 5.4-a) due to the fact that the modeled process noise is constant over time. Compared to bank shots, direct shots accumulate less uncertainty during rollout since collisions with a wall add significant process noise (Figure 5.4-c).

5.3 Fast Contact Planning under Uncertainty

The learned dynamics model enables the prediction of uncertain puck trajectories for contact planning. In particular, we aim to find contact states of the mallet that result in desired puck trajectories after contact. The proposed contact planning module is based on stochastic optimal control, optimizing for the mallet state at contact. We combine the optimization of the contact state with a model-based robot controller that drives the robot to the desired contact state at the desired contact time (cf. Figure 5.2).

5.3.1 Stochastic Optimal Control for Shooting

Given the desired time of contact and the corresponding estimate of the puck state $\boldsymbol{s}_{0-}$ at that time, we pose contact planning for shooting as a stochastic optimal control problem searching for the mallet state $\boldsymbol{x}_0^m, \dot{\boldsymbol{x}}_0^m$ at the time of contact. For this, we aim to maximize a tradeoff between the probability of hitting the goal $\hat{\mathrm{G}}$ and the expected puck speed v_{puck} at the goal line. The expected puck speed is computed as the norm of the mean puck velocity at k_{goal} according to Sec. 5.2.4. While the probability of hitting the goal $\hat{\mathrm{G}}$ does not account for a defending opponent, we use the speed of the puck as a measure of the difficulty of defending against the shot. The stochastic optimal control problem is given as

$$\begin{aligned} \max_{\boldsymbol{x}_0^m, \dot{\boldsymbol{x}}_0^m} \quad & \lambda_1 \hat{\mathrm{G}} + \lambda_2 v_{\mathrm{puck}} \\ \text{s.t.} \quad & \hat{\mathrm{G}} > \beta, \end{aligned} \tag{5.12}$$

where we deploy an additional chance constraint to enforce the probability of hitting the goal to be higher than a threshold β based on the learned stochastic model. The weights λ_1 and λ_2 are used for tuning for the desired behavior. Based on the qualitative comparison illustrated in Figure 5.4, we expect that solely optimizing for the probability of hitting the goal results only in direct shots, as bank shots induce uncertainty. Yet, due to the kinematics of the robot, the puck speed may be increased with bank shots. Thus, depending on the puck state, the tuned objective can produce both straight shots and bank shots, increasing the chances of scoring.

5.3.2 Shooting Angle as Reduced Action Space

The goal of the shooting policy is to find the optimal mallet state at the time of contact, i.e. $\boldsymbol{x}_0^m$ and $\dot{\boldsymbol{x}}_0^m$, respectively. Due to the underlying contact geometry and constraints, for a given puck position $\hat{\boldsymbol{x}}_0^p$ we parameterize the mallet position as a shooting angle $u \in \mathcal{U}$ between the mallet and puck. Note that a shooting angle of $u = 0$ corresponds to a straight shot that is parallel to the side walls of the table. We reduce the dimensionality of the action space further by imposing two heuristic constraints on the mallet velocity $\dot{\boldsymbol{x}}_0^m$: *i)* The mallet velocity at the time of contact aligns with the shooting angle, such that $\dot{\boldsymbol{x}}_0^m = v(\cos u, \sin u)^\top$ with scalar velocity $v > 0$ encoding the norm of the mallet velocity. While this constraint excludes shooting angles that are not aligned with the mallet velocity, it enforces maximum transmission of kinetic energy from the robot to the puck; *ii)* We impose that the norm of the mallet velocity is maximal given a shooting configuration $\boldsymbol{q}_0$ of the robot and velocity limits $\dot{\mathcal{Q}}$ of the joints of the robot,

such that

$$\begin{aligned} v^* &= \max v \\ \text{s.t.} \quad & v\boldsymbol{e}_u = \boldsymbol{J}(\boldsymbol{q}_0)\dot{\boldsymbol{q}}_0, \\ & \dot{\boldsymbol{q}}_0 \in \dot{\mathcal{Q}}. \end{aligned} \tag{5.13}$$

Note that the unit vector $\boldsymbol{e}_u \in \mathbb{R}^3$ encodes the shooting direction including zero contribution in the z-direction. Accordingly, $\boldsymbol{J}(\boldsymbol{q}_0) \in \mathbb{R}^{3\times n_{\text{dof}}}$ corresponds to the Jacobian w.r.t. the Cartesian position of the mallet.

As a result, the shooting angle u is the action that we optimize for. With the imposed constraints, a shooting angle uniquely maps to a mallet state at the time of contact. Thus, in the following, we denote the probability of scoring a goal as a function of the shooting angle with $\Pr\left(\mathrm{G}=1|\hat{\boldsymbol{s}}_{0^-}, u\right)$.

5.3.3 Training an Energy-based Shooting Policy

The long-horizon predictions required to plan shooting actions make it difficult to operate at a rate that is sufficient for agile behavior. We address this challenge by training an energy-based model to reproduce the solutions to the stochastic optimal shooting problem in (5.12) in realtime. Due to implicit policy representation, energy-based models are particularly well-suited for multi-modal solution spaces Florence et al. (2022). In the case of shooting, different modes of the policy include straight shots, single bank shots, and double bank shots.

We train the energy-based model by solving the computationally expensive shooting angle optimization offline and using the results as training data. Namely, we first generate a dataset of shooting angles for N different scenarios, i.e. puck states at the time of contact $\{\boldsymbol{s}_i\}_{i=1}^N$. Due to the one-dimensional parametrization of the action space, we efficiently explore the space of shooting angles for each initial puck state by sampling M candidate shooting angles $\{u_i^j\}_{j=1}^M$. We subsequently compute the stochastic rollout of the puck trajectory as presented in Sec. 5.2.4 and evaluate the objective and chance constraint from (5.12). The best-performing sample $\hat{u}_i$ is then used as a positive example for training, and the remaining $M-1$ samples as negative counter-examples. We finally obtain a dataset of $M \times N$ state-action pairs $\{\boldsymbol{s}_i, \hat{u}_i, \{u_i^j\}_{j=1}^{M-1}\}_{i=1}^N$ and train the energy model $E_\theta(\boldsymbol{s}, u)$ using an InfoNCE-style (van den Oord et al. (2018)) loss

$$\mathcal{L}_{\text{InfoNCE}} = \sum_{i=1}^{N} -\log\left(\tilde{p}_\theta\Big(\hat{u}_i|\boldsymbol{s}_i, \{u_i^j\}_{j=1}^{M-1}\Big)\right), \tag{5.14}$$

Algorithm 2: Shooting policy (EBM inference)

Input: Puck state $\hat{\boldsymbol{s}}_{0^-}$, variance σ, samples $\{\tilde{p}_i, \tilde{u}_i\}_{i=1}^N$
Output: Shooting angle $\hat{u}$, new samples $\{\tilde{p}_i, \tilde{u}_i\}_{i=1}^N$
$\{\tilde{u}_i\}_{i=1}^N \leftarrow\sim$ Multinomial$(N, \{\tilde{p}_i\}_{i=1}^N, \{\tilde{u}_i\}_{i=1}^N)$
$\{\tilde{u}_i\}_{i=1}^N \leftarrow \{\tilde{u}_i\}_{i=1}^N + \sim \mathcal{N}(0, \sigma)$
$\{\tilde{u}_i\}_{i=1}^N \leftarrow$ clip $\{\tilde{u}_i\}_{i=1}^N$ to $\mathcal{U}$
$\{E_i\}_{i=1}^N \leftarrow \{E_\theta(\hat{\boldsymbol{s}}_{0^-}, \tilde{u}_i)\}_{i=1}^N$
$\{\tilde{p}_i\}_{i=1}^N \leftarrow$ softmax$(-\{E_i\}_{i=1}^N)$
$\hat{u} \leftarrow$ argmax$(\{\tilde{p}_i\}, \{\tilde{u}_i\})$

where $\tilde{p}_\theta\left(\hat{u}_i | \boldsymbol{s}_i, \{u_i^j\}_{j=1}^{M-1}\right)$ represents a likelihood with

$$\tilde{p}_\theta\left(\hat{u}_i | \boldsymbol{s}_i, \{u_j^i\}_{j=1}^{M-1}\right) = \frac{e^{-E_\theta(\boldsymbol{s}_i, \hat{u}_i)}}{e^{-E_\theta(\boldsymbol{s}_i, \hat{u}_i)} + \sum_{j=1}^{M-1} e^{-E_\theta(\boldsymbol{s}_i, u_j^i)}}. \tag{5.15}$$

The described loss function reduces energy $E_\theta(\boldsymbol{s}, u)$ for shooting angles that solve the optimization problem in (5.12), while increasing the energy of non-optimal shooting angles. Once the model is trained, this allows us to infer optimal shooting angles using sampling-based optimization.

5.3.4 Online Inference with Warm-Starting

To solve (5.12) given the estimated puck state $\hat{\boldsymbol{s}}_{0^-}$, we search for the state-action pair that minimizes energy, i.e.

$$\hat{u} = \arg\min_{u \in \mathcal{U}} E_\theta(\hat{\boldsymbol{s}}_{0^-}, u). \tag{5.16}$$

For realtime optimization, we leverage direct access to the learned energy landscape of the EBM by executing sampling iterations concurrently with other components of the control loop. This allows us to simultaneously refine contact plans as trajectories are being executed on the robot. The online retrieval of optimal shooting angles is based on the derivative-free optimization procedures used in Florence et al. (2022). As in the offline scenario, we initiate an online shooting action by uniformly sampling N candidate actions. Based on the corresponding energies, candidates are resampled with replacement to warm-start optimization at each following . To converge towards a solution with minimum implicit energy, reductions to the sampling scale σ are applied at each time step, keeping the optimal contact angle $\hat{u}$ as a reference for the mid-level trajectory planner. A full iteration of the EBM optimization is outlined in Algorithm 2. We observe that the learned energy models and utilized optimization procedure efficiently retrieve multimodal contact plans to produce desired behaviors, as shown in Figure 5.5.

5.4 Experiments

This section details the simulated and real-world experiments used to validate our approach in an online contact planning setting. We evaluate the shooting performance of a robot arm controlled by our framework and compare it against state-of-the-art approaches for robot air hockey.

5.4.1 Implementation Details

Data Collection for Puck Dynamics

We use data collected in a physics-based simulator to learn model parameters of puck dynamics as presented in Sec. 5.2. One set of data is collected by randomly moving the robot's end-effector into contact with the puck and the other set of data is collected without moving the robot and by initializing the puck with a high random velocity. In total, the training set consists of 100 episodes with 50 time steps each, which corresponds to a total of 100 seconds of observations of the puck dynamics.

EBM Architecture and Training

The energy-based shooting model consists of a multilayer perceptron with 2 hidden, fully connected layers of 128 neurons each. The model is trained on $N = 3000$ initial puck states, with $M = 100$ action samples. Training required between 500 and 1000 epochs to converge for satisfactory performance using the Adam (Kingma and Ba (2014)) optimizer with a decaying learning rate.

5.4.2 Experimental Setup

The experiment is conducted with a KUKA iiwa14 LBR manipulator equipped with a mallet end-effector that is attached to a passive joint for seamless contact with the table surface. Experiments are carried out in a simulated *MuJoCo* environment and on a real-world setup (cf. Figure 5.1) for evaluation of sim2real transfer. We evaluate three instances of the proposed approach by using different parameters for the chance-constrained optimization problem in (5.12). **Ours #1**: a *conservative* policy that prioritizes accuracy ($\lambda_1 = 1, \lambda_2 = 0, \beta = 0.5$); **Ours #2**: a *balanced* policy that compromises between accuracy and puck speed ($\lambda_1 = 1, \lambda_2 = 0.2, \beta = 0.5$); and **Ours #3**: an *aggressive* policy that prioritizes puck speed ($\lambda_1 = 0, \lambda_2 = 1, \beta = 0.5$). Simulated results are also illustrated in Figure 5.5 for initial puck velocities of zero. We compare the three instances of our contact planner with: **CB**, a baseline that utilizes conventional planning and control methods (Liu et al. (2021)); and **ATACOM**, a reinforcement learning approach for learning a robot policy (Liu et al. (2022)). Note that

	Score	Puck Speed $\left[\frac{\text{m}}{\text{s}}\right]$	Num. Banks
CB	0.51	0.52 ± 0.24	0.00
Atacom	0.90	0.55 ± 0.05	0.00
Ours #1	**0.93**	1.00 ± 0.20	0.00
Ours #2	0.80	1.44 ± 0.63	0.53
Ours #3	0.61	$\mathbf{1.97 \pm 0.49}$	**1.13**

Table 5.1: Simulated experiments.

	Score	Puck Speed $\left[\frac{\text{m}}{\text{s}}\right]$	Num. Banks
	0.49	1.09 ± 0.24	0.00
	0.13	0.66 ± 0.15	0.31
	0.78	1.72 ± 0.20	0.00
	0.60	2.02 ± 0.35	0.37
	0.31	$\mathbf{2.37 \pm 0.50}$	**0.90**

Table 5.2: Real-world experiments.

the ATACOM policy is trained in simulation and deployed in the physical experiment without additional tuning or retraining.

We perform 100 shots with each policy and report the accuracy score, puck speed at the goal line, and the number of bank reflections for successful shots. Each shot is initialized by placing the puck within a grid in front of the robot. Due to imperfect air flow on the air hockey table, the puck moves after release, requiring the controlled robot to adapt for a good shot.

5.4.3 Experimental Results

Recorded metrics are reported in Table 5.1 for the simulated environment and in Table 5.2 for the real-world environment. Compared to **CB** and **ATACOM**, we observe that our framework is capable of achieving higher scoring accuracy and significantly higher puck speeds in both environments. It can be seen that different instances of our policy obtain either a high score or high puck speeds according to the corresponding parameters of the stochastic optimal control problem. For example, when compared to **Ours #1**, it can be seen that **Ours #3** compromises scoring accuracy for faster puck speeds and a high number of bank reflections, potentially making the shots more difficult to defend against. The higher number of bank reflections produced by **Ours #3** indicates that the robot kinematics allow for higher shooting speeds when hitting laterally, at the risk of missing the goal due to uncertainty gained with every bank reflection. Note that the score of **Ours #3** is lower than the score chance threshold $\beta = 0.5$ for this instance. This indicates that the learned model either has an error in the nominal dynamics or expects too little uncertainty gain due to bank reflections. We further note a decrease in performance for all agents due to the *sim2real* gap, with **ATACOM** showing the highest sensitivity to transfer as it requires fine-tuning on the real environment. **CB** shows the least decrease in performance, as it is parameterized for the real system. However, note that the shooting trajectory optimization loop of **CB** is slower than required to run at 50 Hz, making it prone to errors due to the puck moving unpredictably during the shooting motion. Additionally, we note higher puck speeds in the real setting for all agents as a result of differences in real and simulated contact dynamics. Examples of physical shots of all approaches can be found in the supplementary video.

6 Belief-space Planning through Contacts

Publication Note

The material presented in this chapter is adapted from the following publication:

- Jankowski, J., Brudermüller, L., Hawes, N., and Calinon, S. (2024a). Robust pushing: Exploiting quasi-static belief dynamics and contact-informed optimization

L. Brudermüller helped in writing the paper.

Supplementary Material

Video related to this chapter is available at: https://youtu.be/-37Gm7NX6eg.

Non-prehensile manipulation such as pushing is typically subject to uncertain, non-smooth dynamics. However, modeling the uncertainty of the dynamics typically results in intractable belief dynamics, making data-efficient planning under uncertainty difficult. Chapter 5 addressed the problem of planning for the optimal impact through stochastic object dynamics. This chapter dives deep into the mechanics of slow pushing and how uncertainty propagates through such contact dynamics. We focus on the problem of efficiently generating robust open-loop pushing plans using VP-STO (see Chapter 4). First, we investigate how the belief over object configurations propagates through quasi-static contact dynamics. We exploit the simplified dynamics to predict the variance of the object configuration without sampling from a perturbation distribution. In a sampling-based trajectory optimization algorithm, the gain of the variance is constrained in order to enforce the robustness of the plan. Second, we propose an informed trajectory sampling mechanism for drawing robot trajectories that are likely to make contact with the object. This sampling mechanism is shown to significantly improve chances of finding robust solutions, especially when making-and-breaking contacts is required. We demonstrate that the proposed approach is able to synthesize bimanual pushing trajectories, resulting in successful long-horizon pushing maneuvers without exteroceptive feedback such as vision or tactile feedback. We furthermore deploy the proposed approach in a model-predictive control scheme, demonstrating additional robustness against unmodeled perturbations.

Enabling robots to interact with the world physically is a key challenge in robotics. In particular, the ability to move, reorient, or localize objects through contact is a

fundamental capability for robots to perform tasks in unstructured environments. Model-based planning techniques aim to synthesize robot control actions by using a given model to reason over anticipated outcomes of actions. However, there are two core challenges of model-based planning through contacts. First, contact dynamics are inherently discontinuous, i.e. a robot control action may have no effect on the state of a target object if the robot does not make contact. This translates into a vanishing gradient of the associated manipulation objective, making traditional gradient-based optimization difficult.

Second, the mechanics of non-prehensile contacts are subject to uncertainty. This is due to the fact that dynamic effects such as friction between surfaces are difficult to predict, especially when manipulating objects without knowing their physical properties, such as friction coefficients or mass distribution (Lynch and Mason (1995); Dogar and Srinivasa (2011); Ha et al. (2020)). Thus, when generating open-loop plans by assuming an accurate model of the dynamics, those plans are likely to fail as they do not take into account the underlying uncertainty. In Rodriguez (2021), this effect is described with an experiment of repeatedly picking and placing a queen chess piece, which fails if the queen is picked from the top. In contrast, picking the queen from the side stabilizes the repeated pick-and-place process. Uncertainty about the physical parameters of objects is particularly unavoidable when robots interact with objects for the first time. This raises the question of how to let robots deal with this uncertainty autonomously. Deploying deterministic models in fast feedback loops is an implicit approach to compensating for uncertain dynamics, such as done in model-predictive control schemes. Yet they require accurate sensory measurements and achieve robustness against perturbations or modeling errors only by correcting the observed control errors. Modeling uncertainties and reasoning over an anticipated distribution of outcomes, i.e. a belief, is a more explicit way of coping with uncertain dynamics. Planning in belief space enables open-loop execution of plans while still exhibiting robustness against modeled perturbations, thus not requiring sensors. Moreover, control errors are anticipated at planning time and can thus be prevented before they happen, potentially making manipulation under uncertainty more efficient. Closed-loop control in belief space can then be achieved by continuously updating the belief based on observations and subsequent replanning, combining the benefits of reactivity with respect to unmodeled perturbations and the effectiveness of preventing control errors at planning time.

In this chapter, we present an approach for modeling the uncertainty of contact dynamics in order to synthesize robust manipulation behavior. Toward this end, we make the following contributions:

i) We study how a belief over an object's configuration propagates through uncertain contact dynamics. We derive a prediction of the variance of object configurations upon contact, allowing us to anticipate the reduction of uncertainty without sampling

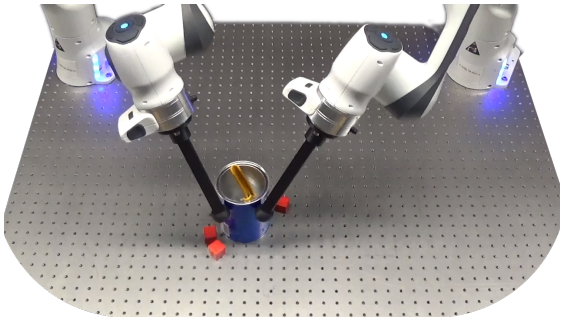


Figure 6.1: Our model-based optimization approach synthesizes bimanual pushing trajectories by controlling the variance of the object configuration, without explicitly modeling contact modes. We show that the robustness of the pushing trajectories is sufficient to successfully push an object over long horizons. The red cubes on the table and the yellow content of the can act as additional perturbations to the contact dynamics.

perturbations to the contact dynamics.

ii) We introduce a contact prior for sampling candidate robot trajectories that are likely to create contacts between the robot and the object.

iii) Last, we propose a sampling-based trajectory optimization algorithm that constrains solutions to be robust based on the predicted variance (*i)*). The informed trajectory distribution (*ii)*) serves as a proposal distribution that guides the sampling-based optimization process.

In real-world experiments we demonstrate that the proposed approach is able to synthesize robust bimanual pushing trajectories in only a few seconds of planning time, consisting of long-horizon (up to 100 seconds) open-loop pushing maneuvers that include making and breaking contacts (cf. Figure 6.1). The experimental results show that the interplay of *i)* and *ii)* is crucial for synthesizing robust behavior. We furthermore show that the proposed planning algorithm is fast enough to be used in a model-predictive control loop, enabling a combination of reactivity and anticipatory robustness. To the best of our knowledge, the proposed algorithm is the first model-based planning approach that is able to synthesize robust plans for contact-rich manipulation without pre-defined manipulation primitives.

6.1 Related Work

6.1.1 Contact-rich Manipulation

To plan and control physical interactions such as contacts, model-based algorithms aim to exploit these models to synthesize manipulation behavior. Optimization-based approaches have shown successful manipulation planning capabilities when the desired behavior can be found via local optimization (Hogan and Rodriguez (2020); Aydinoglu and Posa (2022); Aydinoglu et al. (2022); Cleac'h et al. (2021)) or when contact modes can be represented as a small set of discrete decision variables (Marcucci

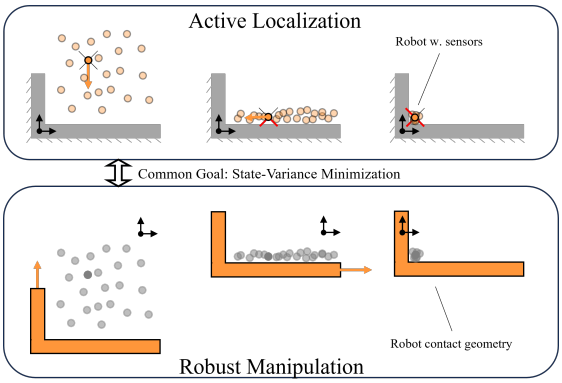


Figure 6.2: Analogies between active robot localization and robust manipulation. Both can be formulated as a belief space planning problem, where the objective is to decrease the uncertainty of the belief over time. While active localization approaches are based on observation models, robust manipulation exploits favorable contact dynamics to achieve the same goal.

et al. (2017); Migimatsu and Bohg (2020); Toussaint et al. (2020, 2022a); Chen et al. (2021)). In the case of local optimization, however, most methods rely on gradients which require a smoothed approximation of the contact dynamics. Moreover, they rely on good initializations, as control actions that do not result in the robot making contact with the object will yield a vanishing gradient of the associated manipulation objective. In contrast, when reformulating the manipulation problem as finding the optimal sequence of contact modes, the difficulty lies in a combinatorial explosion when frequent switching of contact modes is necessary, i.e. making-and-breaking contacts, or if multiple contact points are involved.

Recently, sampling-based planning and control algorithms have been explored for contact-rich manipulation tasks as a gradient-free approach to cope with discontinuous cost landscapes. This offers a framework for combining stochastic sampling and optimization, supporting the search over contact-rich manipulation actions. In Pang et al. (2023), the authors propose to use a quasi-dynamic contact model to efficiently simulate physical interactions during manipulation. Plans are then synthesized with an adaptation of the RRT algorithm. While the planning algorithm is able to generate contact-rich manipulation plans including making and breaking contacts, the underlying contact model is assumed to be accurate, resulting in plans that are not robust to inaccuracies in the contact model. In another stream of research on controlling contact-rich manipulation, sampling-based optimization has been explored in model-predictive control schemes to exploit parallel computing opportunities (Bhardwaj et al. (2022); Howell et al. (2022); Jankowski et al. (2023)). In this chapter, we build upon our previous work on via-point-based stochastic trajectory optimization (*VP-STO*) as a tool for efficiently optimizing robot trajectories without requiring gradients of the manipulation cost with respect to the manipulation action (Jankowski et al. (2023)).

6.1.2 Robust Manipulation

In robotic manipulation, uncertainty is a dominant aspect that arises from the complex and hybrid nature of modeling physical interactions (Rodriguez (2021)). Robust ma-

nipulation aims at exploiting particular contact configurations that naturally reduce errors in manipulation tasks. In Erdmann and Mason (1988), the authors exploit the geometry of a tray with physical boundaries to reorient a tool without sensory feedback. Lynch and Mason (1995) exploit line contacts between a robot and polygonal objects to generate robust pushing plans. In Dogar and Srinivasa (2011), the geometry of a half-open gripper is exploited to actively funnel the probability distribution over object locations between the two fingers and the palm with a push-grasp. Ha et al. (2020) model the uncertainty in an underactuated system with additive noise on the robot control actions and approximate the probability distribution over state trajectories with a normal distribution around a particular contact mode. Logic geometric programming is then used to find the most robust contact mode from a set of pre-defined candidate modes. These strategies have in common that favorable contact geometries are used to create natural contact dynamics that effectively decrease the uncertainty of the manipulation system over time without the need for sensors. However, the above approaches rely on a pre-programmed set of robust behaviors that are tailored to the robot contact geometry.

In this chapter, we achieve robust manipulation as the result of optimizing over an object belief. Belief-space planning in robotics is concerned with modeling the state of the robot and its workspace via probability distributions, e.g. for active localization. Sensor measurements of the robot are then used to reject or confirm possible states based on an observation model to reduce uncertainty. Figure 6.2 illustrates the analogy of active localization and robust manipulation. Both can be formulated as a belief-space planning problem, where the common objective is to decrease the state-variance over time. While the goal is to minimize the uncertainty of the robot's own state in localization problems, robust manipulation aims to minimize the uncertainty of the object state. An even more important difference lies in the way the belief is updated over time. Instead of using observation models to rule out possible states of the robot, robust manipulation exploits natural invariances in the contact dynamics to let possible object states converge to a single state.

6.1.3 Modeling Uncertainty in Contact Dynamics

Modeling the uncertainty in contact dynamics is a key aspect of synthesizing robust robot behavior. In many reinforcement learning approaches (Haarnoja et al. (2019); Schulman et al. (2015, 2017)), domain randomization is a natural way of informing the skill learning process about all the possibilities that may be encountered when moving from simulation to reality. Domain randomization may include probability distributions over parameters of the dynamics model such as friction coefficients, object mass, or object geometry (Andrychowicz et al. (2020); Muratore et al. (2022)). By simulating a large number of combinations of policy samples and domain samples, the policy ideally converges to a behavior that is robust against the uncertainty modeled

through domain randomization. We believe that domain randomization as an interface for modeling uncertainties in physical interactions is the key reason why reinforcement learning techniques show more advanced manipulation skills in the real world such as in-hand manipulation (Handa et al. (2023)) compared to model-based planning and control techniques that typically assume an accurate model. Hence, we aim to bridge the gap between modeling uncertainty in contact dynamics and ad-hoc planning and control techniques that do not require data-inefficient offline training cycles by optimizing over statistical properties of a belief without sampling perturbations.

6.2 Problem Formulation & Approach

We consider an underactuated manipulation system with n^r_{dof} actuated degrees of freedom for the robot, and n^o_{dof} unactuated degrees of freedom for the object to be manipulated. We are interested in controlling the configuration of the object $\boldsymbol{q}^o \in \mathbb{R}^{n^o_{\text{dof}}}$ by executing robot control commands $\boldsymbol{u} \in \mathbb{R}^{n^r_{\text{dof}}}$. The initial object configuration and the contact dynamics are subject to uncertainty, such that the object configuration at time step k is a random variable that is described through the *belief* $b_k = p(\boldsymbol{q}^o_k)$. We formulate the planning problem as a stochastic optimal control problem over a horizon of K time steps:

$$\min_{\boldsymbol{u}_{0:K-1}} \mathrm{E}_{b_K} \left[\left(\boldsymbol{q}^o_K - \boldsymbol{q}^o_{\text{des}} \right)^\top \left(\boldsymbol{q}^o_K - \boldsymbol{q}^o_{\text{des}} \right) \right]. \tag{6.1}$$

We are optimizing for an open-loop control trajectory $\boldsymbol{u}_{0:K-1}$ such that the expected control error at time step K is minimal. In order to evaluate the expected control error in (6.1) given a control trajectory, the belief over object positions is to be propagated through the stochastic contact dynamics. However, as contact dynamics are inherently non-smooth, propagating the belief through contacts in closed form is intractable. Another way to explicitly evaluate the expected cost is to sample a large number of stochastic rollouts. Yet, this is problematic due to the fact that the evaluation of a single robot control trajectory becomes not only inefficient, but also stochastic.

A key to our approach is the separation of the stochastic optimal control problem in (6.1) into a mean control problem and a variance control problem (Okamoto et al. (2018); Shirai et al. (2023)). The objective of the stochastic optimal control problem is separated as follows:

$$\min_{\boldsymbol{u}_{0:K-1}} \left(\mathrm{E}_{b_K} \left[\boldsymbol{q}^o_K \right] - \boldsymbol{q}^o_{\text{des}} \right)^\top \left(\mathrm{E}_{b_K} \left[\boldsymbol{q}^o_K \right] - \boldsymbol{q}^o_{\text{des}} \right) + \mathrm{V}_{b_K} \left[\boldsymbol{q}^o_K \right]. \tag{6.2}$$

We provide a more detailed derivation of the steps from (6.1) to (6.2) in the appendix. The first term in (6.2) corresponds to the mean control problem, which refers to planning for the expected object configuration $\mathrm{E}_{b_K} \left[\boldsymbol{q}^o_K \right]$, i.e. the configuration that is obtained if the mean initial configuration and the nominal contact dynamics are used.

Evaluating the mean control objective does not require the propagation of the belief or the sampling of large numbers of stochastic rollouts of the state. In other words, the mean control problem resembles a deterministic optimal control problem that assumes an accurate dynamics model, while the stochasticity in the original problem is captured by the variance control problem represented by the second term in (6.2). Note that the variance control problem, i.e. minimizing $\mathrm{V}_{b_K}\left[\boldsymbol{q}_K^o\right]$, is independent of the desired object configuration $\boldsymbol{q}_{\mathrm{des}}^o$. The *variance* of the object configuration is defined as:

$$\mathrm{V}_{b_k}[\boldsymbol{q}_k^o] = \mathrm{E}_{b_k}\left[(\boldsymbol{q}_k^o - \mathrm{E}_{b_k}[\boldsymbol{q}_k^o])^\top (\boldsymbol{q}_k^o - \mathrm{E}_{b_k}[\boldsymbol{q}_k^o])\right]. \tag{6.3}$$

Thus, the stochastic optimal control problem can be solved by controlling the nominal object configuration while also steering its variance. The variance is a scalar measure of the second statistical moment of the belief and can be interpreted as the uncertainty that the system has about the object's configuration. However, note that computing the variance at time step K still requires the propagation of the belief or a Monte-Carlo approximation of the stochastic dynamics. In Sec. 6.3, we present an approach that approximates the variance of the object configuration over time without sampling perturbations to the contact dynamics, resulting in efficient and deterministic rollouts of the nominal dynamics and the approximated variance. In Sec. 6.4 and Sec. 6.5, we exploit the approximated variance in a sampling-based trajectory optimization scheme for synthesizing robust robot trajectories.

6.3 Belief Dynamics through Contacts

Given a robot control action and a belief over an object's configuration, we are interested in predicting the mean and the variance of the object's configuration at the consecutive time step. For this, we first introduce a quasi-static model for the object dynamics, i.e. predicting the nominal object configuration given a robot configuration. We then model the uncertainty of the object dynamics through additive perturbations on the object configuration if the robot makes contact with the object. Last, we derive a deterministic prediction of the variance given the current belief, a control action, and the statistical properties of the perturbations.

6.3.1 Stochastic Quasi-Static Dynamics for Pushing

Quasi-static and quasi-dynamic models have been used to simplify the prediction of slow physical interactions between robots and objects (Mason (2001); Koval et al. (2016); Hogan and Rodriguez (2020); Cheng et al. (2021); Pang (2021); Pang et al. (2023)). Both classes of models assume that effects that are related to velocities and accelerations can be neglected as they do not affect the outcome of the prediction. This limits the range of applications to slow interactions such as pushing tasks, insertion tasks, or

in-hand manipulation. Yet, the benefits of quasi-static and quasi-dynamic models are the lower dimensionality of the system state, i.e. half the number of states compared to second-order dynamics models (e.g. Todorov et al. (2012)), and the lower temporal resolution required to compute stable predictions of the system state. Both aspects effectively allow for faster-simulated rollouts of robot plans and thus for more efficient model-based trajectory optimization. Such models oftentimes denote the state as $\boldsymbol{q} = (\boldsymbol{q}^r, \boldsymbol{q}^o)$. It decomposes into the position of the actuated degrees of freedom of the robot $\boldsymbol{q}^r \in \mathbb{R}^{n^r_{\text{dof}}}$, and the position of the unactuated degrees of freedom of the object(s) $\boldsymbol{q}^o \in \mathbb{R}^{n^o_{\text{dof}}}$. The discretized dynamics are consequently given in the form of

$$\begin{pmatrix} \boldsymbol{q}^r_+ \\ \boldsymbol{q}^o_+ \end{pmatrix} = \boldsymbol{f}\left(\begin{pmatrix} \boldsymbol{q}^r \\ \boldsymbol{q}^o \end{pmatrix}, \boldsymbol{u} \right). \tag{6.4}$$

$\boldsymbol{q}^r_+$, $\boldsymbol{q}^o_+$ are the predicted robot and object configurations at the consecutive time step, respectively. In Pang et al. (2023), the input $\boldsymbol{u} \in \mathbb{R}^{n^r_{\text{dof}}}$ is defined as the commanded robot configuration. The robot is assumed to be controlled by a low-level impedance controller (Hogan (1984)), such that the robot can be modeled as an impedance for the contact dynamics. In this chapter, we further simplify the quasi-dynamic contact dynamics in Pang et al. (2023) by assuming infinite stiffness of the controlled robot. As a consequence, contacts with objects are assumed to not affect the robot state itself, but only the configuration of the object. This assumption is particularly realistic when pushing lightweight objects with a stiff robot impedance controller. The high-stiffness assumption induces that the robot is able to reach a desired robot position even when being in contact with an object, i.e. $\boldsymbol{q}^r_+ = \boldsymbol{u}$. The benefit of modeling the contact interactions in such a way is that the joint robot-object dynamics reduce to solely *object dynamics*. This further simplifies the simulation of contacts, as the non-penetration constraint in the dynamics now only applies to the object. This turns the quasi-dynamic contact dynamics into quasi-static contact dynamics, as we remove the dependency on time. We will further exploit this decoupling of the dynamics in the subsequent derivation of transition probabilities for the object. Yet, note that this assumption breaks when contacts between the robot and the environment significantly affect the robot's state, e.g. when trying to move into a solid wall. Furthermore, enclosing grasps, i.e. making contact with one object from two opposing sides, will also break the high-stiffness assumption as the two contact points will affect each other. Due to these limitations, we focus the experiments on planar pushing under uncertainty.

Nominal Object Dynamics

The discretized quasi-static contact dynamics of the object are

$$\begin{pmatrix} \boldsymbol{q}_+^r \\ \boldsymbol{q}_+^o \end{pmatrix} = \begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{q}^o + \delta \boldsymbol{q}^o \end{pmatrix}; \tag{6.5a}$$

$$\delta \boldsymbol{q}^o = \arg\min_{\delta \tilde{\boldsymbol{q}}^o} \delta \tilde{\boldsymbol{q}}^{o\top} \boldsymbol{M}(\boldsymbol{q}^o) \delta \tilde{\boldsymbol{q}}^o, \tag{6.5b}$$

$$\text{s.t.} \quad d(\boldsymbol{u}, \boldsymbol{q}^o + \delta \tilde{\boldsymbol{q}}^o) \geq 0. \tag{6.5c}$$

The control inputs $\boldsymbol{u} \in \mathbb{R}^{n_{\text{dof}}^r}$ to the system dynamics are defined by the commanded joint positions of the robot. In (6.5b), $\boldsymbol{M}(\boldsymbol{q}^o)$ is the inertia matrix of the object with respect to its current configuration $\boldsymbol{q}^o$. Thus, the objective in (6.5b) aims to minimize the work required to overcome the friction between the object's surface and the surface of the environment, e.g. the table the object is placed on. $d(\boldsymbol{q}^r, \boldsymbol{q}^o)$ measures the shortest signed distance between the robot and the object, and thus (6.5c) incorporates the non-penetration constraint, i.e. the robot does not penetrate a rigid object. As the quasi-static robot dynamics in (6.5a) are decoupled from the object configuration, the non-penetration constraint in (6.5c) does not depend on the previous robot state, but only on the control input $\boldsymbol{u}$. This simplification allows us to represent the system dynamics with the next desired robot position as the control input and the object position as the sole state variable. The quasi-static contact dynamics of the object in (6.5a)–(6.5c) can thus be summarized into the nominal forward dynamics of the object configuration:

$$\boldsymbol{q}_+^o = \boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u}). \tag{6.6}$$

In the following, we exploit the simplified mathematical structure of the quasi-static contact dynamics to model uncertainty and to analyze how object belief states propagate through the contact dynamics.

Noisy Object Dynamics

We propose to model the uncertainty in the contact dynamics as additive noise that acts as a perturbation to the nominal quasi-static contact dynamics in (6.6). Hence, the noisy object dynamics are given by

$$\boldsymbol{q}_+^o = \boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u}) + \eta \boldsymbol{w}. \tag{6.7}$$

The perturbation $\boldsymbol{w}$ is assumed to be sampled from a probability distribution $p_{\boldsymbol{w}}$ such that the resulting object configuration satisfies the non-penetration constraint in (6.5c). The noise coefficient η encodes if the robot is in contact with the object and is defined

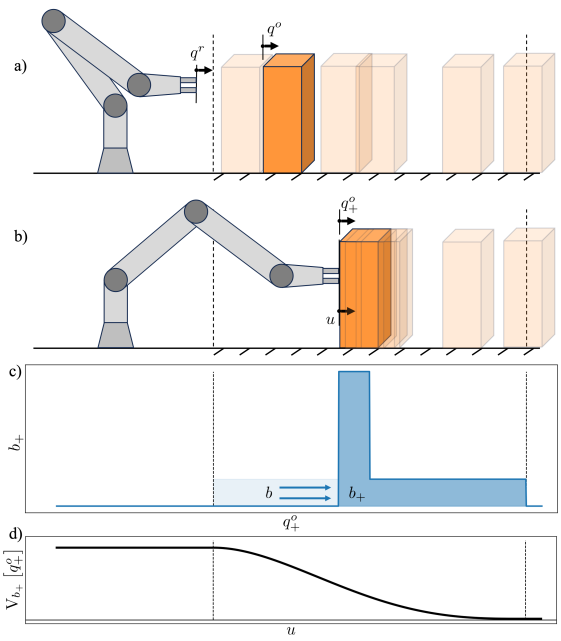


Figure 6.3: Belief dynamics through contact in a one-dimensional example. a) Illustration of object samples from the uniformly distributed initial belief (orange). The ground truth object is depicted in non-transparent orange. b) After executing a push from left to right, all samples that were on the left of the push were pushed by the robot. c) The probability mass that was on the left-hand side of the push (light blue) is now concentrated in a distribution at the contact point according to the perturbation distribution (blue). The probability mass on the right-hand side of the push does not change (blue). d) The variance of the predicted object position is therefore a function of the control action, where a robust control action may decrease the variance over time.

as

$$\eta = \begin{cases} 0 & \text{if } d(\boldsymbol{u}, \boldsymbol{q}^o) > 0 \text{ (no contact)}, \\ 1 & \text{else (contact)}. \end{cases} \tag{6.8}$$

Hence, the model only considers uncertainty in the prediction of the object configuration when the robot is manipulating the object. The presented formulation of the noisy discrete-time contact dynamics in (6.7) captures two distinct modes: one when the robot is in contact with the object ($\eta = 1$) and another when the robot is not in contact with the object ($\eta = 0$). Note that we do not model different contact modes. The contact mode $\eta = 1$ captures arbitrary numbers of contact points with arbitrary contact geometries. The object dynamics can thus be formulated with

$$\boldsymbol{q}^o_+ = \begin{cases} \boldsymbol{q}^o & \text{if } \eta = 0 \text{ (no contact)}, \\ \boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u}) + \boldsymbol{w} & \text{else (contact)}. \end{cases} \tag{6.9}$$

The *no-contact* mode entails that the object configuration does not change. This mode is not subject to uncertainty.

6.3.2 Object Belief Dynamics

In order to analyze how the belief about an object configuration and its associated uncertainty propagates through contact dynamics over time, we use the notation of probabilistic state transitions. Building upon the derived contact dynamics in (6.7), the state of the stochastic system is the object configuration and the action is the next

desired robot configuration. Accordingly, we denote the state transition probability with

$$\boldsymbol{q}_+^o \sim p(\cdot|\boldsymbol{q}^o, \boldsymbol{u}), \tag{6.10}$$

describing the probability distribution over object configurations at the next time step given the commanded robot configuration as well as the object configuration. Note that the random perturbation $\boldsymbol{w}$ that directly acts on the object dynamics in (6.7) is captured by the stochasticity of the transition probability in (6.10). We furthermore denote the probability distribution over object configurations as *belief* $b = p(\boldsymbol{q}^o)$. Given the transition probability in (6.10), a control action $\boldsymbol{u}$ and the object belief b, the resulting belief b_+ can be predicted by marginalizing over the initial object configuration:

$$b_+ = p(\boldsymbol{q}_+^o|\boldsymbol{u}) = \int_{\mathcal{Q}^o} p(\boldsymbol{q}_+^o|\boldsymbol{q}^o, \boldsymbol{u})\ b\ \mathrm{d}\boldsymbol{q}^o. \tag{6.11}$$

This equation represents the *belief dynamics*, which can be used not only for predicting the most likely object configuration but also the variance of the belief after executing action $\boldsymbol{u}$. However, solving the integral in (6.11) is typically intractable for non-linear dynamics and non-Gaussian beliefs. Yet, in the following, we start by showing an example problem for which we can obtain a closed-form solution of (6.11), followed by the introduction of a general approximation.

Example: 1D Box-Pushing. We illustrate the effects of contacts on the belief with a one-dimensional pushing example as illustrated in Figure 6.3. The hand (i.e. the robot) can be controlled directly while the box on the table (i.e. the object) can only be moved by making contact. The quasi-static contact dynamics in this example are simplified to

$$q_+^o = \begin{cases} q^o & \text{if } q^o > u \text{ (no contact)}, \\ u + w & \text{else} \qquad \text{(contact)}. \end{cases} \tag{6.12}$$

These one-dimensional piece-wise linear dynamics move the object to the right-hand side of the contact. The object does not move if no contact has been made. In this example, we consider the perturbation of the contact dynamics to be uniformly distributed, i.e. $w \sim \mathcal{U}_{[0,\alpha]}$. Sub-figures a) and b) illustrate a particular instance of the object being at q^o, which is then moved to q_+^o due to the robot reaching the commanded position u. However, in this example, the initial box position is subject to uncertainty. The initial belief b over box positions is given as a uniform distribution on an interval between box position $\underline{q}^o$ and $\bar{q}^o$, respectively. The interval is indicated by the vertical dashed lines in all sub-figures. The initial belief is

$$b = \mathcal{U}_{[\underline{q}^o, \bar{q}^o]}(q^o). \tag{6.13}$$

Given a control action $u \in [\underline{q}^o, \bar{q}^o]$ as a commanded robot position, we can now predict

the belief after contact by solving (6.11). While solving the integral in (6.11) is typically intractable, this example has a closed-form solution that is given by

$$b_+ = \frac{u - \underline{q}^o}{\bar{q}^o - \underline{q}^o} \mathcal{U}_{[u,u+\alpha]}(q_+^o) + \frac{\bar{q}^o - u}{\bar{q}^o - \underline{q}^o} \mathcal{U}_{[u,\bar{q}^o]}(q_+^o). \tag{6.14}$$

We illustrate the belief dynamics in sub-figure c) by visualizing the initial belief in light blue and the resulting belief in dark blue. It can be seen that the contact dynamics result in a concentration of probability mass around the contact point. This is due to the fact that all probability mass to the left of the contact has moved to the same position interval, i.e. the contact point subject to perturbation. In the extreme case of pushing all the way through the interval of the uniform distribution, i.e. $u = \bar{q}^o$, the object belief is equivalent to the perturbation distribution with the mean shifted to the contact point. The closed-form belief dynamics in (6.14) show that the probability distribution can be controlled by the actions a robot takes. This is underlined by sub-figure d) plotting the variance V of the object position after the push as a function of how far the robot pushed. Note that while in this specific example any control action u that makes contact with the object reduces the variance of the belief over its position, this may not be the case in general. An unfavorable control action could generally also increase the variance of the belief.

6.3.3 Variance Prediction

While the 1D example above showed a closed-form solution to the belief dynamics, this is typically intractable. Consequently, predicting the variance with

$$\mathrm{V}_+ = \mathrm{V}_{b_+}\left[\boldsymbol{q}^o\right], \tag{6.15}$$

in closed form is also intractable. A common approach is to use a Monte-Carlo approximation of the belief dynamics in (6.11) in order to compute the variance of the approximated belief update (Kappen (2015); Shirai et al. (2023)). However, this is problematic since this induces stochasticity in the prediction of the variance V_+. Evaluating whether a control action reduces or increases the variance may thus be subject to uncertainty itself, which introduces numerical issues in downstream optimization techniques. The induced noise furthermore depends on the number of samples used to approximate the belief. Thus, numerical problems and approximation errors may only be avoided by choosing a high number of samples, rendering the underlying planning technique inefficient. In the following, we derive a different way of predicting the variance of object configurations without depending on the transition probability. In other words, we eliminate the need to sample from the belief transition distribution in (6.10) in order to predict the variance given a robot action.

Instead of predicting the updated variance after taking a control action by computing

the variance of the predicted belief as in (6.15), we compute the variance of the noisy object dynamics in (6.7), i.e.

$$\begin{aligned} \mathrm{V}_+ &= \mathrm{V}_{b,p_w}[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u}) + \eta\boldsymbol{w}] \\ &= \mathrm{V}_b[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})] + \mathrm{V}_{b,p_w}[\eta\boldsymbol{w}] + \\ &2\mathrm{E}_{b,p_w}\left[(\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u}) - \mathrm{E}_b[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})])^\top(\eta\boldsymbol{w} - \mathrm{E}_{b,p_w}[\eta\boldsymbol{w}])\right]. \end{aligned} \tag{6.16}$$

At this point, we introduce the assumption that the expectation of the perturbation $\boldsymbol{w}$ is zero, i.e. $\mathrm{E}_{p_w}[\boldsymbol{w}] = \mathbf{0}$. As a result, perturbations are modeled as zero-mean noise in the tangential directions with respect to the contact point and zero noise in the normal direction of the contact. We furthermore note that the contact indicator η and the perturbation are independent, such that the expectation of the product of the contact indicator and the perturbation is zero, i.e. $\mathrm{E}_{b,p_w}[\eta\boldsymbol{w}] = \mathrm{E}_b[\eta]\mathrm{E}_{p_w}[\boldsymbol{w}] = \mathbf{0}$. Consequently, the third term in (6.16) simplifies to

$$\begin{aligned} &\mathrm{E}_{b,p_w}\left[(\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u}) - \mathrm{E}_b[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})])^\top(\eta\boldsymbol{w} - \mathrm{E}_{b,p_w}[\eta\boldsymbol{w}])\right] = \\ &\mathrm{E}_{b,p_w}\left[\eta(\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u}) - \mathrm{E}_b[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})])^\top\boldsymbol{w}\right] = \\ &\mathrm{E}_b\left[\eta(\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u}) - \mathrm{E}_b[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})])\right]^\top \mathrm{E}_{p_w}\left[\boldsymbol{w}\right] = \mathbf{0}. \end{aligned} \tag{6.17}$$

As a result, the predicted variance in (6.16) is equivalent to the sum of the nominally predicted variance and the variance of the induced noise. The predicted variance is thus

$$\mathrm{V}_+ = \mathrm{V}_b\left[\boldsymbol{f}(\boldsymbol{q}^o,\boldsymbol{u})\right] + \mathrm{V}_{b,p_w}\left[\eta\boldsymbol{w}\right]. \tag{6.18}$$

The left-hand term is computed through the nominal object dynamics $\boldsymbol{f}$, thus not including perturbations. The right-hand term is the variance contribution from the noise acting on the object when making contact. With the expectation of p_w being zero, i.e. $\mathrm{E}_{b,p_w}[\eta\boldsymbol{w}] = \mathbf{0}$, the variance of the noise is given with

$$\begin{aligned} &\mathrm{V}_{b,p_w}\left[\eta\boldsymbol{w}\right] = \mathrm{E}_{b,p_w}\left[(\eta\boldsymbol{w})^\top(\eta\boldsymbol{w})\right] = \\ &\mathrm{E}_{b,p_w}\left[\eta^2\boldsymbol{w}^\top\boldsymbol{w}\right] = \mathrm{E}_b\left[\eta^2\right]\mathrm{E}_{p_w}\left[\boldsymbol{w}^\top\boldsymbol{w}\right]. \end{aligned} \tag{6.19}$$

With $\eta = \eta^2 \in \{0,1\}$, the expectation of the squared contact indicator is equal to the expected contact indicator, i.e. $\mathrm{E}_b\left[\eta^2\right] = \mathrm{E}_b\left[\eta\right]$. The expected value of the contact indicator is the probability of the robot making contact with the object given a belief over object positions and a control action. Furthermore, note that due to the zero-mean property of our noise distribution, the expectation of the squared perturbation is equivalent to the variance of the perturbation, i.e. $\mathrm{E}_{p_w}\left[\boldsymbol{w}^\top\boldsymbol{w}\right] = \mathrm{V}_{\boldsymbol{w}}$. Inserting these equalities in (6.19), the variance of the applied perturbation can be expressed with

$$\mathrm{V}_{b,p_w}\left[\eta\boldsymbol{w}\right] = \mathrm{E}_b\left[\eta\right]\mathrm{V}_{\boldsymbol{w}}. \tag{6.20}$$

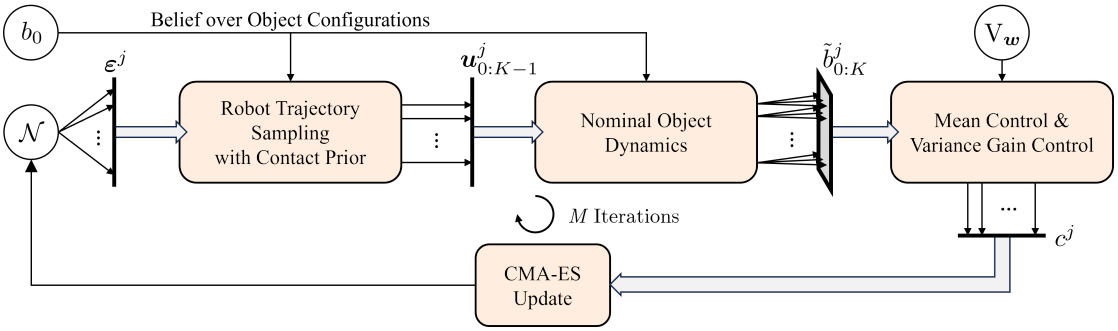


Figure 6.4: Block diagram depicting one iteration of BS-VP-STO. The algorithm starts with sampling a population of latent candidate trajectory variables ε. These are then decoded into robot trajectories $\boldsymbol{q}^r_{0:K}$ using a contact prior. For each candidate trajectory, the object belief is rolled-out using the nominal object dynamics. The variance gain together with the mean control cost is then used to compute the fitness of each candidate trajectory. Finally, the distribution of candidate trajectory variables weighted by fitness is used to update the Gaussian approximation of the probability distribution using CMA-ES. After M iterations, the algorithm returns the best-performing candidate trajectory as a solution.

As a result of this section, we predict the variance of the object configuration with

$$\mathrm{V}_+ = \mathrm{V}_b\left[\boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u})\right] + \mathrm{E}_b\left[\eta\right]\mathrm{V}_{\boldsymbol{w}}. \tag{6.21}$$

The result in (6.21) is central to our contribution, as this allows us to predict the variance of the object configuration, i.e. the uncertainty, based on the variance of the perturbation $\mathrm{V}_{\boldsymbol{w}}$ that is assumed to be a constant value. Especially when those perturbations are constrained to the tangent space of the contact, sampling consistent perturbations involves expensive computations and makes the prediction of the variance stochastic.

Monte-Carlo Approximation of the Nominal Dynamics

We approximate the variance contribution resulting from the nominal contact dynamics using a non-parametric representation of the belief, i.e. particles. We denote the approximated belief as a set of N_p particles $\{{}^i\boldsymbol{q}^o, {}^i\alpha\}_{i=1}^{N_p}$, where each particle consists of a state sample ${}^i\boldsymbol{q}^o$ and a corresponding weight ${}^i\alpha$. The weight of a particle, $0 \leq {}^i\alpha \leq 1$, represents an approximate belief in the corresponding state sample ${}^i\boldsymbol{q}^o$. A prediction step consists of propagating each particle through the nominal contact dynamics, i.e.

$${}^i\boldsymbol{q}^o_+ = \boldsymbol{f}({}^i\boldsymbol{q}^o, \boldsymbol{u}),\ \forall i \in \{1, 2, .., N_p\}. \tag{6.22}$$

Since we do not take any measurements during planning, we assume that the particle weights are constant and equally distributed, such that ${}^i\alpha = {}^1/_{N_p},\ \forall i$. Based on the par-

ticle representation of the belief, we estimate the variance of the object configuration with

$$\hat{\mathrm{V}}_b[\boldsymbol{q}^o] = \frac{1}{N_p} \sum_{i=1}^{N_p} \left({}^i\boldsymbol{q}^o - \boldsymbol{\mu}^o\right)^\top \left({}^i\boldsymbol{q}^o - \boldsymbol{\mu}^o\right). \tag{6.23}$$

The empirical mean object configuration is computed with $\boldsymbol{\mu}^o = {}^1/_{N_p} \sum_{i=1}^{N_p} {}^i\boldsymbol{q}^o$. The probability of making contact $\mathrm{E}_b[\eta]$ is approximated as

$$\hat{\mathrm{E}}_b[\eta] = \frac{1}{N_p} \sum_{i=1}^{N_p} {}^i\eta, \tag{6.24}$$

where ${}^i\eta$ indicates if the i-th particle has an object configuration that is in contact with the robot. As a result, we approximate the overall variance prediction with

$$\hat{\mathrm{V}}_+ = \hat{\mathrm{V}}_b[\boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u})] + \hat{\mathrm{E}}_b[\eta]\, \mathrm{V}_{\boldsymbol{w}}. \tag{6.25}$$

When comparing the predicted variance $\hat{\mathrm{V}}_+$ against the variance of the current time step $\hat{\mathrm{V}}_b[\boldsymbol{q}^o]$, the same set of particles is used to compute the empirical variance as in (6.23). Thus, evaluating the approximated variance dynamics does not involve sampling and is thus deterministic. We exploit this approximation in Sec. 6.4 when optimizing robot trajectories based on the predicted variance.

6.4 Stochastic Optimization for Robust Manipulation

Given our objective to push an object into a desired goal configuration subject to stochastic contact dynamics (cf. (6.2)), this section presents a framework that optimizes for robust robot trajectories directly in the belief space over possible object configurations. This framework extends our previous work *VP-STO* (Jankowski et al. (2023)) to belief-space via-point-based stochastic trajectory optimization (*BS-VP-STO*). Our framework exploits the variance prediction developed in Sec. 6.3 for synthesizing robust manipulation behavior. Due to the quasi-static pushing model in (6.5a) - (6.5c), a robot trajectory is equivalent to a control trajectory, i.e. $\boldsymbol{q}^r_{1:K} = \boldsymbol{u}_{0:K-1}$. Thus, BS-VP-STO is a shooting method aiming at minimizing an objective that depends solely on the object configuration as in (6.1). Due to the non-smooth nature of the contact dynamics and the resulting non-smooth cost function with respect to the optimization variable, we approach the optimization problem with a gradient-free, i.e. zero-order, evolutionary optimization technique.

Figure 6.4 illustrates the optimization loop that is based on zero-order optimization of the variable $\boldsymbol{\varepsilon}$, which uniquely encodes a robot trajectory. At the beginning of the m-th iteration, we sample N_{cand} candidate trajectories from a latent Gaussian distribution

that represents the current solution to the optimization problem:

$$\boldsymbol{\varepsilon}^j \sim \mathcal{N}(\bar{\boldsymbol{\varepsilon}}_m, \boldsymbol{\Sigma}_m),\ \forall j \in \{1, 2, \ldots, N_{\text{cand}}\}. \tag{6.26}$$

The latent variable $\boldsymbol{\varepsilon}_j$ translates to a robot trajectory through an affine mapping g, i.e. $\boldsymbol{u}^j_{0:K-1} = g(\boldsymbol{\varepsilon}_j)$. This affine mapping imposes a contact prior on the sampling of robot trajectories, which is presented in Sec. 6.4.2. Given a candidate robot trajectory $\boldsymbol{u}^j_{0:K-1}$ and an initial belief over object positions $b_0 = p(\boldsymbol{q}^o)$, we compute the nominal belief dynamics. This results in a nominal belief trajectory $\tilde{b}^j_{0:K}$ for each candidate $\boldsymbol{\varepsilon}_j$. Based on the nominal belief trajectory we compute the step-wise predicted variance as developed in Sec. 6.3. The predicted variance is subsequently used in a cost and a constraint to the optimization problem, which is further outlined in Sec. 6.4.1. Together with a cost for controlling the mean of the object configuration, the total cost of each candidate trajectory is used to update the parameters of the latent Gaussian distribution, i.e. $\bar{\boldsymbol{\varepsilon}}_{m+1}, \boldsymbol{\Sigma}_{m+1}$ based on Covariance Matrix Adaptation (CMA-ES) (Hansen (2016)). After M iterations, we return the best-performing sample that returned the lowest cost.

6.4.1 Variance Gain Control

In Sec. 6.3 we derive an approximation of the one-step prediction of the variance of the object configuration. However, this does not enable the prediction of the variance after multiple time steps, which is due to the fact that the prediction of the variance $\hat{\mathrm{V}}_{k+1}$ in (6.25) requires the belief of the previous time step b_k to be known. Therefore, instead of directly controlling the variance at the end of the trajectory $\hat{\mathrm{V}}_K$, we propose to control the predicted variance at each time step. Given a robot trajectory $\boldsymbol{u}^j_{0:K-1}$, we thus compute the nominal belief over object configurations at each time step, i.e. $\tilde{b}_k$, via the nominal object dynamics in (6.6). Given the particle set that approximates the initial belief $b_0 = p(\boldsymbol{q}^o)$ as an input to BS-VP-STO, the nominal belief rollout is computed by applying the nominal forward dynamics to all particles:

$${}^i\boldsymbol{q}^o_{k+1} = \boldsymbol{f}\left({}^i\boldsymbol{q}^o_k, \boldsymbol{u}_k\right),\ \forall i \in \{1, 2, .., N_p\}. \tag{6.27}$$

The one-step prediction of the variance at each time step can then be computed as

$$\begin{aligned} \hat{\mathrm{V}}_{k+1} &= \hat{\mathrm{V}}_{\tilde{b}_k}\left[\boldsymbol{f}(\boldsymbol{q}^o, \boldsymbol{u}_k)\right] + \hat{\mathrm{E}}_{\tilde{b}_k}\left[\eta\right]\mathrm{V}_{\boldsymbol{w}} \\ &= \hat{\mathrm{V}}_{\tilde{b}_{k+1}}\left[\boldsymbol{q}^o\right] + \hat{\mathrm{E}}_{\tilde{b}_k}\left[\eta\right]\mathrm{V}_{\boldsymbol{w}}. \end{aligned} \tag{6.28}$$

In order to quantify the change of uncertainty due to a given control action $\boldsymbol{u}_k$, we are interested in the amount of variance that is gained over one time step. Note that the variance of a continuous random variable is closely related to its differential entropy. While the variance as defined in (6.3) is equivalent to the trace of the covariance matrix, i.e. the sum over all eigenvalues, the upper bound of the differential entropy is monotonic in the determinant of the covariance matrix, i.e. the product of all

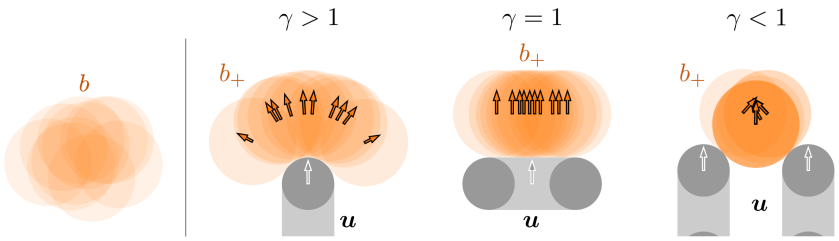


Figure 6.5: Belief dynamics through contact in a two-dimensional example. The three sub-figures on the right illustrate the predicted belief b_+ via samples in orange. All three cases started from the same initial belief b that is depicted in the left-most sub-figure. The visualization of the prediction on the left shows an increase in the variance of the object position as a consequence of pushing with a single contact point ($\gamma > 1$). The second prediction shows a constant variance as a consequence of pushing with a flat contact surface ($\gamma = 1$). The right-most sub-figure shows a decrease in the variance of object position as a consequence of pushing with two contact points ($\gamma < 1$).

eigenvalues (Cover and Thomas (2005)). As a result, the variance of a continuous random variable yields an upper bound for its entropy. Therefore, we introduce a new metric γ, which we call *variance gain*, measuring the relative change of the variance after applying an action $\boldsymbol{u}_k$, i.e.

$$\gamma_k = \frac{\hat{\mathrm{V}}^+_{k+1}}{\hat{\mathrm{V}}_{\tilde{b}_k}[\boldsymbol{q}^o] + \mathrm{V}_{\boldsymbol{w}}} = \frac{\hat{\mathrm{V}}_{\tilde{b}_{k+1}}[\boldsymbol{q}^o] + \hat{\mathrm{E}}_{\tilde{b}_k}[\eta]\,\mathrm{V}_{\boldsymbol{w}}}{\hat{\mathrm{V}}_{\tilde{b}_k}[\boldsymbol{q}^o] + \mathrm{V}_{\boldsymbol{w}}}. \tag{6.29}$$

The variance gain γ is the ratio of the output variance, i.e. the predicted variance $\hat{\mathrm{V}}_{k+1}$, to the input variance, i.e. $\hat{\mathrm{V}}_k + \mathrm{V}_{\boldsymbol{w}}$. Figure 6.5 illustrates three different robot actions resulting in different variance gains. It shows that contact geometry plays a crucial role when planning to make contact between the robot and an object. The variance gain γ reflects that the contact geometry affects the robustness of a contact, e.g. when pushing. The left sub-figure shows that using one finger to push a circular object with an uncertain location results in an increase in variance ($\gamma > 1$). In contrast, the right sub-figure shows that using two fingers with one finger pushing the object towards the other finger results in a decrease in variance ($\gamma < 1$). Using a flat contact surface to push an object with an uncertain location keeps the variance constant as shown in the middle sub-figure ($\gamma = 1$). Note that the variance gain is lower or equal to one for robot actions that have zero probability of making contact with the object. In this case, the belief does not change, i.e. $\hat{\mathrm{V}}_{\tilde{b}_{k+1}}[\boldsymbol{q}^o] = \hat{\mathrm{V}}_{\tilde{b}_k}[\boldsymbol{q}^o]$, since no perturbation is injected into the belief, i.e. $\hat{\mathrm{E}}_{\tilde{b}_k}[\eta]\,\mathrm{V}_{\boldsymbol{w}} = 0$. Thus, a no-contact action results in

$$\gamma_{\mathrm{no-contact}} = \frac{\hat{\mathrm{V}}_{\tilde{b}_k}[\boldsymbol{q}^o]}{\hat{\mathrm{V}}_{\tilde{b}_k}[\boldsymbol{q}^o] + \mathrm{V}_{\boldsymbol{w}}} \leq 1. \tag{6.30}$$

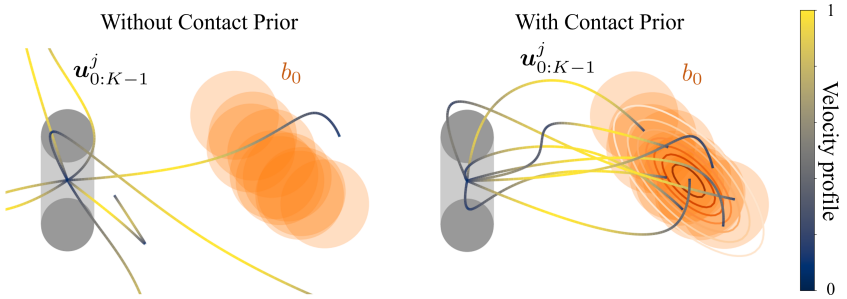


Figure 6.6: Comparison of uninformed and informed sampling of robot trajectories. The left sub-figure shows trajectory samples drawn from a probability distribution computed without a contact prior ($\boldsymbol{Q_q} = \mathbf{0}$). The right sub-figure shows trajectory samples drawn from a probability distribution computed with a contact prior ($\boldsymbol{Q_q} > \mathbf{0}$). The contact prior guides the sampling of robot trajectories towards regions where the robot is likely to make contact with the object given the object belief.

We propose to enforce robustness in the optimization problem by constraining the solution to variance gains smaller or equal to one at all steps, i.e.

$$\gamma_k \leq 1 \quad \forall\, k. \tag{6.31}$$

Due to the zero-order technique that we use for optimizing the robot trajectory, we deploy the robustness constraint as a barrier cost, i.e. a discontinuous cost that is zero if the constraint is satisfied and returns a high value if the constraint is violated. We can further reduce the predicted variance by adding a cost term that is active if the constraint is already satisfied. This is encapsulated in the following cost-term

$$c_{\text{robust}} = \lambda_c \prod_{k=0}^{K-1} e^{-\frac{1-\gamma_k}{K-1}}, \tag{6.32}$$

with a discontinuous cost weight

$$\lambda_c = \begin{cases} 1 & \text{if } \max_k \gamma_k \leq 1, \\ 10^3 & \text{else.} \end{cases} \tag{6.33}$$

The total cost of a candidate robot trajectory is computed as the sum of the variance gain control cost in (6.32) and a task-specific cost c_{task} that computes the deviation of the mean of the object configuration to the desired goal configuration.

6.4.2 Trajectory Sampling with a Contact Prior

The efficiency of sampling-based optimization algorithms depends on the quality of the generated samples. In this section, we present how the object belief can be used to inform the sampling of robot trajectories for manipulation tasks. In general, it is

desirable to sample trajectories with a likelihood that is proportional to the negative cost that can be expected from executing the trajectory. In the following, we denote a robot trajectory with $\boldsymbol{q}^r_{0:K} = [\boldsymbol{q}^r_0, \boldsymbol{q}^r_1, \dots, \boldsymbol{q}^r_K]$, where K corresponds to the number of discretized steps captured by the trajectory. Suppose that the cost is given as a function of the robot trajectory, i.e. $c = f_c(\boldsymbol{q}^r_{0:K})$, then we would like to sample robot trajectories from a corresponding probability distribution with

$$p(\boldsymbol{q}^r_{0:K}) \propto \exp\left(-f_c(\boldsymbol{q}^r_{0:K})\right). \tag{6.34}$$

While we do not have access to such a generative probability distribution, we approximate it with a prior that improves the sample efficiency of the optimization algorithm compared to sampling initial guesses from an uninformed probability distribution. The idea is to sample robot trajectories in regions where the robot is likely to make contact with the object given its belief. This is motivated by the observation that making contact is a necessary precondition for manipulating an object.

Figure 6.6 illustrates the impact of the contact prior on trajectory samples. Without the contact prior, a large portion of the candidate trajectories explores regions in which the robot does not move into the object belief and thus does not contribute to the optimization process.

Via-point-based Trajectory Parameterization

In order to efficiently synthesize robot trajectories in a low-dimensional space, we adopt the via-point-based trajectory representation as in Jankowski et al. (2023). The robot configuration is given with

$$\boldsymbol{q}^r(t) = \boldsymbol{\Phi}_{\text{via}}(t)\boldsymbol{\theta} + \boldsymbol{\phi}_0(t, \boldsymbol{q}^r_0, \dot{\boldsymbol{q}}^r_0), \tag{6.35}$$

where the robot trajectory is parameterized with

$$\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{q}^1_{\text{via}} \\ \vdots \\ \boldsymbol{q}^N_{\text{via}} \end{pmatrix} \in \mathbb{R}^{N \cdot n^r_{\text{dof}}}. \tag{6.36}$$

The trajectory parameter $\boldsymbol{\theta}$ contains N via-points the trajectory passes through. The basis functions $\boldsymbol{\Phi}_{\text{via}}(t)$ enforce that the trajectory passes exactly through the via configurations while smoothly interpolating with minimal acceleration. The basis functions furthermore enforce that the velocity at the end of the trajectory is zero. Note that the last n^r_{dof} elements of $\boldsymbol{\theta}$ are the final robot configuration at the end of the trajectory, i.e. $\boldsymbol{q}^r(T) = \boldsymbol{q}^r_K = \boldsymbol{q}^N_{\text{via}}$. The basis offset $\boldsymbol{\phi}_0(t, \boldsymbol{q}^r_0, \dot{\boldsymbol{q}}^r_0)$ incorporates the initial robot position $\boldsymbol{q}^r_0$ and velocity with $\dot{\boldsymbol{q}}^r_0$. T denotes the duration of the trajectory. We use the time scaling algorithm in Jankowski et al. (2023) for computing the duration of a trajectory

based on a given parameter $\boldsymbol{\theta}$ such that user-defined velocity and acceleration limits are enforced. For implementation details on how to compute the basis functions and offsets, please refer to Jankowski et al. (2022).

In the following, we are interested in computing a Gaussian distribution of the via-points $\boldsymbol{\theta}$ to efficiently sample from. Due to the affine mapping from via-points to robot trajectories in (6.35), this corresponds to sampling from a Gaussian distribution of continuous robot trajectories.

Gaussian Contact Prior

The contact prior is a probability distribution that guides the sampling of robot trajectories towards regions where the robot is likely to make contact with the object. To allow for variations in how to approach the contact with the object, we only consider the final robot configuration of the trajectory to be subject to the contact prior. This is incorporated by exploiting the parameterization of the robot trajectory in (6.35), where the final robot configuration is explicitly given by the last n^r_{dof} elements of $\boldsymbol{\theta}$.

Suppose that a conditional Gaussian distribution

$$p_{\text{c}}(\boldsymbol{q}^r|\boldsymbol{q}^o) = \mathcal{N}(\boldsymbol{f}_{\text{c}}(\boldsymbol{q}^o), \boldsymbol{\Sigma}^{r|o}) \tag{6.37}$$

approximates the probability density of a robot configuration making contact with the object. Furthermore, suppose that the object configuration is Gaussian distributed as well with $\boldsymbol{q}^o \sim \mathcal{N}(\boldsymbol{\mu}^o, \boldsymbol{\Sigma}^o)$. Practically, we find the Gaussian distribution of object configurations by approximating the initial belief b_0 with a Gaussian distribution for computing the contact prior. This lets us compute a probability distribution over robot configurations indicating how likely it is to establish contact between the robot and the object:

$$p_{\text{c}}(\boldsymbol{q}^r) = \mathcal{N}(\boldsymbol{f}_{\text{c}}(\boldsymbol{\mu}^o), \boldsymbol{\Sigma}^{r|o} + \boldsymbol{A}\boldsymbol{\Sigma}^o\boldsymbol{A}^\top), \tag{6.38}$$

with $\boldsymbol{A} = {}^{\partial \boldsymbol{f}_{\text{c}}}/_{\partial \boldsymbol{q}^o}|_{\boldsymbol{\mu}^o}$. The corresponding prior on the trajectory parameter $\boldsymbol{\theta}$ is then given by

$$p_{\text{c}}\left(\boldsymbol{\theta} = \begin{pmatrix} \boldsymbol{q}^1_{\text{via}} \\ \vdots \\ \boldsymbol{q}^N_{\text{via}} \end{pmatrix}\right) = \mathcal{N}\left(\begin{pmatrix} \boldsymbol{0} \\ \vdots \\ \bar{\boldsymbol{q}}_{\text{c}} \end{pmatrix}, \begin{pmatrix} \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{Q}_q \end{pmatrix}^{-1}\right), \tag{6.39}$$

where $\boldsymbol{Q}_q = \left(\boldsymbol{\Sigma}^{r|o} + \boldsymbol{A}\boldsymbol{\Sigma}^o\boldsymbol{A}^\top\right)^{-1}$ describes the precision matrix of the contact prior with respect to the mean contact configuration $\bar{\boldsymbol{q}}_{\text{c}} = \boldsymbol{f}_{\text{c}}(\boldsymbol{\mu}^o)$. We denote the contact prior with

$$p_{\text{c}}(\boldsymbol{\theta}) = \mathcal{N}(\bar{\boldsymbol{\theta}}_{\text{c}}, \boldsymbol{Q}_{\boldsymbol{\theta}}^{-1}). \tag{6.40}$$

Note that the resulting covariance matrix $\boldsymbol{Q}_{\boldsymbol{\theta}}^{-1}$ is degenerated due to zero-precision

values for the via-points except for $\boldsymbol{q}_{\text{via}}^N$. To resolve the degeneration, we regularize the covariance matrix by combining the contact prior with a smoothness prior as described in the following.

Gaussian Contact Prior in Joint Space

Optimizing in the joint space of an articulated robot such as robot arms may be beneficial when kinematic and dynamic limitations are to be considered during planning. Sampling in joint space requires representing the contact prior in joint space as well. Suppose that the robot's end-effector, that is supposed to manipulate the object, has a configuration given by $\boldsymbol{x}^r$ which is computed from the robot's joint positions $\boldsymbol{q}^r$ via forward kinematics $\boldsymbol{x}^r = \boldsymbol{f}_{\text{fk}}(\boldsymbol{q}^r)$. We may adopt the contact prior in (6.38) to formulate a prior distribution over configurations of the end-effector, i.e.

$$p_{\text{c}}(\boldsymbol{x}^r) = \mathcal{N}(\boldsymbol{f}_{\text{c}}(\boldsymbol{\mu}^o), \boldsymbol{\Sigma}^{r|o} + \boldsymbol{A}\boldsymbol{\Sigma}^o\boldsymbol{A}^\top). \tag{6.41}$$

For computing a corresponding Gaussian distribution in joint space, the forward kinematics are linearized around a mean joint position $\bar{\boldsymbol{q}}_{\text{c}}^r$ with

$$\boldsymbol{x}^r \approx \boldsymbol{f}_{\text{fk}}(\bar{\boldsymbol{q}}_{\text{c}}^r) + \boldsymbol{J}(\bar{\boldsymbol{q}}_{\text{c}}^r)\,(\boldsymbol{q}^r - \bar{\boldsymbol{q}}_{\text{c}}^r)\,, \tag{6.42}$$

where $\boldsymbol{J}(\boldsymbol{q}) = {}^{\partial \boldsymbol{x}^r}/_{\partial \boldsymbol{q}^r}|_{\bar{\boldsymbol{q}}_{\text{c}}^r}$ denotes the Jacobian with respect to the end-effector configuration. The mean joint position $\bar{\boldsymbol{q}}_{\text{c}}^r$ can be computed via inverse kinematics with respect to the mean end-effector configuration $\boldsymbol{f}_{\text{c}}(\boldsymbol{\mu}^o)$. Consequently, the Gaussian contact prior for the end-effector can be locally transformed into the joint space, resulting in a Gaussian distribution $p_{\text{c}}(\boldsymbol{q}^r) = \mathcal{N}(\bar{\boldsymbol{q}}_{\text{c}}^r, \boldsymbol{Q}_{\boldsymbol{q}}^{-1})$. The joint space contact precision matrix is computed with

$$\boldsymbol{Q}_{\boldsymbol{q}} = \boldsymbol{J}(\bar{\boldsymbol{q}}_{\text{c}}^r)^\top \left(\boldsymbol{\Sigma}^{r|o} + \boldsymbol{A}\boldsymbol{\Sigma}^o\boldsymbol{A}^\top\right)^{-1} \boldsymbol{J}(\bar{\boldsymbol{q}}_{\text{c}}^r). \tag{6.43}$$

Gaussian Smoothness Prior

The smoothness prior, introduced in our previous work (Jankowski et al. (2023)), incorporates temporal correlations between via-points by computing a Gaussian distribution that expresses a high likelihood for low-acceleration profiles. A typical objective in trajectory optimization is to minimize the integral over squared accelerations of the candidate trajectory, i.e.

$$J_{\text{s}} = \frac{1}{2}\int_0^T \ddot{\boldsymbol{q}}^{r\top}(t)\boldsymbol{R}_{\boldsymbol{q}}\ddot{\boldsymbol{q}}^r(t)dt. \tag{6.44}$$

The positive definite matrix $\boldsymbol{R}_{\boldsymbol{q}}$ encodes the desired smoothing for the individual degrees of freedom. Using the parameterization in (6.35), this objective can be expressed using the via-point parameter $\boldsymbol{\theta}$ and the initial conditions for the trajectory

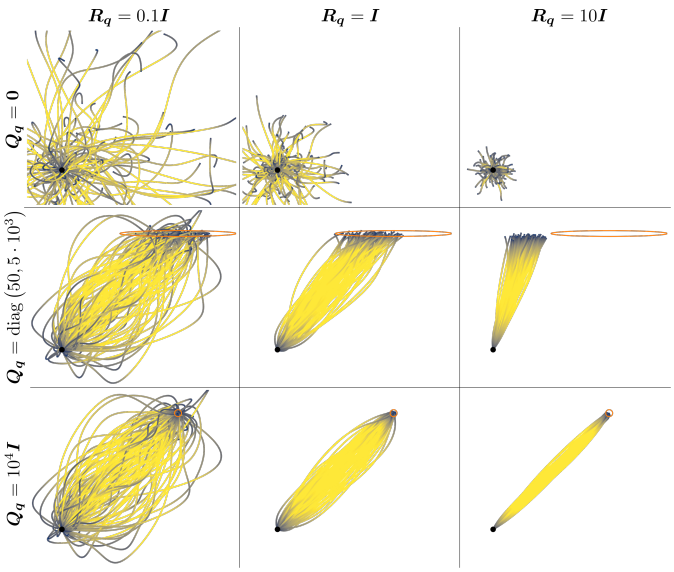


Figure 6.7: Smooth trajectories sampled from the product of the smoothness prior and the contact prior. The contact prior is indicated by the orange ellipses and circles with the mean of the contact prior in the center. The velocity profile of the trajectories is encoded through color with low velocities in blue and high velocities in yellow. Note that all trajectories start and end with exactly zero velocity.

$\boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r$, i.e. $J_\mathrm{s}(\boldsymbol{\theta}, \boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r)$. As a next step, we express the smoothness prior as a probability distribution parameterized with the negative objective in (6.44) with

$$p_\mathrm{s}(\boldsymbol{\theta}, \boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r) \propto e^{-J_\mathrm{s}(\boldsymbol{\theta}, \boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r)}. \tag{6.45}$$

Interestingly, this results in a joint Gaussian distribution over the trajectory parameter and the initial conditions. We then compute a Gaussian smoothness prior on the trajectory parameter by conditioning on the initial conditions, i.e.

$$p_\mathrm{s}(\boldsymbol{\theta} | \boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r) = \mathcal{N}(\bar{\boldsymbol{\theta}}_\mathrm{s}, \boldsymbol{R}_{\boldsymbol{\theta}}^{-1}), \tag{6.46}$$

with the precision matrix being computed with

$$\boldsymbol{R}_{\boldsymbol{\theta}} = \int_0^T \ddot{\boldsymbol{\Phi}}_\mathrm{via}^\top(t) \boldsymbol{R}_{\boldsymbol{q}} \ddot{\boldsymbol{\Phi}}_\mathrm{via}(t) dt. \tag{6.47}$$

Please refer to Section 4.3.1 for the derivation of (6.47) and for details on how to compute the smoothness prior mean $\bar{\boldsymbol{\theta}}_\mathrm{s}$. Note that the smoothness precision matrix $\boldsymbol{R}_{\boldsymbol{\theta}}$ can be computed offline as it does not depend on the trajectory parameter $\boldsymbol{\theta}$.

Product of Gaussian Priors

Given the two priors for making contact and smooth trajectories respectively, we find the informed via-point distribution by fusing the two probabilistic priors via computing the normalized product of the two priors, such that

$$p(\boldsymbol{\theta}) = \mathcal{N}(\bar{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}) \propto p_\mathrm{c}(\boldsymbol{\theta}) p_\mathrm{s}(\boldsymbol{\theta} | \boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r). \tag{6.48}$$

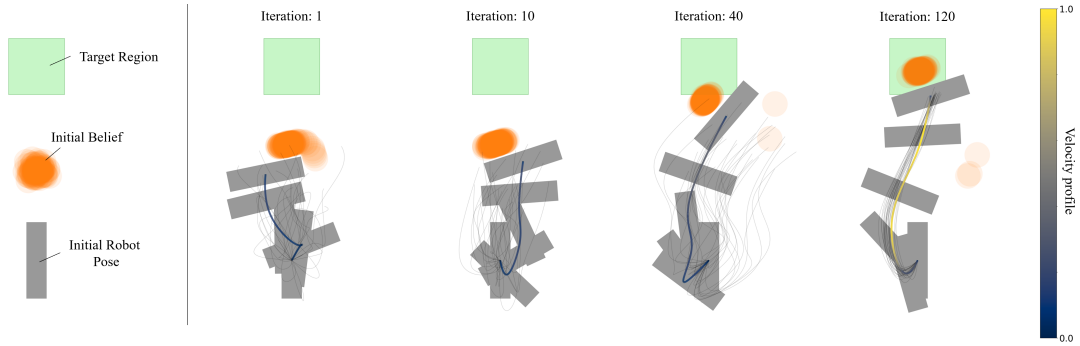


Figure 6.8: BS-VP-STO optimizing the trajectory of a rectangular robot to push a circular object into a goal region subject to uncertain initial object location (represented by a particle-based belief representation) and uncertain contact dynamics. The sub-figures show the best-performing candidate solution with the corresponding velocity profile, as well as the other trajectory samples in light gray, after 1, 10, 40, and 120 iterations from left to right.

The product of two multivariate Gaussians is again a multivariate Gaussian, with the resulting parameters given by

$$\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = (\boldsymbol{Q}_{\boldsymbol{\theta}} + \boldsymbol{R}_{\boldsymbol{\theta}})^{-1}, \tag{6.49a}$$

$$\bar{\boldsymbol{\theta}} = \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{Q}_{\boldsymbol{\theta}}\bar{\boldsymbol{\theta}}_{\mathrm{c}} + \boldsymbol{R}_{\boldsymbol{\theta}}\bar{\boldsymbol{\theta}}_{\mathrm{s}}). \tag{6.49b}$$

Figure 6.7 illustrates trajectories sampled from the product of the contact prior, parameterized by $\boldsymbol{Q}_{\boldsymbol{q}}$, and the smoothness prior, parameterized by $\boldsymbol{R}_{\boldsymbol{q}}$. Within each sub-figure, all trajectories are drawn from a single Gaussian distribution. Note that all trajectories start and end with zero velocity.

Optimizing and Sampling in Latent Space

Given the product of priors, we use the informed generative via-point distribution $p(\boldsymbol{\theta})$ as a probabilistic initial guess for optimizing robot trajectories with CMA-ES. Instead of directly sampling trajectory candidates $\boldsymbol{\theta}$ from an uninformed distribution, e.g. white noise, we sample and optimize for $\boldsymbol{\varepsilon} \in \mathbb{R}^{N n^{r}_{\mathrm{dof}}}$. For a given $\boldsymbol{\varepsilon}$, we compute $\boldsymbol{\theta}$ through an affine transformation as follows:

$$\boldsymbol{\theta} = \bar{\boldsymbol{\theta}} + \boldsymbol{L}_{\boldsymbol{\theta}}\boldsymbol{\varepsilon}. \tag{6.50}$$

Here, $\boldsymbol{L}_{\boldsymbol{\theta}}$ is the Cholesky decomposition of the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. The parameters $\bar{\boldsymbol{\theta}}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$ incorporate the prior as defined in (6.49). The idea of this additional transformation is to decouple the optimization variable $\boldsymbol{\varepsilon}$ from the particular prior. In each iteration m of BS-VP-STO, we obtain the new population of candidate solutions by

sampling N_{cand} robot trajectories via

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\bar{\boldsymbol{\theta}} + \boldsymbol{L}_{\boldsymbol{\theta}}\bar{\boldsymbol{\varepsilon}}_m, \boldsymbol{L}_{\boldsymbol{\theta}}\boldsymbol{\Sigma}_m\boldsymbol{L}_{\boldsymbol{\theta}}^{\top}\right). \tag{6.51}$$

When initializing the CMA-ES distribution as white noise, i.e. $\bar{\boldsymbol{\varepsilon}}_0 = \mathbf{0}$ and $\boldsymbol{\Sigma}_0 = \boldsymbol{I}$, we effectively sample the first population from the informed distribution in (6.49), as inserting the initial parameters into (6.51) yields

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\bar{\boldsymbol{\theta}}, \boldsymbol{L}_{\boldsymbol{\theta}}\boldsymbol{L}_{\boldsymbol{\theta}}^{\top}\right) = \mathcal{N}\left(\bar{\boldsymbol{\theta}}, \boldsymbol{\Sigma}_{\boldsymbol{\theta}}\right). \tag{6.52}$$

Eventually, given a sampled trajectory parameter $\boldsymbol{\theta}$, we find the control trajectory with respect to the system in (6.6) by discretizing the robot trajectory in (6.35), i.e.

$$\boldsymbol{u}_k = \boldsymbol{q}^r\left(t = T\frac{k+1}{K}\right). \tag{6.53}$$

Note that due to the quasi-static model in (6.6), the dynamics can be rolled out with an arbitrary temporal resolution.

Example: Single-horizon Robust 2D Object-Pushing. We showcase the BS-VP-STO pipeline over multiple iterations for a 2D object pushing example, illustrated in Figure 6.8. The task for the robot, a rectangular geometry, is to push the object, a circular geometry, into a target region. We consider two sources of uncertainty in this example: a) The initial position of the object is uncertain, which is reflected by an initial belief; and b) the contact dynamics are uncertain. This task requires exploring contact modes that are robust to these uncertainties, as implicitly done by the presented planning algorithm.

After the first iteration, the best solution corresponds to the robot making contact with its long side. Note that this solution corresponds to the best candidate of the initial population sampled from the product of Gaussian priors, without any CMA-ES updates. Due to the probabilistic contact prior, almost all of the 30 initial candidates bring the robot into contact with the object, enabling an informative sampling of the cost landscape. After 10 iterations of BS-VP-STO, the algorithm found a solution making robust contact with the object while moving it slightly toward the target area. The solution after 40 iterations enables the robot to almost push the object into the target area. After 120 iterations, the algorithm found a solution for pushing the object robustly into the target area, while also optimizing the overall motion duration which is incorporated into c_{task}. The tight distribution of candidate solutions after 120 iterations indicates that CMA-ES has converged.

Algorithm 3: Receding-horizon BS-VP-STO

Input: Robot configuration $\boldsymbol{q}_0^r$ and velocity $\dot{\boldsymbol{q}}_0^r$, object belief b_0, receding horizon length H.
Output: Robot trajectory $\boldsymbol{u}$.

$\boldsymbol{u} \leftarrow \emptyset$
while *task not solved* **do**
  $\boldsymbol{q}_{0:K}^{r*}, \dot{\boldsymbol{q}}_{0:K}^{r*} \leftarrow$ BS-VP-STO$(\boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r, b_0)$
  $\boldsymbol{u}_{0:H-1}^* \leftarrow \boldsymbol{q}_{1:H}^{r*}$
  for $k \leftarrow 0$ *to* $H-1$ **do**
    `// Stochastic rollout`
    ${}^i\boldsymbol{q}_{k+1}^o \sim p(\cdot|{}^i\boldsymbol{q}_k^o, \boldsymbol{u}_k^*),\ \forall i \in \{1, 2, .., N_p\}$
  end
  $\boldsymbol{q}_0^r, \dot{\boldsymbol{q}}_0^r \leftarrow \boldsymbol{q}_H^{r*}, \dot{\boldsymbol{q}}_H^{r*}$
  $b_0 \leftarrow \{{}^i\boldsymbol{q}_H^o\}_{i=1}^{N_p}$
  $\boldsymbol{u} \leftarrow \text{concatenate}\Big(\boldsymbol{u}, \boldsymbol{u}_{0:H-1}^*\Big)$
end

6.5 Receding-horizon BS-VP-STO

Planning a pushing maneuver over a single long horizon is challenging for two reasons: *i)* The dimensionality of the solution space of the optimization problem grows as the solution requires higher expressiveness. In the presented algorithm, this is reflected by an increasing number of via-points N that parameterize the robot trajectory. *ii)* In BS-VP-STO, the belief is rolled out using the nominal object dynamics, i.e. without inducing noise. For robust candidate trajectories, the belief may collapse to a Dirac-delta distribution after k_{collapse} steps, i.e. $\tilde{b}_k = \delta(\boldsymbol{q}_k^o), \forall k \geq k_{\text{collapse}}$, resulting in zero variance. In this case, the variance gain at those time steps is either $\gamma_k = 0$ if the robot does not touch the object, or $\gamma_k = 1$ if the robot touches the object. Thus, the optimization is strongly biased towards not touching the object after k_{collapse} steps.

For these reasons, we propose *receding-horizon* BS-VP-STO, a planning scheme for pushing maneuvers over longer horizons. The scheme alternates between computing a robust push via BS-VP-STO and performing a stochastic rollout of the solution. This allows to plan pushing-maneuvers over multiple shorter horizons while optimizing for task progress, i.e. pushing the object towards the goal, and robustness over a single horizon.

Algorithm 3 sketches the receding-horizon procedure for planning pushing maneuvers. Starting from the initial robot configuration $\boldsymbol{q}_0^r$ and velocity $\dot{\boldsymbol{q}}_0^r$, a given initial belief b_0 and the number of time steps for a receding horizon H, BS-VP-STO is used to generate a robust pushing trajectory $\boldsymbol{q}_{0:K}^{r*}$. In order to update the belief for the subsequent receding horizon, we perform a stochastic rollout of the robust pushing

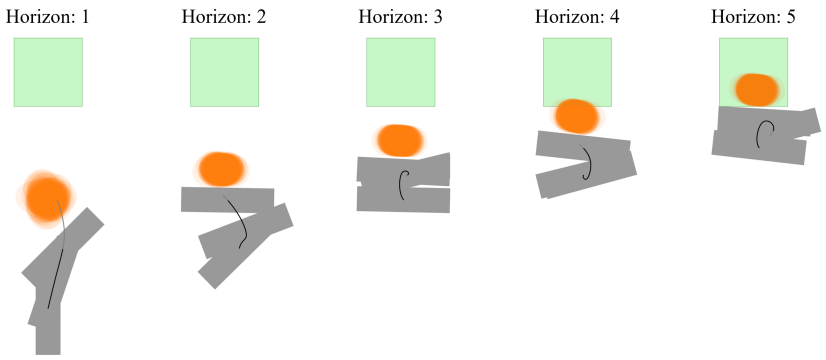


Figure 6.9: Receding-horizon BS-VP-STO optimizing the trajectory of a rectangular robot to push a circular object into a goal region subject to an uncertain initial object location and uncertain contact dynamics. The sub-figures show the robot trajectory optimized over the corresponding receding horizon together with a stochastic rollout of the object dynamics. In each receding horizon, the algorithm optimizes for a tradeoff between pushing progress and robustness. The resulting pushing maneuver is more robust than the solution found when optimizing over a single horizon as in Figure 6.8.

trajectory, i.e. sampling from the transition probability in (6.10). Note that the update of the belief can be extended by taking observations into account. For this, line 9 in Algorithm 3 may be replaced by a Bayesian state estimation update, e.g. a particle filter (Arulampalam et al. (2002)), thus turning the offline planning algorithm into an online re-planning approach. When planning offline, the optimized pushing trajectories of the receding horizons are sequenced to form a single continuous trajectory over a long horizon. This process may be repeated until a task-specific termination criterion is satisfied, e.g. the mean object configuration is within bounds of the target.

Example: Receding-horizon Robust 2D Object-Pushing. In this example, we solve the same problem as in **Example: Single-horizon Robust 2D Object-Pushing**, while iteratively optimizing over multiple receding horizons instead of running BS-VP-STO only once over the full horizon. For this, we only adapt the task-specific cost to reflect the pushing progress toward the goal. We measure this progress by computing the distance $d_k = ||\mathrm{E}[\boldsymbol{q}_k^o] - \boldsymbol{q}_{\mathrm{des}}^o||_2$ between the target area center $\boldsymbol{q}_{\mathrm{des}}^o$ and the mean object position at that time step. We compare the distance at the end of the receding horizon with the distance at the beginning of it in order to make the cost invariant to the absolute distance to the target. Progress is thus defined as $d_0 - d_K$. Consequently, the task-cost c_{task} is defined as follows

$$c_{\mathrm{task}}(\boldsymbol{u}_{0:K-1}) = e^{-\lambda(d_0 - d_K)}. \tag{6.54}$$

Figure 6.9 illustrates each solution of the optimization over multiple shorter horizons. It can be seen that a single-horizon push moves the object belief towards the target area with high probability. The overall pushing maneuver, obtained by sequencing the robot

trajectories of the individual horizons, has no constraints on the number of parameters, i.e. the number of via-points, as the number of receding horizon operations is not fixed but rather tied to a goal check. It is therefore expected to be more expressive than a single-horizon solution and thus more robust.

6.6 Experiments

This section presents robot experiments validating the theory and algorithmic approach developed in this chapter. We use objects that the robot has never interacted with before, with the geometry of the objects being the only information available. In all experiments, we compare the performance of the proposed approach against a baseline that only uses the nominal model of the contact dynamics without considering uncertainties. For this, the planner assumes that the initial object position is known and it uses the nominal object dynamics from Sec. 6.3 as a deterministic dynamics model. Consequently, we run the baseline without the cost and constraints on the variance gain.

6.6.1 Implementation Details

We implement the contact dynamics as in (6.6) for the special case of circular and rectangular shapes in two dimensions. In order to approximate the initial belief via particles, we found that $N_p = 20$ particles are sufficient to generate robust plans. For computing the CMA-ES updates to the Gaussian distribution over candidate trajectories, we use the Python package provided by the authors of Hansen (2016). The planning algorithm was executed on a laptop with an Intel Core i9-14900HX CPU and 32GB of RAM. On average, generating a robust pushing plan for the full target path takes around seven seconds of wall-clock time.

6.6.2 Open-Loop Single-Hand Pushing

The first experiment takes **Example: Receding-horizon Robust 2D Object-Pushing** from simulation into the real world. As an end-effector, the robot uses the rectangular-shaped hand of the Franka robot to push the target object, without considering the fingers for contact. Figure 6.10 illustrates the experimental setup showing the initial robot configuration, the initial object position, and the target object position. We compare trajectories of 2D positions and yaw angles of the hand-generated by our approach (receding-horizon BS-VP-STO) and a baseline approach. The baseline optimizes for efficient trajectories assuming that the nominal contact dynamics accurately predict the object trajectory. To evaluate the robustness of generated plans, we execute the plan to let the robot push the object into the target position and use the same plan for pushing the object back to the initial position. Subsequently, the same plan is executed

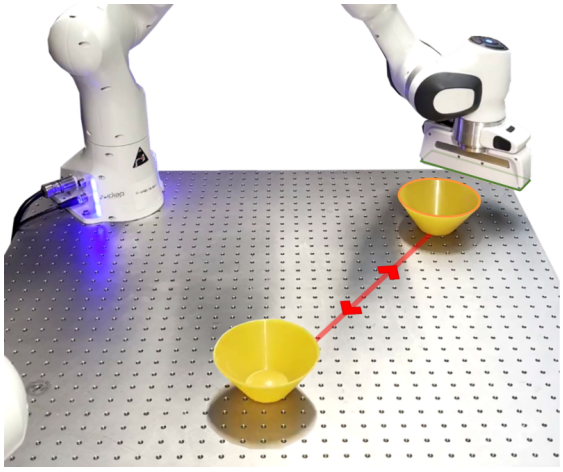


Figure 6.10: Open-loop single-hand pushing experiment: The task for the robot is to use the rectangular geometry of its hand (highlighted in green) to push the object with a circular geometry (highlighted in orange) into a target position without sensory feedback. The experiment consists of repeating the same pushing plan open-loop until the object diverges off the path such that the robot does not make contact anymore.

repeatedly open-loop with the object located where the previous execution ended. We expect that uncertainties in the contact dynamics will lead to deviations from the nominal model, resulting in the accumulation of control errors. Hence, we measure robustness by counting the number of successive runs until the robot loses the object. For a video showing qualitative results, see Extension 1 in the supplementary video. In the following, we report quantitative results.

Deterministic Baseline

As the baseline does not account for uncertainties in the contact dynamics, the resulting optimal trajectory consists of pushing in a straight line toward the target position while keeping the long side of the hand perpendicular to the pushing direction. We execute 10 experiments with the optimal baseline plan, resulting in an average of 6.8 (min. 5, max. 9) successive runs until the robot loses the object. This is the result of deviations from the nominal model due to real-world perturbations such as imperfect friction surfaces and mass distributions, leading to an accumulation of errors in the object positions when pushing a circular object with a flat surface.

Receding-horizon BS-VP-STO

The proposed algorithm uses the same nominal model that was used in the baseline while modeling additional uncertainty. The resulting optimal trajectory is executed in 10 open-loop experiments without additional perturbations to compare the result with the baseline. All experiments have been stopped after 40 successive runs as the system did not show any sign of accumulating errors. This indicates that the optimized trajectory actively controls the uncertainty in the object position by keeping track of the open-loop propagated belief.

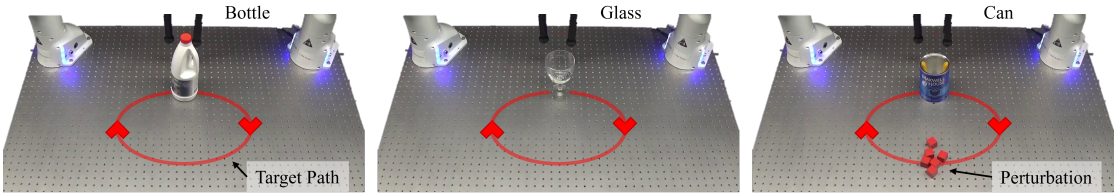


Figure 6.11: Open-loop bimanual pushing experiment: The two manipulators are considered as one bimanual robot with two end-effectors, each equipped with a ball-shaped end-effector. The object (bottle, glass, or can) is placed in front of the robot and the goal is to push the object along the target path (red circle). All objects were chosen due to their circular shape for the sake of a simple implementation of the quasi-static contact dynamics. The initial belief over the object position is Gaussian distributed with a mean equal to the initial object position and a covariance that reflects the uncertainty in object detection. The contact dynamics are modeled probabilistically to reflect uncertainty. For the experiments with the can, we add additional perturbations by placing wooden cubes (red cubes) on the target path and by changing the center of mass to be off-center by placing a heavy tool in the can (yellow content of the can).

6.6.3 Open-Loop Bimanual Pushing

We choose to validate the proposed algorithm with a bimanual pushing task, consisting of two Franka robot arms that are equipped with ball-shaped end-effectors. Note that we treat the two robot arms as one bimanual robot. We plan 2D trajectories for the ball-shaped end-effectors in a plane parallel to the table. The dynamics are modeled in this 2D plane, where the two robots are abstracted as two independent circles. The objects are abstracted as circles as well. Figure 6.11 shows the experimental setup. Initially, the object (bottle, glass, or can) is placed in front of the bimanual robot and the goal for the robot is to push the object along a circular target path. The belief over object positions is initialized with a Gaussian distribution. We furthermore modeled the noise in the contact dynamics with a Gaussian distribution and we tuned the covariance to capture the stochasticity in the contact dynamics. To prevent collisions between the two end-effectors, we include a collision constraint enforcing that the two end-effectors do not touch. We also add a constraint that prevents the robot from crossing its arms. Note that the time, location, and number of contacts are subject to planning, i.e. we do not impose any heuristic that forces the robot to use both end-effectors for pushing. Instead, the proposed cost and constraint on the variance gain drive the optimization algorithm to find stabilizing contact configurations and sequences, such as using both end-effectors for pushing.

Qualitative Planning Results

We illustrate plans generated with the deterministic baseline and with the proposed receding-horizon BS-VP-STO algorithm in Figure 6.12. Each plan is presented with

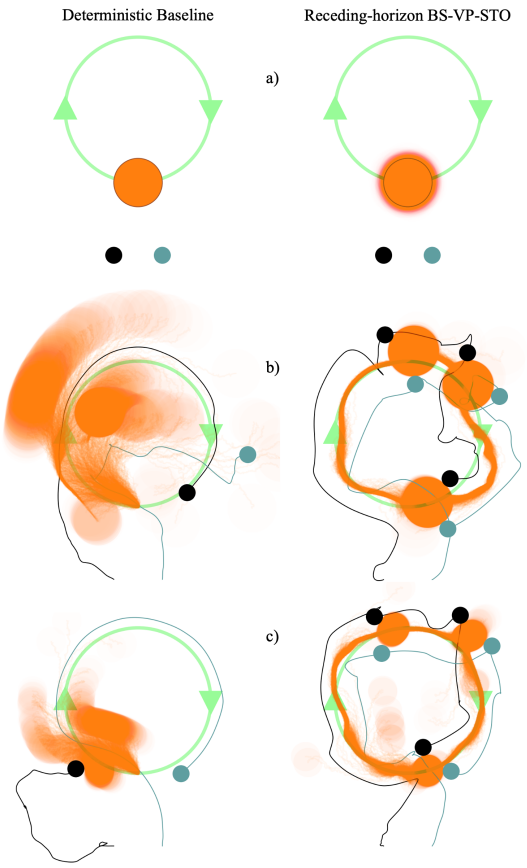


Figure 6.12: Qualitative comparison between the proposed receding-horizon BS-VP-STO algorithm and the deterministic baseline. We perform 1000 stochastic rollouts of each generated plan using the proposed stochastic object dynamics. The left and right end-effectors are visualized with black and blue circles, respectively. The target path is depicted in light green. a) For testing the baseline plan, the object position is initialized with the expected position (orange circle). For the proposed approach, we initialize the object position according to the modeled uncertainty. b) Plans generated for an object with a 5 cm radius, i.e. the bottle and the can. c) Plans generated for an object with a 3 cm radius, i.e. the glass.

1000 stochastic rollouts of the object dynamics used for optimization. For an evaluation of the real-world performance of the plans, please refer to Sec. 6.6.3.

Deterministic Baseline. In all plans generated with the deterministic baseline, the robot uses only one end-effector at a time for pushing the object. This is not surprising as the pushing progress, i.e. the mean control problem, is equally optimized when using one or two end-effectors. Thus, the baseline is not forced to discover the coordination between the two end-effectors and converges to the simpler solution, i.e. using one end-effector. When performing open-loop stochastic rollouts of the baseline plans, the object deviates from the planned trajectory after some time and thus the robot is not able to successfully push the object along the whole target path.

Receding-horizon BS-VP-STO. In contrast, we observe that the robot uses both end-effectors to push the object along the target path when planning with the receding-horizon BS-VP-STO algorithm. Note that the strategy of using two end-effectors for pushing deliberately emerges from planning in belief space. The proposed algorithm discovers the use of two end-effectors by penalizing sampled robot trajectories that result in an increasing uncertainty about the object position, i.e. using one end-effector. At the beginning of the pushing maneuver, the robot performs an action that reduces uncertainty by placing its end-effectors such that they enclose the initial belief. This

effectively brings the particles closer together, resulting in a decreasing variance. The robot then starts to make contact with its two end-effectors side-by-side to push the object along the target path with high probability. After pushing the object along the first half of the circular target path, the no-collision constraint between the two end-effectors forces the robot to break the contact and find a new contact configuration with which it can continue pushing. Consequently, the robot has to move its left end-effector around the object without touching it. Subsequently, the robot makes contact again with the object and continues pushing until it reaches the initial position again. When performing open-loop stochastic rollouts of the robust plans, the robot has a high probability of being successful at pushing the object along the target path despite the perturbations and the lack of feedback.

Quantitative Planning Results & Ablation Studies

In the following, we present ablation studies of the different components of receding-horizon BS-VP-STO. We evaluate *i)* the dependence of the overall algorithmic performance on the number of iterations taken in each BS-VP-STO instance within the receding-horizon setting, *ii)* the relevance of a contact-prior compared to an uninformed proposal distribution, and *iii)* how the receding-horizon BS-VP-STO algorithm scales with the number of degrees of freedom of the robot. We summarize all findings in Figure 6.13 which evaluates the success rate of the receding-horizon BS-VP-STO algorithm, i.e. if the planning algorithm finds a valid solution for the robot pushing an object with 5 cm radius. A plan is considered successful if the mean of the object belief is within 1 cm tolerance to the target location and if the plan is valid with respect to the robustness constraints on the variance gain. The planning algorithm is aborted after 500 iterations of receding-horizon BS-VP-STO and the plan is considered a failure. Sub-figure b) on the right-hand side in Figure 6.12 illustrates a successful plan for the problem considered for the ablations. We instantiate the planning problem with a varying number M of iterations for one BS-VP-STO instance within the receding-horizon scheme. For each M, we run receding-horizon BS-VP-STO 50 times and measure the success rate of the planning algorithm.

Impact of the Number of BS-VP-STO Iterations. Figure 6.13 illustrates the statistics of success over the number of BS-VP-STO iterations. We observe that the success rate increases with the number of iterations M and reaches around 100% success rate with $M = 4$ BS-VP-STO iterations. Note that running BS-VP-STO for one iteration poses a special case of the algorithm since no CMA-ES update is performed, resulting in only sampling an initial candidate population and picking the best-performing candidate. This procedure in fact corresponds to the predictive sampling algorithm introduced in Howell et al. (2022).

The planning time increases linearly with the number of iterations. For reference,

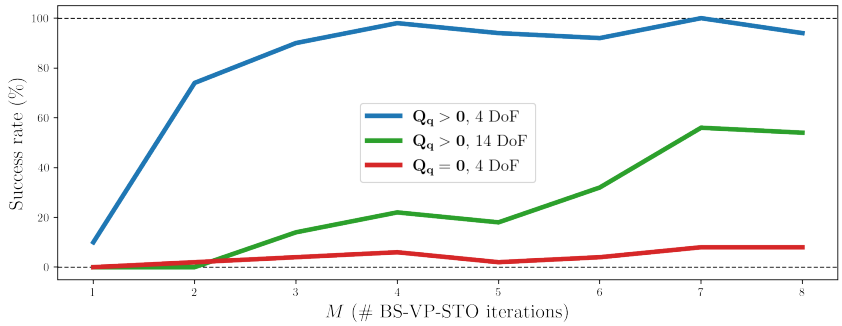


Figure 6.13: Success rates of receding-horizon BS-VP-STO over the number of BS-VP-STO iterations M. We compare the performance of the planning algorithm with the contact prior $\boldsymbol{Q_q} > \mathbf{0}$ and without $\boldsymbol{Q_q} = \mathbf{0}$; and we evaluate the scalability of the planning algorithm to many degrees of freedom (14 DoF). A planning run is considered successful if a valid solution is found within 500 single-horizon optimizations.

performing one BS-VP-STO iteration for four degrees of freedom takes approximately 0.01 seconds of wall-clock time. In this setup, we execute the pushing plan for an execution horizon H that corresponds to 0.2 seconds. Thus, it would be possible to run the planning algorithm in an online receding horizon fashion with up to $M = 20$ BS-VP-STO iterations per receding horizon.

Impact of the Contact Prior. To evaluate the impact of the contact prior, we set the contact precision matrix in BS-VP-STO to $\boldsymbol{Q_q} = \mathbf{0}$ (cf. (6.39)). This corresponds to uninformed trajectory samples as depicted in the left sub-figure of Figure 6.6. In Figure 6.13 we observe that the success rate of the algorithm without the contact prior is significantly lower than the success rate of the algorithm with the contact prior. This indicates that the contact prior is improving the efficiency of the planning algorithm to find robust manipulation actions in a few iterations. Computing the contact prior requires a matrix inversion to compute $\boldsymbol{Q_q}$ with dimensionality equivalent to the number of degrees of freedom. Since this operation has to be performed only once for a single horizon, the computational overhead of the contact prior is negligible.

Scalability to Many Degrees of Freedom. Last, we evaluate the scalability of the proposed planning algorithm to many degrees of freedom by moving from planning in the 2D plane to planning in the joint space of the bimanual robot, which has 14 degrees of freedom (seven degrees of freedom per robot arm). Coordinating all degrees of freedom adds complexity to the planning problem, while at the same time increasing the dimensionality of the search space. In Figure 6.13 we observe that the success rate for a given number of iterations M drops when increasing the complexity of the problem, requiring more iterations to discover the required coordination between the 14 joints. While the contact prior in joint space (cf. Sec. 6.4.2) imposes coordination between the seven joints of the individual arms, BS-VP-STO is still required to find joint trajectories such that the two end-effectors touch the object at the right place at

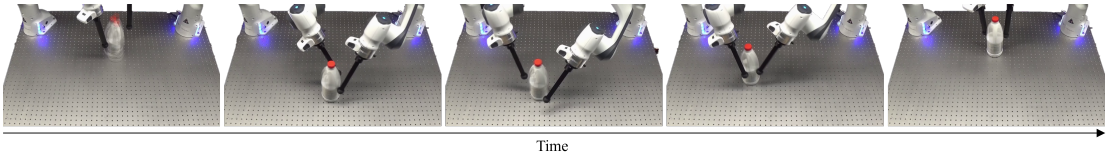


Figure 6.14: Snapshots of the robot behavior synthesized with the receding-horizon BS-VP-STO algorithm and executed on the real bimanual system. Each snapshot shows an overlay of five experiments that were conducted with different initial positions of the bottle. The robot successfully pushed the bottle along the target path in all five experiments. At the beginning, i.e. in the first image, the robot encloses the initial belief with its two end-effectors such that robust pushing is possible. After moving along the first half of the circular target path, i.e. in the third image, the robot re-positions its two end-effectors to continue pushing the bottle along the target path while avoiding collisions between the two arms.

the right time. The planning time per iteration is not significantly increased compared to planning for four degrees of freedom since only an additional forward kinematics computation for the belief rollout is required. However, note that the contact dynamics are still modeled using the ball-shaped abstraction of the end-effector. Modeling the whole kinematic chain of the robot arms as contact geometries adds additional computational complexity to the planner.

Real-world Pushing Results

Figure 6.14 shows snapshots of the robot behavior planned with the receding-horizon BS-VP-STO algorithm. In addition, a full video of the experiment can be found in Extension 2 in the supplementary video. We executed the planned robot trajectories for each of the three objects (bottle, glass, can), where we conducted five experiments for each object by placing the object at different initial positions. Out of 15 executed plans, the robot successfully pushed the object along the target path in 14 experiments. The robot failed to push the glass in one experiment, where the object was too far away from the mean initial position such the robot did not enclose the object during the initial uncertainty-reducing action.

We furthermore evaluated the deterministic baseline planner with the same three objects, while placing the objects only at the expected location. In all three experiments, the actual object position deviated from the planned object position after a few pushes, resulting in the robot losing contact with the object and thus failing to push the object along the target path. We show a video example of the motion generated by the baseline in Extension 2 in the supplementary video.

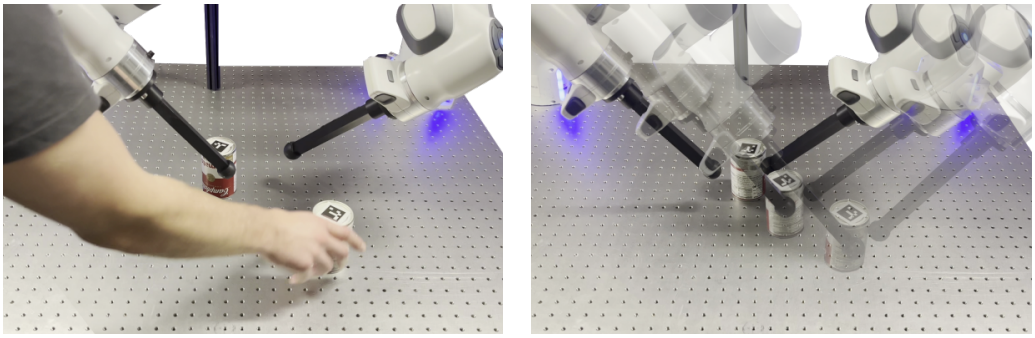


Figure 6.15: Closed-loop bimanual pushing experiment: The left image shows the external perturbations applied by manually moving the object. A camera continuously provides noisy observations of the object's position to the closed-loop controller. The right image shows how the proposed approach is able to generate robust plans in realtime, enabling uncertainty-aware model-predictive control loops.

6.6.4 Closed-Loop Bimanual Pushing

Last, we show that receding-horizon BS-VP-STO can be used in a closed control loop, gaining additional robustness against out-of-distribution disturbances. We use the same bimanual robot setup as in Sec. 6.6.3. However, instead of pushing an object along a target path, the task is to push the object to the center of the table while its position is perturbed by a human. In addition, noisy observations of the object's position are provided by a camera for closing the loop. We deploy a particle filter for continuously updating the belief based on both the stochastic rollout and noisy observations of the object as described in Sec. 6.5. We qualitatively compare the resulting behavior of the robot when being controlled with receding-horizon BS-VP-STO against the behavior when using the deterministic nominal model in a model-predictive controller. Both controllers run at a control rate of 5 Hz with $M = 6$ BS-VP-STO iterations per control step. Figure 6.15 illustrates the external perturbations applied to the object and the resulting robot behavior when controlled with our proposed robust approach. A video of the resulting behavior of both control approaches can be found in Extension 3 in the supplementary video.

We observe that the robot robustly pushes the object back to the center of the table using both end-effectors when controlled with the proposed approach. The additional state estimation enables the robot to also react to out-of-distribution perturbations, re-generating robust plans given noisy measurement updates. For the deterministic baseline, we observe that the continuous feedback enables the robot to maintain contact with the object and to generate consistent pushes. However, we observe that the robot only uses one end-effector for pushing as also observed for the open-loop experiment in Sec. 6.6.3. In contrast to our approach, this leads to larger control errors during pushing that need to be corrected, resulting in the deterministic baseline taking more time to accurately bring the object back to the center of the table.

6.7 Discussion

In this chapter, we investigated the problem of planning robust manipulation actions subject to stochastic contact dynamics. The quasi-static model used to predict contact dynamics is a simplification of the real-world contact dynamics tailored to the particular problem of slow pushing. In particular, reducing the dynamics from joint robot-object dynamics to solely object dynamics enables efficient reasoning over belief dynamics. However, the provided model excludes other categories of manipulation tasks involving effects such as grasping objects. We leave it to future work to investigate how other manipulation dynamics can be reduced to object-only dynamics.

Furthermore, we have shown that informed prior distributions for sampling candidate actions are beneficial, if not necessary, for sampling-based optimization for contact-rich manipulation. We used the product of Gaussian priors to bias the sampling towards smooth and contact-making trajectories. As the evolutionary optimization algorithm CMA-ES is based on sampling from and iterating on Gaussian distributions, we incorporated our Gaussian-distributed prior to initialize CMA-ES. Yet, we see great potential in the use of non-Gaussian priors that are further optimized with a stochastic optimization algorithm such as BS-VP-STO. Especially when the system has many degrees of freedom such as for two robot arms or articulated robotic hands, sampling from a proposal distribution that captures possibly non-Gaussian correlations between the degrees of freedom, e.g. correlations between fingers, is expected to be a key to scalable, realtime control through contacts.

Last, we show that a manipulation task such as pushing, which is typically approached with high-bandwidth closed-loop control, can also be stabilized by planning appropriate open-loop actions that deliberately optimize for robustness. However, if the robot is not able to anticipate perturbations or the statistical properties of the perturbations, feedback is required to be able to stabilize the manipulation. This opens the question of the *optimal* combination of open-loop robust planning and continuous feedback. Closing the loop may be done by re-planning robust actions if the observations deviate from the planned belief, e.g. in a low-bandwidth feedback loop.

7 Conclusion

This thesis presents theoretical and algorithmic contributions towards controlling objects through contact with a robot manipulator. They aim to address two main challenges that arise from the task of controlling objects through contacts: the discontinuity of contact dynamics and the uncertainty about object properties. The presented review of state-of-the-art approaches to controlling objects through contacts (cf. Section 1.2) shows that efficient exploration of the contact space is crucial for *making-and-breaking* contacts. In related work, this is achieved by discretizing the contact space and combining discrete and continuous optimization, *i.e. ad-hoc* planning. Another approach we see in related work is to train a policy in simulation using reinforcement learning and then transfer it to the real world, *i.e.* learning-based control. This thesis aims to bridge the gap between *ad-hoc* planning techniques that do not require long training cycles and learning-based approaches that scale well with the complexity of the problem. Based on the insight that stochastic optimization, for instance using CMA-ES, provides a useful tool for overcoming discontinuous and non-convex optimization landscapes induced by the contact dynamics, we develop a stochastic trajectory optimization algorithm (cf. Chapter 4). The parameterization of the underlying robot trajectories based on a few via-points enables setting probabilistic priors to the optimization problems such as smoothness or setting a probabilistic prior on the robot making contact with the object of interest. We show that contact-inducing probabilistic priors are particularly useful to efficiently explore the space of contacts in order to enable the robot manipulator to make and break contacts in realtime.

Besides the discontinuity inherent to contact dynamics, another key challenge on the way towards robots that autonomously manipulate their environment is the uncertainty in the dynamics. Manipulating new, unseen objects naturally leads to high uncertainty in the dynamics of the object before interacting with it since physical properties are unknown in general. This uncertainty can be accounted for by means of belief-space planning and control. We show that even for complex dynamics such as contacts, the belief over the object as part of the underactuated system can be predicted efficiently and deterministically (cf. Chapter 6). With such a model of the

belief dynamics, robots are able to generate manipulation actions that are robust to this natural uncertainty.

The theoretical and algorithmic contributions are evaluated in a series of experiments on real robots. We demonstrate that the robot manipulator performs dynamic hand-overs, which requires avoiding early contact with the object even when the object is moving. In another experiment, the robot pushes a box towards a moving target. Since the target is moving, the robot is required to efficiently reason about first breaking contact with the object, moving around the object and then making contact with the object again. We furthermore demonstrate a robot manipulator playing air hockey, which requires the robot to quickly reason over collisions between its end-effector and the puck and to adapt its shooting trajectory in realtime. Last, we show a robot with two independent end-effectors manipulate objects robustly in the presence of uncertainty in the dynamics of the object. These experiments demonstrate robotic manipulation skills that push the boundaries of current model-based planning and control approaches, indicating that efficient sampling-based control is a promising direction for reasoning through stochastic models of contact dynamics, as claimed in the thesis statement (cf. Section 1.4).

7.1 Limitations

This work presents novel contributions towards the goal of robots being able to dexterously interact with their environment. However, there is still a large number of challenges to be addressed until robots can autonomously operate in unstructured environments, such as tidying up people's homes or cooking their dinner. In the following, we discuss the major limitations of the presented work.

7.1.1 Limitations of Current Models of Contact Dynamics

All model-based planning and control approaches are limited by the underlying model. A major limitation of current models of contact dynamics is the tradeoff between accuracy (small time steps) and computational efficiency (large time steps) due to the underlying numerical integration of differential equations. This tradeoff typically prohibits long-horizon planning through contacts in realtime, even if the contact interaction is simple, for instance when an air hockey puck is bouncing off a wall (cf. Chapter 5).

Current models of contact dynamics derived from physics typically assume rigid bodies with known object poses and geometries. However, the gap between robots perceiving the world (mostly) through cameras and this simplified representation of the world is large. In lab environments, this gap is often circumvented by specifically using rigid

objects with known geometries and by deploying motion capture systems to track those objects. While such conditions facilitate the focused development of control algorithms based on available models, they easily break in real-world scenarios. If the model does not capture physical effects such as *deformation*, the robot will not be able to exploit such effects that might even be necessary for the task at hand. Potential future work is discussed in Section 7.2.1.

7.1.2 Limitations of Gaussian Trajectory Priors

The presented work uses Gaussian trajectory priors to encode prior knowledge about the task, such as encoding probabilities for making contact with an object. The Gaussian structure enables sampling and optimizing in a latent space. However, to control more complex contact geometries, a single Gaussian distribution has to have a large variance to cover the space of possible contacts, which makes sampling-based optimization algorithms less efficient. Tasks such as in-hand manipulation with robotic hands may require non-Gaussian priors that cover the multimodality inherent to the problem to efficiently explore promising contacts. In Section 7.2.2, we discuss future work on training generative models to combine informative trajectory priors with sampling-based optimization.

7.1.3 Limitations of Local Optimization

By using zero-order optimization algorithms for trajectory optimization, we increase chances of finding a desired local minimum compared to gradient-based algorithms. However, the additional exploration does not guarantee to find the global minimum of the non-convex optimization problem. In this thesis, we studied tasks where the local minima typically correspond to contact modes. For example, touching an object from one side or not touching the object form two different local minima with respect to object-centric objectives. The presented work uses exploration to discover the best contact mode. However, this requires the manipulation task to be simple enough such that the next best contact mode is reflected by the largest change in the objective function. While this holds for pushing objects closer to a target pose, it likely does not hold for manipulation tasks with sparse rewards or costs, i.e. where early decisions have a large impact on the final outcome while the final outcome is not reflected in the immediate reward. An example for such a task is for a robot to grasp a book that is lying flat on a table such that the robot cannot grasp the book in that configuration. The robot has to first push the book to the edge of the table to create a gap between the book and the table, and then grasp the book. Such tasks require robots to plan in a more global, abstract space than robot trajectories. This naturally extends to interconnecting local optimization with global planning algorithms, which is studied in the field of *task and motion planning*, e.g. by Garrett et al. (2021); Xue et al. (2024).

7.2 Future Work

The presented work opens up a number of directions for future research. In the following, we discuss the most promising directions.

7.2.1 Physics-based Learning of Stochastic Contact Dynamics

The presented work shows that reasoning over models of contact dynamics in realtime is crucial for robots to robustly manipulate objects through contacts. However, as discussed in Section 7.1.1, current models of contact dynamics derived from physics are limited to simplifying assumptions, *e.g.* rigid-body contacts, and by the tradeoff between accuracy and computational efficiency. If those limitations were to be lifted, model-based control algorithms for contact-rich manipulation could be applied to a wider range of objects while possibly running at higher frequencies. Learned *world models* (Ha and Schmidhuber (2018); Wu et al. (2023); Sakagami et al. (2023)) may help to overcome the practical limitations of discretizing and solving differential equations. However, learning such models typically requires large amounts of data and the state representation is often not interpretable, which makes the definition of the reward/cost functions challenging. Especially when trained on images and/or videos, learned world models that also capture the contact dynamics potentially predict physically infeasible interactions. A promising direction for future work is to combine physics-based models with learned dynamics models to guide model learning towards physically plausible interactions. However, it remains an open question how the state-based representation of physics-based models can be transferred to learned models with typically abstract representations of the world.

Furthermore, we have shown that stochastic models of contact dynamics are crucial to achieve robust manipulation when interacting with novel objects. Thus, future work on learning models of contact dynamics should enable predicting not only the most likely outcome of a contact interaction but also a distribution over possible outcomes as shown for learning collision models in Chapter 5. Last, learned models of contact dynamics should be able to predict the outcome of contact interactions with an arbitrary resolution of time. Ideally, a single model can be used to predict the dynamics with both a coarse resolution and higher uncertainty, as well as with a fine resolution and lower uncertainty. Such scalable models of contact dynamics would open up opportunities for an efficient more coarse exploration of the contact space and for a more refined optimization of discovered contact modes and sequences.

7.2.2 Learning Generative Trajectory Priors

The presented work follows the idea of stochastic optimization, *i.e.* sampling candidate solutions from a proposal distribution and refining the proposal distribution given

the evaluated candidates. Yet, as discussed in Section 7.1.2, a Gaussian proposal distribution may not be expressive enough for discovering and optimizing for more complex contact interactions. Hence, an interesting direction for future work is to learn a proposal distribution using a generative model, for instance a diffusion model (Chi et al. (2024)). If data has been collected on a wide range of meaningful examples of contact interactions, sampling from a learned proposal distribution may cover the space of promising contacts in a more suitable way than a Gaussian distribution. Yamada et al. (2024) recently approached this problem training a diffusion model on play data capturing physical interactions of a robotic hand with a deformable object. By continuously sampling multiple sequences of robot actions from the learned model and selecting the best sequence based on the simulated rollout of the sequences, the resulting performance improves in contrast to to taking only one sample. However, it is an open question what an iterative optimization algorithm within the latent space of a generative model could look like beyond selecting the best-performing sample. Such a combination of model-based planning and learning-based proposal distributions potentially achieves stronger generalization to new scenarios than either of the two approaches alone.

7.2.3 Reasoning over Tactile Feedback

Many contributions presented in this thesis are concerned with generating robot actions for manipulating objects through contact. We typically assume full observability of the environment through cameras. However, cluttered environments or occlusions may make it difficult for the robot to accurately perceive the environment. Moreover, cameras may not be able to capture local details of the environment, such as if an object in a robotic hand is slipping. Tactile feedback can provide complementary information to cameras. For instance, a robot that has to pick an object from a cluttered box should be able to rely on tactile feedback to detect features of the object. Similarly to controlling contact-rich manipulation, tactile-based state estimation suffers from the discontinuous nature of contacts: if the robot does not touch the environment, the tactile measurement is independent of the state of the robot and the environment, *e.g.* a force sensor returns zero for both of the following two scenarios: *i)* if the robot is very far away from an object, and *ii)* if the robot is almost touching an object. Estimating the object state upon contact is further constrained to a contact manifold, *i.e.* the set of robot and object states that have a distance equal to zero without penetration. Thus, algorithms for reasoning over tactile feedback have to take such discontinuities and constraints into account. Many state estimation techniques work with the assumption that similar states lead to similar observations (*e.g.* in terms of Euclidean distance), which does not hold for tactile feedback.

An interesting direction for future work is to combine Bayesian state estimation with advanced sampling techniques that are able to capture the set of states that possibly

explain a tactile observation, similar to Nagami and Schwager (2024). In combination with the belief-space planning and control approach presented in Chapter 6, robots may be able to generate manipulation actions that seek to reduce the state uncertainty through informative tactile feedback and by exploiting favorable contact dynamics.

A Appendix to Chapter 6

Separation of the Expected Cost

Suppose that the state $\boldsymbol{x} \in \mathbb{R}^{n_x}$ that is to be controlled is a random variable that is distributed with $\boldsymbol{x} \sim p$. Furthermore suppose that the task is described by a quadratic cost of the state and a desired state $\boldsymbol{x}_{\mathrm{des}}$, i.e.

$$J_{\mathrm{det}}(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}})^{\top} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}}) . \tag{A.1}$$

The corresponding stochastic optimal control objective is given by the expectation of the quadratic cost of the state:

$$\begin{aligned} J_{\mathrm{sto}} &= \mathrm{E}_p \left[J_{\mathrm{det}}(\boldsymbol{x}) \right] \\ &= \mathrm{E}_p \left[(\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}})^{\top} (\boldsymbol{x} - \boldsymbol{x}_{\mathrm{des}}) \right] \\ &= \mathrm{E}_p \left[\boldsymbol{x}^{\top} \boldsymbol{x} \right] - 2\mathrm{E}_p \left[\boldsymbol{x} \right]^{\top} \boldsymbol{x}_{\mathrm{des}} + \boldsymbol{x}_{\mathrm{des}}^{\top} \boldsymbol{x}_{\mathrm{des}}. \end{aligned} \tag{A.2}$$

In the following, we denote the mean of the state with

$$\bar{\boldsymbol{x}} = \mathrm{E}_p \left[\boldsymbol{x} \right] . \tag{A.3}$$

Furthermore, the variance of the state is defined as the expectation of the squared deviations from the mean:

$$\begin{aligned} \mathrm{V}_p \left[\boldsymbol{x} \right] &= \mathrm{E}_p \left[(\boldsymbol{x} - \bar{\boldsymbol{x}})^{\top} (\boldsymbol{x} - \bar{\boldsymbol{x}}) \right] , \\ &= \mathrm{E}_p \left[\boldsymbol{x}^{\top} \boldsymbol{x} + \bar{\boldsymbol{x}}^{\top} \bar{\boldsymbol{x}} - 2\bar{\boldsymbol{x}}^{\top} \boldsymbol{x} \right] , \\ &= \mathrm{E}_p \left[\boldsymbol{x}^{\top} \boldsymbol{x} \right] + \bar{\boldsymbol{x}}^{\top} \bar{\boldsymbol{x}} - 2\bar{\boldsymbol{x}}^{\top} \mathrm{E}_p \left[\boldsymbol{x} \right] , \\ &= \mathrm{E}_p \left[\boldsymbol{x}^{\top} \boldsymbol{x} \right] - \bar{\boldsymbol{x}}^{\top} \bar{\boldsymbol{x}}. \end{aligned} \tag{A.4}$$

As a result, the stochastic optimal control objective can be rewritten in terms of the deterministic quadratic cost of the mean state and the variance of the state with respect

to its probability distribution p:

$$\begin{aligned} J_{\mathrm{sto}} &= \bar{\boldsymbol{x}}^{\top}\bar{\boldsymbol{x}} - 2\bar{\boldsymbol{x}}^{\top}\boldsymbol{x}_{\mathrm{des}} + \boldsymbol{x}_{\mathrm{des}}^{\top}\boldsymbol{x}_{\mathrm{des}} + \mathrm{V}_p\left[\boldsymbol{x}\right] \\ &= \left(\bar{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{des}}\right)^{\top}\left(\bar{\boldsymbol{x}} - \boldsymbol{x}_{\mathrm{des}}\right) + \mathrm{V}_p\left[\boldsymbol{x}\right] \\ &= J_{\mathrm{det}}(\bar{\boldsymbol{x}}) + \mathrm{V}_p\left[\boldsymbol{x}\right]. \end{aligned} \tag{A.5}$$

Bibliography

Ajaykumar, G., Stiber, M., and Huang, C.-M. (2021). Designing user-centric programming aids for kinesthetic teaching of collaborative robots. *Robotics and Autonomous Systems*, page 103845.

Akgun, B., Cakmak, M., Jiang, K., and Thomaz, A. L. (2012a). Keyframe-based learning from demonstration. *International Journal of Social Robotics*, 4(4):343–355.

Akgun, B., Cakmak, M., Yoo, J. W., and Thomaz, A. L. (2012b). Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 391–398.

AlAttar, A., Rouillard, L., and Kormushev, P. (2019). Autonomous air-hockey playing cobot using optimal control and vision-based bayesian tracking. In *International Conference Towards Autonomous Robotic Systems (TAROS)*.

Albu-Schäffer, A., Ott, C., and Hirzinger, G. (2007). A unified passivity-based control framework for position, torque and impedance control of flexible joint robots. *The International Journal of Robotics Research*, 26(1):23–39.

Andrychowicz, O. M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2020). Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

Atrom, K. J. (1971). *Introduction to stochastic control theory*. Mathematics in science and engineering. Elsevier, Burlington, MA.

Aydinoglu, A. and Posa, M. (2022). Real-time multi-contact model predictive control via admm. In *International Conference on Robotics and Automation (ICRA)*, page 3414–3421.

Aydinoglu, A., Sieg, P., Preciado, V. M., and Posa, M. (2022). Stabilization of complementarity systems via contact-aware controllers. *IEEE Transactions on Robotics*, 38(3):1735–1754.

Aydinoglu, A., Wei, A., Huang, W.-C., and Posa, M. (2024). Consensus complementarity control for multi-contact mpc.

Bangura, M. and Mahony, R. (2014). Real-time Model Predictive Control for Quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780.

Bentivegna, D. C., Atkeson, C. G., and Cheng, G. (2004a). Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47(2-3):163–169.

Bentivegna, D. C., Atkeson, C. G., Ude, A., and Cheng, G. (2004b). Learning to act from observation and practice. *International Journal of Humanoid Robotics*, 1(04):585–611.

Betts, J. T. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming, Second Edition*. Society for Industrial and Applied Mathematics, second edition.

Bhardwaj, M., Sundaralingam, B., Mousavian, A., Ratliff, N., Fox, D., Ramos, F., and Boots, B. (2022). STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation. In *Conference on Robot Learning (CoRL)*, pages 750–759.

Billard, A. G., Calinon, S., Dillmann, R., and Schaal, S. (2008). Robot programming by demonstration. In Siciliano, B. and Khatib, O., editors, *Handbook of Robotics*, pages 1371–1394. Springer, Secaucus, NJ, USA.

Bishop, B. E. and Spong, M. W. (1999). Vision based control of an air hockey playing robot. *IEEE Control Systems Magazine*, 19(3).

Büchler, D., Guist, S., Calandra, R., Berenz, V., Schölkopf, B., and Peters, J. (2022). Learning to play table tennis from scratch using muscular robots. *IEEE Transactions on Robotics*.

Byravan, A., Boots, B., Srinivasa, S. S., and Fox, D. (2014). Space-time functional gradient optimization for motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 6499–6506.

Calinon, S. (2016). A tutorial on task-parameterized movement learning and retrieval. *Intelligent Service Robotics*, 9:1–29.

Calinon, S. (2019a). Learning from demonstration (programming by demonstration). In Ang, M. H., Khatib, O., and Siciliano, B., editors, *Encyclopedia of Robotics*. Springer.

Calinon, S. (2019b). Mixture models for the analysis, edition, and synthesis of continuous time series. In Bouguila, N. and Fan, W., editors, *Mixture Models and Applications*, pages 39–57. Springer.

Calinon, S., Bruno, D., and Caldwell, D. G. (2014). A task-parameterized probabilistic model with minimal intervention control. In *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, pages 3339–3344, Hong Kong, China.

Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. (2015). Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set. *IEEE Robotics Automation Magazine*, 22(3):36–52.

Chen, C., Culbertson, P., Lepert, M., Schwager, M., and Bohg, J. (2021). Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8262–8268.

Cheng, X., Huang, E., Hou, Y., and Mason, M. T. (2021). Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526.

Chernova, S. and Thomaz, A. L. (2014). Robot learning from human teachers. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 8(3):1–121.

Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., and Song, S. (2024). Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*.

Chuck, C., Qi, C., Munje, M. J., Li, S., Rudolph, M., Shi, C., Agarwal, S., Sikchi, H., Peri, A., Dayal, S., et al. (2024). Robot air hockey: A manipulation testbed for robot learning with reinforcement learning. *arXiv preprint arXiv:2405.03113*.

Cleac'h, S. L., Howell, T., Yang, S., Lee, C.-Y., Zhang, J., Bishop, A., Schwager, M., and Manchester, Z. (2021). Fast contact-implicit model-predictive control.

Coumans, E. (2015). Bullet physics simulation. In *ACM SIGGRAPH 2015 Courses*, SIGGRAPH '15, New York, NY, USA. Association for Computing Machinery.

Cover, T. M. and Thomas, J. A. (2005). *Differential Entropy*, chapter 8, pages 243–259. John Wiley & Sons, Ltd.

Dogar, M. and Srinivasa, S. (2011). A framework for push-grasping in clutter. In Hugh Durrant-Whyte, N. R. and Abbeel, P., editors, *Proceedings of Robotics: Science and Systems (RSS '11)*, pages 65–73. MIT Press.

Erdmann, M. and Mason, M. (1988). An exploration of sensorless manipulation. *IEEE Journal on Robotics and Automation*, 4(4):369–379.

Fishman, A., Paxton, C., Yang, W., Fox, D., Boots, B., and Ratliff, N. (2020). Collaborative interaction models for optimized human-robot teamwork. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11221–11228.

Florence, P., Lynch, C., Zeng, A., Ramirez, O. A., Wahid, A., Downs, L., Wong, A., Lee, J., Mordatch, I., and Tompson, J. (2022). Implicit behavioral cloning. In *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 158–168.

Garrett, C. R., Chitnis, R., Holladay, R., Kim, B., Silver, T., Kaelbling, L. P., and Lozano-Pérez, T. (2021). Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(Volume 4, 2021):265–293.

Goldshtein, M. and Tsiotras, P. (2017). Finite-horizon covariance control of linear time-varying systems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 3606–3611.

Gomez-Gonzalez, S., Neumann, G., Schölkopf, B., and Peters, J. (2020). Adaptation and robust learning of probabilistic movement primitives. *IEEE Transactions on Robotics (T-Ro)*, 36(2):366–379.

Graesdal, B. P., Chia, S. Y. C., Marcucci, T., Morozov, S., Amice, A., Parrilo, P. A., and Tedrake, R. (2024). Towards tight convex relaxations for contact-rich manipulation.

Ha, D. and Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Ha, J.-S., Driess, D., and Toussaint, M. (2020). A probabilistic framework for constrained manipulations and task and motion planning under uncertainty. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6745–6751.

Haarnoja, T., Moran, B., Lever, G., Huang, S. H., Tirumala, D., Humplik, J., Wulfmeier, M., Tunyasuvunakool, S., Siegel, N. Y., Hafner, R., et al. (2024). Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. (2019). Soft actor-critic algorithms and applications.

Handa, A., Allshire, A., Makoviychuk, V., Petrenko, A., Singh, R., Liu, J., Makoviichuk, D., Van Wyk, K., Zhurkevich, A., Sundaralingam, B., and Narang, Y. (2023). Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5977–5984.

Hansen, N. (2016). The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*.

Hogan, F. R. and Rodriguez, A. (2020). Reactive planar non-prehensile manipulation with hybrid model predictive control. *The International Journal of Robotics Research*, 39(7):755–773.

Hogan, N. (1984). Impedance control: An approach to manipulation. In *1984 American Control Conference*, pages 304–313.

Howell, T., Gileadi, N., Tunyasuvunakool, S., Zakka, K., Erez, T., and Tassa, Y. (2022). Predictive sampling: Real-time behaviour synthesis with mujoco. *arXiv preprint arXiv:2212.00541*.

Huang, J. and Cakmak, M. (2017). Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pages 453–462. IEEE.

Igeta, K. and Namiki, A. (2015). A decision-making algorithm for an air-hockey robot that decides actions depending on its opponent player's motions. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1840–1845. IEEE.

Igeta, K. and Namiki, A. (2017). Algorithm for optimizing attack motions for air-hockey robot by two-step look ahead prediction. In *IEEE/SICE International Symposium on System Integration*, pages 465–470.

Ijspeert, A. J., Nakanishi, J., and Schaal, S. (2002). Learning attractor landscapes for learning motor primitives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1547–1554.

Jankowski, J., Brudermüller, L., Hawes, N., and Calinon, S. (2023). VP-STO: Via-point-based stochastic trajectory optimization for reactive robot behavior. *International Conference on Robotics and Automation (ICRA)*.

Jankowski, J., Brudermüller, L., Hawes, N., and Calinon, S. (2024a). Robust pushing: Exploiting quasi-static belief dynamics and contact-informed optimization.

Jankowski, J., Girgin, H., and Calinon, S. (2021). Probabilistic adaptive control for robust behavior imitation. *IEEE Robotics and Automation Letters*, 6(2):1997–2004.

Jankowski, J., Marić, A., Liu, P., Tateo, D., Peters, J., and Calinon, S. (2024b). Energy-based contact planning under uncertainty for robot air hockey.

Jankowski, J., Racca, M., and Calinon, S. (2022). From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control. *IEEE Robotics and Automation Letters*, 7(2):3242–3249.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). STOMP: Stochastic trajectory optimization for motion planning. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574.

Kappen, H. J. (2015). Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162:1244–1266.

Kawato, M., Gandolfo, F., Gomi, H., and Wada, Y. (1994). Teaching by showing in kendama based on optimization principle. In *International Conference on Artificial Neural Networks*, pages 601–606. Springer.

Kicki, P., Liu, P., Tateo, D., Bou-Ammar, H., Walas, K., Skrzypczyński, P., and Peters, J. (2023). Fast kinodynamic planning on the constraint manifold with deep neural networks. *IEEE Transactions on Robotics*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kober, J. and Peters, J. (2008). Policy search for motor primitives in robotics. *Advances in neural information processing systems*, 21.

Kobilarov, M. (2012). Cross-entropy motion planning. *The International Journal of Robotics Research*, 31(7):855–871.

Koval, M. C., Pollard, N. S., and Srinivasa, S. S. (2016). Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. *The International Journal of Robotics Research*, 35(1-3):244–264.

Koç, O., Maeda, G., and Peters, J. (2018). Online optimal trajectory generation for robot table tennis. *Robotics and Autonomous Systems*, 105:121–137.

Kraft, D. (1988). *A software package for sequential quadratic programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR.

Liu, P., Bou-Ammar, H., Peters, J., and Tateo, D. (2024). Safe reinforcement learning on the constraint manifold: Theory and applications. *arXiv preprint arXiv:2404.09080*.

Liu, P., Tateo, D., Ammar, H. B., and Peters, J. (2022). Robot reinforcement learning on the constraint manifold. In *Conference on Robot Learning*, pages 1357–1366. PMLR.

Liu, P., Tateo, D., Bou-Ammar, H., and Peters, J. (2021). Efficient and reactive planning for high speed robot air hockey. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 586–593. IEEE.

Lozano-Perez, T. (1983). Robot programming. *Proceedings of the IEEE*, 71(7):821–841.

Lynch, K. and Park, F. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge Univeristy Press.

Lynch, K. M. and Mason, M. T. (1995). Stable pushing: Mechanics, controllability, and planning. *The International Journal of Robotics Research*, 15:533 – 556.

Manchester, Z., Doshi, N., Wood, R. J., and Kuindersma, S. (2019). Contact-implicit trajectory optimization using variational integrators. *The International Journal of Robotics Research*, 38(12-13):1463–1476.

Marcucci, T., Deits, R., Gabiccini, M., Bicchi, A., and Tedrake, R. (2017). Approximate hybrid model predictive control for multi-contact push recovery in complex environments. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 31–38.

Mason, M. T. (2001). *Mechanics of Robotic Manipulation*. MIT Press.

Migimatsu, T. and Bohg, J. (2020). Object-centric task and motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 5(2):844–851.

Mülling, K., Kober, J., and Peters, J. (2011). A biomimetic approach to robot table tennis. *Adaptive Behavior*, 19(5):359–376.

Muratore, F., Ramos, F., Turk, G., Yu, W., Gienger, M., and Peters, J. (2022). Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9.

Nagami, K. and Schwager, M. (2024). State estimation and belief space planning under epistemic uncertainty for learning-based perception systems. *IEEE Robotics and Automation Letters*, 9(6):5118–5125.

Namiki, A., Matsushita, S., Ozeki, T., and Nonami, K. (2013). Hierarchical processing architecture for an air-hockey robot system. In *2013 IEEE International Conference on Robotics and Automation*, pages 1187–1192. IEEE.

Nikandrova, E., Laaksonen, J., and Kyrki, V. (2014). Towards informative sensor-based grasp planning. *Robotics and Autonomous Systems*, 62(3):340–354. Advances in Autonomous Robotics — Selected extended papers of the joint 2012 TAROS Conference and the FIRA RoboWorld Congress, Bristol, UK.

Okamoto, K., Goldshtein, M., and Tsiotras, P. (2018). Optimal covariance control for stochastic systems under chance constraints. *IEEE Control Systems Letters*, 2(2):266–271.

Pang, T. (2021). A convex quasistatic time-stepping scheme for rigid multibody systems with contact and friction. *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6614–6620.

Pang, T., Suh, H. J. T., Yang, L., and Tedrake, R. (2023). Global planning for contact-rich manipulation via local smoothing of quasi-dynamic contact models. *IEEE Transactions on Robotics*, 39(6):4691–4711.

Paraschos, A., Daniel, C., Peters, J., and Neumann, G. (2018). Using probabilistic movement primitives in robotics. *Autonomous Robots*, 42.

Pignat, E., Silvério, J., and Calinon, S. (2020). Learning from demonstration using products of experts: Applications to manipulation and task prioritization. *arXiv:2010.03505*.

Platt, R., Kaelbling, L., Lozano-Perez, T., and Tedrake, R. (2017). Efficient planning in non-gaussian belief spaces and its application to robot grasping. In *Robotics Research*, pages 253–269. Springer.

Platt, R., Tedrake, R., Kaelbling, L. P., and Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*.

Ploeger, K., Lutter, M., and Peters, J. (2021). High acceleration reinforcement learning for real-world juggling with binary rewards. In *Conference on Robot Learning*, pages 642–653. PMLR.

Ploeger, K. and Peters, J. (2022). Controlling the cascade: Kinematic planning for n-ball toss juggling. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1139–1144. IEEE.

Posa, M., Cantu, C., and Tedrake, R. (2014). A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81.

Ratliff, N. D., Issac, J., Kappler, D., Birchfield, S., and Fox, D. (2018). Riemannian motion policies. *arXiv:1801.02854*.

Rodriguez, A. (2021). The unstable queen: Uncertainty, mechanics, and tactile feedback. *Science Robotics*, 6(54):eabi4667.

Rosmann, C., Makarow, A., Hoffmann, F., and Bertram, T. (2017). Time-Optimal nonlinear model predictive control with minimal control interventions. In *Proc. IEEE Conference on Control Technology and Applications (CCTA)*, pages 19–24.

Rubinstein, R. Y. (1999). The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190.

Sakagami, R., Lay, F. S., Dömel, A., Schuster, M. J., Albu-Schäffer, A., and Stulp, F. (2023). Robotic world models—conceptualization, review, and engineering best practices. *Frontiers in Robotics and AI*, 10.

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., Pan, J., Patil, S., Goldberg, K., and Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France. PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Shetty, S., Xue, T., and Calinon, S. (2024). Generalized policy iteration using tensor approximation for hybrid control. In *The Twelfth International Conference on Learning Representations*.

Shimada, H., Kutsuna, Y., Kudoh, S., and Suehiro, T. (2017). A two-layer tactical system for an air-hockey-playing robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Shirai, Y., Jha, D. K., and Raghunathan, A. U. (2023). Covariance steering for uncertain contact-rich systems. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7923–7929.

Steinmetz, F., Nitsch, V., and Stulp, F. (2019). Intuitive task-level programming by demonstration through semantic skill recognition. *IEEE Robotics and Automation Letters*, 4(4):3742–3749.

Tadokoro, K., Fukuda, S., and Namiki, A. (2022). Development of air hockey robot with high-speed vision and high-speed wrist. *Journal of Robotics and Mechatronics*, 34(5):956–964.

Taitler, A. and Shimkin, N. (2017). Learning control for air hockey striking using deep reinforcement learning. In *International Conference on Control, Artificial Intelligence, Robotics Optimization*.

Tedrake, R. and the Drake Development Team (2019). Drake: Model-based design and verification for robotics.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE.

Toussaint, M., Ha, J.-S., and Driess, D. (2020). Describing physics for physical reasoning: Force-based sequential manipulation planning. *IEEE Robotics and Automation Letters*, 5(4):6209–6216.

Toussaint, M., Harris, J., Ha, J.-S., Driess, D., and Hönig, W. (2022a). Sequence-of-constraints mpc: Reactive timing-optimal control of sequential manipulation. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13753–13760.

Toussaint, M., Harris, J., Ha, J.-S., Driess, D., and Hönig, W. (2022b). Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation. *arXiv preprint arXiv:2203.05390*.

Van den Broeck, L., Diehl, M., and Swevers, J. (2011). Model predictive control for time-optimal point-to-point motion control. *IFAC Proceedings Volumes*, 44(1):2458–2463.

van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748.

von Drigalski, F., Joshi, D., Murooka, T., Tanaka, K., Hamaya, M., and Ijiri, Y. (2021). An analytical diabolo model for robotic learning and control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4055–4061. IEEE.

Weiss, A., Igelsboeck, J., Calinon, S., Billard, A. G., and Tscheligi, M. (2009). Teaching a humanoid: A user study on learning by demonstration with HOAP-3. In *Proc. IEEE Intl Symp. on Robot and Human Interactive Communication (Ro-Man)*, pages 147–152, Toyama, Japan.

Wieber, P.-B., Tedrake, R., and Kuindersma, S. (2016). Modeling and control of legged robots. In *Springer handbook of robotics*, pages 1203–1234. Springer.

Williams, G., Aldrich, A., and Theodorou, E. A. (2017). Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40(2):344–357.

Williams, G., Drews, P., Goldfain, B., Rehg, J. M., and Theodorou, E. A. (2016). Aggressive driving with model predictive path integral control. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440.

Wrede, S., Emmerich, C., Grünberg, R., Nordmann, A., Swadzba, A., and Steil, J. (2013). A user study on kinesthetic teaching of redundant robots in task and configuration space. *J. Hum.-Robot Interact.*, 2(1):56–81.

Wu, P., Escontrela, A., Hafner, D., Abbeel, P., and Goldberg, K. (2023). Daydreamer: World models for physical robot learning. In Liu, K., Kulic, D., and Ichnowski, J., editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 2226–2240. PMLR.

Xue, T., Razmjoo, A., and Calinon, S. (2024). D-lgp: Dynamic logic-geometric program for reactive task and motion planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 14888–14894.

Yamada, J., Zhong, S., Collins, J., and Posner, I. (2024). D-cubed: Latent diffusion trajectory optimisation for dexterous deformable manipulation. *arXiv preprint arXiv:2403.12861*.

Yan, L., Stouraitis, T., Moura, J., Xu, W., Gienger, M., and Vijayakumar, S. (2024). Impact-aware bimanual catching of large-momentum objects. *IEEE Transactions on Robotics*, 40:2543–2563.

Zaidi, Z., Martin, D., Belles, N., Zakharov, V., Krishna, A., Lee, K. M., Wagstaff, P., Naik, S., Sklar, M., Choi, S., et al. (2023). Athletic mobile manipulator system for robotic wheelchair tennis. *IEEE Robotics and Automation Letters*, 8(4):2245–2252.

Zhang, Z., Tomlinson, J., and Martin, C. (1997). Splines and linear control theory. *Acta Math. Appl*, 49:1–34.

Zheng, D., Ridderhof, J., Tsiotras, P., and Agha-mohammadi, A.-a. (2022). Belief space planning: a covariance steering approach. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11051–11057.

Zhou, Y., Gao, J., and Asfour, T. (2019). Learning via-point movement primitives with inter- and extrapolation capabilities. In *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, pages 4301–4308.

Zucker, M., Ratliff, N., Dragan, A. D., Pivtoraiko, M., Klingensmith, M., Dellin, C. M., Bagnell, J. A., and Srinivasa, S. S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193.

Montreux, Switzerland
July 18, 2024
jankowskijulius@gmail.com

Julius Jankowski

Education

École Polytechnique Fédérale de Lausanne (EPFL) & Idiap Research Institute 2020 – 2024
PhD in Robotics, advised by Dr Sylvain Calinon & Prof. Jean-Philippe Thiran.
Thesis: "A Stochastic Approach to Contact-rich Manipulation"

Technical University of Munich (TUM) 2017 – 2019
MSc in Robotics, Cognition, Intelligence, thesis supervised by Prof. Daniel Rixen, Grade 1.0 (highest).
Thesis: "Optimal Trajectory Planning for Redundant Manipulators"

Leibniz University of Hannover 2013 – 2017
BSc in Mechatronics, thesis advised by Prof. Sami Haddadin, Grade 1.0 (highest).
Thesis: "Implementation of an Assistive Grasp Control for the Upper Limb Prosthesis μlimb based on 3D-Object Detection and Visual Servoing"

Work Experience

Amazon Robotics 2023 – 2024
Applied Scientist Intern (Tools: Python) *Berlin, Germany*

Planning through contacts for robotic picking of warehouse items from cluttered bins.

Franka Emika - Research & Development 2018 – 2020
Research Engineer (Tools: C, C++, ROS, Git) *Munich, Germany*

Developed a robust, time-invariant path tracking controller for a robot manipulator (Master's Thesis). Deployed and tested an autonomous navigation stack for a differential-drive mobile robot. Developed and integrated an impedance controller for a torque-controlled differential-drive robot.

DLR (German Aerospace Center) - Institute of Robotics and Mechatronics 2018
Research Engineer (Tools: Matlab/Simulink) *Oberpfaffenhofen, Germany*

Implemented and tested a momentum-based observer for sensorless force estimation with a robot manipulator.

Academic Activities

Research Visit	Prof. Perla Maiolino, Soft Robotics Lab, University of Oxford	State Estimation through Touch with a Tactile Skin
Workshop	Lightning Talk and Poster Presentation	Compliant Robot Manipulation (ICRA 2022)
Teaching	Teaching Assistance	Robotics, UniDistance Suisse
Teaching	Master Project Supervision (EPFL)	6 Completed

Awards

1st Place at the Robot Air Hockey Challenge (NeurIPS 2023 competition) 2023
1st Place at the RoboCup Junior World Cup (Discipline: Rescue) 2008

Publications

Articles under Review

1. Jankowski, J., Brudermüller, L., Hawes, N. and Calinon, S. (2024) *Robust Pushing: Exploiting Quasi-static Belief Dynamics and Contact-informed Optimization.* submitted to the International Journal of Robotics Research (IJRR).
2. Brudermüller, L., Berger, G., Jankowski, J., Bhattacharyya, R., Calinon, S., Jungers, R. and Hawes, N. (2024) *CC-VPSTO: Chance-Constrained Via-Point-based Stochastic Trajectory Optimisation for Safe and Efficient Online Robot Motion Planning.* submitted to IEEE Transactions on Robotics (T-RO).
3. Jankowski, J., Marić, A., Liu, P., Tateo, D., Peters, J. and Calinon, S. (2024) *Energy-based Contact Planning under Uncertainty for Robot Air Hockey.* submitted to IEEE Robotics and Automation Letters (RA-L).

Peer-reviewed Journal Articles

1. Jankowski, J., Racca, M. and Calinon, S. (2022) *From Key Positions to Optimal Basis Functions for Probabilistic Adaptive Control.* RA-L.
2. Jankowski, J., Girgin, H. and Calinon, S. (2021) *Probabilistic Adaptive Control for Robust Behavior Imitation.* RA-L.
3. Lembono, T.S., Pignat, E., Jankowski, J. and Calinon, S. (2021) *Learning Constrained Distributions of Robot Configurations with Generative Adversarial Networks.* RA-L.

Peer-reviewed Conference Proceedings

1. Jankowski, J.*, Brudermüller, L.*, Hawes, N. and Calinon, S. (2023) *VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior.* ICRA. *Authors contributed equally.
2. Ewerton, M., Villamizar, M., Jankowski, J., Calinon, S. and Odobez, J.-M. (2023) *A Multitask and Kernel approach for Learning to Push Objects with a Target-Parameterized Deep Q-Network.* IROS.
3. Girgin, H., Jankowski, J. and Calinon, S. (2022) *Reactive Anticipatory Robot Skills with Memory.* ISRR.
4. Wittmann, J.*, Jankowski, J.*, Wahrmann, J. and Rixen, D.J. (2020) *Hierarchical Motion Planning Framework for Manipulators in Human-Centered Dynamic Environments.* RO-MAN. *Authors contributed equally.
5. Garofalo, G., Mansfeld, N., Jankowski, J. and Ott, C. (2019) *Sliding Mode Momentum Observers for Estimation of External Torques and Joint Acceleration.* ICRA.