From Classical Mechanics to Pattern Recognition: Model-Algorithm Co-design via Efficient Approximations

Thesis by
Albert Hao Li

In Partial Fulfillment of the Requirements for the
Degree of
Control and Dynamical Systems



CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2027
Defended August 14, 2026

© 2027

Albert Hao Li
ORCID: 0000-0001-8264-7868

All rights reserved except where otherwise noted

ACKNOWLEDGEMENTS

Five years ago, I came to Caltech seeking a contemplative place where I could contribute my own little piece of human knowledge to the world. As it turns out, this is harder than it seems! I lost and found my way many times and feel an immense sense of gratitude to so many people who could offer comfort and guidance through my highest highs, lowest lows, and all the ordinary days in between.

My family has always provided unconditional support and the freedom to explore, succeed, and make my own mistakes—even when I didn't deserve it. I often reflect on my implausible luck growing up with so many chances to succeed and grow, which I attribute to the dedication of my parents and my upbringing. The challenges that I have faced are dwarfed by the challenges of their journey to America as immigrants with almost nothing to their name except sheer perseverance. To my parents Jim and Donna and my sister Shirley: thank you, I love you.

My path to academia started at Berkeley, where I joined a chance lunch with my friends Philipp Wu and Rachel Thomasson, who told me about the cool research they were doing between classes. This led me to explore robotics research at various labs on campus, where I became convinced that I should go to grad school, first at Stanford and then at Caltech. Our annual brunch catch-ups have been some of my most anticipated hangouts over these past few years. Thanks for leading a horse to water and for all the fun times in Soda and beyond.

Caltech is a tiny place full of great people. When I was a young grad student, I was in a deep, deep rut until I serendipitously moved offices to the third floor of Annenberg, where I could endlessly bother senior grad students Andrew Taylor and Victor Dorobantu with my stupid questions on a whiteboard. These early conversations gave me a lot of confidence that I had at least one good idea in me, and jump-started my first project. At the same time, I was incredibly lucky that Preston Culbertson came over as a postdoc in the mechanical engineering department looking for a student to explore ideas in robotic manipulation with. Preston was, in many ways, my academic anchor at Caltech for many years—I learned so much in numerical optimization, coding, control theory, statistics, computer vision, the philosophy of robotics, life, and so on from the hours that we spent on talking on Slack or in the lab. These conversations blossomed into the core of my thesis work on efficient algorithms for robotic planning, control, and manipulation. Thank you so much.

I had a great time spitballing, laughing, and growing with friends in CMS and MCE. Special shoutout to Sabera Talukder, Bingliang Zhang, Wenda Chu, Jennifer Sun, John Lathrop, Ivan Jimenez, Uriah Israel, Noel Csomay-Shanklin, Zach Olkin, Will Compton, Gary Yang, and many more, who kept my sanity over lunches and dinners, karaoke, and trips to tall buildings to watch fireworks.

Lastly, I have so much gratitude for the faculty in CMS that have made Caltech such a special place, and especially my advisors Richard Murray and Yisong Yue. In particular, I want to thank Yisong for supporting me early on in my PhD by throwing a few thousand dollars into my first project (even without knowing me!) and continuing to be incredibly supportive as my research evolved (and became way more compute-hungry). I am grateful that we have a relationship where we can be completely frank with each other with complete trust. I look forward to many more years of laughter—and serious intellectual engagement—with you, even beyond Caltech.

ABSTRACT

Exact descriptions of real systems are rarely computationally tractable, and tractable descriptions are rarely exact. The manner in which one approximates system behavior, computational outcomes, or patterns in data is therefore of primary concern. In turn, the choices made during such approximations dictate which algorithms exist for system analysis, how fast they run, and what guarantees they can carry. The form this design problem takes depends on the origin of the model. When the governing laws are known, an approximation can be constructed deliberately, and the designer chooses which structure it exposes. When only a numerical surrogate is available, its errors are inherited rather than chosen, and the burden shifts to algorithms that can absorb them. When no known governing principles exist at all, the approximation itself must be learned from data, and it is valuable only insofar as it serves the algorithms that consume it. This thesis develops methods exploring all these regimes, beginning from highly-specific challenges motivated by mechanical systems in robotic manipulation and ending with general-purpose models of high-dimensional data like natural videos. Across these settings, recurring themes include computational tractability, robustness to uncertainty, the degree to which physical priors can be leveraged, and the question of how systems and data should be represented. By designing each approximation around the algorithm that consumes it, this thesis achieves state-of-the-art results in a variety of fields ranging from robotic manipulation to video understanding. These include the fastest multi-finger grasp synthesis method of its time, the first real-time sampling-based controllers competitive with reinforcement learning for contact-rich manipulation, and video-text encoders that outperform models trained on significantly more video data.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Albert H. Li, Preston Culbertson, Joel W. Burdick, and Aaron D. Ames. "FRoGGeR: Fast Robust Grasp Generation via the Min-Weight Metric". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 6809–6816. DOI: 10.1109/IROS55552.2023.10341806.
A.H.L. conceived the project, developed the theory, implemented the algorithms, ran all experiments, and wrote the manuscript.

[2] Albert H. Li, Preston Culbertson, and Aaron D. Ames. "Toward An Analytic Theory of Intrinsic Robustness for Dexterous Grasping". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 2992–2999. DOI: 10.1109/IROS58592.2024.10802185.
A.H.L. conceived the project, developed the theory, implemented the algorithms, ran all experiments, and wrote the manuscript.

[3] Tyler Ga Wei Lum, Albert H. Li, Preston Culbertson, Krishnan Srinivasan, Aaron D. Ames, Mac Schwager, and Jeannette Bohg. "Get a Grip: Multi-Finger Grasp Evaluation at Scale Enables Robust Sim-to-Real Transfer". In: *Proceedings of The 8th Conference on Robot Learning (CoRL)*. Vol. 270. PMLR, 2025, pp. 5405–5433.
A.H.L. helped conceive the project, contributed to the core codebase, conducted all hardware experiments, conducted the analytic simulation baseline experiments, and helped write the manuscript. Note: T.G.W.L. and A.H.L. contributed equally.

[4] Albert H. Li, Preston Culbertson, Vince Kurtz, and Aaron D. Ames. "DROP: Dexterous Reorientation via Online Planning". In: *2025 IEEE International Conference on Robotics and Automation (ICRA)*. 2025, pp. 14299–14306. DOI: 10.1109/ICRA55743.2025.11128433.
A.H.L. conceived the project, wrote the core codebase, designed and built the hardware, ran all experiments, and wrote the manuscript.

[5] Albert H. Li, John Zhang, Jan Brüdigam, Brandon Hung, Aaron D. Ames, Jiuguang Wang, Simon Le Cleac'h, and Preston Culbertson. "Judo: A User-Friendly Open-Source Package for Sampling-Based Model Predictive Control". In: *2026 IEEE International Conference on Robotics and Automation (ICRA)*. 2026.
A.H.L. redesigned and optimized the presented codebase, ran the hardware experiments for the cube rotation experiments, wrote and conducted the simulated benchmarks, and wrote the manuscript.

[6] Albert H. Li, Ivan Dario Jimenez Rodriguez, Joel W. Burdick, Yisong Yue, and Aaron D. Ames. "KALIKO: Kalman-Implicit Koopman Operator Learning For Prediction of Nonlinear Dynamical Systems". In: *2026*

IEEE International Conference on Robotics and Automation (ICRA). 2026.
A.H.L. conceived the project, developed the core algorithm, implemented the core Bayesian filtering code, performed the reconstruction and eigenfunction analysis, ran all simulated experiments, and wrote the manuscript. Note: A.H.L. and I.D.J.R. contributed equally.

[7] Albert H. Li, Bingliang Zhang, Wenda Chu, Yuan Sui, Hongkai Zheng, Sophia Stiles, Kailen Hargenrader, Yisong Yue, and Sabera J. Talukder. "Simmer: A Scalable Pretraining Recipe for Video-Text Encoders". In: Under review. 2026.
A.H.L. conceived of the method for synthetically enriching the textual labels, generated, iterated, and processed the corresponding data, designed the core initial infrastructure, helped plan and design the core experiments, set up the evaluation pipeline, and wrote a significant portion of the final manuscript. Note: A.H.L, B.Z., and W.C. contributed equally.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . v
Published Content and Contributions . . . vi
Table of Contents . . . vii
Chapter I: Introduction . . . 1
1.1 Recurring Themes . . . 2
1.2 Thesis Outline . . . 6

I Analytic Models for Robust Multi-finger Grasping 11
Chapter II: Fast Robust Grasp Generation via the Min-Weight Metric . . . 12
2.1 What Makes a Good Grasp? . . . 12
2.2 Preliminaries . . . 15
2.3 The Min-Weight Metric . . . 20
2.4 Fast Robust Grasp Generation . . . 28
2.5 Experimental Results . . . 31
2.6 Discussion . . . 37
Chapter III: A New Theory of Intrinsic Robustness and Probabilistic Grasping 39
3.1 Intrinsic vs. Extrinsic Robustness . . . 39
3.2 Preliminaries . . . 41
3.3 A Theory of Intrinsically-Robust Grasping . . . 42
3.4 Towards Probabilistic Force Closure . . . 51
3.5 Experimental Validation . . . 60
3.6 Conclusion . . . 62

II Physics Simulators as Models for Contact-rich Manipulation 64
Chapter IV: Robust Real-World Grasping via Simulation at Scale . . . 65
4.1 Learning Grasp Evaluators from Physics Simulation . . . 66
4.2 Preliminaries . . . 68
4.3 Related Work . . . 68
4.4 Dataset Construction and Grasp Synthesis . . . 70
4.5 Experiments . . . 84
4.6 Conclusion . . . 92
Chapter V: Simulation-based Predictive Control for Contact-rich Manipulation 94
5.1 Introduction . . . 94
5.2 Background and Preliminaries . . . 95
5.3 The DROP Algorithm . . . 97

5.4 Experiments . . . 106
5.5 Open-Source Tooling for Sampling-Based MPC . . . 111
5.6 Further Results Enabled by Fast Experimentation . . . 115
5.7 Conclusion . . . 118
5.8 Appendix: Implementation Details . . . 119
5.9 Appendix: A Primer on Factor Graphs and Incremental Smoothing . 126

III Data-driven Modeling for General Temporal Data 128
Chapter VI: Kalman-Implicit Koopman Operator Learning . . . 129
6.1 Introduction . . . 130
6.2 Mathematical Preliminaries . . . 134
6.3 The Kalman-Implicit Koopman Operator . . . 137
6.4 Analyzing KALIKO Embeddings . . . 140
6.5 Experiments . . . 144
6.6 Conclusion . . . 150
Chapter VII: Bernoulli-Quantized Tokenization . . . 151
7.1 Introduction . . . 151
7.2 Preliminaries . . . 154
7.3 Bernoulli-Quantized Tokenization . . . 157
7.4 Experiments . . . 158
7.5 Ablations . . . 171
7.6 Qualitative Analyses . . . 175
7.7 Conclusion . . . 177
7.8 Appendix: UCR15 Anomaly Detection Results . . . 177
Chapter VIII: Simmer: A Scalable Pretraining Recipe for Video-Text Encoders 179
8.1 Introduction . . . 179
8.2 SIMMER: A Training Recipe for Granular Video-Text Encoders . . . 179
8.3 Experiments . . . 179
8.4 Conclusion . . . 179
8.5 Appendix: Implementation and Evaluation Details . . . 179
Bibliography . . . 180

Chapter 1

INTRODUCTION

A dominant paradigm for building engineered systems is to construct a *model* of the system of interest and pair it with an *algorithm* that operates on it. There exists a tight coupling between the two: a model's structure dictates which algorithms are available and how well they perform. For example, linear dynamical systems admit linear stabilizing feedback controllers [1], problems that can be parameterized as convex optimization programs can be solved using efficient interior-point methods [2], and reduced-order models enable real-time computation that would be intractable at full scale [3]. Conversely, powerful general-purpose algorithms can motivate the design of models that are compatible with them. For instance, the success of gradient-based optimization has spurred an enormous effort to develop differentiable model parameterizations (e.g., in deep learning [4]).

However, in practice, systems of interest may exhibit structural properties that inhibit the ability to efficiently perform computation with them. In these settings, the practitioner must often *approximate*—constructing a model that is simpler than the true system to recover a new structure that enables a larger algorithm class for solving the task at hand. Towards this, the chief concern of this thesis is the study of such model approximations and their interaction with algorithmic design for prediction and control. We are motivated by problems in robotic manipulation, representation learning, and their intersection, since these domains introduce a diversity of structural priors (or the lack thereof) that in turn motivate a diversity of model approximations and algorithmic choices rooted in classical mechanics, optimization, and machine learning.

We begin with multi-finger grasping, a classical and well-studied problem whose rich geometric structure allows us to construct tractable convex approximations of grasp quality metrics, enabling state-of-the-art grasp synthesis and providing fertile ground for resulting theoretical analysis. We then study physics simulators as general-purpose numerical approximators of physical dynamical systems, permitting us to study robotic control problems beyond multi-finger grasping. Here, the approximate model is given rather than designed, and the challenge shifts to *compensating for* its errors: we study algorithms for either learning models from or controlling with an

imperfect simulator, applying them not only to grasping but also to more dynamic tasks like in-hand reorientation and loco-manipulation. Finally, we consider settings where analytic modeling is no longer viable: the data of interest may come from systems too complex to describe from first principles, or may expose no tractable notion of state to build a model around, even when the underlying physics is known. In this final part, we instead *learn* model approximations from data, developing methods for representation learning and prediction on general temporal data across a diverse suite of domains.

As we move from analytic mechanics to data-driven techniques, the amount of domain-specific structure that can be leveraged for approximation and algorithm design diminishes. Despite this, fundamental challenges like computational efficiency, robustness to uncertainty, and compatibility with downstream decision-making remain the same. The manner in which we address these challenges must adapt as the available structure changes, and the question of how to do so is the central question of this thesis.

This turn from hand-derived mechanics toward data-driven modeling is not unique to the present work. It reflects a broader movement that reshaped robotics over the years this research was conducted, in which many manipulation problems once approached through carefully constructed models of contact are instead attacked by scaling up data and learning the relevant models directly. The physics-simulator methods of this thesis develop one form of this idea, a discriminative one, in which large-scale simulated experience trains a model that evaluates candidate actions rather than generating them. A parallel and now-dominant line of work treats the problem yet more directly, fitting generative policies to demonstrations through imitation learning [5, 6] and training vision-language-action models on large, heterogeneous robot datasets [7, 8, 9, 10, 11, 12, 13, 14] in pursuit of general-purpose foundation models for robot control. The methods developed here are a precursor to this shift, making the same wager on scale and data-driven modeling while retaining more of the classical structure that the fully data-driven approaches set aside.

1.1 Recurring Themes

In service of this broad goal, this thesis explores several recurring themes, which we briefly survey here.

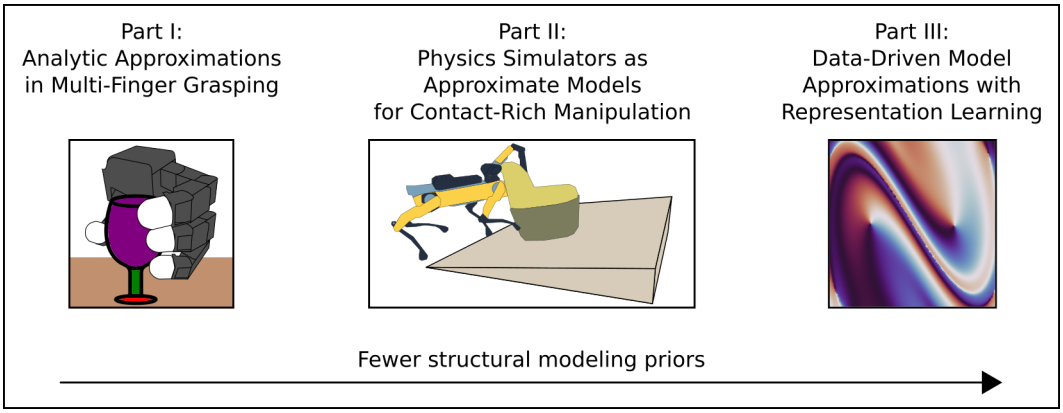


Figure 1.1: This thesis studies the interplay between model approximation and algorithmic design for prediction and control. It begins by developing *analytic* models for multi-finger grasping, continues with *physics simulators* as general-purpose numerical approximators of physical dynamical systems, and concludes with *data-driven* models for general temporal data. Along this progression, the structural priors that can be leveraged for approximation and algorithm design gradually diminish. As a result, a diverse range of methods are studied for exploiting the approximations in each chapter.

Table 1.1: Each part of the thesis explores the recurring themes below in different ways.

	Part I: Grasping	**Part II: Simulation**	**Part III: Data-driven Models**
Computational Efficiency	Runtime (SoTA grasp synthesis speed)	Resource (massively parallel simulation)	Sample + Runtime (cross-domain generalization and efficient prediction)
Physics Priors	Analytic mechanics	Physics-based simulation	Learned from data
Uncertainty & Robustness	Theoretical bounds	Systematic real-world experiments	Stochastic encoding and Bayesian filtering
Representation & State Estimation	Signed distance fields	Point clouds, NeRFs	Learned tokens, Koopman embeddings

1.1.1 Computational Efficiency

A critical consideration is the effect that an approximation has on the *computational efficiency* of the system. Computing relevant quantities fast enough to be useful (whether for real-time control or for large-scale offline planning) is a practical concern that shapes many of the forthcoming modeling decisions explored below. Several notions of efficiency are relevant.

Runtime efficiency characterizes the speed at which a sub-problem can be solved, which is critical when algorithms must run in real time. For instance, in robotics and control, model predictive controllers solve an optimization problem at each control

step over some finite horizon [15]. The temporal coarseness and horizon length are tunable parameters that allow the user to adjust the trade-off between runtime efficiency and solution quality. An algorithm's runtime efficiency becomes a crucial bottleneck when deployed on real-world systems.

Resource efficiency characterizes the computational resources (e.g., number of cores, available memory) required to solve a problem. Sampling-based control methods, for instance, require rolling out many candidate trajectories in parallel, demanding massively-parallelized simulation (either online or offline) [16, 17]. Such techniques have been influential across many fields, from robotic control [18] to reinforcement learning [19] to large language model inference [20].

Sample efficiency characterizes the amount of data required to build or calibrate a model. First-principles models derived from known physics require no data but may be inaccurate; purely data-driven models can be expressive but may require large datasets to train. Between these extremes lies a spectrum of approaches that embed physical priors like symmetries, conservation laws, or known dynamics into data-driven models to reduce their data requirements [21]. This thesis explores several points along this spectrum.

These notions of efficiency interact in nontrivial ways. A first-principles model may be highly sample-efficient, requiring no training data, but computationally expensive to evaluate, limiting its runtime efficiency. A learned surrogate may be cheap to query at inference time but require significant resources to train and large datasets to fit. Similarly, massively parallel rollouts can improve wall-clock runtime but at the cost of resource efficiency, demanding specialized hardware. Throughout this thesis, the choice of model and algorithm is shaped by which axes of efficiency matter most for the task at hand.

1.1.2 Modeling Physical Phenomena

A powerful advantage of modeling *physical* systems is the availability of strong priors derived from natural laws, like Newton's second law of motion. However, the degree to which these priors can be leveraged varies considerably, and several concerns arise in practice.

First, some physical phenomena resist accurate modeling across all scales and operating regimes. Friction models, for instance, range from the elementary Coulomb model to highly nonlinear, rate-and-state-dependent models studied in tribology [22], where behavior depends on fine effects like microscale dynamics.

When a simple model is used in a regime where it breaks down, the question becomes how to detect and compensate for its shortcomings.

Second, even when highly accurate models are available, computing quantities of interest exactly may only be possible numerically, and may be expensive. For example, certifying force closure of a robotic grasp requires checking whether a convex hull in wrench space contains the origin [23]. The Navier-Stokes equations fully describe fluid motion [24], yet resolving turbulent flows numerically is notoriously expensive [25]. While it is clear that such cases demand tractable approximations, *which one* to use depends greatly on the approximation's properties and algorithmic implementation.

Third, some phenomena are governed by known physics, but the available data may be too sparse or lack the measurements needed to make physical priors actionable. For example, the "weather" dataset commonly used in the time series forecasting literature [26] records only a handful of meteorological variables at a single station, far too little information to meaningfully construct a physics-based atmospheric model. In such settings, one must fall back on purely data-driven approaches.

1.1.3 Uncertainty and Robustness

Because the models we employ are imperfect, a natural consideration is how to analyze or accommodate their errors, which this thesis studies from three complementary angles.

The first is theoretical: when an approximation is introduced for computational convenience, one can analyze or bound its error using, for example, optimality certificates derived from convexity [2]. Characterizing the tightness of such approximations reveals when they can be trusted, providing a principled basis for trading off performance and computational tractability.

The second angle is empirical: when closed-form guarantees are unavailable, robustness can still be assessed by systematically stress-testing an algorithm under varying conditions by perturbing inputs, injecting noise, or transferring from simulation to real hardware (e.g., as in [27]).

The third is algorithmic: rather than improving the model itself, one can design methods that are inherently robust to uncertainty in the data they operate on, for instance by filtering noisy observations through a principled state estimator [28]. This shifts the burden from model accuracy to algorithmic resilience, reducing dependence on any single model's fidelity.

1.1.4 Representation and State Estimation

The notion of "state" is often taken for granted when reasoning about dynamical systems. In rigid body dynamics, the state of a kinematic chain is fully described by its joint positions and velocities [29], and this is sometimes sufficient for reasoning about the motion of a robot in isolation of its environment. However, when physical interactions like contact are involved, the state must also encode properties of the environment with which the robot interacts. For example, in robotic manipulation, an object's geometry determines the space of feasible grasps and the contact forces that can be applied, which in turn affects downstream controller design.

This makes the choice of *geometric* representation an integral part of algorithm design. Whether an object is described by a mesh, a signed distance field [30], a point cloud, or an implicit neural representation [31] has a direct effect on the efficiency and performance of downstream planning and control algorithms. Closely related is the question of *how* to estimate an object's geometry from sensor data.

More broadly, when we move to purely data-driven settings, classical notions of state may not apply at all. For example, for arbitrary time series with no known corresponding physical model, there is no canonical state space to appeal to. In these settings, the state must be *learned* from data, and the quality of this learned representation directly determines the quality of downstream prediction and control. This observation motivates the final part of the thesis, which asks how to discover compact, predictive representations for general temporal data.

1.2 Thesis Outline

The remainder of this thesis is organized into the following three parts.

- **Part I: Analytic Models for Robust Multi-finger Grasping.**
  - **Chapter 2: Fast Robust Grasp Generation via the Min-Weight Metric.** This chapter introduces the *min-weight metric*, a novel measure of grasp robustness that can be computed via a linear program. The min-weight metric is a computationally efficient approximation of the classical Ferrari-Canny [32] ε metric: its sign indicates force closure, its magnitude correlates with grasp robustness, and crucially, it is almost everywhere differentiable, enabling gradient-based grasp synthesis through a bilevel optimization program. We formulate grasp synthesis as a constrained optimization program over the robot's configura-

tion space, enforcing kinematic feasibility and collision avoidance while maximizing robustness. In both simulated and real-world experiments, the min-weight metric demonstrates state-of-the-art runtime efficiency while also exhibiting high grasping success rates. By designing an approximation with the right mathematical structure, we gain access to efficient gradient-based algorithms that would be unavailable for the original, non-differentiable metric.

  - **Chapter 3: A New Theory of Intrinsic Robustness and Probabilistic Grasping.** This chapter develops a formal theory of *intrinsic robustness* for dexterous grasping: robustness to uncertainties in object pose and geometry that affect grasp planning *before* execution, as opposed to the classical notion of extrinsic robustness (resistance to disturbances after grasping). The central result is a geometric sufficient condition guaranteeing that a nominally *force closure* grasp remains force closure under perturbations to the contact friction cones [23]. A direct consequence is that the min-weight metric from the preceding chapter provides a provable lower bound on the Ferrari-Canny metric, giving theoretical justification for its maximization as a grasp robustness metric. Building on this foundation, we introduce an analytic framework for computing the probability of force closure when surface normals at the contacts are uncertain. Where the preceding chapter constructs the approximation, this chapter characterizes when it can be trusted, pairing the worst-case bound with empirical evidence of its tightness and developing a probabilistic model of geometric uncertainty that informs grasp selection.

- **Part II: Physics Simulators as Models for Contact-rich Manipulation.**

  - **Chapter 4: Robust Real-World Grasping via Simulation at Scale.** This chapter marks the transition from constructing approximations analytically to compensating for the errors of an existing one. Here, the approximate model is a GPU-accelerated physics simulator, and the central hypothesis is that a discriminative *grasp evaluator* (here, a neural network trained to predict the probability of grasp success from perceptual data) can dramatically improve multi-finger grasping when trained on simulated data at sufficient scale. To test this, we constructed the largest existing dataset to date of millions of simulated grasps with visual sensing data and noisy success labels from perturbed simulation trials.

Grasp evaluators trained on this dataset are used to filter and refine candidate grasps generated by any upstream sampler, using sampling-based optimization over the learned success landscape. Hardware experiments demonstrate that evaluator-based methods achieve very high success rates over a wide array of object geometries and significantly outperform both analytic and generative baselines. The sim-to-real gap (the inevitable mismatch between the simulator's approximation and the true physics) is systematically studied empirically with a large number of ablations in both simulation and hardware.

  - **Chapter 5: Simulation-based Predictive Control for Contact-rich Manipulation.** This chapter extends the physics simulation paradigm from grasping to dynamic, contact-rich manipulation. We present DROP (Dexterous Reorientation via Online Planning), a system that performs in-hand cube reorientation using sampling-based model-predictive control (MPC) with a physics simulator as the predictive model. Unlike the dominant reinforcement learning paradigm, which trains policies offline via domain randomization, DROP plans online: at each control step, candidate action sequences are rolled out in parallel through MuJoCo, and the best-performing sequence is executed. A vision-based state estimation pipeline combining learned keypoint detection, fixed-lag smoothing, and a collision-aware corrector closes the loop on real hardware. The key insight is that drawing a diversity of samples at each time step naturally compensates for the simulator's approximation errors, since a high sample count regularizes model imperfections. DROP achieves hardware performance comparable to state-of-the-art RL methods while retaining the flexibility to adapt to new task specifications without retraining. We further present `judo`, an open-source software package that provides a unified framework for prototyping, benchmarking, and deploying sampling-based MPC controllers, and demonstrate its use on additional hardware platforms including quadrupedal loco-manipulation.

- **Part III: Data-driven Modeling for General Temporal Data.**

  - **Chapter 6: Kalman-Implicit Koopman Operator Learning.** This chapter enters the regime where the modeler receives only raw trajectories, with no analytic state description available to the learner, and

asks whether classical structure can be retained within learned approximations in a way that preserves both expressiveness and computational tractability. We present Kalman-Implicit Koopman Operator (KALIKO) learning, a method that learns a *linear* dynamical approximation of nonlinear systems by using the Kalman filter as an implicit encoder. Rather than training an explicit nonlinear encoder to map observations into a latent space where dynamics are linear, KALIKO uses Kalman filtering and smoothing to infer latent Koopman observables directly from data, training the system end-to-end. The resulting approximated Koopman dynamics inherits several desirable properties: prediction reduces to matrix multiplication in the lifted space (enabling fast inference), the embedded Kalman filter provides a principled manner in which to learn dynamics from noisy data, and the resulting models are naturally compatible with the zeroth-order sampling-based control algorithms explored in the previous chapters. Experiments on canonical nonlinear dynamical systems show that KALIKO recovers theoretically expected Koopman eigenfunctions, while experiments on high-dimensional PDE-generated wave data and a closed-loop control task demonstrate superior prediction and control performance relative to many neural network and transformer baselines. This chapter balances the expressiveness of data-driven modeling with the computational efficiency of classical methods in Bayesian state estimation.

– **Chapter 7: Bernoulli-Quantized Tokenization.** This chapter discards the last piece of structure retained by the preceding chapter, asking how to learn compact, informative representations of continuous time series data in a purely data-driven manner. Drawing inspiration from the success of discrete tokenization in large language models, we develop a Bernoulli-quantized tokenization scheme that maps continuous time series into discrete token sequences without assuming any particular physical structure. We evaluate the learned discrete approximation directly on reconstruction-driven tasks like time series imputation and anomaly detection, isolating the quality of the representation itself from any downstream model that might consume it. Critically, we demonstrate that the choice to use *stochastic* representation learning methods during training drastically improves the effective representational efficiency of the quantized "vocabulary" by encouraging exploration in

the embedding space, which directly translates to improved downstream performance on a diverse suite of data domains.

– **Chapter 8: A Scalable Pretraining Recipe for Video-Text Encoders.** The final chapter extends data-driven representation learning to video, the richest data stream considered in this thesis, where the relevant structure is semantic rather than dynamical and the supervision comes from paired natural language rather than reconstruction. We present Simmer, a scalable pretraining recipe for video-text encoders organized around two complementary roles for text. During pretraining, captions provide broad *coverage*: each video is augmented with many diverse synthetic captions, and this many-to-one supervision is absorbed by a multi-vector contrastive objective that lets different captions align with different components of the video representation. During a final *cooldown* stage, captions instead provide fidelity, with training annealed on a small corpus of human-sourced dense captions. Controlled pilot studies show these components are complementary, and scaling the recipe to 10M videos yields state-of-the-art performance across five video-text tasks and three model scales while using 2–60× fewer videos than prior large-scale models. Echoing the rest of the thesis, the supervision signal and the objective that consumes it must be designed together.

Part I

Analytic Models for Robust Multi-finger Grasping

Chapter 2

FAST ROBUST GRASP GENERATION VIA THE MIN-WEIGHT METRIC

Multi-finger grasping is a classical problem in robotics, with roots tracing back nearly five decades [33, 34]. We begin with a standard setting in which a fixed-base, fully-actuated serial manipulator must be controlled to pick a rigid object from the environment (e.g., off of a tabletop). This problem has rich geometric structure: contact mechanics, wrench spaces, and convex hulls provide a precise mathematical framework for reasoning about when a grasp is robust. However, the standard robustness metrics arising from this structure are non-differentiable and expensive to optimize, limiting the algorithms that can operate on them. This chapter illustrates the first mode of approximation studied in this thesis: *constructing* a computationally tractable model that preserves the essential properties of the original while unlocking a more powerful algorithm class. Concretely, we develop a linear programming (LP) relaxation of the classical Ferrari-Canny [32] grasp metric that is almost everywhere differentiable, enabling fast gradient-based grasp synthesis that would be infeasible with the original metric.

2.1 What Makes a Good Grasp?

Traditional methods [35] for grasp synthesis focus largely on optimizing analytic quality metrics (e.g., the *largest inscribed ball metric* [36, 32, 37], which we call the *Ferrari-Canny metric* as in [38]), which use hand-object contact points to measure a grasp's robustness to external perturbations. However, these methods suffer some drawbacks in practice: traditional metrics are often hard to optimize and may be non-differentiable, meaning grasps must be synthesized with slower sampling-based methods, like simulated annealing with rejection sampling when constraints are violated [39]. Further, they are sensitive to the object geometry (i.e., the surface location and normals), which requires detailed object models to be constructed offline, e.g., by using object scanning rigs [40, 41].

The focus of this chapter is the synthesis of multi-finger *precision grasps* for dexterous hands. Compared to power grasps, precision grasps are more useful for manipulation tasks that require delicate or accurate movements, such as tool use or

bin packing. We assume that each finger of the end-effector contacts a rigid object in exactly one location, and we consider the problem of quickly refining an initial pose for a dexterous hand into a robust precision grasp for a particular object. Our goals are twofold. First, we seek to generate these grasps quickly (in seconds rather than minutes for the methods available at the time [35, 42, 43]) while enforcing kinematic and collision constraints. Second, we seek to balance common trade-offs of grasp synthesis methods in terms of performance, speed, and interpretability.

2.1.1 Related Work

At the time this research was conducted, the majority of recent work in the grasping literature concerned *parallel-jaw* grasps, which admit simple parameterizations due to the low number of DOFs and ease of control [44, 45, 46, 47, 48]. Multifinger, dexterous grasping introduces numerous challenges, as the grasp parameterization must specify the states of each finger, and the high dimensionality of this representation demands more fine-grained control and better sensing. Moreover, the complex kinematics and potential for self-collisions complicate the search for feasible grasps. In turn, the problem of synthesizing dexterous grasps, particularly using data-driven methods, had received far less attention than parallel-jaw grasps.

Many classical methods for multifinger grasp synthesis ignore kinematic and collision constraints and only optimize for contact location by leveraging analytic grasp metrics [49]. The GraspIt! simulator explicitly considers these constraints but simplifies the problem by searching a lower-dimensional space of "eigengrasps" using simulated annealing [35]. This approach has several downsides, including the need to define eigengrasps for new hands (which is highly non-trivial), and slow computational speed, often only returning the results of a solve (which may not even have succeeded!) in about five minutes [42]. Overall, analytic metrics have two main drawbacks: (i) their usage is typically slow, often due to non-smoothness, and (ii) they demand high-fidelity estimates of object geometry and contact locations, diminishing their efficacy [38].

One reason for these challenges is the difficulty in casting the nonlinear constrained grasp optimization problem in a computationally tractable way. Among methods addressing this challenge, collision constraints are typically penalized instead of enforced, leading synthesized grasps to have high amounts of infeasible interpenetration [47, 43, 50]. Other attempts at solving the unrelaxed problem are computationally prohibitive (e.g., minutes to hours in [51]).

This chapter focuses on mitigating the slow speed of analytic metrics with the view that, despite their drawbacks, these metrics still provide a useful and interpretable framework for grasping that is agnostic to robot model, object representation, and quality of available data. The core strategy is to design an approximation with the right mathematical structure so that efficient gradient-based algorithms become available. To that end, we build on prior works that formulate differentiable approximations of analytic force closure measures to recover robust grasps on any multifinger arm/hand system using bilevel optimization.

These prior works propose methods of varying complexity, including solving and differentiating a sequence of linear programs (LPs) [52], a sequence of semidefinite programs (SDPs) [53], a sum of squares program [54], or a single SDP that only approximates force closure [50, 47]. In contrast, we propose in Sec. 2.3 a single LP whose optimal value mathematically indicates force closure and whose maximization yields robust grasps.

The method of Wu *et al.* [55] is most similar to ours as they also propose solving a bilevel optimization program with smooth collision constraints. However, instead of optimizing for robustness, they solve a feasibility problem and impose a force closure constraint parameterized as a quadratic program while training a conditional variational autoencoder (CVAE) to output performant initial grasps. We compare our method's formulation to theirs in Sec. 2.5.

We do not compare against other common analytic metrics in the literature [49] for two reasons. First, for a fair comparison of their runtime efficiency independent of implementation, they should be differentiable and agnostic to object and task, eliminating many choices such as independent contact regions or task-based methods. Second, among metrics satisfying these desiderata (e.g., the minimum singular value of the grasp matrix), the optimized grasps are not guaranteed to satisfy force closure, so they cannot provide robustness guarantees *even under perfect conditions*.

2.1.2 Contributions

The main contribution of this chapter is the formulation of grasp synthesis/refinement as a nonlinear optimization problem that leverages a novel, almost-everywhere differentiable approximation of the Ferrari-Canny metric: the *min-weight metric*. The end result is FRoGGeR (**F**ast **Ro**bust **G**rasp **Ge**nerato**r**), a framework for fast robust grasp generation. The optimization problem underlying FRoGGeR can, due to the properties of the min-weight metric, be solved efficiently using commercial

solvers. Additionally, FRoGGeR allows us to explicitly enforce constraints on grasps while harnessing the speed of gradient-based optimization.

FRoGGeR is designed to be compatible with existing methods in the grasping community. For instance, while FRoGGeR can synthesize grasps with no prior knowledge, it may also refine infeasible or suboptimal grasps computed by learning-based methods. Further, this chapter represents object geometry implicitly using *signed distance fields* (SDFs), which means we can leverage existing work that learns SDFs of objects from sensor data [30]. To allow the use of mesh-based object representations, we also present practical approximations of the SDF, its gradient, and its Hessian computable using only the mesh.

In summary, the contributions are as follows:

- FRoGGeR: a fast, robust grasp generator built on the *min-weight metric*, a novel, almost-everywhere differentiable approximation of the Ferrari-Canny grasp metric with a numerically efficient gradient;
- a practical procedure based on nonlinear optimization with an open-source implementation[1] that generates feasible grasps on the order of seconds; and
- numerical experiments and simulations comparing grasps generated by our method to prior work, thereby demonstrating the speed of the proposed approach.

In addition, we lay the groundwork for a theoretical understanding of modeling grasps, their uncertainty, and robustness properties, which is the main advantage of studying this classical, but constrained, problem setting. The next chapter builds directly on this foundation by proving that the min-weight metric lower bounds the Ferrari-Canny metric and developing a probabilistic theory of grasp robustness under geometric uncertainty.

2.2 Preliminaries

2.2.1 Mathematical Notation and Preliminaries

We consider a fixed-base, fully-actuated serial manipulator and dexterous hand with n_f fingers contacting the object. Denote by n the total DOFs of the system and $\boldsymbol{q} \in \mathcal{Q} \subset \mathbb{R}^n$ the manipulator's generalized positions. We let $\mathrm{FK}_i(\boldsymbol{q})$ and $\mathbf{J}_i(\boldsymbol{q})$ denote the

[1] Open-source implementation available at github.com/alberthli/frogger.

forward kinematics map and the corresponding Jacobian of some prescribed contact point i. For convenience, we denote the hand Jacobian as $\mathbf{J}_h = \text{blkdiag}(\mathbf{J}_1, \ldots, \mathbf{J}_{n_f})$.

Our goal is to manipulate a rigid object O with surface ∂O and body frame $\{O\}$. The pose of a frame $\{Z\}$ with respect to a frame $\{Y\}$ is expressed $\mathbf{T}_{YZ} \in \text{SE}(3)$, and similarly, $\mathbf{R}_{YZ} \in \text{SO}(3)$ represents the relative orientation of $\{Z\}$ with respect to $\{Y\}$. The robot base frame is denoted $\{B\}$.

We are now ready to mathematically formalize the notion of grasps.

Definition 2.1 (Feasible Configuration)**.** *A configuration $\boldsymbol{q}$ is* feasible *if it results in no self-collisions and no robot-object collisions; yields a stable grasp according to some metric of interest; and places the fingers on the surface of the object (i.e., it satisfies $\text{FK}_i(\boldsymbol{q}) \in \partial O$ for all $i \in \{1, \ldots, n_f\}$).*

Definition 2.2 (Grasp)**.** *A* grasp *is a pair $(\boldsymbol{q}^*, \mathbf{T}_{BO})$, where $\boldsymbol{q}^*$ is a feasible configuration.*

Definition 2.3 (Grasp Map)**.** *The* grasp map $\mathbf{G}(\boldsymbol{q})$ *is a configuration-dependent linear operator that maps a vector of finger contact forces $\mathbf{F}_C \in \mathbb{R}^{3n_f}$, expressed in their local contact frames, to an applied wrench on the object $\boldsymbol{w}_O \in \mathbb{R}^6$.*

The above definitions imply a minimal procedure for synthesizing multi-finger grasps: given an object at pose $\mathbf{T}_{BO}$, we can solve a constrained optimization program (or feasibility program) to compute a grasp at configuration $\boldsymbol{q}^*$.

First, we must specify a finger-object contact model, which will govern the necessary conditions for grasp stability. In this thesis, we assume a *Coulomb friction model*, which asserts that there is no finger-object slip if the contact forces remain in the *friction cone*. Mathematically, the no-slip condition states that the magnitude of the tangent contact forces does not exceed the static coefficient of friction times the (strictly positive) normal contact force: $\left\|\mathbf{F}_C^T\right\| \leq \mu F_C^N$. As is common in the literature, we use a *pyramidal approximation* [23] of the friction cone with n_s sides (see Fig. 2.1). This under-approximates the true friction cone in exchange for representing the no-slip condition using a linear, rather than conic, constraint. Concretely, the set of no-slip contact forces for the i-th finger is

$$\mathcal{F}_i = \left\{ \mathbf{F}_{Ci} \in \mathbb{R}^3 \;\middle|\; F_{C\,i}^N > 0,\ \left\|\mathbf{F}_{Ci}^T\right\| \leq \mu F_{C\,i}^N \right\}, \tag{2.1}$$

and the pyramidal approximation with n_s edges replaces $\mathcal{F}_i$ with

$$\hat{\mathcal{F}}_i = \text{conv}\left(\left\{ \boldsymbol{f}_i^{(1)}, \ldots, \boldsymbol{f}_i^{(n_s)} \right\} \right), \tag{2.2}$$

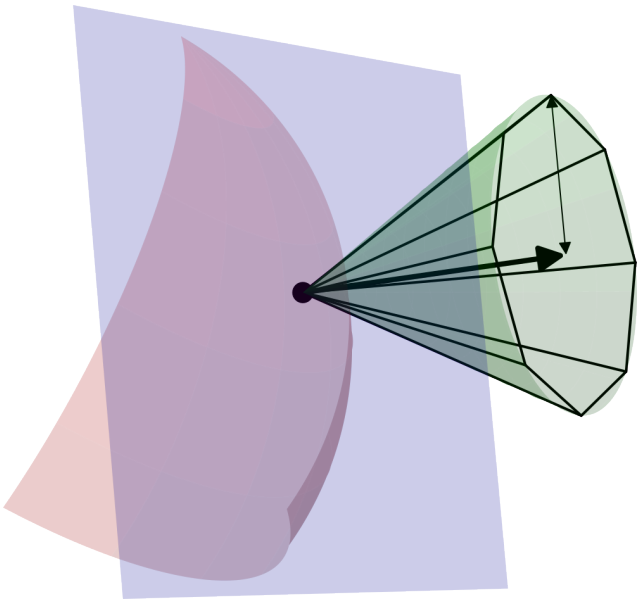


Figure 2.1: The friction cone (green) and its pyramidal approximation with $n_s = 8$ sides (black lines). The edges of the pyramid are the *basis forces*, which span the set of contact forces satisfying the no-slip condition in the set $\hat{\mathcal{F}}$. The pyramidal approximation is a conservative (inner) approximation of the true cone, converting the conic no-slip constraint into a linear one.

where $\boldsymbol{f}_i^{(1)}, \ldots, \boldsymbol{f}_i^{(n_s)}$ are the *basis forces* for the i-th finger, equally spaced on the boundary of the friction cone. The full set of no-slip contact forces across all fingers is then $\hat{\mathcal{F}} = \hat{\mathcal{F}}_1 \times \cdots \times \hat{\mathcal{F}}_{n_f}$.

Since the friction pyramid is convex, contact forces that are convex combinations of its edges also satisfy the no-slip condition, so the basis forces span the space of no-slip contact forces. Each basis force $\boldsymbol{f}_i^{(j)}$ at contact point $\boldsymbol{x}_i = \mathrm{FK}_i(\boldsymbol{q})$ induces a *basis torque* $\boldsymbol{\tau}_i^{(j)} = \boldsymbol{x}_i \times \boldsymbol{f}_i^{(j)}$. Concatenating each basis force with its corresponding basis torque yields a *basis wrench* $\boldsymbol{w}_k = \left(\boldsymbol{f}_i^{(j)}, \boldsymbol{\tau}_i^{(j)}\right) \in \mathbb{R}^6$. This map from basis forces to basis wrenches is precisely the action of the *grasp map* $\mathbf{G}(\boldsymbol{q})$, which produces $n_w = n_f \cdot n_s$ basis wrenches $\mathcal{W} = \{\boldsymbol{w}_k\}_{k=1}^{n_w}$, whose convex hull defines a (sub)set of generalized forces that can be applied to the object from the set of no-slip forces. We let the elements of $\mathcal{W}$ form the columns of the *wrench matrix*

$$\mathbf{W} = \begin{bmatrix} \boldsymbol{w}_1 & \ldots & \boldsymbol{w}_{n_w} \end{bmatrix} \in \mathbb{R}^{6 \times n_w},$$

a construction which will be very useful in forthcoming analyses.

The concept of basis forces and wrenches are crucial, because as we shall see, properties like grasp stability and quality are typically expressed in terms of the

forces or wrenches applied onto the object. We emphasize that the basis forces and wrenches depend on the positions of finger-object contacts, so they are themselves functions of the configuration $\boldsymbol{q}$, though this dependence is suppressed above for notational clarity. Therefore, reasoning explicitly about grasp stability by way of the basis wrenches allows us to formulate the grasp synthesis problem in a manner amenable to numerical optimization.

2.2.2 Force Closure

Force closure is one of the most classical notions of grasp stability. We say a grasp is *force closure* if (assuming infinite control authority) it can resist any external perturbations applied to the object. Formally, we have the following definition.

Definition 2.4 (Force Closure [23, Def. 5.2])**.** *A grasp* $(\boldsymbol{q}, \mathbf{T}_{\text{BO}})$ *is* force closure *if given any external wrench* $\boldsymbol{w}_{\text{E}}$ *applied to the object, there exists contact forces* $\mathbf{F}_{\text{C}}$ *such that* $\mathbf{G}(\boldsymbol{q})\mathbf{F}_{\text{C}} = -\boldsymbol{w}_{\text{E}}$.

Force closure is a concept fundamentally about the *directions* of perturbations. Observe that since $\mathbf{G}$ is a linear operator, if a grasp is not force closure, there exists some direction $\boldsymbol{d} \in \mathbb{R}^6$ such that for arbitrarily small $\beta \in \mathbb{R}$, there exists no solution $\mathbf{F}$ satisfying $\mathbf{GF} = -\beta\boldsymbol{d}$. Therefore, infinitesimal disturbance wrenches applied to the object in the direction $\boldsymbol{d}$ in wrench space would cause the grasp to destabilize. It stands to reason that force closure is a necessary property which must be satisfied by *any* stable grasp. This makes it a natural foundation for the development of grasp quality metrics and the derivation of the main results of this chapter.

2.2.3 Grasp Metrics

Numerous works over the last few decades have proposed analytic grasp quality metrics as a means for grasp synthesis. We will primarily focus on the canonical *Ferrari-Canny metric* [32], which is known by many other names, including the *epsilon metric* [38] and the *largest inscribed ball metric* [37]. In this chapter, we denote it as ϵ.

As discussed, the force closure property indicates that external perturbations along every direction in wrench space can be resisted given infinite control authority. Of course, in reality, control authority is limited, so there exists a direction $\boldsymbol{d}_{\text{min}}$ in wrench space that is the most sensitive to perturbations. That is, among all possible directions in wrench space, the smallest force that can destabilize the grasp is achieved by applying a disturbance to the object along $\boldsymbol{d}_{\text{min}}$.

This quantity is precisely what the (L_1) Ferrari-Canny metric measures, and in particular, this value is equivalent to the radius of the largest origin-centered ball that can be inscribed in the convex hull of basis wrenches conv ($\mathcal{W}$) [32, Sec. 4.2]. We remark that the original derivation of this metric was motivated by the realistic assumption of finite control authority: the grasp that achieves the maximum L_1 Ferrari-Canny metric is the one that minimizes the total sum of applied finger forces on the object [32, Sec. 4.4]. There also exists an L_∞ Ferrari-Canny metric, which instead corresponds to the grasp that minimizes the maximum finger force [32, Sec. 4.3]; from now on, we will set aside the L_∞ version and refer to the L_1 metric simply as the Ferrari-Canny metric.

The Ferrari-Canny metric ϵ associates a grasp's quality with its *robustness*. While this is the setting assumed in the rest of this chapter, we briefly remark that there are many other grasp metrics in the literature, such as manipulability-based metrics, task space metrics, contact region metrics, and more. For a thorough review of these alternative quality measures, we refer the reader to the excellent summary in [49].

2.2.4 Computational Considerations

With a grasp quality metric in hand, we now turn to the question of computation. As it turns out, even *evaluating* ϵ quickly is somewhat nontrivial, since it requires computing the convex hull of a finite set of points by parameterizing its facets with hyperplanes, then returning the smallest bias as the radius of the largest inscribed ball [56]. While this single computation is fairly simple in isolation, it suffers from a few serious drawbacks.

First, in an iterative optimization algorithm, repeatedly computing a convex hull is relatively slow, which hinders the usage of the Ferrari-Canny metric for real-world usage. Second, the process of constructing the convex hull is non-differentiable, which means that typical grasp quality optimization algorithms must rely on sampling-based methods. Because there are many constraints to consider (some of which are equality constraints), it is typical to use rejection sampling during optimization, which for highly non-convex problems can be very sample-inefficient (especially for precision grasps where infeasible grasps are much more common than feasible ones [35]).

To illustrate this point in more detail, it is productive to consider the mechanical steps required to evaluate the Ferrari-Canny metric.

1. For each finger i, compute the fingertip position $\boldsymbol{x}_i = \text{FK}_i(\boldsymbol{q})$ via forward kinematics.

2. At each contact point, compute a pyramidal friction cone approximation as a function of the surface normal. The n_s edges of this cone are the *basis forces* $\boldsymbol{f}_i^{(1)}, \ldots, \boldsymbol{f}_i^{(n_s)}$, and as explained above, the action of the grasp map yields corresponding *basis wrenches* $\boldsymbol{w}_k \in \mathcal{W}$.

3. Compute the convex hull $\text{conv}(\mathcal{W})$ over all n_w basis wrenches. Check whether the origin lies in the hull; if it does not, the grasp is not force closure, and the notion of robustness is undefined. Otherwise, continue. Both the process of computing the hull and the binary origin membership check are non-differentiable.

4. Compute the radius of the largest origin-centered ball inscribed in $\text{conv}(\mathcal{W})$. This radius is the Ferrari-Canny metric ϵ.

Due to the aforementioned non-differentiability, this "standard model" of grasp evaluation is not amenable to gradient-based optimization methods. Using methods from differentiable optimization [57], we seek to address this using the methods developed in this chapter.

2.3 The Min-Weight Metric

We now present the first major result of this thesis. As discussed in Sec. 2.2.3 and 2.2.4, the Ferrari-Canny metric provides a principled, analytic measure of grasp robustness, but suffers from computational bottlenecks that impose challenges for constrained multi-finger grasp synthesis. This motivates the forthcoming discussion, which aims to develop a simple approximate measure of grasp robustness amenable to fast robust grasp generation.

We begin by re-examining the force closure condition. Following the derivation of the Ferrari-Canny metric, it is well-known that a sufficient condition for force closure is for the origin to be contained in the convex hull of basis wrenches, i.e., $\mathbf{0} \in \text{conv}(\mathcal{W})$ [32, Sec. 4.2]. To check this condition, we can equivalently solve the

following linear feasibility program at some configuration $\boldsymbol{q}$:

$$\begin{aligned} \text{find} \quad & \boldsymbol{\alpha} && \text{(2.3a)} \\ \text{subject to} \quad & \mathbf{W}(\boldsymbol{q})\boldsymbol{\alpha} = \mathbf{0}, && \text{(2.3b)} \\ & \mathbb{1}_{n_w}^{\top}\boldsymbol{\alpha} = 1, && \text{(2.3c)} \\ & \boldsymbol{\alpha} \geq \mathbf{0}, && \text{(2.3d)} \end{aligned}$$

where $\mathbb{1}_{n_w}$ is the vector of all ones of length n_w. Constraints (2.3c)-(2.3d) simply assert that $\boldsymbol{\alpha}$ is a vector of valid convex weights and constraint (2.3b) states that the origin lies in the hull of the columns of $\mathbf{W}$, which are composed of the elements of $\mathcal{W}$.

The key idea of the min-weight metric is to relax constraint (2.3d) by allowing negative weights $\boldsymbol{\alpha}$. If the minimum weight in $\boldsymbol{\alpha}$ is non-negative, the feasibility program (2.3) is satisfied, so $\mathbf{0} \in \text{conv}(\mathcal{W})$. This motivates the following LP, whose optimal solution is parameterized by the configuration $\boldsymbol{q}$:

$$\begin{aligned} \ell^*(\boldsymbol{q}) = \underset{\boldsymbol{\alpha}\in\mathbb{R}^{n_w},\, \ell\in\mathbb{R}}{\text{maximize}} \quad & \ell && \text{(2.4a)} \\ \text{subject to} \quad & \mathbf{W}(\boldsymbol{q})\boldsymbol{\alpha} = \mathbf{0} && \text{(2.4b)} \\ & \mathbb{1}_{n_w}^{\top}\boldsymbol{\alpha} = 1 && \text{(2.4c)} \\ & \boldsymbol{\alpha} \geq \ell\mathbb{1}_{n_w}. && \text{(2.4d)} \end{aligned}$$

The min-weight metric exhibits several desirable properties.

First, $\ell^*(\boldsymbol{q})$ is defined even when the configuration $\boldsymbol{q}$ is not associated with a feasible, force closure grasp. This simple property is key: the Ferrari-Canny metric ϵ is undefined when a grasp is not force closure, because in this case, the origin does not lie in conv $(\mathcal{W})$, so vacuously, there does not exist any origin-centered ball inscribed in conv $(\mathcal{W})$. Thus, ϵ cannot be used to refine infeasible grasps into feasible ones. In contrast, we can optimize $\boldsymbol{q}$ such that $\ell^*(\boldsymbol{q})$ increases until positive, at which point it clearly satisfies force closure.

Second, computing ℓ^* is substantially easier than computing ϵ and all other approximations of ϵ discussed in Sec. 2.2.4, since evaluating it requires only solving a single LP. Moreover, we shall soon see that using differentiable optimization programs, we can also compute gradients of the min-weight metric efficiently.

Lastly, we can formally view the min-weight metric as the *metrization* of a grasp's force closure status in a continuous manner via the following result.

Theorem 2.1 (Force Closure Metrization)**.** *Assume that there exists some subset of $\mathcal{W}$ containing seven affinely independent basis wrenches. Then, problem* (2.4) *always has a feasible solution.*

Further, assume that $\operatorname{int}(\operatorname{conv}(\mathcal{W})) \neq \emptyset$. *Then, the optimal solution* $(\boldsymbol{\alpha}^*, \ell^*)$ *satisfies the following properties.*

(i) ***Non-Force Closure.*** *The min-weight metric is strictly negative if and only if the origin is not in the convex hull of basis wrenches:*

$$\ell^*(\boldsymbol{q}) < 0 \iff \mathbf{0} \notin \operatorname{conv}(\mathcal{W}).$$

(ii) ***Robust Force Closure.*** *The min-weight metric is strictly positive if and only if the origin is in the convex hull of basis wrenches:*

$$\ell^*(\boldsymbol{q}) > 0 \iff \mathbf{0} \in \operatorname{int}(\operatorname{conv}(\mathcal{W})).$$

(iii) ***Equilibrium.*** *The min-weight metric is exactly 0 if and only if the origin lies on the boundary of the convex hull of basis wrenches:*

$$\ell^*(\boldsymbol{q}) = 0 \iff \mathbf{0} \in \partial \operatorname{conv}(\mathcal{W}),$$

where ∂S denotes the boundary of a set S.

Proof. For the feasibility claim, it suffices to show that given the assumptions of the theorem, there always exists $\boldsymbol{\alpha}$ satisfying (2.4b) and (2.4c), since we can always set the value of ℓ to the minimum element of $\boldsymbol{\alpha}$. Let $\mathbf{W}' \in \mathbb{R}^{6\times 7}$ denote a submatrix of $\mathbf{W}$ with 7 affinely independent columns and $\boldsymbol{\alpha}' \in \mathbb{R}^7$ the associated subvector of weights in $\boldsymbol{\alpha}$. Besides these 7 elements, set all other elements in $\boldsymbol{\alpha}$ to 0. Consider the following ones-augmented wrench matrix and zeros vector:

$$\overline{\mathbf{W}}' = \begin{bmatrix} \mathbf{W}' \\ \mathbb{1}^\top \end{bmatrix} \in \mathbb{R}^{7\times 7}, \quad \bar{\mathbf{0}} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \in \mathbb{R}^7.$$

The columns of $\mathbf{W}'$ are affinely independent in $\mathbb{R}^6$ if and only if the columns of $\overline{\mathbf{W}}'$ are linearly independent in $\mathbb{R}^7$ [58, Exercise 1.1]. Thus, $\overline{\mathbf{W}}'$ is invertible, so there always exists a solution $\boldsymbol{\alpha}'$ satisfying $\overline{\mathbf{W}}'\boldsymbol{\alpha}' = \bar{\mathbf{0}}$. Equivalently, observe that $\mathbf{W}'\boldsymbol{\alpha}' = \mathbf{0}$ and $\mathbb{1}_7^\top \boldsymbol{\alpha}' = 1$, which implies that $\mathbf{W}\boldsymbol{\alpha} = \mathbf{0}$ and $\mathbb{1}_m^\top \boldsymbol{\alpha} = 1$, since we set all

the other entries of $\boldsymbol{\alpha}$ to 0. Since we can always make this choice given the affine independence assumption, the feasibility claim follows.

Proof of (i). A negative optimal min-weight metric value $\ell^*(\boldsymbol{q}) < 0$ implies the feasibility program (2.3) has no solution. By construction of the feasibility program, this equivalently implies that $\mathbf{0} \notin \text{conv}(\mathcal{W})$.

Proof of (ii). A positive optimal min-weight metric value $\ell^*(\boldsymbol{q}) > 0$ is achieved if and only if there exists a set of convex weights with strictly positive part $\boldsymbol{\alpha} \succ \mathbf{0}$ such that $\mathbf{W}\boldsymbol{\alpha} = \mathbf{0}$. Since we assume $\text{conv}(\mathcal{W})$ has nonempty interior, its relative interior is its interior. Thus, we can invoke the result that the assumption $\mathbf{0} \in \text{int}(\text{conv}(\mathcal{W}))$ holds if and only if $\exists \boldsymbol{\alpha} \succ \mathbf{0}$ such that $\mathbf{W}\boldsymbol{\alpha} = \mathbf{0}$ [58, Exercise 3.1].

Proof of (iii). Follows immediately from (i) and (ii), since $\mathbb{R}^6$ is clearly disjointly partitioned by $\text{conv}(\mathcal{W})^{\text{C}}$, $\text{int}(\text{conv}(\mathcal{W}))$, and $\partial\,\text{conv}(\mathcal{W})$, where $\mathcal{S}^{\text{C}}$ denotes the complement of a set $\mathcal{S}$. □

Remark 2.1. *The number 7 appears in the theorem statement because it is 1 larger than 6, the dimension of the wrench space, and the affine independence property in the proofs of Theorem 2.1 require one element more than the dimension of the ambient space.*

The assumptions of Thereom 2.1 are mild. The existence of 7 affinely independent basis wrenches simply means that these wrenches form the vertices of some non-degenerate simplex in wrench space, i.e., the set of feasible wrenches for a grasp has some nonzero volume (notice that this assumption says nothing about whether the origin is contained in $\text{conv}(\mathcal{W})$, so we are not just vacuously assuming a nontrivial property like force closure). This assumption is only violated in edge cases, such as only using one finger, all fingers contacting at exactly the same point, or the coefficient of friction attaining a value of 0. Moreover, the affine independence condition clearly implies the second assumption that $\text{int}(\text{conv}(\mathcal{W})) \neq \emptyset$, since the hull has nonzero volume, but we state this second assumption explicitly because it clarifies the proof.

Given these assumptions, Thereom 2.1 states that the sign of ℓ^* indicates force closure, justifying its maximization. Heuristically, very negative values of ℓ^* indicate a grasp is far from force closure while very positive values indicate the origin lies well within $\text{conv}(\mathcal{W})$ (see Fig. 2.2). This motivates using ℓ^* as an approximate measure of robustness.

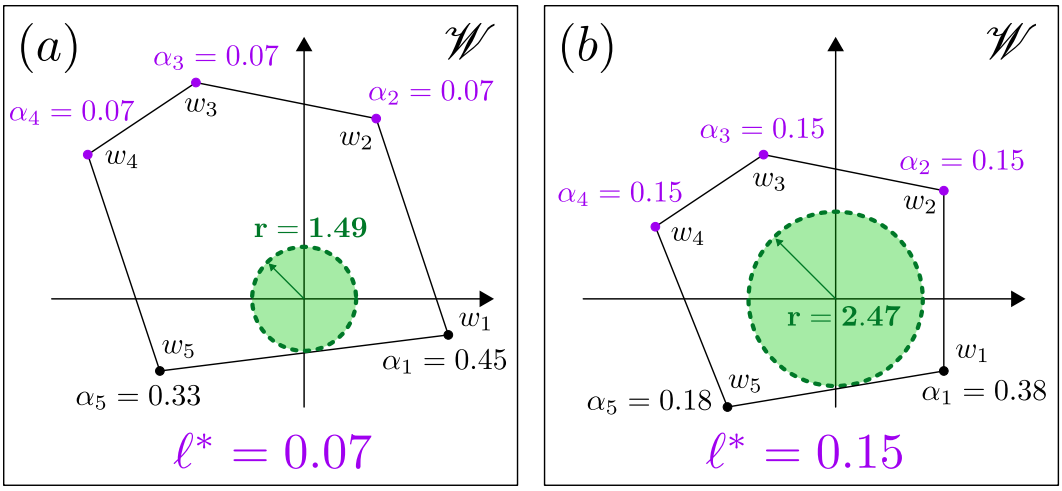


Figure 2.2: Toy examples of the min-weight metric. The weights α_i are associated with wrenches $w_i \in \mathcal{W} \subseteq \mathscr{W}$. The minimum weights are highlighted purple. The largest inscribed ball about the origin in each convex hull is highlighted green. We empirically observe that the minimum weight ℓ^* is strongly correlated with the radius of this ball, i.e., the Ferrari-Canny metric ϵ.

We emphasize that Theorem 2.1 is fundamentally a statement about the *sign* of ℓ^*, making the binary concept of force closure continuous. It does not by itself, for example, assert that a value of $\ell^* = 0.2$ corresponds to a "better" grasp than one with $\ell^* = 0.1$ in any precise sense with respect to some grasp robustness metric. However, empirically, we observe that ℓ^* and the Ferrari-Canny metric ϵ are strongly correlated: maximizing ℓ^* seems to improve a lower bound on ϵ (see Fig. 2.3). In particular, some positive multiple of ℓ^* appears to uniformly lower bound ϵ across force closure grasps. Proving this rigorously turns out to be nontrivial: in simple abstract cases, one can construct examples where the largest possible value of ℓ^* corresponds to arbitrarily small inscribed radii. We return to this observation in Chapter 3, where we develop the convex-analytic tools to make this claim mathematically precise.

Since $\ell^* \leq 1/n_w$, the *normalized min-weight metric* $\bar{\ell}^* = n_w \ell^*$ is well-defined with $\bar{\ell}^* \in [0, 1]$ for force closure grasps, allowing us to specify the constraint $\bar{\ell}^* \geq k_\ell$ for some desired lower bound $k_\ell \in [0, 1]$ on grasp robustness. In experiments, we use $k_\ell = 0.3$.

Finally, as in many classical metrics, ℓ^* is not invariant to the choice of object frame [37, Ch. 13.5]. While methods exist that address this issue (see [49]), we do not explore them here.

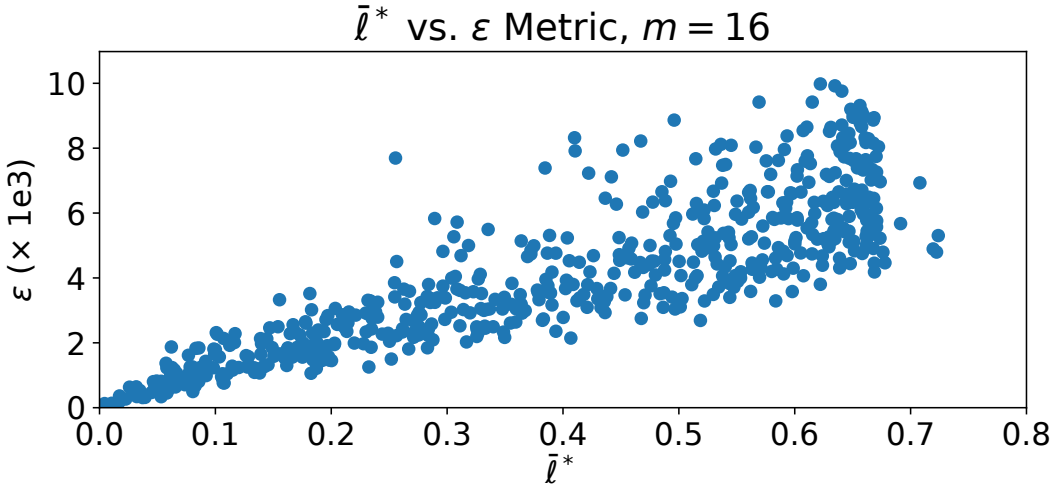


Figure 2.3: Normalized min-weight metric $\bar{\ell}^*$ vs. Ferrari-Canny metric ϵ for 600 total feasible force closure grasps on 6 objects from the YCB dataset [41] generated using FRoGGeR for $n_w = 16$ basis wrenches. We report a Pearson correlation of 0.865 and observe a clear improvement in the worst-case ϵ value as $\bar{\ell}^*$ increases. Because $\bar{\ell}^*$ is bounded above, we typically observe a ceiling in its value (here, ≈ 0.7), which degrades the correlation as grasp robustness increases. However, ℓ^* is very informative in the regime where grasps are barely force closure (bottom left), i.e., where they are least robust.

2.3.1 Computing $\nabla \ell^*(\boldsymbol{q})$ with Differentiable Optimization

The key insight enabling gradient-based grasp synthesis is that the solution to a convex optimization program is, under mild conditions, an implicit function of the problem data. Concretely, consider a convex program parameterized by $\boldsymbol{\theta}$, and let $\boldsymbol{z}^*(\boldsymbol{\theta})$ collect the optimal primal and dual variables. At optimality, the KKT conditions define a residual $\boldsymbol{F}(\boldsymbol{z}^*, \boldsymbol{\theta}) = \boldsymbol{0}$. If certain regularity conditions hold (e.g., strict complementarity and linear independence of active constraints), the implicit function theorem guarantees that $\boldsymbol{z}^*$ is locally differentiable with respect to $\boldsymbol{\theta}$, with Jacobian

$$\frac{d\boldsymbol{z}^*}{d\boldsymbol{\theta}} = -\left[\frac{\partial \boldsymbol{F}}{\partial \boldsymbol{z}}\right]^{-1} \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{\theta}}.$$

This observation, formalized for quadratic programs by Amos and Kolter [57] and extended to general convex programs by Agrawal et al. [59], allows us to differentiate *through* an optimization program by implicitly differentiating its KKT system.

We apply this technique to compute $\nabla \ell^*(\boldsymbol{q})$ where it is defined and exploit the resulting structure for fast evaluation.

For brevity, let $\boldsymbol{x} = (\boldsymbol{\alpha}, \ell)$ and express (2.4) as

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \boldsymbol{c}^\top \boldsymbol{x} && \text{(2.5a)} \\ \text{subject to} \quad & \mathbf{A}_{\text{eq}} \boldsymbol{x} = \boldsymbol{b}_{\text{eq}} && \text{(2.5b)} \\ & \mathbf{A}_{\text{in}} \boldsymbol{x} \preceq \mathbf{0}. && \text{(2.5c)} \end{aligned}$$

Let $\boldsymbol{\lambda}$ and $\boldsymbol{\nu}$ denote the Lagrange multipliers associated with the inequality and equality constraints, respectively. Following [57], we write the stationarity, complementary slackness, and primal feasibility conditions for (2.4):

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \mathbf{H}_3 \end{bmatrix} := \begin{bmatrix} \boldsymbol{c} + \mathbf{A}_{\text{in}}^\top \boldsymbol{\lambda} + \mathbf{A}_{\text{eq}}^\top \boldsymbol{\nu} \\ \boldsymbol{\lambda} \odot (\mathbf{A}_{\text{in}} \boldsymbol{x}) \\ \mathbf{A}_{\text{eq}} \boldsymbol{x} - \boldsymbol{b}_{\text{eq}} \end{bmatrix}, \tag{2.6}$$

where $\odot$ denotes the Hadamard product. Solving $\mathbf{H} = \mathbf{0}$ is necessary and sufficient for the optimal solution of any LP [2, Sec. 5.5.3]. Let $D_{(\cdot)} f$ and $\partial_{(\cdot)} f$ denote the total and partial Jacobians of a vector-valued function f with respect to variables $(\cdot)$, respectively. Since $\mathbf{H} = \mathbf{0}$ at the optimal solution, by implicit differentiation,

$$\begin{aligned} & D_{\boldsymbol{q}} \mathbf{H}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*, \boldsymbol{q}) = \mathbf{0} \\ & \implies \partial_{(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})} \mathbf{H} \cdot D_{\boldsymbol{q}} (\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)(\boldsymbol{q}) + \partial_{\boldsymbol{q}} \mathbf{H} = \mathbf{0}. \end{aligned} \tag{2.7}$$

The partial derivative $\partial_{\boldsymbol{q}} \mathbf{H}$ can be computed explicitly or with autodifferentiation through the wrench matrix $\mathbf{W}(\boldsymbol{q})$, which is derived from the grasp map $\mathbf{G}(\boldsymbol{q})$. In particular, $\nabla \ell^*(\boldsymbol{q})^\top$ is given by the last row of $D_{\boldsymbol{q}} \boldsymbol{x}^*(\boldsymbol{q})$, which we can compute via the following result.

Proposition 2.1 (Gradient Exploit)**.** *Let* $(\cdot)^\dagger$ *denote the Moore-Penrose pseudoinverse. Then,*

$$D_{\boldsymbol{q}} \boldsymbol{x}^*(\boldsymbol{q}) = \begin{bmatrix} D_{\boldsymbol{q}} \boldsymbol{\alpha}^*(\boldsymbol{q}) \\ \nabla \ell^*(\boldsymbol{q})^\top \end{bmatrix} = - \begin{bmatrix} \operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}} \\ \mathbf{A}_{\text{eq}} \end{bmatrix}^\dagger \begin{bmatrix} \mathbf{0} \\ \partial_{\boldsymbol{q}} \mathbf{H}_3 \end{bmatrix}. \tag{2.8}$$

Proof. We proceed by differentiating each block row of the KKT system (2.6) with respect to $\boldsymbol{q}$.

Step 1: Compute $\partial_{(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})} \mathbf{H}$. Differentiating each $\mathbf{H}_i$ with respect to the primal-dual variables $(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$, we obtain

$$\partial_{(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})} \mathbf{H} = \begin{bmatrix} \partial_{\boldsymbol{x}} \mathbf{H}_1 & \partial_{\boldsymbol{\lambda}} \mathbf{H}_1 & \partial_{\boldsymbol{\nu}} \mathbf{H}_1 \\ \partial_{\boldsymbol{x}} \mathbf{H}_2 & \partial_{\boldsymbol{\lambda}} \mathbf{H}_2 & \partial_{\boldsymbol{\nu}} \mathbf{H}_2 \\ \partial_{\boldsymbol{x}} \mathbf{H}_3 & \partial_{\boldsymbol{\lambda}} \mathbf{H}_3 & \partial_{\boldsymbol{\nu}} \mathbf{H}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{A}_{\text{in}}^\top & \mathbf{A}_{\text{eq}}^\top \\ \operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}} & \operatorname{diag}(\mathbf{A}_{\text{in}} \boldsymbol{x}^*) & \mathbf{0} \\ \mathbf{A}_{\text{eq}} & \mathbf{0} & \mathbf{0} \end{bmatrix},$$

where the entries follow from the definitions:

- $\mathbf{H}_1 = \boldsymbol{c} + \mathbf{A}_{\text{in}}^\top \boldsymbol{\lambda} + \mathbf{A}_{\text{eq}}^\top \boldsymbol{\nu}$ does not depend on $\boldsymbol{x}$;
- $\mathbf{H}_2 = \boldsymbol{\lambda} \odot (\mathbf{A}_{\text{in}} \boldsymbol{x})$ has $\partial_{\boldsymbol{x}} \mathbf{H}_2 = \operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}}$ and $\partial_{\boldsymbol{\lambda}} \mathbf{H}_2 = \operatorname{diag}(\mathbf{A}_{\text{in}} \boldsymbol{x}^*)$;
- $\mathbf{H}_3 = \mathbf{A}_{\text{eq}} \boldsymbol{x} - \boldsymbol{b}_{\text{eq}}$ gives $\partial_{\boldsymbol{x}} \mathbf{H}_3 = \mathbf{A}_{\text{eq}}$.

Step 2: Write the block-row IFT system. The implicit differentiation equation (2.7) expands to three block rows:

$$\begin{aligned} &\text{Row 1 (stationarity):} && \mathbf{A}_{\text{in}}^\top D_{\boldsymbol{q}} \boldsymbol{\lambda}^* + \mathbf{A}_{\text{eq}}^\top D_{\boldsymbol{q}} \boldsymbol{\nu}^* = -\partial_{\boldsymbol{q}} \mathbf{H}_1, \\ &\text{Row 2 (comp. slack.):} && \operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^* + \operatorname{diag}(\mathbf{A}_{\text{in}} \boldsymbol{x}^*) D_{\boldsymbol{q}} \boldsymbol{\lambda}^* = -\partial_{\boldsymbol{q}} \mathbf{H}_2, \\ &\text{Row 3 (primal feas.):} && \mathbf{A}_{\text{eq}} D_{\boldsymbol{q}} \boldsymbol{x}^* = -\partial_{\boldsymbol{q}} \mathbf{H}_3. \end{aligned} \tag{2.9}$$

Step 3: Simplify Row 2 via complementary slackness. Since $\mathbf{A}_{\text{in}}$ does not depend on $\boldsymbol{q}$, we have $\partial_{\boldsymbol{q}} \mathbf{H}_2 = \mathbf{0}$. Furthermore, by complementary slackness, each constraint index i satisfies either $\lambda_i^* > 0$ with $(\mathbf{A}_{\text{in}} \boldsymbol{x}^*)_i = 0$ (active), or $\lambda_i^* = 0$ with $(\mathbf{A}_{\text{in}} \boldsymbol{x}^*)_i < 0$ (inactive). We examine Row 2 component-wise:

$$\lambda_i^* \, (\mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^*)_i + (\mathbf{A}_{\text{in}} \boldsymbol{x}^*)_i \, (D_{\boldsymbol{q}} \boldsymbol{\lambda}^*)_i = 0, \qquad \forall \, i.$$

For active constraints ($\lambda_i^* > 0$, $(\mathbf{A}_{\text{in}} \boldsymbol{x}^*)_i = 0$), the second term vanishes, so $\lambda_i^* (\mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^*)_i = 0$, which implies $(\mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^*)_i = 0$. Multiplying by λ_i^* preserves the zero. For inactive constraints ($\lambda_i^* = 0$), the first term vanishes directly. In both cases, $\lambda_i^* (\mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^*)_i = 0$, so

$$\operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}} D_{\boldsymbol{q}} \boldsymbol{x}^* = \mathbf{0}. \tag{2.10}$$

Step 4: Assemble and solve. Stacking (2.10) and Row 3, we obtain

$$\underbrace{\begin{bmatrix} \operatorname{diag}(\boldsymbol{\lambda}^*) \mathbf{A}_{\text{in}} \\ \mathbf{A}_{\text{eq}} \end{bmatrix}}_{\boldsymbol{\Omega}} D_{\boldsymbol{q}} \boldsymbol{x}^* = \begin{bmatrix} \mathbf{0} \\ -\partial_{\boldsymbol{q}} \mathbf{H}_3 \end{bmatrix}.$$

Applying the Moore-Penrose pseudoinverse of $\boldsymbol{\Omega}$ to both sides yields

$$D_{\boldsymbol{q}} \boldsymbol{x}^*(\boldsymbol{q}) = -\boldsymbol{\Omega}^\dagger \begin{bmatrix} \mathbf{0} \\ \partial_{\boldsymbol{q}} \mathbf{H}_3 \end{bmatrix},$$

which is the desired result. Note that Row 1 determines the dual sensitivities $D_{\boldsymbol{q}} \boldsymbol{\lambda}^*$ and $D_{\boldsymbol{q}} \boldsymbol{\nu}^*$ but is not needed for computing $D_{\boldsymbol{q}} \boldsymbol{x}^*$. □

Proposition 2.1 allows efficient computation of $\nabla \ell^*(\boldsymbol{q})$, especially when n_w is "small" ($\lesssim 40$). For example, when $n_w = 16$ (e.g., using a square pyramidal approximation for a 4-fingered hand), we find that $\ell^*(\boldsymbol{q})$ and $\nabla \ell^*(\boldsymbol{q})$ can be computed together in about 3.9ms with `cvxpylayers` [59], whereas using our exploit, they are computed in 0.18ms, a 22× speedup.

We remark that if ℓ^* is locally Lipschitz with respect to the constraint matrix parameters $\mathbf{A}_{\text{eq}}$ and $\mathbf{A}_{\text{in}}$, it is differentiable everywhere but a set of measure 0 by a theorem of Rademacher (see [60] for discussion). The gradient is defined when (2.4) has unique primal/dual optima, and in this case, the inverse of $\partial_{(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\nu})}\mathbf{H}$ exists, so $\nabla \ell^*(\boldsymbol{q})$ is computable. The Lipschitz condition can always be satisfied by removing degenerate constraints, so we assume it.

2.4 Fast Robust Grasp Generation

2.4.1 The Grasp Refinement Problem

FRoGGeR *refines* a candidate grasp configuration $\boldsymbol{q}_0$ into a locally optimal one $\boldsymbol{q}^*$ by solving the following nonlinear bilevel optimization program (recalling that $\bar{\ell}^* = n_w \ell^*$):

$$\begin{aligned} \underset{\boldsymbol{q}}{\text{maximize}} \quad & \ell^*(\boldsymbol{q}) \\ \text{subject to} \quad & \boldsymbol{q}_{\min} \preceq \boldsymbol{q} \preceq \boldsymbol{q}_{\max} \\ & \bar{\ell}^*(\boldsymbol{q}) \geq k_\ell \\ & \underset{i}{\text{FK}}(\boldsymbol{q}) \in \partial O, \ i = 1, \ldots, n_f \\ & \text{No (non-finger/object) collision.} \end{aligned} \tag{FRoGGeR}$$

To enforce joint limits, we constrain the robot configuration $\boldsymbol{q}$ to lie between minimum and maximum values $\boldsymbol{q}_{\min}$ and $\boldsymbol{q}_{\max}$. Further, we enforce that the fingertips lie on the object surface ∂O and that no rigid bodies are interpenetrating.

To express these constraints mathematically, we first parameterize ∂O as the 0-level set of a twice-differentiable signed distance field (SDF) $s : \mathbb{R}^3 \to \mathbb{R}$, which reports the distance of query points $\boldsymbol{x} \in \mathbb{R}^3$ to ∂O, with $s(\boldsymbol{x}) < 0$ for all points in O:

$$s(\boldsymbol{x}) = \begin{cases} -\operatorname{dist}(\boldsymbol{x}, \partial O), & \boldsymbol{x} \in O, \\ +\operatorname{dist}(\boldsymbol{x}, \partial O), & \boldsymbol{x} \notin O. \end{cases}$$

Second, we consider every possible pair of geometries we would like to prevent from colliding and parameterize the collision status using the differentiable constraints

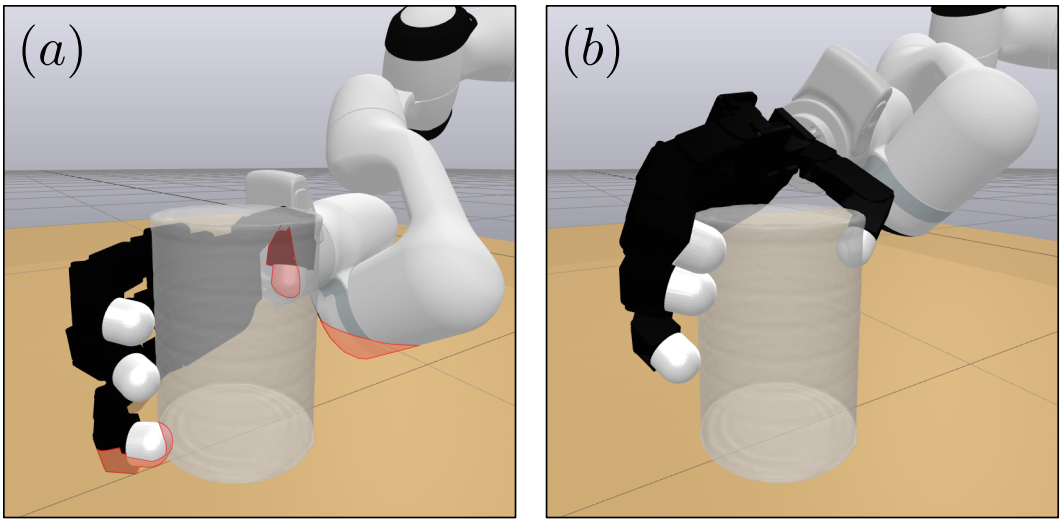


Figure 2.4: FRoGGeR quickly refines infeasible dexterous grasps into kinematically-feasible, collision-free ones using gradient-based nonlinear optimization. We leverage the min-weight metric presented in Sec. 2.3 to maximize refined grasp robustness. **(a)** A grasp $\boldsymbol{q}_0$ sampled using a heuristic with collisions between the object, robot, and table (highlighted red). **(b)** The feasible, robust refined grasp $\boldsymbol{q}^*$. In this example, the entire process—from sampling to refinement—took just 0.816 seconds. The median computation time over all experiments was 0.834 seconds.

$\sigma\left(o_A^{(j)}, o_B^{(j)}; \boldsymbol{q}\right),\ j = 1, \ldots, n_p$, where n_p is the number of collision pairs and σ is an SDF between two geometries, at least one of whose state depends smoothly on $\boldsymbol{q}$. For pair j, we enforce a minimum safety margin of $d_j > 0$ unless it is a finger-object pair, for which we specify $d_j < 0$ to allow a small amount of interpenetration. Thus, we can express (FRoGGeR) formally as

$$\begin{aligned}
\text{maximize} \quad & \ell^*(\boldsymbol{q}) && \text{(2.11a)}\\
\text{subject to} \quad & \boldsymbol{q}_{\min} \preceq \boldsymbol{q} \preceq \boldsymbol{q}_{\max} && \text{(2.11b)}\\
& \bar{\ell}^*(\boldsymbol{q}) \geq k_\ell && \text{(2.11c)}\\
& s(\underset{i}{\mathrm{FK}}(\boldsymbol{q})) = 0,\ i = 1, \ldots, n_f && \text{(2.11d)}\\
& \sigma\left(o_A^{(j)}, o_B^{(j)}; \boldsymbol{q}\right) \geq d_j,\ j = 1, \ldots, n_p. && \text{(2.11e)}
\end{aligned}$$

2.4.2 Gradients of the Constraint Functions

To use gradient-based methods, we must compute the gradients of the objective and each constraint. The gradient of constraint (2.11d) is immediately given by $\mathbf{J}_i^\top(\boldsymbol{q})\nabla s(\mathrm{FK}_i(\boldsymbol{q}))$ for $i = 1, \ldots, n_f$. For constraint (2.11e), we compute for each pair of geometries (o_A, o_B) the *witness points* $(\boldsymbol{p}_A, \boldsymbol{p}_B)$: the two points of furthest penetration if the pair is colliding, or the two closest points otherwise. Let $I_c = 1$

indicate collision of a pair and $I_c = 0$ otherwise. Then,

$$\begin{aligned} \sigma\left(o_A, o_B; \boldsymbol{q}\right) &= (-1)^{I_c+1} \left\|\boldsymbol{p}_A - \boldsymbol{p}_B\right\| \\ \implies \nabla_{\boldsymbol{q}} \sigma\left(o_A, o_B; \boldsymbol{q}\right) &= (-1)^{I_c} \left(\mathbf{J}_B^\top - \mathbf{J}_A^\top\right) \hat{\boldsymbol{n}}_{AB}, \end{aligned}$$

where $\mathbf{J}_A$ and $\mathbf{J}_B$ are the Jacobians at witness points A and B and $\hat{\boldsymbol{n}}_{AB}$ is the unit vector from $\boldsymbol{p}_A$ to $\boldsymbol{p}_B$.

We use `Drake` [61] to compute witness points for all geometry pairs in a scene. To speed up computation, we represent nonconvex bodies as a union of convex polytopes computed using V-HACD [62]. To reduce the number of checked pairs, `Drake` culls distant pairs using a broadphase algorithm, and we set the associated gradients to $\mathbf{0}$.

2.4.3 Candidate Grasp Generation

To obtain initial configurations $\boldsymbol{q}_0$, we use a heuristic sampler that noisily aligns the palm with the axes of the object's oriented bounding box with probabilities proportional to the box side lengths, motivated by observations of preferred human grasps [42]. The width of the fingertips is set by the appropriate axis of the bounding box, and the palm is placed roughly 4cm from the object. To obtain the configuration variables, we solve an inverse kinematics (IK) problem, but do not enforce collision constraints or force the fingertips to lie on the object surface. Thus, we only consider infeasible candidate grasps. To solve (2.11), we use the `NLopt` [63] implementation of SLSQP [64].

2.4.4 Object Surface Representations

The SDF representation is convenient for reasoning about both collision and geometric properties, since the outward-pointing surface normal at a point $\boldsymbol{p} \in \partial O$ is given by $\nabla s(\boldsymbol{p})$ and principal curvatures can be computed from the Hessian $\nabla^2 s(\boldsymbol{p})$. However, supplying the true SDF s is non-trivial. Some approaches learn this representation [30], while most use the object's mesh or a point cloud [44, 54]. Since these representations are all widely used, it is desirable for grasp synthesis methods to be compatible with any of them.

In the case of a learned or analytical SDF, computing the requisite gradients can be done via autodifferentiation. If provided a dense enough point cloud, the Poisson surface reconstruction algorithm can return a watertight mesh [65]. Therefore, we focus on the case of meshes, which is both the most common and the most challenging representation.

We assume that there exists a true smooth SDF s and denote the approximation computed with the object mesh as $\tilde{s}$. Given a closest point $\boldsymbol{p}'$ to $\boldsymbol{p}$, the gradient is

$$\nabla\tilde{s}(\boldsymbol{p}) = \text{sign}(\tilde{s}(\boldsymbol{p}))\frac{\boldsymbol{p}-\boldsymbol{p}'}{\|\boldsymbol{p}-\boldsymbol{p}'\|}, \tag{2.12}$$

where when $\tilde{s}(\boldsymbol{p}) = 0$, $\nabla\tilde{s}(\boldsymbol{p})$ is the mesh normal at $\boldsymbol{p}'$.

To compute whether a grasp is (robustly) force closure, we must compute the grasp map $\mathbf{G}(\boldsymbol{q})$, which depends on the contact frames associated with fingertip positions $\boldsymbol{p} \in \mathbb{R}^3$ [32]. The normal component of each contact frame is the inward-pointing surface normal, i.e., $\hat{\boldsymbol{n}}(\boldsymbol{p}) = -\nabla s(\boldsymbol{p})$. Therefore, to differentiate any objective or constraint that depends on measures of force closure, we require $\nabla^2 s(\boldsymbol{p})$.

However, when the object is parameterized as a mesh, the surface is piecewise flat, so $\nabla^2\tilde{s}(\boldsymbol{p}) \equiv \mathbf{0}$ wherever it is defined even if $\nabla^2 s(\boldsymbol{p}) \neq \mathbf{0}$. We propose a coarse but efficient finite-difference approximation. Let $\boldsymbol{p}_1 = \boldsymbol{p} \in \mathbb{R}^3$ be an arbitrary query point. Fix a small constant $\delta > 0$, randomly select 2 unit vectors $\boldsymbol{d}_2, \boldsymbol{d}_3 \in \mathbb{R}^3$, and define $\boldsymbol{p}_k = \boldsymbol{p}_1 + \delta\boldsymbol{d}_k$ for $k = 2, 3$. By (2.12), we have $\nabla\tilde{s}(\boldsymbol{p}_k)$ for all $k = 1, 2, 3$. Fix $\boldsymbol{d}_1 = \nabla s(\boldsymbol{p}_1)$.

Using our perturbation directions and finite-difference approximations of the directional derivatives, $\boldsymbol{y}_k = (\nabla s(\boldsymbol{p}_1 + \delta\boldsymbol{d}_k) - \nabla s(\boldsymbol{p}_1))/\delta$, we solve

$$\boldsymbol{\Sigma}\begin{bmatrix}\boldsymbol{d}_1 & \boldsymbol{d}_2 & \boldsymbol{d}_3\end{bmatrix} = \begin{bmatrix}\boldsymbol{y}_1 & \boldsymbol{y}_2 & \boldsymbol{y}_3\end{bmatrix}$$

for $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, noting that $\boldsymbol{y}_1 = \mathbf{0}$ since $\nabla s(\boldsymbol{p})$ is unchanging close to the surface along $\nabla s(\boldsymbol{p})$ and $\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3$ are linearly independent with probability 1. We then symmetrize: $\nabla^2\tilde{s}(\boldsymbol{p}) = (\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^\top)/2$. Empirically, setting δ to roughly 10 times the average mesh edge length yields accurate enough gradients for grasp refinement.

2.5 Experimental Results

We evaluate FRoGGeR on a combination of simulated and real-world grasping experiments. In simulation, we compare against a competitive baseline to validate the benefits of robustness-aware grasp synthesis. For the real-world experiments, we deploy FRoGGeR on hardware using a Franka Research 3 with an Allegro hand, demonstrating that the improvements observed in simulation transfer to physical grasping.

We use the 7-DOF Franka Research 3 and 16-DOF 4-fingered Allegro hand. The system is mounted on a flat tabletop and each target object is spawned with a

fixed initial pose over all trials for repeatability, since the arm/hand configuration is allowed to vary arbitrarily.

2.5.1 Simulation Results

We evaluate the robustness of FRoGGeR by executing 20 "shaky pickups" per object in simulation using `Drake`. To do so, we generate a pick trajectory where the end-effector is lifted 10cm in 1s and then held for 1.5s. We add sinusoidal perturbations to this trajectory with amplitude 3mm and varying frequency in all spatial axes 0.25s after the pick begins until the end of the simulation. A pick fails if either (1) the object rotates by more than 30° or if the object deviates from the pick trajectory by more than 7.5cm at any point; or (2) the total grasp synthesis time exceeds 1 minute.

Initial configurations $\boldsymbol{q}_0$ are generated using the heuristic sampler described in Sec. 2.4.3. Our choice to use this coarse sampling heuristic instead of a more performant method is intentional, as our goal is to evaluate FRoGGeR's robustness to the quality of the initial guess. We control for the resulting decrease in performance by evaluating the relative performance of our method versus the baseline under identical conditions. We chose to synthesize 4-finger grasps to capture the full dexterity of the Allegro hand.

We compare FRoGGeR to the baseline of Wu *et al.* [55], which only enforces force closure without optimizing for robustness. The only difference between our method and the baseline is that in (2.11), FRoGGeR maximizes $\ell^*(\boldsymbol{q})$ with constraint $\bar{\ell}^*(\boldsymbol{q}) \geq 0.3$, while the baseline has no objective and (2.11c) is replaced with a bilevel force closure equality constraint. Otherwise, the same IK routines, sampler, collision geometries, and controller were used.

2.5.1.1 Object Data Processing

We only present results on objects parameterized as meshes. When supplied with analytical SDFs or well-trained deep SDFs, our method was generally both fast and performant. We use meshes to demonstrate our approach on non-smooth object representations and to validate the usefulness of the Hessian approximation from Sec. 2.4.4.

The objects used in our experiments are from a pruned subset of the YCB dataset [41]. First, we removed all objects that were too large, small, or thin to reasonably grasp with 4 fingers from a flat table, as well as deformable or multibody objects. Second, since the YCB meshes are not watertight, we attempted to reprocess them

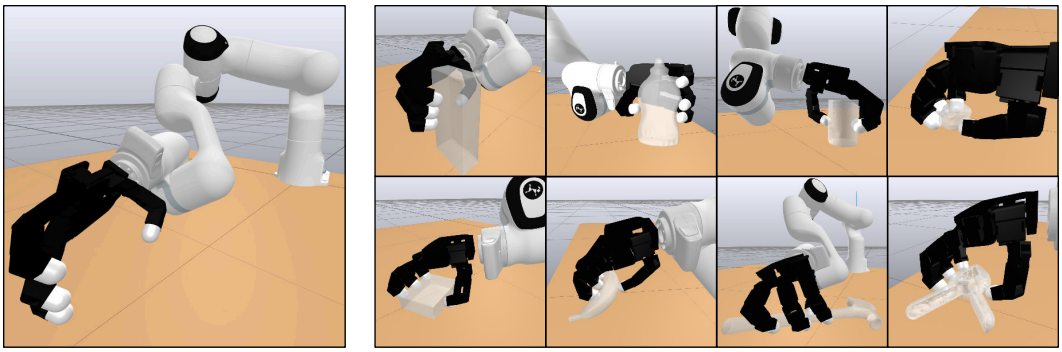


Figure 2.5: **Left.** A representative pre-shaped grasp $\boldsymbol{q}_0$ from our heuristic sampler. **Right.** Example refined grasps $\boldsymbol{q}^*$ from our experiments. We can produce robust grasps for highly varying objects. Top row: sugar box, mustard bottle, soup can, strawberry. Bottom row: pudding box, banana, hammer, large clamp.

by densely sampling points on each mesh and running Poisson reconstruction. Of these, we removed objects for which we could not produce watertight meshes due to poor data quality (e.g., from transparency, thin walls, etc.). We note that FRoGGeR works even on non-watertight meshes of adequate quality, but we take this step to eliminate the effect of poor meshes on our results. In total, we test on 43 objects belonging to three categories: *spheroids*, like fruits and balls; *boxes/cylinders*, like food containers, cans, or large cups; and *adversarial* objects with irregular geometry, like tools or very flat/long objects.

For simplicity, we set the friction coefficient to be $\mu = 0.7$ for all objects, which is reasonable for the rubbery Allegro fingertips on mostly plastic objects. We also assumed a uniform density of 150 kg/m^3 and computed masses using the volume of the processed meshes. The optimizer assumed a more conservative friction coefficient of $\mu = 0.5$ and the controller was provided the mass.

2.5.1.2 Results and Discussion

We report values related to the quality of the grasp (pick success, ϵ metric value, and $\bar{\ell}^*$) as well as values regarding the runtime of each method in Table 2.1. We find that overall, FRoGGeR outperforms the baseline in terms of pick success by 20 percentage points and yields ϵ values that are roughly twice as high.

We also found that overall, FRoGGeR was ~16× faster at generating grasps than the baseline, a result of ~3× faster single solve times and ~15× fewer number of solves required to produce a feasible grasp. This is consistent with the observation by Wu *et al.* that their method struggles to converge to feasible solutions when $\boldsymbol{q}_0$

Category	Method	% conv. ↑	% pick succ. ↑	ϵ ($\times 10^3$) ↑	norm. ℓ^* ↑	t/solve (s) ↓	# solves ↓	total t (s) ↓
sphere (240 total)	baseline	12.9% (31/240)	83.9% (26/31)	2.7 (1.6, 3.6)	0.32 (0.19, 0.39)	0.67 (0.44, 1.00)	68 (61, 73)	27.2 (13.5, 36.2)
	FRoGGeR	**97.9%** (235/240)	**95.3%** (224/235)	**4.9** (4.2, 5.5)	**0.67** (0.55, 0.72)	**0.31** (0.18, 0.49)	**2** (1, 4)	**0.57** (0.30, 1.2)
box/cyl (320 total)	baseline	52.8% (169/320)	68.6% (116/169)	2.2 (0.1, 3.8)	0.20 (0.09, 0.36)	0.79 (0.41, 1.29)	42 (13, 53)	13.4 (5.5, 31.1)
	FRoGGeR	**100%** (320/320)	**81.6%** (261/320)	**4.8** (3.8, 5.9)	**0.58** (0.44, 0.65)	**0.15** (0.09, 0.24)	**3** (2, 5)	**0.87** (0.44, 1.6)
adversarial (300 total)	baseline	61.7% (185/300)	43.8% (81/185)	1.8 (0.6, 2.9)	0.18 (0.08, 0.29)	0.79 (0.47, 1.25)	29 (10, 55)	12.7 (5.8, 24.5)
	FRoGGeR	**100%** (300/300)	**63.0%** (189/300)	**4.3** (3.5, 5.1)	**0.53** (0.42, 0.63)	**0.18** (0.11, 0.30)	**3** (2, 9)	**1.0** (0.49, 3.3)
overall (860 total)	baseline	44.8% (385/860)	58.0% (223/385)	2.0 (0.8, 3.4)	0.19 (0.09, 0.32)	0.73 (0.44, 1.15)	50 (17, 63)	13.8 (5.8, 29.5)
	FRoGGeR	**99.4%** (855/860)	**78.8%** (674/855)	**4.6** (3.7, 5.5)	**0.58** (0.45, 0.66)	**0.21** (0.12, 0.36)	**3** (1, 6)	**0.83** (0.39, 1.9)

Table 2.1: **Simulation results.** For each of 43 objects, we try to generate 20 feasible grasps and evaluate them for both methods. In each cell, the top entry is the baseline and the bottom is FRoGGeR. The better result is bolded. ↑ and ↓ denote whether higher or lower values are better. Statistics are reported as the median and interquartile range over converged runs and all times are reported in seconds. A run *converges* if it yields a feasible grasp in under 1 minute. We find that our method almost always quickly converges to a feasible grasp while the baseline succeeds under 45% of the time. Further, over converged grasps, our method outperforms the baseline on a "shaky pickup" task in every category and by 20 percentage points overall. Our method's superior robustness is reflected in its ϵ values, which are about twice as high as the baseline. FRoGGeR's fast convergence is a result of both solving each optimization problem faster as well as requiring significantly fewer attempts before finding a feasible grasp. Note that the total time reported also includes time spent solving IK problems when sampling initial configurations $\boldsymbol{q}_0$ (unreported, since the IK problem solved is the same for both methods).

is infeasible, which requires an expensive IK pre-solve [55]. In contrast, FRoGGeR retains superior speed and feasibility rate even with a coarse IK procedure, which suggests that our formulation is also robust to poor candidate grasps. In particular, only 5 runs (all on one object) timed out using our method, whereas over half of the runs timed out for the baseline.

One explanation for this gap is that the baseline force closure equality constraint's gradient vanishes at force closure, which yields a constraint geometry that is difficult to satisfy. Since we do not demand that $\boldsymbol{q}_0$ satisfies (2.11d), we observed that the optimization often terminated unsuccessfully satisfying only one of the equality constraints. In contrast, our constraint (2.11c) has non-zero gradients even in force closure, which we conjecture is better-posed numerically, and in particular, allows FRoGGeR to converge for a larger set of candidate grasps $\boldsymbol{q}_0$ than the baseline.

We remark that our reported baseline pick success values are significantly lower than those reported by Wu *et al.* [55], which we attribute to adding shaking to the pick trajectory. When these perturbations were smaller or nonexistent, we typically observed much higher baseline pick success rates, which supports our hypothesis

that enforcing only non-robust force closure yields grasps that are brittle in practice.

One of the limitations of our method is that the ϵ metric often prefers grasps where the fingertips lie on edges or corners, since these regions are typically farther from the center of the object (roughly where we place the object frame), yielding larger moment arms. Moreover, these regions allow a grasp to direct forces in "non-robust" directions with little change, drawing solutions to them. This yields unstable grasps in practice, since small deviations in the positions of the fingertips produce large changes in the contact conditions, which commonly occurs in dynamic scenarios.

Edge-seeking behavior was the most common failure mode of both methods, which is reflected by the poor performance on many adversarial objects with less low-curvature area on which to grasp. This behavior was also observed on objects in the box/cyl category, which explains the worse performance compared to spheroids. However, failures often occurred for the baseline even when no fingers were placed on edges.

Finally, we find that the overall performance of both FRoGGeR and the baseline was highly sensitive to the sampled initial conditions. For instance, if the initial width of the fingertips was not guided by object bounding boxes, both methods suffered in terms of runtime and grasp quality, as enforcing surface constraints became harder. This motivates the use of data-driven methods in identifying candidate grasps that may be synergistic with the refinement process.

2.5.2 Hardware Results

We now present real-world grasping results. These experiments not only demonstrate the utility of FRoGGeR in the real world, they also provide some evidence regarding the *robustness* of the synthesized grasps relative to the baseline with imperfect object reconstructions.

The hardware setup consists of an Allegro hand mounted on a Franka Research 3 arm. We employed an eye-in-hand setup with a Zed camera to capture images used to train a NeRF [31] of the scene. Only monocular RGB was used. A mesh was extracted from the trained NeRF by running the marching cubes algorithm on a density level set, which we chose empirically to be 17.5 for all objects.

For both FRoGGeR and the baseline of Wu *et al.*, we synthesized 40 grasps on 8 objects (see Fig. 2.6). We used a simple position-based controller: for an optimal solution $\boldsymbol{q}^*$, we defined new target positions for each finger 2cm into the surface, computed the corresponding joint configuration via inverse kinematics, and tracked



Figure 2.6: **Left.** Hardware setup. We synthesized grasps on eight objects for an Allegro hand mounted on a Franka Research 3. From left to right: 3D-printed part, goblet, box, mug, cube, bottle, conditioner, and apple. We employed an eye-in-hand setup with a Zed camera to capture images used to train a NeRF prior to grasping. Only monocular RGB was used. **Right.** Representative FRoGGeR-synthesized grasps.

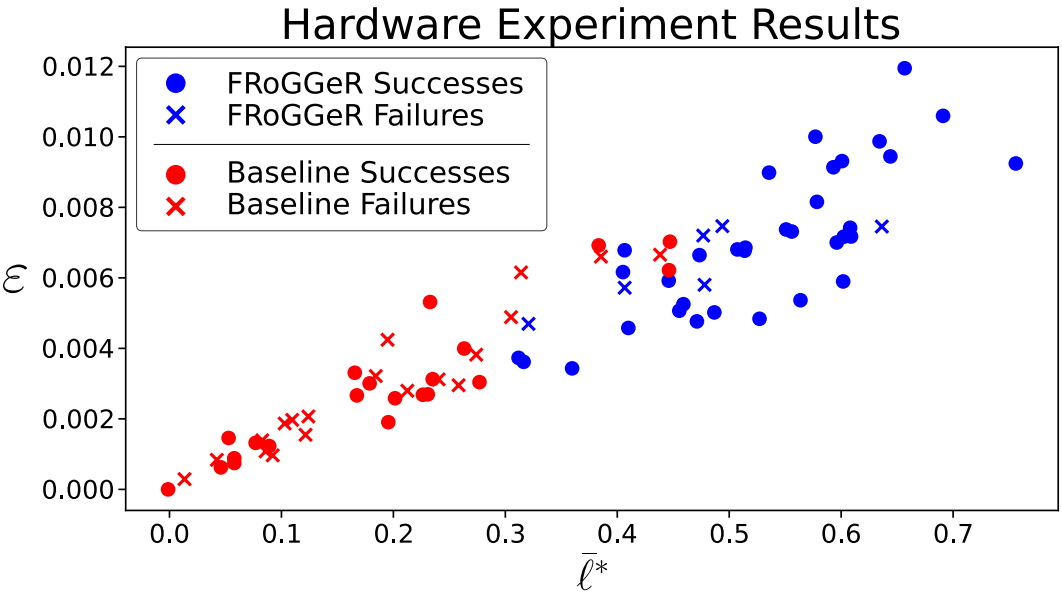


Figure 2.7: Hardware results. FRoGGeR-synthesized grasps are of higher quality and have a much higher success rate (34/40) than the baseline, which only ensures force closure and does not optimize for grasp quality, resulting in many low-quality grasps and a lower success rate (21/40). The relation between $\bar{\ell}^*$ and ϵ supports the observation that ℓ^* lower bounds ϵ up to a positive multiplicative constant.

it with proportional control. After grasping, the object was picked up 20cm and held in place. A grasp was said to fail if it was not successfully picked or if after picking, there was significant rotation. Trials where the object was struck during the pre-grasp motion due to perception errors were excluded (~15% of attempts).

The quantitative results are summarized in Fig. 2.7. FRoGGeR-synthesized grasps succeeded in 34/40 trials (85%), while grasps synthesized using the baseline succeeded in 21/40 (52.5%). Baseline grasps tended to have low values of both ϵ and $\bar{\ell}^*$.

Qualitatively, the dominant failure mode for both methods was attempting grasps on regions where grasp quality was sensitive to shape error. For instance, two failures for FRoGGeR grasps were caused by trying to grab a tea bottle on a sloped area near the top, and many of the baseline failures were caused by trying to grab edges of objects and slipping.

These hardware results validate two key findings: first, that the improvements in robustness observed in simulation transfer to real-world grasping, even in the presence of significant perception noise; and second, that maximizing ℓ^* yields grasps that are empirically more robust to intrinsic uncertainties arising from imperfect object reconstruction via NeRFs.

2.6 Discussion

This chapter demonstrated that constructing the right approximation can unlock algorithmic capabilities that are unavailable for the original problem. The min-weight metric ℓ^*, an LP relaxation of the Ferrari-Canny metric ϵ, is almost everywhere differentiable and cheap to evaluate, enabling fast gradient-based grasp synthesis via bilevel optimization. The resulting system, FRoGGeR, generates feasible, robust precision grasps in seconds while explicitly enforcing kinematic and collision constraints. Experiments showed that ℓ^* is empirically correlated with ϵ and that FRoGGeR yields grasps that are significantly more robust to dynamic perturbations than a baseline that only enforces force closure. FRoGGeR also demonstrated superior solve times and feasibility rates (and in fact, at the time of its publication in 2023, was the first instance of a sub-second grasp synthesis method in the broader multi-finger grasping literature), owing to the favorable numerical properties of the LP formulation compared to the QP-based force closure constraint, whose gradient vanishes at the constraint boundary.

Several limitations point toward future work and the developments in subsequent chapters. FRoGGeR is restricted to precision grasps with a fixed number of contact points, and generalizing to broader grasp types would increase its applicability. More fundamentally, FRoGGeR relies on accurate knowledge of the object's geometry, and the hardware experiments revealed that grasp quality is sensitive to object representation (since this directly affects the estimates of the object surface normals). The next chapter addresses this directly by developing a theory of intrinsic robustness that formally characterizes how much geometric uncertainty a grasp can tolerate, providing theoretical justification for ℓ^* as a robustness proxy and extending it into

a probabilistic framework for uncertainty-aware grasp synthesis.

Chapter 3

A NEW THEORY OF INTRINSIC ROBUSTNESS AND PROBABILISTIC GRASPING

3.1 Intrinsic vs. Extrinsic Robustness

The preceding chapter constructed the min-weight metric ℓ^* as a computationally tractable LP relaxation of the Ferrari-Canny metric ϵ and demonstrated its effectiveness for fast grasp synthesis. A natural follow-up question is: how much is lost in this approximation? This chapter addresses that question by developing a formal theory of *intrinsic robustness* for dexterous grasping, characterizing when and why the min-weight approximation preserves meaningful robustness guarantees.

To motivate the notion of intrinsic robustness, we will consider one specific source of intrinsic uncertainty. Recall that the analytic grasp metrics studied in Chapter 2 assume perfect knowledge of an object's geometry. In practice, this assumption

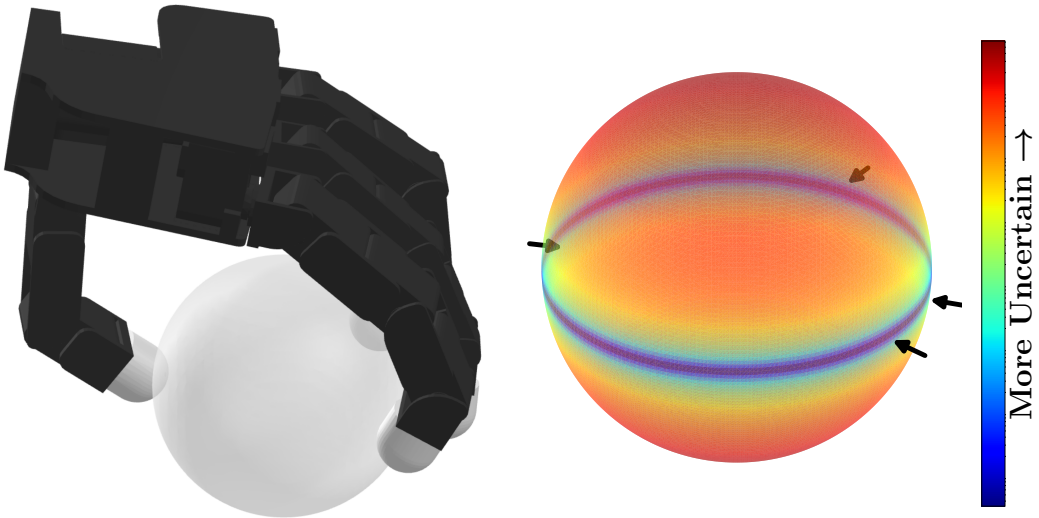


Figure 3.1: This chapter develops an analytic theory of uncertainty-aware grasping, which enables an analysis of the uncertainty tolerance of classical grasp metrics, and the development of new, probabilistic approaches. Above we plot a grasp obtained by optimizing our proposed probabilistic metric, PONG, on a toy example with varying surface normal uncertainty. While there exist many robust grasps for the true geometry, the grasp synthesized using our probabilistic metric places the fingertips in the minimum-uncertainty region.

rarely holds: uncertainties in object pose and geometry affect grasp planning *before* execution. We call such uncertainties *intrinsic*, in contrast to the classical notion of *extrinsic* robustness, which measures resistance to disturbances applied *after* a grasp is established (e.g., an unexpected force on the object [32]). Poor plans due to intrinsic uncertainty create brittle grasps, so intrinsic uncertainties ultimately reduce extrinsic robustness. Despite this, prior to the work in this chapter, mathematical attempts to characterize intrinsic robustness have been limited. The aim of this chapter is to provide some new theory to analyze it.

Our main contribution is a geometric sufficient condition guaranteeing that a nominally force-closure grasp remains force-closure under perturbations to its contact friction cones (Theorem 3.1). This immediately proves that ϵ, originally designed to characterize extrinsic robustness, also quantifies intrinsic robustness (Corollary 3.1). We apply this result in two ways.

First, we prove that the min-weight metric ℓ^* from Chapter 2 uniformly lower bounds (a positive multiple of) ϵ (Theorem 3.2), providing theoretical justification for the LP approximation: maximizing ℓ^* improves worst-case guarantees on the standard ϵ metric. We validated this in Section 2.5 by comparing ℓ^*-synthesized grasps against a competitive baseline on dozens of hardware trials, where they demonstrated superior performance.

Second, we develop a novel *analytic* notion of probabilistic force closure and an uncertainty-aware metric called *probabilistic object normals for grasping* (PONG). Maximizing PONG on objects with uncertain geometry leads to probabilistically robust grasps that concentrate fingertip placements in low-uncertainty regions (e.g., Fig. 3.1). In simulation, we show that ℓ^* performs comparably to PONG on uncertain objects, suggesting that the lower bound from Theorem 3.2 is not conservative in practice.

3.1.1 Related Work

While this chapter presents an analytic treatment of intrinsic uncertainty, prior work has studied it from data-driven perspectives. For example, [66] notes that ϵ is sensitive to uncertainty in object pose and estimates force closure probability via Monte Carlo simulation to rank grasps by robustness. *Gaussian process implicit surfaces* (GPISs) [67] have represented uncertain geometry for parallel-jaw grasping [68], dexterous grasping with tactile sensors [69], and control synthesis [70]. But, GPISs are difficult to supervise from sensor data (requiring known signed-distance labels),

scale poorly with data quantity, and lack expressivity (due to strong smoothness priors imposed by usual kernels).

An alternate approach is to directly learn a probabilistic metric from data. For example, Dex-Net 2.0 predicts the robustness of a batch of planar parallel-jaw grasps, selecting the most robust one [44]. Similar works have optimized multi-finger grasps with gradient-based optimization by leveraging the differentiability of neural networks [71, 72]. As discussed in Chapter 2, many works have also proposed differentiable approximations of analytic metrics or the force closure condition for gradient-based grasp synthesis, but it is typically not understood how well or whether these approximations preserve grasp robustness.

3.2 Preliminaries

We recall and extend the notation introduced in Chapter 2. As before, we consider a fixed-base, fully-actuated serial manipulator with configuration $\boldsymbol{q} \in \mathcal{Q}$ and n_f fingers contacting a compact rigid object $\mathcal{O} \subset \mathbb{R}^3$ with surface $\partial\mathcal{O}$. The i-th fingertip contacts the object at $\boldsymbol{x}^i = \mathrm{FK}_i(\boldsymbol{q})$ with inward-pointing surface normal $\boldsymbol{n}^i$. As detailed in Sec. 2.2.1, each contact has a pyramidal friction cone with n_s sides whose edges are the basis forces; the resulting $n_w = n_f n_s$ basis wrenches are collected in $\mathcal{W} = \{\boldsymbol{w}_l\}_{l=1}^{n_w}$.

This chapter introduces some additional notation. For a finite set of points $\{\boldsymbol{z}_l\}$ indexed by l, we write $C_{\boldsymbol{z}} := \mathrm{conv}(\{\boldsymbol{z}_l\})$ as shorthand for the convex hull. Define the *wedge* operator $[\cdot]_\times : \mathbb{R}^3 \to \mathfrak{so}(3)$ such that $[\boldsymbol{a}]_\times \boldsymbol{b} := \boldsymbol{a} \times \boldsymbol{b}$ for $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$. Optimal solutions to optimization programs are denoted with asterisks. A set is *degenerate* if it has zero volume. $B_r(\boldsymbol{c})$ denotes a ball of radius r centered at $\boldsymbol{c}$.

We index each basis wrench $\boldsymbol{w}_j^i$ by finger i and pyramid edge j; for brevity, we often combine these into a single index l. Let $\mathcal{I} = \{1, \ldots, n_f\}$, $\mathcal{J} = \{1, \ldots, n_s\}$, and $\mathcal{L} = \{1, \ldots, n_w\}$ denote the corresponding index sets.

We emphasize that the basis wrenches depend on the contact points $\boldsymbol{x}^i$ and surface normals $\boldsymbol{n}^i$, so we always implicitly have $\boldsymbol{w}_l = \boldsymbol{w}_l(\boldsymbol{q})$. We refer to basis wrench uncertainty equivalently as *friction cone uncertainty*, since the friction cone at $\boldsymbol{x}^i$ is determined by $\boldsymbol{n}^i$. This dependence on $\boldsymbol{x}^i$ and $\boldsymbol{n}^i$ is precisely how intrinsic uncertainty (in object geometry) propagates into a grasp's robustness properties.

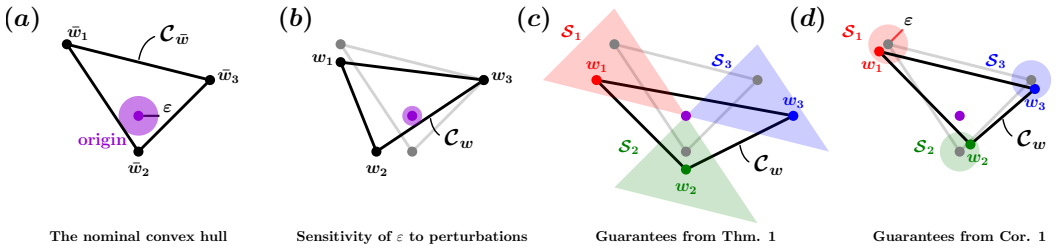


Figure 3.2: **(a)** A set of nominal basis wrenches and their hull $C_{\bar{w}}$ with a Ferrari-Canny metric of ϵ. **(b)** The value of ϵ is sensitive to some perturbations (e.g., $\boldsymbol{w}_2$) but not others (e.g., $\boldsymbol{w}_1$). We derive guarantees on the size of allowable perturbations such that $\mathbf{0} \in C_w$. **(c)** The guarantees from Theorem 3.1. The shaded regions $\mathcal{S}_i$ indicate areas where $\boldsymbol{w}_i$ may lie, independent of other basis wrenches, while still guaranteeing $\mathbf{0} \in C_w$. Note how the $\mathcal{S}_i$ are anisotropic. **(d)** The Ferrari-Canny metric ϵ also provides (strictly worse) guarantees as a result. Note this cartoon shows a planar wrench space, which is actually $\mathbb{R}^6$.

3.3 A Theory of Intrinsically-Robust Grasping

Recall that a grasp is force closure if it can generate Coulomb-compliant forces on the object to resist any disturbance wrench. A sufficient condition for force closure is that the origin is contained in the convex hull of a grasp's basis wrenches, i.e., $\mathbf{0} \in C_w$ [73]. We say that such basis wrenches *certify* force closure.

Since the basis wrenches depend on contacts $\boldsymbol{x}^i$ and surface normals $\boldsymbol{n}^i$, force closure is a function of both an object's geometry and a grasp's contacts. Thus, intrinsic uncertainties generate uncertainty in a grasp's basis wrenches, also reducing extrinsic robustness. In this section, we analyze how much uncertainty a nominally force closure grasp can tolerate while still remaining force closure.

3.3.1 Characterizing a Grasp's Uncertainty Tolerance

Consider a nominal set of basis wrenches $C_{\bar{w}}$ and the convex hull of a perturbed version, C_w (see Fig. 3.2a/b). For a given grasp, the value of ϵ may be sensitive or insensitive to such perturbations. If the perturbations are "large," the origin will leave C_w, so we cannot certify force closure. Let $\boldsymbol{w}_l$ and $\bar{\boldsymbol{w}}_l$ denote elements of C_w and $C_{\bar{w}}$ respectively. The following result bounds the size of tolerable perturbations in terms of $C_{\bar{w}}$.

Theorem 3.1. *If* $\boldsymbol{w}_l - \bar{\boldsymbol{w}}_l \in -C_{\bar{w}}$ *for all* $l \in \mathcal{L}$*, then* $\mathbf{0} \in C_w$.

Proof. Suppose for the sake of contradiction that $\mathbf{0} \notin C_w$. Then, by the separating hyperplane theorem, there exists $\boldsymbol{a}$ such that $\boldsymbol{a}^\top \boldsymbol{w} > 0$ for all $\boldsymbol{w} \in C_w$. Further, there must exist some $k \in \mathcal{L}$ satisfying $\boldsymbol{a}^\top \bar{\boldsymbol{w}}_k \leq \boldsymbol{a}^\top \bar{\boldsymbol{w}}_l$ for all $l \in \mathcal{L}$.

Since $\boldsymbol{w}_l - \bar{\boldsymbol{w}}_l \in -C_{\bar{\boldsymbol{w}}}$ for each l, there exists a set of convex weights $\boldsymbol{\alpha} \in \mathbb{R}^{n_w}$ satisfying $\boldsymbol{\alpha} \geq \mathbf{0}$ and $\sum_{l=1}^{n_w} \alpha_l = 1$, such that

$$\boldsymbol{w}_k - \bar{\boldsymbol{w}}_k = -\sum_{l=1}^{n_w} \alpha_l \bar{\boldsymbol{w}}_l. \tag{3.1}$$

Thus,

$$\begin{aligned} \boldsymbol{w}_k &= \bar{\boldsymbol{w}}_k + (\boldsymbol{w}_k - \bar{\boldsymbol{w}}_k) \\ &= \bar{\boldsymbol{w}}_k - \sum_{l=1}^{n_w} \alpha_l \bar{\boldsymbol{w}}_l, \end{aligned} \tag{3.2}$$

which implies

$$\begin{aligned} \boldsymbol{a}^\top \boldsymbol{w}_k &= \boldsymbol{a}^\top \bar{\boldsymbol{w}}_k - \sum_{l=1}^{n_w} \alpha_l \boldsymbol{a}^\top \bar{\boldsymbol{w}}_l \\ &\leq \boldsymbol{a}^\top \bar{\boldsymbol{w}}_k - \boldsymbol{a}^\top \bar{\boldsymbol{w}}_k \sum_{l=1}^{n_w} \alpha_l \\ &= 0. \end{aligned} \tag{3.3}$$

But $\boldsymbol{a}^\top \boldsymbol{w}_k > 0$ since $\boldsymbol{w}_k \in C_{\boldsymbol{w}}$, contradicting (3.3). □

We view $\boldsymbol{w}_l - \bar{\boldsymbol{w}}_l$ as the deviation of the l^{th} basis wrench from some nominal value. If all deviations are respectively contained in $-C_{\bar{\boldsymbol{w}}}$, then by Theorem 3.1, the origin lies in the convex hull $C_{\boldsymbol{w}}$ of the true, perturbed basis wrenches, i.e., *the grasp is force closure*. See Fig. 3.2c for visual intuition. Further, we can relate this result to the Ferrari-Canny metric ϵ directly, resulting in the following corollary.

Corollary 3.1. *Let ϵ denote the Ferrari-Canny metric for a nominal set of basis wrenches $\overline{\mathcal{W}}$. If the true (i.e., perturbed) basis wrenches $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{n_w}$ satisfy*

$$\boldsymbol{w}_l \in B_\epsilon(\bar{\boldsymbol{w}}_l), \ \forall l \in \mathcal{L}, \tag{3.4}$$

then the true grasp is force closure.

Proof. By definition of the Ferrari-Canny metric, we have

$$B_\epsilon(\mathbf{0}) \subset C_{\bar{\boldsymbol{w}}}. \tag{3.5}$$

Therefore, for any $l \in \mathcal{L}$, if $\boldsymbol{w}_l \in B_\epsilon(\bar{\boldsymbol{w}}_l)$, we have

$$\boldsymbol{w}_l - \bar{\boldsymbol{w}}_l \in B_\epsilon(\mathbf{0}) \subset -C_{\bar{\boldsymbol{w}}}. \tag{3.6}$$

The result follows by Theorem 3.1, which gives $\mathbf{0} \in C_{\boldsymbol{w}}$, so the true grasp is force closure. □

Though ϵ represents a grasp's sensitivity to external disturbance wrenches, Corollary 3.1 shows that ϵ also captures a grasp's *intrinsic* robustness (e.g., to object pose/geometry) implicitly through friction cone uncertainty. We analyze concrete ways to explicitly use the relation between object geometry and basis wrenches in Sec. 3.4.1.

See Fig. 3.2d for an illustration of Corollary 3.1.

3.3.2 Bounding ϵ with the Min-Weight Metric ℓ^*

Recall from Chapter 2 that the min-weight metric ℓ^* (Sec. 2.3) encourages the origin to lie far inside $\mathcal{C}_{\boldsymbol{w}}$ by maximizing its minimum convex weight, and is defined by the LP

$$\begin{aligned} \ell^* = \underset{\boldsymbol{\alpha}\in\mathbb{R}^{n_w},\ell\in\mathbb{R}}{\text{maximize}} \quad & \ell \\ \text{subject to} \quad & \mathbf{W}\boldsymbol{\alpha} = \mathbf{0} \\ & \mathbb{1}_{n_w}^\top \boldsymbol{\alpha} = 1 \\ & \boldsymbol{\alpha} \geq \ell \mathbb{1}_{n_w}, \end{aligned} \tag{3.7}$$

where $\mathbf{W} = \mathbf{W}(\boldsymbol{q})$ is the matrix whose l^{th} column is $\boldsymbol{w}_l \in \mathcal{W}$ and $\mathbb{1}_{n_w} \in \mathbb{R}^{n_w}$ is the vector of all 1s.

In Chapter 2 (Fig. 2.3), it was empirically observed that (an unknown positive scaling K of) ℓ^* seemed to lower bound ϵ uniformly on force closure grasps. If true, this would imply that ℓ^* captures intrinsic robustness by Corollary 3.1, since $B_{K\ell^*}(\mathbf{0}) \subseteq B_\epsilon(\mathbf{0})$. In this section, we leverage Theorem 3.1 to formally prove this claim for "sufficiently regular" grasps.

Before doing so, we remark that every compact set $\mathcal{X}$ has a largest (not necessarily origin-centered) inscribing ball whose radius is called the *Chebyshev radius*, which we denote δ. We use this to impose some desired regularity on the types of grasps we consider.

Definition 3.1. *Let $P(\mathcal{X})$ denote the power set of $\mathcal{X}$. The set $\mathcal{S}_\mathcal{O} \subset P(\mathbb{R}^6)$ is the set of all sets of basis wrenches $\mathcal{W}$ with cardinality n_w such that*

1. *$\mathcal{W}$ is possible to generate on the manipuland $\mathcal{O}$, and*
2. *the Chebyshev radii of $\mathcal{C}_{\boldsymbol{w}}$, denoted $\delta(\mathcal{W})$, are uniformly bounded: $\delta(\mathcal{W}) \geq \delta_{\min} > 0$ for all $\mathcal{W} \in \mathcal{S}_\mathcal{O}$.*

Condition (2) is mild; for every grasp, $\delta \geq \epsilon$, so Definition 3.1 excludes grasps that are poor by virtue of a small Chebyshev radius up to a fixed level $\delta_{\min}$. This condition is satisfied in practice, since we synthesize grasps by lower bounding their desired quality. This gives the following.

Theorem 3.2. *For all basis wrenches $\mathcal{W} \in \mathcal{S}_O$ on a compact manipuland O, there exists a grasp-independent constant $K(O) > 0$ uniformly satisfying $K\ell^*(\mathcal{W}) \leq \epsilon(\mathcal{W})$.*

3.3.3 Proof of Theorem 3.2

To prove Theorem 3.2, we require some intermediate results. First, consider the following characterization of ball containment in a convex hull (rephrased from [53, Eqn. (27)]).

Lemma 3.1. *Fix some $r \geq 0$ and points $\mathcal{W} = \{\boldsymbol{w}_l\}_{l=1}^{n_w} \subset \mathbb{R}^6$. Then, $B_r(\mathbf{0}) \subset C_{\boldsymbol{w}}$ if and only if*

$$b \geq r \tag{3.8}$$

for every pair $(\boldsymbol{a}, b) \in \mathbb{R}^6 \times \mathbb{R}$ that satisfies

$$\begin{aligned} \boldsymbol{a}^\top \boldsymbol{w}_l + b \geq 0, \ \forall l \in \mathcal{L}, \\ \boldsymbol{a}^\top \boldsymbol{a} = 1. \end{aligned} \tag{3.9}$$

Using Lemma 3.1, we can reformulate the Ferrari-Canny metric as the following optimization program.

Lemma 3.2. *When $\mathbf{0} \in C_{\boldsymbol{w}}$ (i.e., the grasp is force closure), the Ferrari-Canny metric can be expressed*

$$\begin{aligned} \epsilon = \underset{\boldsymbol{a}\in\mathbb{R}^6, b\in\mathbb{R}}{\text{minimize}} \quad & b \\ \text{subject to} \quad & \boldsymbol{a}^\top \boldsymbol{w}_l + b \geq 0, \ \forall l \in \mathcal{L}, \\ & \boldsymbol{a}^\top \boldsymbol{a} = 1. \end{aligned} \tag{3.10}$$

Proof. Program (3.10) is motivated by Lemma 3.1. The choice $r = \epsilon$ proves Lemma 3.2 by contradiction.

Let $(\boldsymbol{a}^*, b^*)$ be an optimal solution for (3.10) and define the halfspace

$$\mathcal{H} := \{\boldsymbol{x} \mid \boldsymbol{x}^\top \boldsymbol{a}^* + b^* \geq 0\}. \tag{3.11}$$

First, we show $b^* \geq 0$. Suppose that $b^* < 0$. Then,

$$\mathbf{0}^\top \boldsymbol{a}^* + b^* = b^* < 0, \tag{3.12}$$

so $\mathbf{0} \notin \mathcal{H}$. However, for $(\boldsymbol{a}^*, b^*)$ to be feasible, we must have that $C_{\boldsymbol{w}} \subset \mathcal{H}$ by the first constraint of (3.10). This contradicts the assumption that $\mathbf{0} \in C_{\boldsymbol{w}}$, so we conclude

$$b^* \geq 0. \tag{3.13}$$

Now, suppose for the sake of contradiction that $b^* \neq \epsilon$. Since $(\boldsymbol{a}^*, b^*)$ satisfies Lemma 3.1, we have

$$\begin{aligned} B_{b^*}(\mathbf{0}) &\subset C_{\boldsymbol{w}}, \\ \|\boldsymbol{a}^*\|_2 &= 1. \end{aligned} \tag{3.14}$$

Further, by definition,

$$B_\epsilon(\mathbf{0}) \subset C_{\boldsymbol{w}}. \tag{3.15}$$

Consider two cases.

- **Case:** $b^* < \epsilon$**.** Because $b^* < \epsilon$ and $B_\epsilon(\mathbf{0}) \subset C_{\boldsymbol{w}}$, we must have

  $$\begin{aligned} -b^* \boldsymbol{a}^* &\in B_{b^*}(\mathbf{0}) \subset \operatorname{int}(C_{\boldsymbol{w}}), \\ -\epsilon \boldsymbol{a}^* &\in B_\epsilon(\mathbf{0}) \subset C_{\boldsymbol{w}}. \end{aligned} \tag{3.16}$$

  By convexity, the point

  $$\boldsymbol{y} = -\frac{(b^* + \epsilon)}{2} \boldsymbol{a}^* \tag{3.17}$$

  lies in $C_{\boldsymbol{w}}$. First, we must have $\boldsymbol{y} \notin \mathcal{H}$, since

  $$\begin{aligned} \boldsymbol{y}^\top \boldsymbol{a}^* + b^* &= \frac{1}{2}(b^* - \epsilon) \\ &< 0, \end{aligned} \tag{3.18}$$

  where $b^* - \epsilon < 0$ by assumption. However, $\boldsymbol{y} \in C_{\boldsymbol{w}}$, so there must exist convex weights $\boldsymbol{\alpha}$ such that

  $$\boldsymbol{y} = \sum_l \alpha_l \boldsymbol{w}_l. \tag{3.19}$$

This implies that $\boldsymbol{y} \in \mathcal{H}$, because

$$\begin{aligned} \boldsymbol{y}^\top \boldsymbol{a}^* + b^* &= \sum_l \alpha_l \boldsymbol{w}_l^\top \boldsymbol{a}^* + \sum_l \alpha_l b^* \\ &= \sum_l \alpha_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^* \right) \\ &\geq 0 \end{aligned} \tag{3.20}$$

since $(\boldsymbol{a}^*, b^*)$ is a feasible solution to (3.10). Equations (3.18) and (3.20) contradict, so we cannot have $b^* < \epsilon$.

- **Case: $b^* > \epsilon$.** Since ϵ is the radius of the largest ball inscribed in $C_{\boldsymbol{w}}$, there exists a facet of $C_{\boldsymbol{w}}$ with outward pointing unit normal $\boldsymbol{d}^*$ such that

$$\epsilon \boldsymbol{d}^* \in \partial C_{\boldsymbol{w}}. \tag{3.21}$$

This defines a supporting hyperplane of $C_{\boldsymbol{w}}$:

$$\mathcal{P}_{\boldsymbol{d}^*} := \{\boldsymbol{x} \mid -\boldsymbol{x}^\top \boldsymbol{d}^* + \epsilon = 0\}. \tag{3.22}$$

Since $C_{\boldsymbol{w}}$ is the intersection of all halfspaces containing $\mathcal{W}$, and every facet of $C_{\boldsymbol{w}}$ is generated by a supporting hyperplane of $C_{\boldsymbol{w}}$, the pair $(-\boldsymbol{d}^*, \epsilon)$ is a feasible solution for (3.10). That is,

$$\begin{aligned} -\boldsymbol{d}^{*\top} \boldsymbol{w}_l + \epsilon \geq 0, \ \forall l \in \mathcal{L}, \\ \boldsymbol{d}^{*\top} \boldsymbol{d}^* = 1. \end{aligned} \tag{3.23}$$

However, since $(-\boldsymbol{d}^*, \epsilon)$ achieves objective value $\epsilon < b^*$, this contradicts that $(\boldsymbol{a}^*, b^*)$ is optimal for (3.10), completing the proof.

□

3.3.3.1 The Min-Weight LP Dual

The key ingredient for comparing programs (3.10) and (3.7) is to derive the dual of the min-weight LP, which reveals a structural similarity between the two programs. In particular, the dual of the min-weight LP takes a form that closely resembles the nonconvex program (3.10), differing only in the quadratic constraint $\boldsymbol{a}^\top \boldsymbol{a} = 1$ being replaced by a linear normalization constraint. This connection is what ultimately allows us to relate the two optimal values.

Lemma 3.3. *By strong duality, the min-weight metric ℓ^* defined in* (3.7) *satisfies*

$$\begin{aligned} \ell^* = \underset{\boldsymbol{\nu}\in\mathbb{R}^6,\varphi\in\mathbb{R}}{\text{minimize}} \quad & \varphi \\ \text{subject to} \quad & \boldsymbol{\nu}^\top \boldsymbol{w}_l + \varphi \geq 0, \ \forall l \in \mathcal{L} \\ & \sum_l \left(\boldsymbol{\nu}^\top \boldsymbol{w}_l + \varphi\right) = 1. \end{aligned} \tag{3.24}$$

Proof. Recall the definition of the min-weight metric:

$$\begin{aligned} \ell^* = \underset{\boldsymbol{\alpha},\ell}{\text{maximize}} \quad & \ell \\ \text{subject to} \quad & \mathbf{W}\boldsymbol{\alpha} = \mathbf{0} \\ & \mathbb{1}^\top \boldsymbol{\alpha} = 1 \\ & \boldsymbol{\alpha} \geq \ell \mathbb{1}. \end{aligned} \tag{3.25}$$

First, note that the min-weight metric can be rewritten as

$$\begin{aligned} \ell^* = -\underset{\boldsymbol{\alpha},\ell}{\text{minimize}} \quad & -\ell \\ \text{subject to} \quad & \mathbf{W}\boldsymbol{\alpha} = \mathbf{0} \\ & \mathbb{1}^\top \boldsymbol{\alpha} = 1 \\ & \boldsymbol{\alpha} \geq \ell \mathbb{1}. \end{aligned} \tag{3.26}$$

The Lagrangian of the minimization, with dual variables $\boldsymbol{\lambda} \geq \mathbf{0}$, $\boldsymbol{\nu}$, and φ, can be written as

$$\mathfrak{L} = -\ell + \boldsymbol{\lambda}^\top(\ell\mathbb{1} - \boldsymbol{\alpha}) + \boldsymbol{\nu}^\top \mathbf{W}\boldsymbol{\alpha} + \varphi(\mathbb{1}^\top \boldsymbol{\alpha} - 1). \tag{3.27}$$

Collecting terms in the primal variables ℓ and $\boldsymbol{\alpha}$, we obtain

$$\mathfrak{L} = \ell(-1 + \boldsymbol{\lambda}^\top \mathbb{1}) + \boldsymbol{\alpha}^\top(-\boldsymbol{\lambda} + \mathbf{W}^\top \boldsymbol{\nu} + \varphi\mathbb{1}) - \varphi. \tag{3.28}$$

The dual function is obtained by minimizing the Lagrangian over the primal variables $(\ell, \boldsymbol{\alpha})$. For the Lagrangian to be bounded below, the coefficients of ℓ and $\boldsymbol{\alpha}$ must vanish, giving

$$\begin{aligned} -1 + \boldsymbol{\lambda}^\top \mathbb{1} &= 0, \\ -\boldsymbol{\lambda} + \mathbf{W}^\top \boldsymbol{\nu} + \varphi\mathbb{1} &= \mathbf{0}. \end{aligned} \tag{3.29}$$

Subject to these conditions, the dual function evaluates to

$$g(\boldsymbol{\lambda}, \boldsymbol{\nu}, \varphi) = \begin{cases} -\varphi, & \text{if } \boldsymbol{\lambda}^\top \mathbb{1} = 1, \ \mathbf{W}^\top \boldsymbol{\nu} + \varphi\mathbb{1} = \boldsymbol{\lambda}, \\ -\infty, & \text{else.} \end{cases} \tag{3.30}$$

Thus, we have the dual program, and by strong duality,

$$\begin{aligned} \ell^* = -\underset{\boldsymbol{\lambda},\boldsymbol{\nu},\varphi}{\text{maximize}} \quad & -\varphi \\ \text{subject to} \quad & \mathbb{1}^\top \boldsymbol{\lambda} = 1 \\ & \boldsymbol{\lambda} = \mathbf{W}^\top \boldsymbol{\nu} + \varphi \mathbb{1} \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{3.31}$$

Rewriting the dual as a minimization and eliminating $\boldsymbol{\lambda}$ yields the desired result. □

We remark on the structural similarity between programs (3.10) and (3.24). Both minimize a scalar over a set defined by linear inequalities involving the basis wrenches. In program (3.10), the search direction $\boldsymbol{a}$ is constrained to the unit sphere ($\boldsymbol{a}^\top \boldsymbol{a} = 1$), whereas in the dual (3.24), the normalization is instead enforced by a linear equality constraint on the sum

$$\sum_l (\boldsymbol{\nu}^\top \boldsymbol{w}_l + \varphi) = 1.$$

This relaxation from a quadratic to a linear constraint is precisely what makes the min-weight metric efficiently computable via an LP, and the proof of Theorem 3.2 exploits this relationship by constructing a feasible dual solution from an optimal solution to (3.10).

3.3.3.2 Main Proof

We are now ready to prove Theorem 3.2.

Proof. We only consider the case where $\mathbf{0} \in C_w$, since otherwise, there is no origin-centered inscribed ball of C_w. This allows us to invoke Lemma 3.2. Consider an arbitrary set of basis wrenches $\mathcal{W} \in \mathcal{S}_O$ and suppose $(\boldsymbol{a}^*, b^*)$ is an optimal solution to (3.10), so that

$$\|\boldsymbol{a}^*\|_2 = 1. \tag{3.32}$$

Define the pair $(\boldsymbol{\nu}, \varphi)$ by

$$\begin{aligned} \boldsymbol{\nu} &= \frac{\boldsymbol{a}^*}{\sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^* \right)}, \\ \varphi &= \frac{b^*}{\sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^* \right)}. \end{aligned} \tag{3.33}$$

This pair is clearly feasible for (3.24). Dividing by $\sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^*\right)$ is well-defined, since if the sum were not strictly positive, it would imply that $C_{\boldsymbol{w}}$ is degenerate and contained in the hyperplane

$$\mathcal{P} = \left\{\boldsymbol{x} \mid \boldsymbol{x}^\top \boldsymbol{a}^* + b^* = 0\right\}, \tag{3.34}$$

which would imply $\mathcal{W} \notin \mathcal{S}_O$.

By strong duality and the feasibility[1] of $(\boldsymbol{\nu}, \varphi)$, we have

$$\begin{aligned} \ell^* &= \varphi^* \\ &\leq \frac{b^*}{\sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^*\right)} \\ &= \frac{\epsilon}{\sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^*\right)}, \end{aligned} \tag{3.35}$$

where $b^* = \epsilon$ by Lemma 3.2. Rearranging, we obtain

$$\ell^* \cdot \sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^* + b^*\right) \leq \epsilon. \tag{3.36}$$

So, if $K > 0$ exists such that

$$K \leq \sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^*(\mathcal{W}) + b^*(\mathcal{W})\right) \tag{3.37}$$

uniformly for all $\mathcal{W} \in \mathcal{S}_O$, the proof follows.

Let $\delta(\mathcal{W}) > 0$ be the Chebyshev radius of $C_{\boldsymbol{w}}$. We show if $\boldsymbol{v} \in C_{\boldsymbol{w}}$ is a point in $C_{\boldsymbol{w}}$ farthest from $\mathcal{P}$, then

$$\boldsymbol{v}^\top \boldsymbol{a}^* + b^* \geq 2\delta. \tag{3.38}$$

Suppose for the sake of contradiction that

$$\boldsymbol{v}^\top \boldsymbol{a}^* + b^* < 2\delta. \tag{3.39}$$

Let $\boldsymbol{c}$ be a Chebyshev center. Then, the distance of $\boldsymbol{c}$ to $\mathcal{P}$ must satisfy

$$\boldsymbol{c}^\top \boldsymbol{a}^* + b^* \geq \delta, \tag{3.40}$$

[1] Since the primal (3.7) is a maximization, the dual (3.24) is a minimization and dual-feasible values *upper bound* the optimal dual and primal values.

or $\boldsymbol{c}$ could not be the center of a ball inscribing $C_{\boldsymbol{w}}$. Let

$$\boldsymbol{y} := \boldsymbol{c} + \delta \boldsymbol{a}^* \in B_\delta(\boldsymbol{c}) \subset C_{\boldsymbol{w}}. \tag{3.41}$$

The distance of $\boldsymbol{y}$ to the plane $\mathcal{P}$ is given by

$$\begin{aligned} \boldsymbol{y}^\top \boldsymbol{a}^* + b^* &= \boldsymbol{c}^\top \boldsymbol{a}^* + \delta \cdot (\boldsymbol{a}^*)^\top \boldsymbol{a}^* + b^* \\ &= \delta + \boldsymbol{c}^\top \boldsymbol{a}^* + b^* \\ &\geq 2\delta \\ &> \boldsymbol{v}^\top \boldsymbol{a}^* + b^*, \end{aligned} \tag{3.42}$$

where the last inequality is by assumption, contradicting that $\boldsymbol{v}$ is a farthest point in $C_{\boldsymbol{w}}$ from $\mathcal{P}$. Thus,

$$\boldsymbol{v}^\top \boldsymbol{a}^* + b^* \geq 2\delta. \tag{3.43}$$

Now, there exists an extreme point of $C_{\boldsymbol{w}}$ that is a farthest point in $C_{\boldsymbol{w}}$ from $\mathcal{P}$, and $\mathcal{W}$ contains all extreme points of $C_{\boldsymbol{w}}$ by construction. Thus, there exists $\boldsymbol{w} \in \mathcal{W}$ such that

$$\boldsymbol{w}^\top \boldsymbol{a}^* + b^* \geq 2\delta. \tag{3.44}$$

By Def. 3.1, we assume a $\delta_{\min}$ such that

$$\begin{aligned} 2\delta_{\min} &\leq 2\delta(\mathcal{W}) \\ &\leq \max_{\boldsymbol{w} \in \mathcal{W}} \boldsymbol{w}^\top \boldsymbol{a}^*(\mathcal{W}) + b^*(\mathcal{W}) \\ &\leq \sum_l \left(\boldsymbol{w}_l^\top \boldsymbol{a}^*(\mathcal{W}) + b^*(\mathcal{W}) \right), \end{aligned} \tag{3.45}$$

so the choice $K = 2\delta_{\min}$ completes the proof. □

3.4 Towards Probabilistic Force Closure

The previous analysis applied the framework of intrinsic robustness and Theorem 3.1 to characterize the min-weight metric. We now explore another application, and develop the idea of *probabilistic force closure*, wherein we use the relation between object geometry and basis wrenches to apply our theory explicitly when we have a *belief distribution* over the object geometry.

Suppose the basis wrenches $\mathcal{W}$ associated with a grasp are a random variable generated by, e.g., uncertainty in the contact locations $\boldsymbol{x}^i$, the surface normals $\boldsymbol{n}^i(\boldsymbol{x}^i)$

of O, parameters γ of the object $O(\gamma)$, etc. Then, one can attempt to compute or lower bound the probability of force closure.

Let $\overline{W}$ and $\mathcal{W}$ be *nominal* and *random* sets of basis wrenches respectively with convex hulls $C_{\bar{w}}$ and C_w. By Theorem 3.1, we can bound the probability of force closure:

$$\begin{aligned}\mathbb{P}[\boldsymbol{w}_l - \bar{\boldsymbol{w}}_l \in -C_{\bar{w}},\ \forall l \in \mathcal{L}] &\leq \mathbb{P}[\mathbf{0} \in C_w] \\ &\leq \mathbb{P}[\text{force closure}].\end{aligned} \tag{3.46}$$

If we can compute or bound the LHS of (3.46), then probabilistically-robust grasps may be synthesized. However, as-is, (3.46) is not very useful, since (a) it requires a probability measure to be defined over O, and (b) the relevant uncertainty sets (e.g., $-C_{\bar{w}}$ in Theorem 3.1 or $B_\epsilon(\bar{\boldsymbol{w}}_l)$ in Corollary 3.1) must be easily integrable with respect to this measure.

3.4.1 Probabilistic Object Normals for Grasping

In light of these challenges, we study a special case where the contact locations $\boldsymbol{x}^i$ are known but the surface normals $\boldsymbol{n}^i$ are random and modeled by a degenerate Gaussian distribution. This gives rise to a method we call *probabilistic object normals for grasping* (PONG).

Let the known contacts be denoted $X = \{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^{n_f}\}$ and consider the set of normals that induce basis wrenches certifying force closure. We call this the *force closure set*:

$$\mathfrak{N}(X) := \left\{ \{\boldsymbol{n}^i\}_{i=1}^{n_f} \mid \mathbf{0} \in C_w \right\}. \tag{3.47}$$

Suppose the normals are jointly distributed with density p and that they are mutually independent, so that the joint density factorizes:

$$p(\boldsymbol{n}^1, \ldots, \boldsymbol{n}^{n_f}) = \prod_{i=1}^{n_f} p(\boldsymbol{n}^i). \tag{3.48}$$

Then, we can compute the probability that $\mathbf{0} \in C_w$ as follows:

$$\mathbb{P}[\mathbf{0} \in C_w] = \int_{\mathfrak{N}(X)} \prod_{i=1}^{n_f} p(\boldsymbol{n}^i) d\boldsymbol{n}^i. \tag{3.49}$$

Though p is factorizable, the integral itself is not, since the integration variables are coupled by the domain $\mathfrak{N}$, and C_w depends on all of the random normals. Further,

$\mathfrak{N}$ is difficult to parameterize; it has no closed form since it is implicitly defined by the condition $\mathbf{0} \in C_{\boldsymbol{w}}$, where hull membership is checked by solving an LP [73].

Thus, our strategy is to derive an *approximate force closure set* $\mathcal{A} \subseteq \mathfrak{N}$ in tandem with a choice of density function p for which the integral over $\mathcal{A}$ is known. Since $\mathcal{A} \subseteq \mathfrak{N}$, integrating p over $\mathcal{A}$ yields a lower bound on (3.49).

3.4.2 Random Normals, Forces, and Wrenches

We model each random normal $\boldsymbol{n}^i$ as the sum of a deterministic *mean normal vector* denoted $\bar{\boldsymbol{n}}^i$ and a *random perturbation vector* $\Delta\boldsymbol{n}^i = \mathcal{T}_i \boldsymbol{z}$, where $\mathcal{T}_i = \begin{bmatrix} \bar{\boldsymbol{t}}_1^i & \bar{\boldsymbol{t}}_2^i \end{bmatrix} \in \mathbb{R}^{3\times 2}$ is a basis for the tangent plane at $\bar{\boldsymbol{n}}^i$, and $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \Sigma^i)$ is a zero-mean Gaussian random vector in $\mathbb{R}^2$.

We parameterize the normals in this way for two reasons. First, all perturbations in the direction of $\bar{\boldsymbol{n}}^i$ do not change its direction, and thus do not contribute to uncertainty in its orientation. Second, the planar restriction allows us to integrate a Gaussian density in the plane over a polytope using Green's Theorem [74, Prop. 1].

We remark that with this construction, the random normals $\boldsymbol{n}^i$ will not have unit norm. However, since friction cones are invariant under scaling, as long as the random basis forces $\boldsymbol{f}_j^i$ remain Coulomb-compliant, we can still certify force closure with the induced random basis wrenches.

We now construct a random friction pyramid about the random normal $\boldsymbol{n}^i$ such that its edges are always Coulomb-compliant. First, consider the *mean* basis wrenches $\bar{\boldsymbol{w}}_j^i$ with force and torque components $\bar{\boldsymbol{f}}_j^i$ and $\bar{\boldsymbol{\tau}}_j^i$. We represent $\bar{\boldsymbol{f}}_j^i$ as the sum of $\bar{\boldsymbol{n}}^i$ and a tangent component $\left(\bar{\boldsymbol{f}}_j^i\right)_t$ of length μ. We compute $\left(\bar{\boldsymbol{f}}_j^i\right)_t$ using a unit length *generator* $\bar{\boldsymbol{g}}_j^i(\bar{\boldsymbol{n}}^i) \in \mathbb{R}^3$ of our choice orthogonal to $\bar{\boldsymbol{n}}^i$ such that

$$\left(\bar{\boldsymbol{f}}_j^i\right)_t = \mu \cdot \left(\bar{\boldsymbol{g}}_j^i \times \bar{\boldsymbol{n}}^i\right). \tag{3.50}$$

For example, for some arbitrary $\boldsymbol{v} \neq \bar{\boldsymbol{n}}^i$, we could pick

$$\bar{\boldsymbol{g}}_j^i(\bar{\boldsymbol{n}}^i) = \frac{\boldsymbol{v} \times \bar{\boldsymbol{n}}^i}{\|\boldsymbol{v} \times \bar{\boldsymbol{n}}^i\|_2}. \tag{3.51}$$

Thus, we can write

$$\begin{aligned} \bar{\boldsymbol{f}}_j^i &= \bar{\boldsymbol{n}}^i + \mu \cdot \left(\bar{\boldsymbol{g}}_j^i \times \bar{\boldsymbol{n}}^i \right) \\ \implies \bar{\boldsymbol{w}}_j^i &= \underbrace{\begin{bmatrix} \left(I + \mu \cdot \left[\bar{\boldsymbol{g}}_j^i \right]_\times \right) \\ \left[\boldsymbol{x}^i \right]_\times \left(I + \mu \cdot \left[\bar{\boldsymbol{g}}_j^i \right]_\times \right) \end{bmatrix}}_{:= T_j^i(\bar{\boldsymbol{n}}^i)} \bar{\boldsymbol{n}}^i, \end{aligned} \tag{3.52}$$

and similarly, $\boldsymbol{w}_j^i = T_j^i \boldsymbol{n}^i$, which satisfies $\left\| (\boldsymbol{f}_j^i)_t \right\|_2 \leq \mu \left\| \boldsymbol{n}^i \right\|$ (in contrast, recall that $\left\| \left(\bar{\boldsymbol{f}}_j^i \right)_t \right\|_2 = \mu \left\| \boldsymbol{n}^i \right\|$). To see this, let ψ_j^i be the angle between $\bar{\boldsymbol{g}}_j^i$ and $\boldsymbol{n}^i$. Then,

$$\left\| (\boldsymbol{f}_j^i)_t \right\|_2 = \mu \left\| \bar{\boldsymbol{g}}_j^i \right\| \left\| \boldsymbol{n}^i \right\| \sin(\psi_j^i) \leq \mu \left\| \boldsymbol{n}^i \right\|, \tag{3.53}$$

ensuring Coulomb-compliance of the random basis forces.

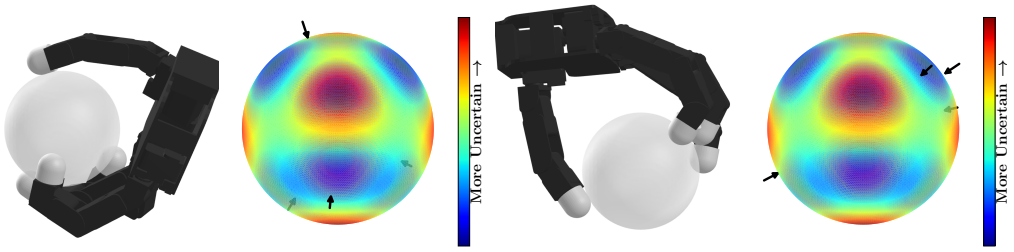


Figure 3.3: More examples of grasps on a synthetically uncertain spherical manipuland. The surface normal tangent covariance at each point on the sphere is (proportional to) an isotropic, monotonically increasing function of the real part of the $(2, 4)$-spherical harmonic. The synthesized grasps tend to avoid placing the fingertips on the red uncertain regions while remaining kinematically feasible. They were synthesized in 6.47 and 4.80 seconds respectively.

3.4.3 Deriving an Approximate Force Closure Set

Next, we define disjoint sets $\mathcal{A}^i$ whose union defines a set $\mathcal{A} \subseteq \mathfrak{N}$. Combining Theorem 3.1 with the definition of $\mathfrak{N}$ and the relation $\boldsymbol{w}_j^i = T_j^i \boldsymbol{n}^i$ from Sec. 3.4.2, if $\mathcal{A}^i$ satisfies

$$\mathcal{A}^i \subseteq \{ \boldsymbol{n}^i \mid T_j^i(\boldsymbol{n}^i - \bar{\boldsymbol{n}}^i) \in -C_{\bar{\boldsymbol{w}}},\ \forall j \in \mathcal{J} \}, \tag{3.54}$$

then we have

$$\boldsymbol{n}^i \in \mathcal{A}^i,\ \forall i \in \mathcal{I} \implies (\boldsymbol{n}^1, \ldots, \boldsymbol{n}^{n_f}) \in \mathfrak{N}. \tag{3.55}$$

Letting $\mathcal{A} = \bigcup_{i=1}^{n_f} \mathcal{A}^i$, we therefore have the bound

$$\begin{aligned} \prod_{i=1}^{n_f} \int_{\mathcal{A}^i} p(\boldsymbol{n}^i) d\boldsymbol{n}^i &= \int_{\mathcal{A}} \prod_{i=1}^{n_f} p(\boldsymbol{n}^i) d\boldsymbol{n}^i \\ &\leq \int_{\mathfrak{N}} \prod_{i=1}^{n_f} p(\boldsymbol{n}^i) d\boldsymbol{n}^i. \end{aligned} \tag{3.56}$$

We can factorize the integral this way since $C_{\bar{\boldsymbol{w}}}$ in (3.54) depends only on the *mean* normals $(\bar{\boldsymbol{n}}^1, \ldots, \bar{\boldsymbol{n}}^{n_f})$, which are fixed parameters of the known uncertainty distributions. In contrast, in (3.47), $C_{\boldsymbol{w}}$ (and thus $\mathfrak{N}$) depends jointly on all of the *random* normals, which are the integration variables.

Unfortunately, the set on the RHS of (3.54) still cannot be expressed in closed form. Thus, we let $\mathcal{A}^i$ conservatively approximate it by computing a polytope with large volume satisfying (3.54) that is parameterized by its vertices.

To do this, for each finger, we fix k search directions $\boldsymbol{d}^{i,k} \in \mathbb{R}^3$ in the tangent plane to O at $\boldsymbol{x}^i$ with index set $\mathcal{K}$. We compute the longest step $\theta^{i,k} \geq 0$ that can be taken in this direction while satisfying (3.54) and denote the point $\theta^{i,k}\boldsymbol{d}^{i,k}$ as $\boldsymbol{v}^{i,k}$. We then let $\mathcal{A}^i$ be the convex hull over all $\boldsymbol{v}^{i,k}$.

Conveniently, this search can be expressed as a linear program. Define $\overline{\mathbf{W}} \in \mathbb{R}^{6\times n_w}$, whose columns are the nominal basis wrenches. Then, to compute each $\boldsymbol{v}^{i,k}$, we can solve the following LP for each $(i,k) \in \mathcal{I} \times \mathcal{K}$ in parallel:

$$\begin{aligned} \underset{\substack{\theta^{i,k}\in\mathbb{R} \\ \left\{\boldsymbol{\alpha}_j^{i,k}\right\}_{j=1}^{n_s} \subset \mathbb{R}^{n_w}}}{\text{maximize}} \quad & \theta^{i,k} && \text{(3.57a)} \\ \text{subject to} \quad & \theta^{i,k} \geq 0 && \text{(3.57b)} \\ & \boldsymbol{\alpha}_j^{i,k} \geq \mathbf{0}, \ \forall j \in \mathcal{J} && \text{(3.57c)} \\ & \mathbb{1}_{n_w}^\top \boldsymbol{\alpha}_j^{i,k} = 1, \ \forall j \in \mathcal{J} && \text{(3.57d)} \\ & T_j^i\left(\theta^{i,k}\boldsymbol{d}^{i,k}\right) = -\overline{\mathbf{W}}\boldsymbol{\alpha}_j^{i,k}, \ \forall j \in \mathcal{J}. && \text{(3.57e)} \end{aligned}$$

Constraint (3.57b) enforces nonnegativity of the scaling along $\boldsymbol{d}^{i,k}$, constraints (3.57c) and (3.57d) enforce that the $\boldsymbol{\alpha}_j^{i,k}$ are valid convex weights, and constraint (3.57e) enforces that the random normal $\boldsymbol{n}^i = \bar{\boldsymbol{n}}^i + \theta^{i,k}\boldsymbol{d}^{i,k}$ must lie in $\mathcal{A}^i$.

Therefore, after solving (3.57), we can compute

$$L_{FC} := \prod_{i=1}^{n_f} \int_{\mathcal{A}^i(\theta,\boldsymbol{d})} p(\boldsymbol{n}^i) d\boldsymbol{n}^i, \tag{3.58}$$

which bounds the probability of force closure.

3.4.4 Integrating Planar Gaussians Over Polytopes

The computation of (3.58) requires integrating a Gaussian probability density over a planar polytope. The following result, adapted from [74, Prop. 1], makes this precise.

Proposition 3.1. *Let $\mathscr{P}$ be a planar polygon with n_v vertices $\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{n_v} \in \mathbb{R}^2$ ordered counterclockwise with $\boldsymbol{y}^{n_v+1} := \boldsymbol{y}^1$. Let $\boldsymbol{Z}$ be a bivariate Gaussian random variable with mean $\boldsymbol{\mu}$ and covariance $\Sigma = \operatorname{diag}(\sigma_1^2, \sigma_2^2)$. Then,*

$$\mathbb{P}[\boldsymbol{Z} \in \mathscr{P}] = \frac{1}{\sigma_2\sqrt{8\pi}} \sum_{m=1}^{n_v} D^m \int_0^1 A^m(r) B^m(r) dr, \tag{3.59}$$

where the terms above are

$$\begin{aligned} D^m &:= y_2^{m+1} - y_2^m, \\ A^m(r) &:= \exp\left(-\frac{1}{2\sigma_2^2}\left[(1-r)y_2^m + r y_2^{m+1} - \mu_2\right]^2\right), \\ B^m(r) &:= \operatorname{erf}\left(\frac{(1-r)y_1^m + r y_1^{m+1} - \mu_1}{\sigma_1\sqrt{2}}\right). \end{aligned}$$

The proof of Proposition 3.1 closely follows the one presented in [74, Proposition 1]. First, we recall Green's Theorem. Note that in the following, we use y_1, y_2 to denote scalar components of the 2D integration variable $\boldsymbol{y}$.

Theorem 3.3 (Green's Theorem)**.** *Let $\mathcal{D}$ be a closed region in the plane with piecewise smooth boundary. Let $P(y_1, y_2)$ and $Q(y_1, y_2)$ be continuously differentiable functions defined on an open set containing $\mathcal{D}$. Then,*

$$\begin{aligned} &\oint_{\partial\mathcal{D}} P(y_1, y_2) dy_1 + Q(y_1, y_2) dy_2 \\ &\quad = \iint_{\mathcal{D}} \left(\frac{\partial Q}{\partial y_1} - \frac{\partial P}{\partial y_2}\right) dy_1 dy_2. \end{aligned} \tag{3.60}$$

Second, we prove a useful intermediate result.

Lemma 3.4. *The following holds:*

$$\begin{aligned} &\int \exp(-(ay^2 + 2by + c)) dy \\ &\quad = \frac{1}{2}\sqrt{\frac{\pi}{a}} \exp\left(\frac{b^2 - ac}{a}\right) \operatorname{erf}\left(\sqrt{a}y + \frac{b}{\sqrt{a}}\right) + \textit{const.} \end{aligned} \tag{3.61}$$

Proof. Recall that

$$\operatorname{erf}(y) = \frac{2}{\sqrt{\pi}} \int_0^y \exp(-t^2) dt. \tag{3.62}$$

We differentiate the RHS of (3.61) with respect to y and by the Fundamental Theorem of Calculus, we have

$$\begin{aligned} &\frac{1}{\sqrt{a}} \exp\left(\frac{b^2 - ac}{a}\right) \cdot \sqrt{a} \exp\left(-\left(\sqrt{a}y + \frac{b}{\sqrt{a}}\right)^2\right) \\ &\quad = \exp\left(\frac{b^2 - ac}{a}\right) \exp\left(-\left(ay^2 + 2by + \frac{b^2}{a}\right)\right) \\ &\quad = \exp(-(ay^2 + 2by + c)), \end{aligned} \tag{3.63}$$

which is the integrand of the LHS, proving the claim. □

We are now ready to prove Proposition 3.1.

Proof of Proposition 3.1. We have that

$$\begin{aligned} &(\boldsymbol{y} - \boldsymbol{\mu})^\top \Sigma^{-1} (\boldsymbol{y} - \boldsymbol{\mu}) \\ &\quad = \frac{1}{\sigma_1^2} y_1^2 - \frac{2}{\sigma_1^2} \mu_1 y_1 + \left[\frac{1}{\sigma_2^2}(y_2 - \mu_2)^2 + \frac{\mu_1^2}{\sigma_1^2}\right]. \end{aligned} \tag{3.64}$$

Letting

$$\begin{aligned} a &= \frac{1}{2\sigma_1^2}, \\ b &= \frac{-\mu_1}{2\sigma_1^2}, \\ c &= \frac{1}{2}\left[\frac{1}{\sigma_2^2}(y_2 - \mu_2)^2 + \frac{\mu_1^2}{\sigma_1^2}\right], \end{aligned} \tag{3.65}$$

and applying Lemma 3.4 to the bivariate Gaussian density function $f(y_1, y_2)$, we have

$$\begin{aligned} &\int f(y_1, y_2) dy_1 \\ &= \frac{1}{2\sigma_2\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{y_2 - \mu_2}{\sigma_2}\right)^2\right) \operatorname{erf}\left(\frac{y_1 - \mu_1}{\sigma_1\sqrt{2}}\right) \\ &\quad + \text{const.} \end{aligned} \tag{3.66}$$

Applying Theorem 3.3, we see that for the choice $P = 0$ and $Q = \int f(y_1, y_2)dy_1$,

$$\begin{aligned}&\iint_{\mathcal{D}} f(y_1, y_2)dy_1dy_2 \\ &= \oint_{\partial\mathcal{D}} \frac{1}{2\sigma_2\sqrt{2\pi}} \exp\left(-\frac{(y_2-\mu_2)^2}{2\sigma_2^2}\right) \operatorname{erf}\left(\frac{y_1-\mu_1}{\sigma_1\sqrt{2}}\right) dy_2.\end{aligned} \tag{3.67}$$

To evaluate the contour integral, we split the contour up into the line segments formed by connecting the n_v extreme points of $\mathcal{D}$ in counterclockwise order. A point $\boldsymbol{y}$ on the m^{th} segment can be expressed

$$\boldsymbol{y} = (1-r)\begin{bmatrix} y_1^m \\ y_2^m \end{bmatrix} + r\begin{bmatrix} y_1^{m+1} \\ y_2^{m+1} \end{bmatrix},\ r \in [0,1]. \tag{3.68}$$

Performing this change of variables and letting $\boldsymbol{y}^{n_v+1} = \boldsymbol{y}^1$,

$$\begin{aligned}&\iint_{\mathcal{D}} f(y_1, y_2)dy_1dy_2 \\ &\quad = \frac{1}{\sigma_2\sqrt{8\pi}} \sum_{m=1}^{n_v} D^m \int_0^1 A^m(r)B^m(r)dr,\end{aligned} \tag{3.69}$$

where

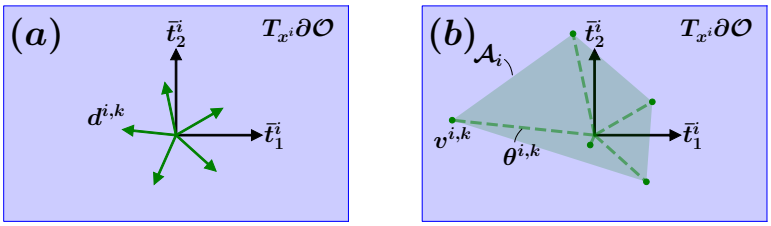
$$\begin{aligned} D^m &:= y_2^{m+1} - y_2^m, \\ A^m(r) &:= \exp\left(-\frac{1}{2\sigma_2^2}\left[(1-r)y_2^m + ry_2^{m+1} - \mu_2\right]^2\right), \\ B^m(r) &:= \operatorname{erf}\left(\frac{(1-r)y_1^m + ry_1^{m+1} - \mu_1}{\sigma_1\sqrt{2}}\right). \end{aligned}$$

Finally, noting that $\mathbb{P}[\boldsymbol{Z} \in \mathcal{D}] = \iint_{\mathcal{D}} f(y_1, y_2)dy_1dy_2$ completes the proof. □

3.4.5 Efficient Computation of L_{FC}

3.4.5.1 Efficiently Solving Batches of Vertex LPs

Program (3.57) suggests solving a batch of $n_f \cdot n_v$ linear programs in parallel to compute the scaling values $\theta^{i,k} \in \mathbb{R}$. In fact, since the constraints of (3.57) decompose across the friction cone index j (the only coupling is through the shared scalar $\theta^{i,k}$), we can further parallelize the computation. Concretely, for a fixed index



Fixed search directions $d^{i,k}$. **Points $v^{i,k}$ and convex hull $\mathcal{A}_i$.**

Figure 3.4: Visualization of the vertex LP construction. For each finger i and search direction $\boldsymbol{d}^{i,k}$, the LP (3.57) computes the farthest vertex $\boldsymbol{v}^{i,k} = \theta^{i,k}\boldsymbol{d}^{i,k}$ in the tangent plane satisfying the containment condition (3.54). The convex hull of these vertices forms the polytope $\mathcal{A}^i$ over which the Gaussian density is integrated.

triple (i, j, k), let $\theta_j^{i,k}$ be the optimal value of the smaller LP

$$\begin{aligned}
\underset{\theta_j^{i,k}\in\mathbb{R},\, \boldsymbol{\alpha}_j^{i,k}\in\mathbb{R}^{n_w}}{\text{maximize}} \quad & \theta_j^{i,k} && \text{(3.70a)}\\
\text{subject to} \quad & \theta^{i,k} \geq 0 && \text{(3.70b)}\\
& \boldsymbol{\alpha}_j^{i,k} \geq \mathbf{0} && \text{(3.70c)}\\
& \mathbb{1}^\top \boldsymbol{\alpha}_j^{i,k} = 1 && \text{(3.70d)}\\
& \left(\theta^{i,k}\boldsymbol{d}^{i,k}\right) T_j^i = -\overline{\mathbf{W}}\boldsymbol{\alpha}_j^{i,k}. && \text{(3.70e)}
\end{aligned}$$

Then, by the separability of the constraints, the original optimum satisfies

$$\theta^{i,k} = \min_{j=1,\ldots,n_s} \theta_j^{i,k}, \tag{3.71}$$

which allows us to solve $n_f \cdot n_v \cdot n_s$ independent (and smaller) LPs rather than $n_f \cdot n_v$ coupled ones.

To actually solve many LPs in a batched manner, we use a custom port of the `quantecon` implementation of the simplex method.[2] Because this implementation is in JAX [75], it can be run on both CPU or GPU without any additional modification. Due to certain parts of the computation stack being CPU-bound, we choose to compute the bound entirely on CPU.

3.4.5.2 Differentiating L_{FC}

In order to maximize the bound in (3.58) in a gradient-based nonlinear optimization program, we must compute the gradient of L_{FC} with respect to the robot configuration $\boldsymbol{q}$. To accomplish this, we need three major components:

[2] `https://github.com/alberthli/jax_simplex`

- differentiating through the numerical integration scheme used to evaluate the expressions in Proposition 3.1 with respect to the polygon vertices $\boldsymbol{v}^{i,k}$;
- differentiating through the vertex LP in (3.57) (or the more efficient program (3.70)) with respect to the wrench matrix $\mathbf{W}(\boldsymbol{q})$;
- differentiating through the parameters $\mathbf{W}$, $\bar{\boldsymbol{n}}^i$, $\{\bar{\boldsymbol{t}}_1^i, \bar{\boldsymbol{t}}_2^i\}$, and $\{\sigma_1^i, \sigma_1^i\}$ with respect to the configuration $\boldsymbol{q}$.

To differentiate through the numerical integration, we use the open source package `torchquad` [76] designed to differentiate numerical quadrature methods. All subexpressions in Proposition 3.1 can be implemented in JAX, which yields the desired gradient in a straightforward manner.

To differentiate the optimal value of $\theta^{i,k}$ in (3.70) with respect to the robot configuration $\boldsymbol{q}$, we use implicit differentiation of the KKT conditions. The analytical gradient can be computed quickly in a nearly identical way to the one used by FRoGGeR in Chapter 2, since here we also solve a linear program [73, Prop. 1]. For the case of quadratic programs and more general programs, see [57].

We compute the gradients with respect to the uncertainty distribution parameters directly using JAX. However, due to the special structure of (3.70), we compute the gradients of $\mathbf{W}$ with respect to the distribution parameters completely analytically, which in practice leads to a large speedup. Because deriving these gradients is extremely tedious and involves an inordinate amount of algebraic manipulation, we omit these details, though we did test the correctness of our implementation against the outputs of JAX's autodifferentiation.

3.5 Experimental Validation

3.5.1 Toy Examples

We first demonstrate PONG on two toy examples. In both, we let the manipuland O be a sphere of radius 0.05m, and we define (unnormalized) distributions over $\boldsymbol{x} \in \partial O$ as follows:

$$\begin{aligned} &\bar{\boldsymbol{n}}(\boldsymbol{x}) = \boldsymbol{x}/\|\boldsymbol{x}\|_2\,, \\ &\sigma_1^2 = 100x_3^2, \quad \sigma_2^2 = 0.01\exp\left(Y_4^2(\theta(\boldsymbol{x}), \phi(\boldsymbol{x}))\right), \end{aligned} \tag{3.72}$$

where the mean of both is $\bar{\boldsymbol{n}}(\boldsymbol{x})$ and they have isotropic tangent variances defined by σ_1^2 and σ_2^2 respectively.

The first distribution assigns low uncertainty at the equator of the object, symmetrically increasing it towards the poles. The second assigns uncertainties based on the (2,4)-spherical harmonic Y_4^2, where $\theta(\boldsymbol{x}), \phi(\boldsymbol{x})$ are the azimuthal and polar angles of $\boldsymbol{x} \in \partial O$. Visualizations of grasps generated with each are shown in Fig. 3.1 and Fig. 3.3 respectively.

We note the qualitative difference that the uncertainty distribution makes, even though the manipuland's geometry is identical in each case. This demonstrates that accounting for uncertainty in grasp synthesis can yield vastly different solutions: for example, a vision system could report higher uncertainty for regions of the object that are occluded, which in turn could encourage grasping visible regions.

3.5.2 Comparative Analysis on Realistic Meshes

While PONG-synthesized grasps on toy examples showed some promise, to use it on more realistic examples, we need a distribution supported on ∂O. In the absence of a natural choice, we define a *curvature-regularized* one. At $\boldsymbol{x} \in \partial O$, the *shape operator* $\mathfrak{S}_{\boldsymbol{x}} : T_{\boldsymbol{x}}\partial O \to T_{\boldsymbol{x}}\partial O$ is defined as the directional derivative $\mathfrak{S}_{\boldsymbol{x}}(\boldsymbol{v}) := -D_{\boldsymbol{v}}N(\boldsymbol{x})$, where N is the unit normal at $\boldsymbol{x}$. The eigenpairs of $\mathfrak{S}_{\boldsymbol{x}}$, $(\kappa_1, \boldsymbol{v}_1), (\kappa_2, \boldsymbol{v}_2) \in \mathbb{R} \times T_{\boldsymbol{x}}\partial O$, are the *principal curvatures and directions* at $\boldsymbol{x}$ respectively [77]. See Fig. 3.5 for a visualization.

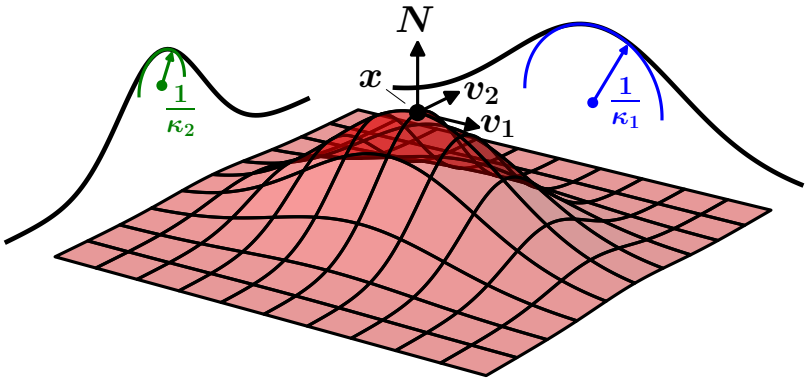


Figure 3.5: The shape operator at a point $\boldsymbol{x} \in \partial O$ encodes the local curvature of the surface. The principal curvatures κ_1, κ_2 and directions $\boldsymbol{v}_1, \boldsymbol{v}_2$ characterize how the surface normal $N(\boldsymbol{x})$ varies along each tangent direction. Regions of high curvature are assigned greater uncertainty in the PONG distribution, since small positional errors induce large changes in the surface normal.

Since $N(\boldsymbol{x}) = \nabla s(\boldsymbol{x})/\|\nabla s(\boldsymbol{x})\|_2$, we have

$$\mathfrak{S}_{\boldsymbol{x}}(\boldsymbol{v}) = -\left(I - \frac{\nabla s(\boldsymbol{x})\nabla s(\boldsymbol{x})^\top}{\|\nabla s(\boldsymbol{x})\|_2^2}\right)\frac{\nabla^2 s(\boldsymbol{x})}{\|\nabla s(\boldsymbol{x})\|_2}\boldsymbol{v}. \tag{3.73}$$

Letting the directions $\boldsymbol{v}_1, \boldsymbol{v}_2$ form our tangent basis at each contact $\boldsymbol{x}^i$ and (some increasing function of) the magnitude of κ_1, κ_2 form our variances, we can define a curvature-sensitive uncertainty distribution with parameters

$$\begin{aligned} \bar{\boldsymbol{n}}^i &:= -\left[\nabla_{\boldsymbol{x}} s(\boldsymbol{x}^i) / \left\|\nabla_{\boldsymbol{x}} s(\boldsymbol{x}^i)\right\|_2\right], \\ \bar{\boldsymbol{t}}_m^i &:= \boldsymbol{v}_m, \ m \in \{1,2\}, \\ \left(\sigma_m^i\right)^2 &:= \log\left(K_{\text{curv}} \cdot |\kappa_m| + h\right), \ m \in \{1,2\}, \end{aligned} \tag{3.74}$$

where $K_{\text{curv}} > 0$ relates curvature to uncertainty and $h > 0$ captures prior uncertainty at $\boldsymbol{x}$. Intuitively, we assign greater uncertainty to high curvatures, since slight errors in position will lead to large changes in the surface normal.

Since ℓ^* captures intrinsic uncertainty, we study whether it performs comparably to PONG even though it only provides conservative guarantees on uncertainty-awareness per Corollary 3.1 without reasoning about the bound (3.46), and find that it does. We synthesize 2000 grasps on 5 objects (a camera, teacup, rotated Rubik's cube, tape dispenser, and teddy bear) and in simulation study pick success as a function of metric.

All 2000 grasps are synthesized using the FRoGGeR program from Chapter 2 but with objective $L_{FC}(\boldsymbol{q})$ and evaluated on ℓ^* and L_{FC}. We observe two key trends (see Fig. 3.6). Based on the theory of Sec. 3.3, we expect fewer failures as each metric rises, which we observe. Further, because each metric is conservative, we expect the ratio of successes to failures to rise with the metric, which is again confirmed. This provides evidence that the min-weight metric is empirically probabilistically sensitive to certain uncertainties in object geometry.

3.6 Conclusion

This chapter completes the theoretical arc of Part I by analyzing the approximation constructed in Chapter 2. Theorem 3.1 establishes a geometric sufficient condition for intrinsic robustness, showing that a nominally force-closure grasp remains force-closure as long as perturbations to each basis wrench lie within the nominal convex hull. The min-weight bound (Theorem 3.2) then provides formal justification for using ℓ^* as a fast proxy for ϵ: maximizing the LP relaxation provably improves a worst-case lower bound on ϵ, and hence on intrinsic robustness, which matches the hardware experiments in Chapter 2 confirming that ℓ^*-synthesized grasps are empirically robust even under significant perception noise. Building on these guarantees, the PONG framework extends the theory into a probabilistic setting, enabling

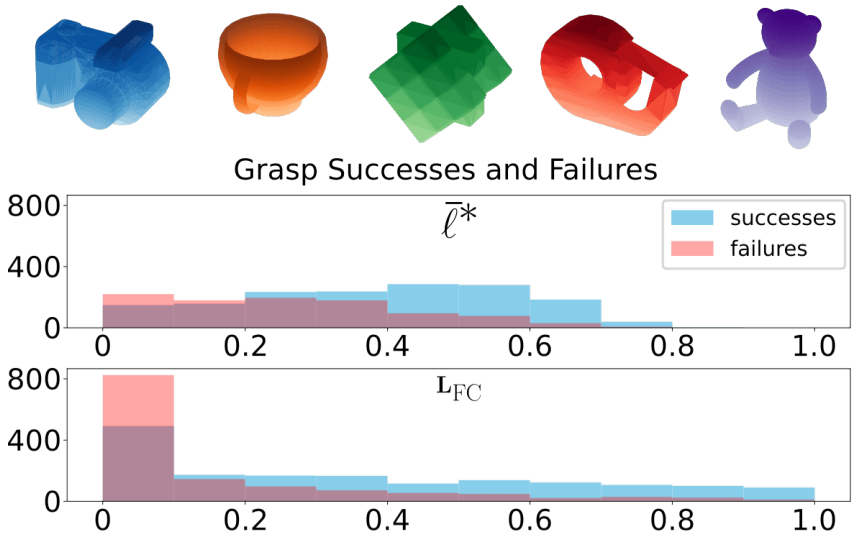


Figure 3.6: **Top.** 5 simulated objects. **Bottom.** Success/failure histograms on 2000 simulated grasps for $\bar{\ell}^*$ and L_{FC}.

grasp synthesis that explicitly accounts for geometric uncertainty by integrating over stochastic surface normal distributions. The simulation studies in this chapter provided evidence that the min-weight bound is not overly conservative in practice and showed that PONG concentrates fingertip placements in low-uncertainty regions, paving the way for incorporating richer stochastic models of object geometry into grasp planners.

Together, Chapters 2 and 3 illustrate the power of working with structured analytic models: the rich geometry of contact mechanics enables both the construction of a tractable approximation and a rigorous characterization of what is lost. However, the analytic approach studied in this part is inherently limited to quasi-static reasoning about isolated grasps. Real-world manipulation involves dynamic contact, imperfect actuation, and complex interactions that resist closed-form analysis. The next part of this thesis shifts from constructing and analyzing approximations to *compensating for* the errors of an existing one: a physics simulator. Rather than deriving models from first principles, we use simulation as a black-box approximate model and develop algorithms that are robust to the inevitable gap between simulated and real-world physics. In exchange, physics simulation allows us to consider a larger class of contact-rich manipulation tasks, as well as a way to scale up the insights of this part of the thesis.

Part II

Physics Simulators as Models for Contact-rich Manipulation

Chapter 4

ROBUST REAL-WORLD GRASPING VIA SIMULATION AT SCALE

The preceding chapters developed analytic grasp metrics and showed that structured approximations (i.e., the min-weight LP relaxation, the PONG probabilistic framework, etc.) can enable fast, provably robust grasp synthesis. However, in the real world, object geometry is uncertain, contact dynamics are complex, and the gap between model and reality is often the dominant source of failure. The previously-studied methods reason only about quasi-static contact mechanics using a fairly coarse model of contact, assumptions on the model which may be too strict for more general contact-rich tasks.

Part II studies a fundamentally different approach to modeling: rather than constructing approximate models derived directly from analytic mechanical models, we use physics simulators to model more complex physical interactions over large-scale simulations. We can leverage simulated priors effectively in both *offline* settings by collecting large-scale datasets of simulated physical interactions, and in *online* settings using model predictive control to achieve highly contact-rich manipulation with neither explicit reliance on analytic models nor pre-trained policies.

While simulators like Isaac Gym [17] and MuJoCo [78] internally operate simply by integrating Lagrangian mechanical models under the hood, they also enable a few key properties. First, simulation frameworks may come with semi-realistic visual rendering, which allows us to collect *paired* vision-state data (e.g., RGB images that correspond to a specific robot configuration in its environment). Second, they typically expose an interface for tuning the "hardness" of the physics model, which allows the contact approximation to be deliberately coarsened when numerical stability and speed matter more than fidelity. This flexibility also underlies *domain randomization* [79], the prevailing strategy for compensating for the *simulation-to-reality* (sim-to-real) gap by randomizing physical parameters during offline policy training. The methods in this part confront the same gap by different means, through scale in the offline setting and through feedback in the online one. Lastly, while Part I reasoned about grasping quasi-statically, simulation allows us to supervise on the full *dynamic* outcome of an interaction, and later in this part, to plan through

those dynamics online.

4.1 Learning Grasp Evaluators from Physics Simulation

To put Part II in context of Part I, we reconsider the problem of grasp synthesis from a different perspective. Where Part I constructed grasp quality metrics by carefully approximating the mechanics of contact, here we *learn* an evaluator directly by supervision from the *outcome* (e.g., success or failure) of robot-environment contact using a physics simulator as the approximate model. We give up closed-form structure and the guarantees it affords in exchange for a model that implicitly captures richer physics (friction, deformation, multi-body dynamics). This comes at the cost of requiring large-scale data and empirical—rather than analytic—validation.

In light of this viewpoint, we can recast the FRoGGeR method introduced in Part I is as a *coarse-to-fine* approach for grasp synthesis, in which some generative *sampler* proposes grasp candidates (e.g., the "heuristic" sampler introduced in 2.4.3) and a discriminative *evaluator* refines them using some grasp quality metric (e.g., the FRoGGeR program in 2.4. See Table 4.1 for other examples from the literature).

Data-driven evaluators [71, 72, 80] offer an alternative to both the analytic evaluators of Chapter 2 and the probabilistic modeling of Chapter 3: by conditioning on realistic perceptual data, they avoid the need for ground-truth object models entirely. Such evaluators have proven effective for parallel-jaw grasping [81, 44, 82, 83]. However, their application to multi-finger grasping has seen considerably less success, leading many works to instead study purely generative approaches [47, 84, 85, 86, 87]. This chapter asks: can the evaluator-based approach, which succeeded for parallel-jaw grasping, also work for dexterous hands if given the right data?

We argue that the answer is yes, provided the training data satisfies three key criteria: (i) positive *and negative* grasp examples with high-quality labels, (ii) corresponding perceptual data (e.g., point clouds) for training measurement-conditioned models, and (iii) sufficient scale to generalize to the real world. For parallel-jaw grasping, the seminal work DexNet 2.0 [44] demonstrated that these conditions could produce high-performance grasping methods by training on over 6.7M simulated grasps with top-down images and grasp quality labels. For multi-finger grasping, however, no existing dataset satisfies all three criteria: recent large datasets were designed for generative models and typically lack either negative examples or perceptual data (Table 4.2), while prior evaluator-based works use small, closed-source datasets of tens of thousands of grasps (see the historical trend in Fig. 4.15 of Section 4.5.2).

Table 4.1: **Comparison of dexterous grasp synthesis methods.** SDF = Signed Distance Field, BPS = Basis Point Set, CVAE = Conditional Variational Autoencoder, EBM = Energy-Based Model.

Method	Sampler	Evaluator	Obj. Data
GraspIt! [35]	Heuristic	Analytic (Ferrari-Canny)	Shape primitives
FRoGGeR [73]	Heuristic	Analytic (Min-weight)	Mesh, SDF
Wu et al. [55]	Learned (CVAE)	Analytic (Force closure)	Point cloud
DexGraspNet [47]	Heuristic	Analytic (Force closure proxy)	Mesh
Lu et al. [71]	Heuristic	Learned (CNN)	RGB-D
FFHNet [88]	Learned (CVAE)	Learned (Classifier)	BPS
DexDiffuser [89]	Learned (Diffusion)	Learned (Classifier)	BPS
UniDexGrasp [84]	Learned (EBM)	Learned (RL)	Point cloud
Ours	Any	Learned (Success prob.)	BPS, NeRF

We test our hypothesis by training multiple grasp evaluators on a large dataset of purely simulated grasps and perceptual data, then deploying them zero-shot on real hardware over a diverse set of objects. Our experiments show that evaluator-based synthesis methods provide a clear performance improvement over purely generative methods [84, 88, 89, 55], that these gains are uniquely attributable to the evaluators, and that dataset scale is crucial for real-world performance. In summary, the contributions of this chapter are as follows.

An open-source, large-scale dataset. Our dataset consists of 3.5M grasps (for each of the Allegro and LEAP [90] hands) on 4.3K unique objects at multiple scales, which at the time this research was conducted, was larger (per-hand) than all existing multi-finger grasp datasets [47, 86, 91]. We release synthetic images of these objects from multiple views, corresponding point clouds, and NeRFs trained over these images. We include over 250 successful and failed grasps for each object and soft labels indicating quantities such as the probability of grasp success (PGS).

Comprehensive simulated and real-world evaluations. We test our evaluators trained on our dataset on thousands of simulated and hundreds of real-world grasps on the Allegro hand, demonstrating state-of-the-art performance. We show that grasp evaluators provide a clear, uniquely attributable performance improvement over evaluator-free methods. Further, these results are the first to quantify the positive effect that dataset scale has on grasp success rates in the real world.

4.2 Preliminaries

The notation in this part is self-contained and does not carry over the symbols of Part I. For example, O here denotes a learned object *representation* (rather than the object set of Part I) and $\mathcal{G}$ denotes a grasp (rather than a grasp map). As elsewhere in this thesis, bold lowercase symbols denote vectors (e.g., the configuration $\boldsymbol{q}$) and bold uppercase symbols denote matrices and transforms (e.g., the pose $\mathbf{T}$).

4.2.1 Grasp Parameterization

Let $\{O\}$ be the object frame with its origin at the centroid of a given object and let $\{H\}$ be the hand frame with its origin at the base of the middle finger. Let n_j and n_f denote the number of hand joints and fingers, respectively. We parameterize a grasp by its pregrasp pose and its subsequent grasping fingertip trajectories. Concretely, we define a grasp as $\mathcal{G} = (\mathbf{T}_{OH}, \boldsymbol{q}, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n_f})$, where $\mathbf{T}_{OH} \in \mathrm{SE}(3)$ is the pre-grasp wrist pose in the $\{O\}$ frame, $\boldsymbol{q} \in \mathbb{R}^{n_j}$ is the pre-grasp joint configuration of the hand, and each $\boldsymbol{d}_i \in S^2$ is the direction in the $\{O\}$ frame that the i^{th} fingertip moves towards to execute the grasp. This representation is object-agnostic, and is easy to execute on real hardware by finding a collision-free trajectory to the pre-grasp pose, using inverse kinematics to compute target joint angles to achieve the desired grasp fingertip directions, and using a PD controller to move the fingertips along this trajectory. This grasp parameterization does not rely on contact points from a ground-truth object model.

4.2.2 NeRF Probability Density Field

Neural Radiance Fields (NeRFs) [31] are trained on a set of posed images $\{(\mathbf{I}_k, \mathbf{T}_k)\}$, where $\mathbf{I}_k \in \mathbb{R}^{W \times H \times 3}$ is the image and $\mathbf{T}_k \in \mathrm{SE}(3)$ is its associated camera pose, to model a specific scene as a 6D, vector-valued continuous function that maps a position $\boldsymbol{p} \in \mathbb{R}^3$ and a normalized viewing direction $\boldsymbol{d} \in \mathbb{R}^3$ to a differential density $\sigma \in \mathbb{R}$ and an emitted color $r, g, b \in \mathbb{R}$. This is implemented as a density network $\sigma(\boldsymbol{p})$, which is only a function of position, and a color network $c(\boldsymbol{p}, \boldsymbol{d})$, which is a function of both position and viewing direction.

4.3 Related Work

Chapters 2 and 3 studied *analytic* grasp synthesis methods that optimize metrics derived from contact mechanics [92, 35, 93, 55, 73, 94]. As discussed there, these methods require accurate object geometry and are sensitive to perception noise [95, 96]. For a broader review, see [97, 23].

When generating grasps in the real world, grasp planners must use *real-world object representations* like RGB images, depth images, or point clouds. Such representations are ultimately approximations of the true object geometry, and it is natural to ask when one would prefer to use one over the other.

For example, a recent promising representation is the basis point set (BPS) [98], a fixed-length feature vector containing the minimal distances to a fixed set of basis points. BPSs are attrative because the length of the object representation is fixed, which simplifies downstream model training on these representations. Yet another popular representation is the Neural Radiance Field (NeRF) [31], which is used for novel view synthesis given an initial dataset of view images. NeRFs have proven useful for grasp generation, representing affordances, and semantic grasping [99, 100, 101, 102, 103, 104, 105].

All of these representations (and others), have thus far only been studied for parallel-jaw grippers, which have limited use for complex manipulation tasks. One reason for this emphasis is that parallel-jaw grasps can be parameterized with a single element of SE(3), and it is clear that the relevant NeRF features lie between the jaws. In contrast, the parameterization of multi-finger grasps is not as straightforward. We address this challenge below.

Data-driven grasp synthesis methods, as extensively reviewed in [48, 95], leverage grasp datasets to train grasp planners that take real-world perceptual information as input. Many works have used simulation to generate large parallel-jaw grasping datasets [46, 44], which have shown to be highly effective [106, 44]. However, generating large-scale datasets for multi-fingered hands is much more complex due to the high number of degrees of freedom and ambiguities in grasp parameterization. Despite this, recent works have released large-scale dexterous grasping datasets, as shown in Table 4.2, such as MultiDex [91], DexGraspNet [47], and Grasp'D-1M [86]. One key limitation of these datasets is that they only contain positive examples, so they cannot be used to train evaluator models that can distinguish between successful and unsuccessful grasps. One counterexample is MultiGripperGrasp [107], which uses GraspIt! [35] to generate grasps that are then evaluated in parallel simulation and provides a ranking of grasp goodness. However, it is currently limited to a small set of 345 objects, does not contain visual observations, and does not account for table collisions, which are important for most grasping applications. Many methods use a combination of sampling and evaluator models, such as Mayer et al. [88] and Weng et al. [89], who propose learning both to sample and evaluate multi-

Table 4.2: **Dexterous grasp dataset comparison.** PGS = probability of grasp success.
(*) denotes total number of grasps divided by number of hands in the dataset.

Dataset	Grasps	Labels	Objects	Code + Data	Observations	Has Table
MultiDex [91]	~87k*	+ve only	58	Yes	-	Possible
Grasp'D-1M [86]	~333k*	+ve only	2.3k	No	Visual	Yes
DexGraspNet [47]	1.32M	+ve only	5.3k	Yes	-	No
MultiGripperGrasp [107]	~2.8M*	Ranked	345	Yes	-	No
Get a Grip (Ours)	3.5M	PGS	4.3k	Yes	Visual	Yes

fingered grasps using basis point sets as input. However, they needed to generate their own dataset of 180K grasps to train these models. Another line of works focuses on learning differentiable grasp evaluators that can optimize grasps through backpropagation [54, 108, 71, 109]. These works also created their own small datasets on the order of 10k grasps across about 100 objects. The dataset introduced in this chapter seeks to aid future investigations of evaluator-based grasping by mitigating the substantial effort required to generate a new one.

4.4 Dataset Construction and Grasp Synthesis

4.4.1 Dataset Generation

Our dataset contains objects with associated grasps, RGB images, point clouds, NeRFs, and labels (illustrated in Figure 4.1). Each precision grasp $\mathcal{G} = (\mathbf{T}_{OH}, \boldsymbol{q}, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{n_f})$ is stored together with three smooth labels as a tuple $\left(\mathcal{G}^k, y^k_{\text{coll}}, y^k_{\text{pick}}, y^k_{\text{PGS}}\right)$, where each $y^k_* \in [0, 1]$: y_{coll} denotes a grasp with unwanted collisions, y_{pick} that the simulated pick was successful, and y_{PGS} the probability of grasp success, defined as the conjunction of the two.

Our grasp generation pipeline is heavily motivated by Wang et al. [47], but requires key modifications, including (1) placing objects on a table with consistent gravity (instead of floating with different gravity directions) to better match the real world, (2) switching from a binary success label to a continuous success probability label evaluated over multiple trials, (3) modifying evaluation and simulation procedures to improve sim-to-real transfer, and (4) restricting contact points to fingertips. At a high level, we first generate hundreds of grasps on each object from a diverse object set given by DexGraspNet [47], yielding an initial dataset of about 2M unlabeled grasps. We then simulate these grasps in Isaac Gym [17] to compute the success-probability labels. Finally, we augment the dataset by perturbing each of the more-successful grasps several times and re-evaluating them. Because the

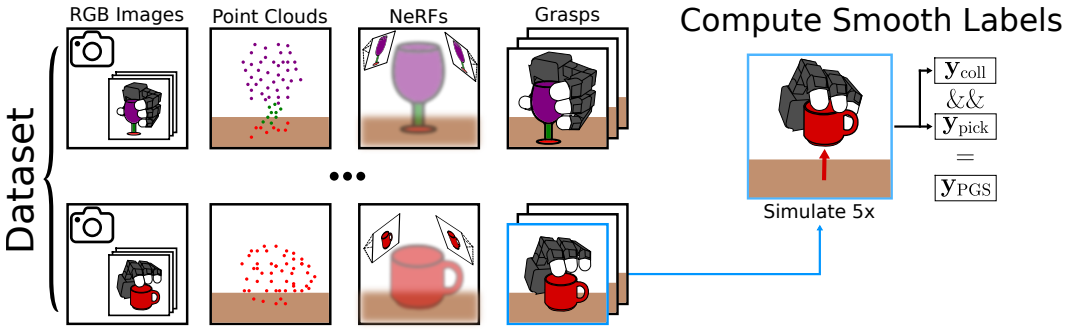


Figure 4.1: **Data and label generation.** Our dataset supplies the components for robust sim-to-real transfer. For each object in the dataset, we provide RGB images of it from several random views, a full point cloud, and trained NeRF weights, which allows our data to be compatible with most vision-based methods. We also supply hundreds of grasps per object, each of which is simulated in Isaac Gym multiple times with slight wrist pose perturbations. Averaging, this yields three smooth regression targets indicating the probability of (1) unwanted collisions, (2) simulated pick success, and (3) grasp success, the logical conjunction of (2) and (3).

initial dataset has many more failed grasps than successful ones, this augmentation yields a more balanced label distribution, and the combined dataset contains over 3.5M examples. We describe each stage in turn below.

Step 1: Grasp Generation. We follow the sampling strategy from Wang et al. [47] by defining an energy function $E(\mathcal{G})$ that is the sum of various penalty terms on a grasp and sampling low-energy grasps from the distribution $p(\mathcal{G}) \propto \exp(-E(\mathcal{G}))$ using Langevin sampling. The samples are initialized by spawning the hand on some inflated level set of the object mesh such that the palm faces the object with a canonical open-hand joint configuration, then perturbing the configuration with noise. Our weighted energy function (dependence on $\mathcal{G}$ omitted) is

$$E = E_{\text{fc}} + w_{\text{dis}}E_{\text{dis}} + w_{\text{joints}}E_{\text{joints}} + w_{\text{pen}}E_{\text{pen}} + w_{\text{spen}}E_{\text{spen}} + w_{\text{tpen}}E_{\text{tpen}},$$

where E_{fc} is the differentiable force closure estimator introduced in [47] that encourages force closure grasps; E_{dis} encourages hand-to-object proximity; E_{joints} penalizes joint violations; E_{pen} and E_{spen} penalize hand-object and self penetration respectively; and E_{tpen} is a new energy term that penalizes hand-table penetration, added to keep the grasps above the table. Without this term, many optimized grasps would converge to grasps that fully cage the object from above and below in a way that would not be possible in the real world due to table collisions.

Compared to the energy function in DexGraspNet [47], we make three changes. First, we sample *precision grasps* rather than power grasps, since this synergizes

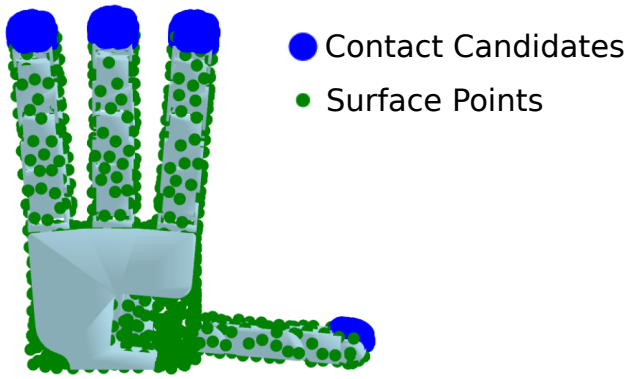


Figure 4.2: **Visualization of the Allegro hand contact candidates and 500 surface points.** Contact candidates are used to compute E_{dis}, encouraging hand-object proximity. Surface points are used to compute E_{tpen}, which penalizes hand-table penetration to keep the grasps above the table.

with our grasp motion parameterization: we specify *contact candidates* only on the fingertips, which act as attractors to the object surface during generation, while the *surface points* used to penalize hand-object and hand-table collisions still cover the whole hand (see Figure 4.2). We sample 500 surface points uniformly from each link's collision geometry, distributed proportionally to mesh surface area, and densely sample 128 contact candidates over the "front" of each Allegro fingertip, parameterized by a spherical section centered on the (spherical) fingertip center with radius 1.2cm, polar sweep ±54°, and azimuthal sweep 54° to 126°. The local fingertip frame has its $+x$ axis pointing "palm out" and its $+z$ axis aligned with the fingers. Second, we recover pre-grasp poses with the fingertips 1.5cm off the surface instead of on it, since the extra clearance aids collision-free motion planning. For execution, we also compute a *post-grasp* configuration with the fingertips 3.5cm below the object surface. Third, we penalize hand-table collisions for kinematic feasibility: letting $\boldsymbol{p}_{\text{table}}(\cdot)$ return the closest point on the table, the table-penetration energy is

$$E_{\text{tpen}} = \sum_{\boldsymbol{p} \in P(\mathcal{H})} \max\Big((\boldsymbol{p} - \boldsymbol{p}_{\text{table}}(\boldsymbol{p})) \cdot \hat{\boldsymbol{g}}, 0 \Big),$$

where $P(\mathcal{H})$ is a set of predefined points on the hand $\mathcal{H}$ and $\hat{\boldsymbol{g}}$ is the gravity vector. Finally, the grasp directions $\{\boldsymbol{d}_i\}$ point from each fingertip to the closest point on the mesh. Specifying them eliminates ambiguity in grasp execution and aids reproducibility. Table 4.3 lists all grasp generation parameters.

Step 2: Grasp Evaluation. We evaluate each generated grasp in Isaac Gym [17].

Parameter	Value
n_c_per_finger	5
w_fc	0.5
w_dis	500
w_pen	300.0
w_spen	100.0
w_joints	1.0
w_ff	3.0
w_fp	0.0
w_tpen	100.0

Parameter	Value
switch_prob	0.5
mu	0.98
step_size	0.005
stepsize_period	50
starting_temp	18
annealing_period	30
tempdecay	0.95

Parameter	Value
jitter	0.1
dist_lower	0.2
dist_upper	0.3
theta_lower	$-\pi$ / 6
theta_upper	π / 6

Table 4.3: **Grasp generation parameters.** Variables names taken from DexGraspNet [47]. Left: Energy function parameters. Middle: Optimization parameters. Right: Initialization parameters.

The object is first spawned and allowed to settle. We then spawn the hand at the pre-grasp pose, assume fingertip i moves 5cm along $\boldsymbol{d}_i$, and compute a corresponding closed-hand configuration $\boldsymbol{q}_{\text{close}}$ via inverse kinematics. We drive $\boldsymbol{q} \rightarrow \boldsymbol{q}_{\text{close}}$ with PD control to execute the grasp, then lift the object 20cm off the table with gravity enabled. We label each grasp by whether (1) the pre-grasp was collision-free and (2) the pick succeeded without the hand-object pose deviating significantly, yielding y_{coll} and y_{pick}. We need both, as Isaac Gym's physics may exploit nonphysical hand-object penetration to achieve a pick, which we forbid. We then set $y_{\text{PGS}} = y_{\text{coll}} \wedge y_{\text{pick}}$ to indicate a true, physically-achievable success. Concretely, a pick is a *simulation success* ($y_{\text{pick}} = 1$) only if (1) the hand contacts the object on at least 3 links, (2) the palm and proximal links do not touch the object, (3) the object's position relative to the hand changes by no more than 10cm throughout the lift, and (4) its orientation changes by no more than 45 degrees about each Euler angle. It is *collision-free* ($y_{\text{coll}} = 1$) only if the hand contacts neither the table nor the object during the pre-grasp pose.

To obtain non-binary labels for regression, we re-simulate each grasp 5 times under small wrist-pose perturbations (translations drawn uniformly from [-5mm, 5mm]

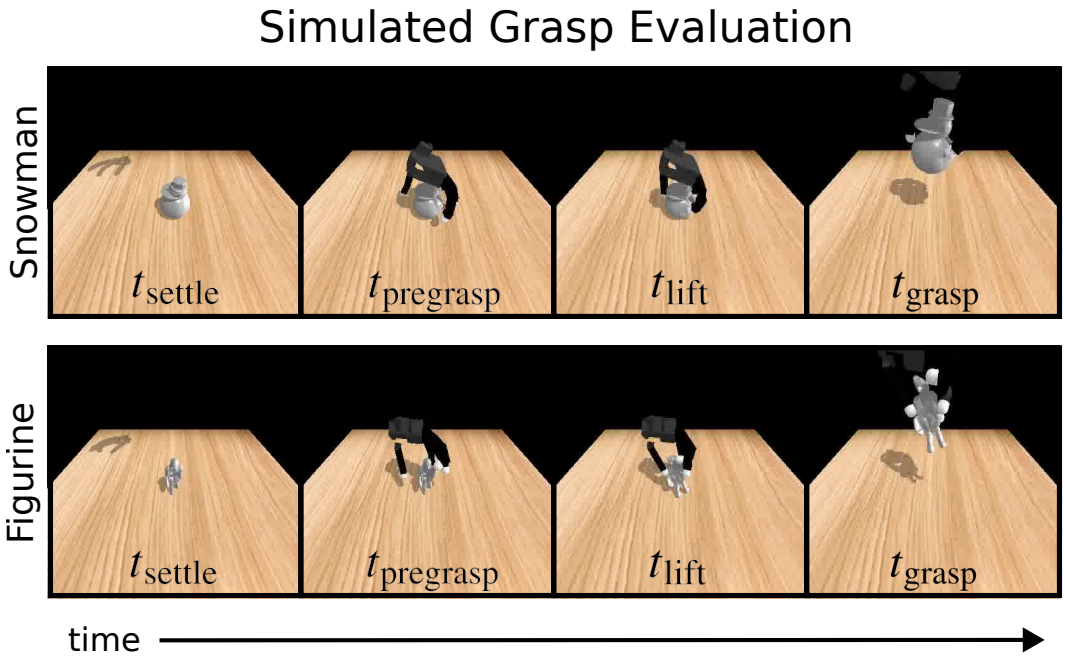


Figure 4.3: **Visualization of grasp evaluation in simulation on a snowman object (top) and a figurine object (bottom).** After the object settles on the table (t_{settle}), the Allegro hand is moved to the pre-grasp position (t_{pregrasp}), executes the grasp (t_{grasp}), and lifts the object (t_{lift}). At t_{pregrasp}, we compute y_{coll} by checking for hand collisions with the object and table. At t_{lift}, we compute y_{pick} by checking if the object pose relative to the hand significantly changed from t_{pregrasp}.

per axis and roll, pitch, and yaw from [-2.5°, 2.5°]) and average the outcomes. The resulting y_{PGS} is supervised to match an empirical success probability rather than a measure of classifier confidence, which has been found to help learn robust grasps [66, 38]. In contrast with [47], we evaluate feasibility in a tabletop setting rather than a floating environment, which improves sim-to-real transfer by accounting for table penetrations, and spawning the hand in a pre-grasp (rather than in-contact) pose eliminates non-physical contact behavior that otherwise destabilizes the labels. For the simulation itself, the timestep is 1/60 seconds while the integrator solves at 1/120 seconds for numerical stability using the Truncated Gauss-Seidel (TGS) solver with 8 position and 8 velocity iterations per step. The contact offset ("force at a distance") is applied from 1mm of separation, substantially lower than the 1cm used by DexGraspNet [47], which reduces unwanted non-physical collisions. Geometries use Isaac Gym's default convex decomposition, and each object is dropped from 2cm above the table and allowed to settle before evaluation begins (the pipeline is visualized in Figure 4.3).

Across the full 3.5M-grasp dataset, the label medians and interquartile ranges are

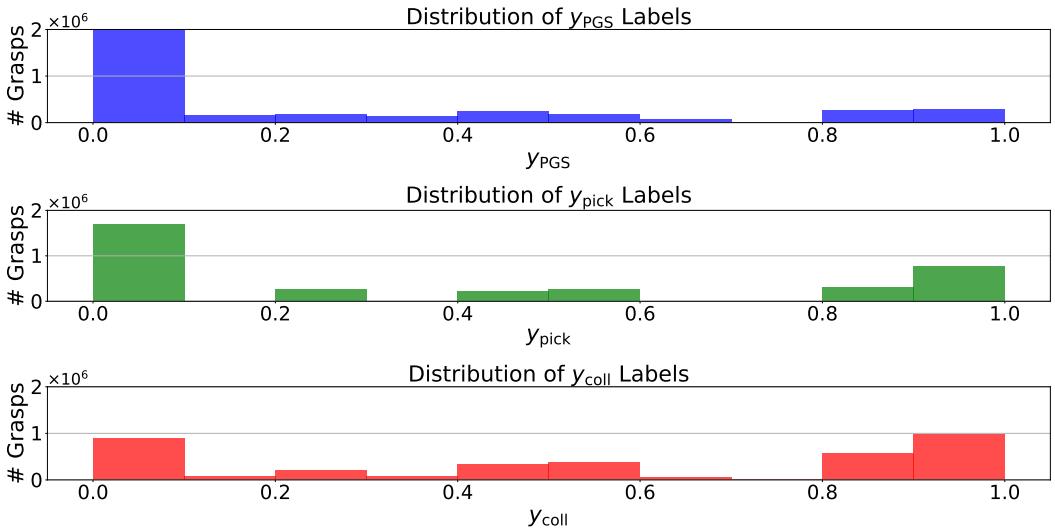


Figure 4.4: **Dataset label distribution across 3.5M grasps.**

y_{PGS}: 0.0 (0.0, 0.48), y_{pick}: 0.2 (0.0, 0.8), and y_{coll}: 0.08 (0.6, 1.0). Figure 4.4 shows the full distributions.

Step 3: Grasp Augmentation. Because only 2-20% of generated grasps per object are successful, we augment the dataset with additional positive examples: we take every nominal grasp satisfying $y_{\text{PGS}} \geq 0.9$, perturb it, re-evaluate it in Isaac Gym, and add the result to the dataset. Unlike the wrist-only label-smoothing perturbations of Step 2, these augmentation perturbations are larger and act on the full grasp (wrist pose, finger joints, and fingertip directions) to promote data diversity. We draw them with Halton sequence sampling [110], which produces quasi-random, *low-discrepancy* sequences that cover the input space more uniformly than standard random sampling, over the following ranges: wrist translation [-5mm, 5mm] per axis; wrist orientation [-2.5°, 2.5°] per roll/pitch/yaw axis; finger joints [-0.05rad, 0.05rad] per angle; and fingertip directions [-10°, 10°] about each of the two axes orthogonal to the nominal direction. Each high-success grasp is perturbed five times on the same object and twice more on *different, randomly sampled* objects. The cross-object perturbations typically fail, but we find they guard against overfitting and provide useful signal about object geometry.

Object dataset and NeRFs. Our objects are a strict subset of DexGraspNet's [47]: of its 5.3K unique objects we keep 4.3K, because our tabletop setting requires a canonical "up" direction, so many objects were manually reoriented. We keep an object only if, after reorientation, it settles when dropped from 2cm. That is, it does not tip over within 200 Isaac Gym timesteps, the real part of its relative quaternion

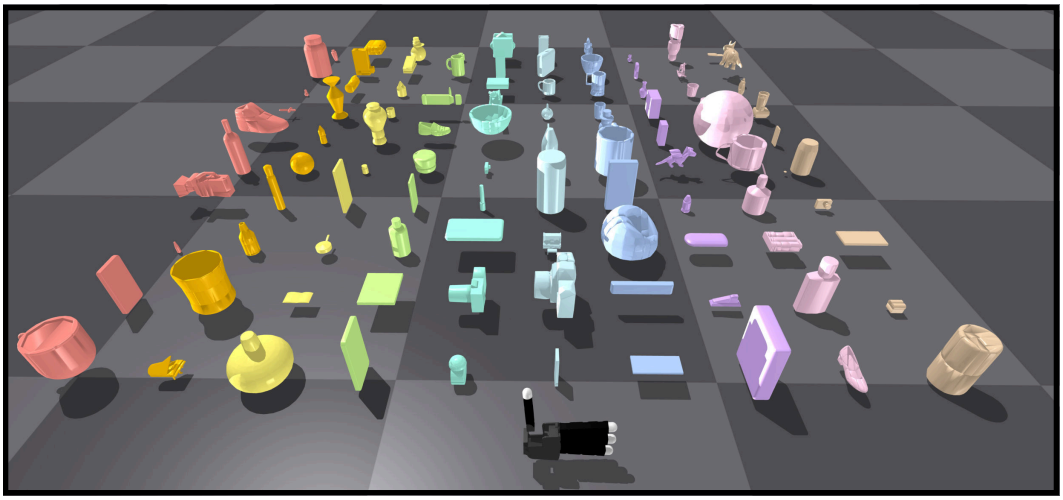


Figure 4.5: **Visualization of a random subset of 100 objects out of the 212 test objects.** The Allegro hand is shown at the bottom for scale. The object dataset is a subset of the DexGraspNet dataset's meshes [47]. See their work for more details.

stays above 0.95 throughout, and its translation stays within 1mm per axis and its roll, pitch, and yaw within 1e-2 radians for 15 consecutive timesteps. All objects use a uniform friction coefficient of 0.9 and a constant density of 500 kg/m^3 (as in [47]), so masses are determined entirely by volume, as in other works [46, 47, 91]. Each object is nominally centered and randomly scaled so its maximum extent lies between 0.1 and 0.3m. To add scale diversity while keeping the dataset under 2.5TB, roughly 1.7K of the objects are duplicated at three distinct scales (with max extents drawn from $U_{\text{small}}(0.05, 0.1)$, $U_{\text{med}}(0.1, 0.2)$, and $U_{\text{large}}(0.2, 0.3)$) while the rest are used at a single scale, yielding about 7.7K objects. The large storage footprint comes from the 3D NeRF density grids used by the NeRF evaluator (Section 4.4.3). For each object we train a NeRF by uniformly sampling 100 images over a spherical cap above it (radius 0.45m, polar angle drawn from $U(0, \frac{\pi}{4})$) and training for 400 iterations with `nerfstudio` [111, 112]. Full NeRF, point-cloud, and mesh extraction details are given in Section 4.4.2. Figure 4.5 visualizes a subset of the objects.

Dataset splits and compute. We split the 4305 unique object meshes into 3981 (92.5%) train, 216 (5%) validation, and 108 (2.5%) test objects, ensuring each mesh appears in only one split. Accounting for multiple scales, this yields 7695 objects (7108 train, 375 validation, 212 test) and 3,531,098 grasps in total (3,261,228/170,128/99,742 across the three splits). Generating the full dataset took roughly 4 days on 4 Nvidia A100 GPUs: about 1 day each for grasp generation and label generation, and 2 days for image generation, NeRF training, and point-cloud

extraction.

4.4.2 Object Representations

The grasp evaluators take in one of three object representations, all derived from the per-object NeRFs trained during dataset construction (Section 4.4.1).

Neural radiance fields. We train NeRFs on both simulated and real-world data. The simulated-data procedure is described in Section 4.4.1. For real-world data, we instead collect 100 images by moving the wrist-mounted camera along a hard-coded trajectory of 3 spirals around the object, then train for 400 iterations with `nerfstudio` [111, 112]. In both cases we train without scene contraction, pose auto-scaling, centering, or reorientation, and use a scale factor of 1.0 so the NeRF's coordinate space matches the input data. All other parameters use the default `nerfacto` settings. Figure 4.6 compares the resulting NeRFs: the two are qualitatively similar, and both exhibit floater artifacts [113] that downstream models must either filter out or tolerate.

Basis point sets. Methods that represent the object as a basis point set (BPS) [98] use the following procedure.

1. Train a NeRF as described above.
2. Sample a point cloud with 5000 points using `nerfstudio` [112] by rendering out depth images with opacity exceeding 0.5 in the axis-aligned bounding box parameterized by the lower and upper bounds $[-0.2m, -0.2m, 0.0m] \times [0.2m, 0.2m, 0.3m]$ using a z-up convention.
3. Preprocess the point cloud to remove most outliers and floaters. First, we use `open3d` [114] to remove statistical outliers (`nb_neighbors=20`, `std_ratio=2.0`) and radius outliers (`nb_points=16`, `radius=0.05`). Next, we construct an undirected graph of the remaining points, where points are nodes and an edge exists between two nodes if the points are within 1mm. Floaters are removed by keeping the largest connected component.
4. Generate a random set of 4096 basis points within a sphere of radius 0.3m, centered 0.15m above the table. This set of basis points remains fixed for all experiments. We utilize `bps` [98] for basis point set operations.
5. Compute basis point set values by calculating the distance from each basis point to its closest point in the point cloud.

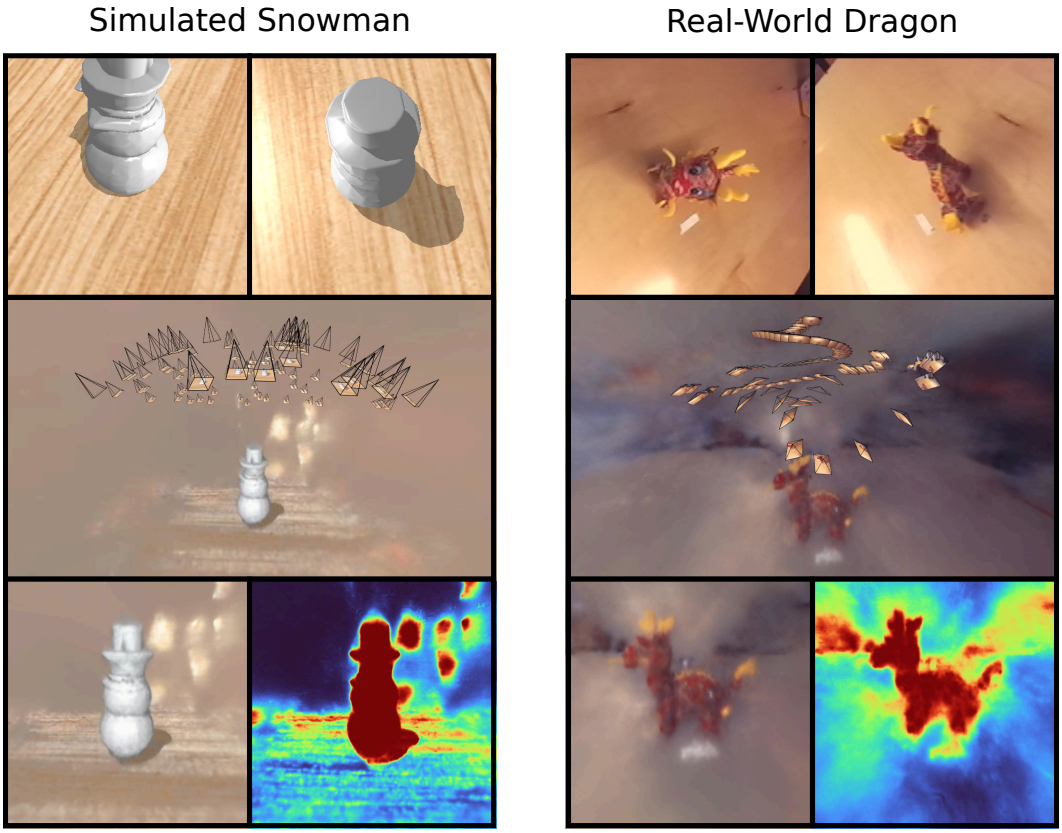


Figure 4.6: **Comparison between NeRFs trained on simulated snowman data and real-world dragon data.** This demonstrates the qualitative similarity between the two NeRFs. First row: two of the RGB images used for NeRF training. Second row: Camera poses used for NeRF training. Third row: NeRF-rendered RGB image and accumulation image.

Figure 4.7 shows an example of a point cloud and BPS for both a simulated and real-world object.

Meshes. We also extract triangle meshes for two purposes: the analytic FRoGGeR baseline, and collision-free motion planning from the start pose to the pre-grasp pose (used by all methods). For both, the mesh is generated by the following procedure.

1. Train a NeRF as described above.
2. Use `scikit-image` [115] to perform marching cubes, which extracts a 2D surface mesh from a 3D volume with a NeRF density level set of 15 within a bounding box of $[-0.2m, -0.2m, 0.0m] \times [0.2m, 0.2m, 0.3m]$ (with z being the up-direction).

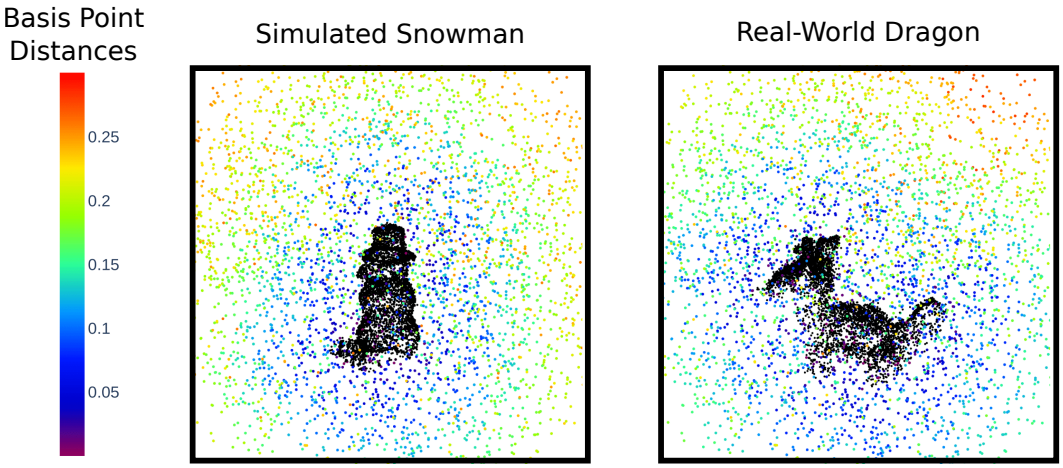


Figure 4.7: **Visualization of a point cloud (5000 black points) and a basis point set (4096 points colored by distance to the point cloud).** These are generated by a NeRF for both a simulated snowman object (left) and a real-world dragon object (right).

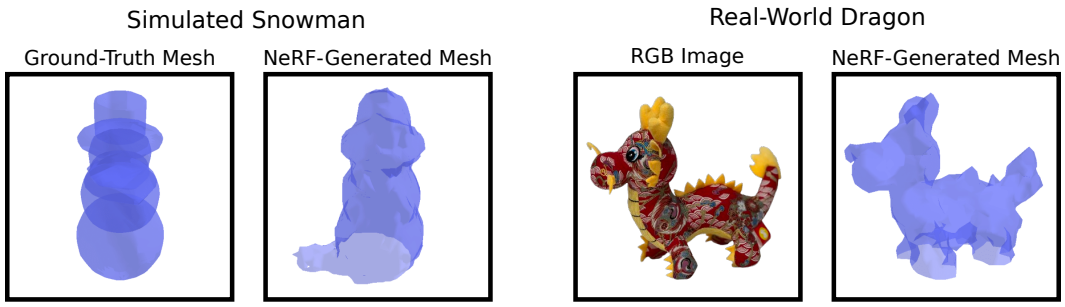


Figure 4.8: **We use NeRFs to generate meshes for two purposes: analytic grasp planning methods (FRoGGeR) and collision-free motion planning from the start pose to the pre-grasp pose (all methods).** Left: Comparison between the ground-truth mesh and NeRF-generated mesh of a simulated snowman object, where the extra vertices at the bottom are due to artifacts from shadows/lighting effects. Right: Comparison between an RGB image and the NeRF-generated mesh of a real-world dragon object. We do not have a ground-truth mesh for the real-world dragon object.

3. Use `trimesh` [116] to remove floaters by keeping only connected components with at least 31 edges.

Figure 4.8 shows examples of NeRF-generated meshes. We found these parameters to be reasonable for all test objects.

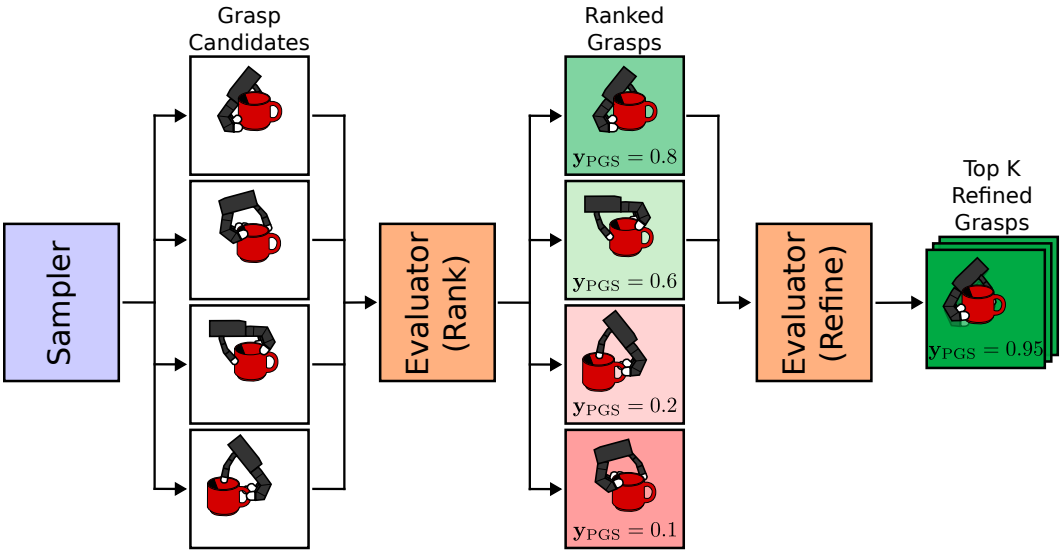


Figure 4.9: **Our grasp planners consist of a *sampler* and an *evaluator*.** First, the sampler generates a batch of grasp candidates. The evaluator ranks them, and the top K grasps are refined using sampling-based optimization, where the objective is given by the evaluator.

4.4.3 Grasp Synthesis with Learned Evaluators

Let the choice of object representation, e.g., a point cloud, be denoted O. The grasp evaluator is then a neural network $\hat{y}_{\text{coll}}, \hat{y}_{\text{pick}}, \hat{y}_{\text{PGS}} = f_\varphi(\mathcal{G}, O)$ trained by minimizing an equally-weighted L2 loss over all three labels. At inference time, grasp synthesis proceeds in three steps: (1) a large batch of candidate grasps $\mathcal{T}$ is drawn from a sampler; (2) an initial evaluation culls all but the top K grasps by predicted success $\hat{y}_{\text{PGS}}$ into a smaller set $\mathcal{S} \subseteq \mathcal{T}$ (we use $K = 5$ in simulation, where all are refined and executed, and $K = 40$ on hardware, where all are refined and only the best is executed); and (3) we iteratively refine every grasp in $\mathcal{S}$ and finally return $\mathcal{G}^* = \arg\max_{\mathcal{G}\in\mathcal{S}} \hat{y}_{\text{PGS}}(\mathcal{G})$. Figure 4.9 illustrates this sample-and-refine process.

Refinement is sampling-based: at iterate i we draw a noisy perturbation $\Delta\mathcal{G}^{(i)}$ and accept it, setting $\mathcal{G}^{(i+1)} = \mathcal{G}^{(i)} + \Delta\mathcal{G}^{(i)}$, only if it improves the predicted success, $\hat{y}_{\text{PGS}}(\mathcal{G}^{(i)} + \Delta\mathcal{G}^{(i)}) > \hat{y}_{\text{PGS}}(\mathcal{G}^{(i)})$. Otherwise, $\mathcal{G}^{(i+1)} = \mathcal{G}^{(i)}$. The perturbations are zero-mean Gaussian, with standard deviations of 5mm per wrist-translation coordinate, 0.05 in $\mathfrak{so}(3)$ for the wrist and grasp orientations (mapped to SO(3) via the exponential map), and 0.01 radians per joint angle. We run 50 refinement iterations. Other refinement strategies, such as gradient-based or alternative sampling-based methods, could also be used. We do not maximize $\hat{y}_{\text{coll}}$ or $\hat{y}_{\text{pick}}$ at inference, since they are regressed only to provide an additional signal at train time.

To feed grasps to the networks as fixed-length vectors, we represent the grasp $\mathcal{G}$ in flattened form as $\boldsymbol{g} = (\boldsymbol{p}, \boldsymbol{r}, \boldsymbol{q}, \boldsymbol{d}_1, ..., \boldsymbol{d}_{n_f}) \in \mathbb{R}^{9+n_j+3n_f}$, where $\boldsymbol{p} \in \mathbb{R}^3$ is the wrist position, $\boldsymbol{r} \in \mathbb{R}^6$ the wrist orientation represented by a continuous 6-D rotation vector [117], $\boldsymbol{q} \in \mathbb{R}^{n_j}$ the pre-grasp joint configuration, and $\boldsymbol{d}_i \in \mathbb{R}^3$ the direction in which the i^{th} fingertip moves during the grasp.

We study object representations O that can be computed with an eye-in-hand camera, which we move along a spiral trajectory to obtain diverse views and high-quality reconstructions (see the right side of Figure 4.6). Using such favorable visual features lets us isolate the contribution of the evaluator and dataset from low-quality real-world measurements. The two representations we use, basis point sets and NeRFs (Section 4.4.2), have both proven promising for parallel-jaw grasping, giving us two distinct evaluators to compare. We sample and clean a point cloud from each NeRF for use in training and evaluation, and describe the samplers and evaluators in detail below.

Diffusion sampler. Figure 4.10 shows the diffusion sampler architecture, inspired by Weng et al. [89]. As their implementation was not publicly available at the time of publication, we re-implemented their architecture to the best of our ability from the textual description. We use an embedding dimension of 128 and a sequence length of 4 for all attention modules, where the sequence length is created by reshaping the outputs of the fully-connected layers to $(4, 128)$. We use Denoising Diffusion Implicit Models (DDIM) [118] with a linear scheduler on the noise variance β that moves from 0.0001 to 0.02 over 1000 diffusion timesteps (Figure 4.11 shows the train and validation loss curves). Although the sampler is trained only on successful grasps, it can still produce unsuccessful ones, as also observed in prior work [89, 88]. Its most common failure modes are (1) grasps that are "close" to successful but with fingers slightly too close to or far from the object, and (2) grasps that are substantially too far from the object.

Fixed-dataset sampler. To disentangle the importance of the learned evaluator from that of the diffusion sampler, we also use a simple baseline that stores a fixed set of 4349 grasps from the training set: one high-success grasp ($y_{\text{PGS}} \geq 0.9$) per training object, with none stored for objects lacking one. Because these grasps come from training objects, they are not tailored to the unseen test objects used in simulation or the real-world objects used on hardware.

BPS evaluator. Figure 4.12 shows the architecture used for the BPS evaluator. We used the official implementation from Mayer et al. [88], which is the same

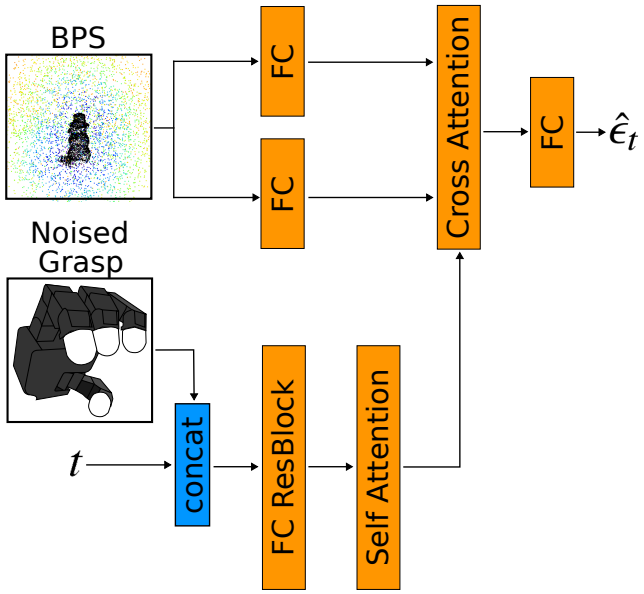


Figure 4.10: **Diffusion Sampler architecture.** Our model takes in a basis point set, a noised grasp, and a diffusion timestep. The noised grasp and the diffusion timestep are processed into a query using a self-attention block. The basis point set is processed into a key-value pair. These are embedded together using a cross-attention block and used to compute the predicted noise. This is a similar architecture to the one used by Weng et al. [89].

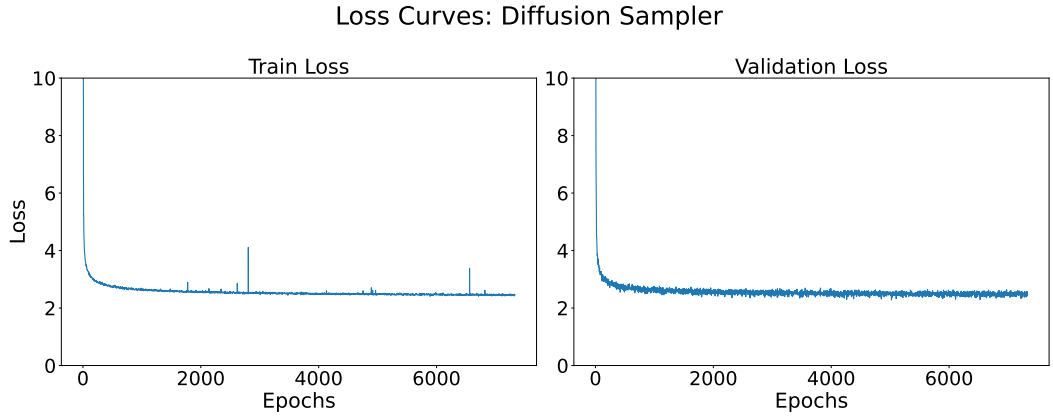


Figure 4.11: **Train and validation loss curves for the diffusion sampler.**

architecture used by Weng et al. [89].

NeRF evaluator. The NeRF evaluator instead uses a grasp representation that leverages NeRF features directly (Figure 4.13). A key factor for grasp success is the surface geometry at the contact points, which is difficult to model from images, as few fast and reliable surface reconstruction methods are adequate for grasp planning. We instead hypothesize that NeRF features capture accurate estimates of the object

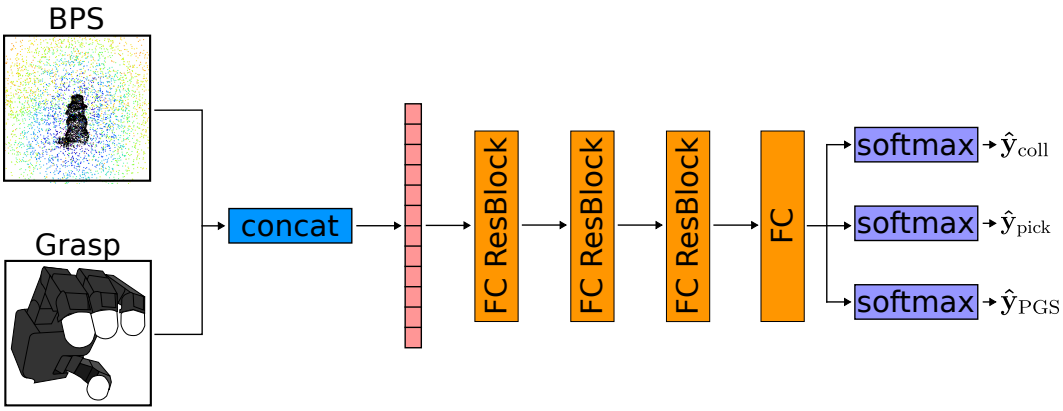


Figure 4.12: **BPS Evaluator architecture.** Our model takes in a basis point set and a grasp, which are concatenated together and then passed through three fully-connected resblocks and a final fully-connected layer, which returns logits for 3 labels: whether (i) there are unwanted collisions in the scene, (ii) the pick succeeded in the simulator, and (iii) both (i) and (ii) are true. This is the same architecture used by Mayer et al. [88] and Weng et al. [89].

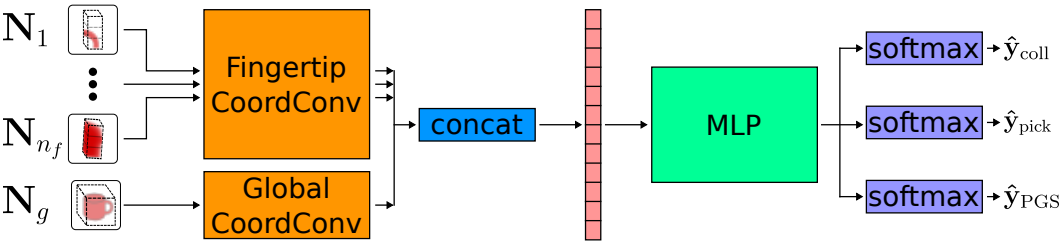


Figure 4.13: **NeRF Evaluator architecture.** Our model takes in local and global NeRF feature grids, passing them through 3D CoordConv networks and an MLP, which returns logits for 3 labels: whether (i) there are unwanted collisions in the scene, (ii) the pick succeeded in the simulator, and (iii) both (i) and (ii) are true. The Fingertip CoordConv weights are shared for all fingertips.

surface. Centered at finger i's position $\boldsymbol{p}_i$, a square of side length 0.06m is swept along $\boldsymbol{d}_i$ by 0.08m to recover a rectangular prism, discretized into a 4D tensor $\mathbf{N}_i \in \mathbb{R}^{4\times31\times31\times41}$ whose first channel is the NeRF density and last three are spatial coordinates. These grid dimensions overapproximate both the fingertip size and the grasp depth so the captured geometry is not too local. To also capture global object geometry, we generate a grid of side length 0.4m centered on the estimated object centroid, $\mathbf{N}_g \in \mathbb{R}^{4\times41\times41\times41}$, and the centroid is estimated by assuming uniform mass density and integrating over spatial regions with NeRF density exceeding 15.0. The local grids approximate the fingertip swept volumes along the grasp directions, while the global grid is a fixed-size summary of the object's overall geometry.

FRoGGeR baseline. For all experiments that use FRoGGeR [73, 94], we use

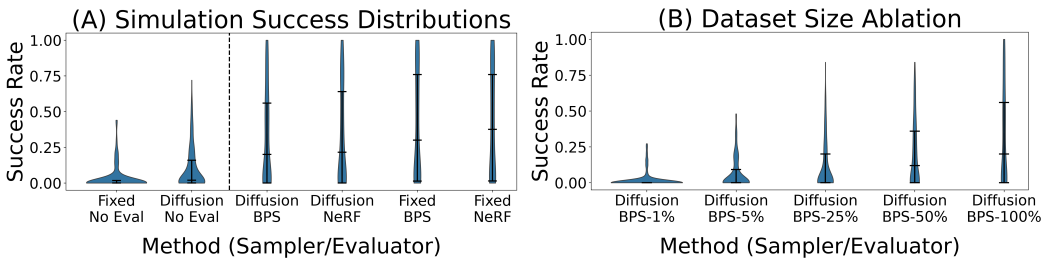


Figure 4.14: **Violin plots showing sim results over hundreds of unseen test objects.** The short lines mark the median and IQR. "Diffusion" vs. "Fixed" refers to sampling initial grasp $\mathcal{G}^{(0)}$ from a diffusion model or a fixed set of grasps. "NeRF" vs. "BPS" refers to whether the object O is given by NeRF or basis point set features. **(A)** Independent of the choice of sampler or object representation, evaluator refinement (right 4 columns) drastically improves planned grasp quality compared to sampling without refinement (left 2 columns). **(B)** Ablations on dataset size used to train the evaluator, which demonstrates a strong correlation between dataset size and grasp planner performance. The percentages indicate the fraction of the dataset used to train the evaluator.

the open-source implementation at `https://github.com/alberthli/frogger`, planning over the meshes extracted in Section 4.4.2. We plan grasps using the full fixed-base hand-arm system so that generated grasps account for the arm's kinematic reachability, resulting in fewer downstream motion planning failures. Of the 100 real-world pick attempts using FRoGGeR, 49 failed: 16 from planned grasps that failed to lift the object, and 33 from planning issues in which no grasp could be found within the timeout.

4.5 Experiments

The central hypothesis of this work is that learned grasp evaluators can achieve robust sim-to-real transfer for multi-finger grasp planners, but only if the associated data are large-scale and account for perceptual data modalities that can be measured in the real world. We now test this hypothesis on the dataset introduced above. We also ablate other modeling choices like object representation or sampler choice to measure the performance increase uniquely attributable to the evaluator. Lastly, we measure the effect of dataset scale on evaluator performance.

4.5.1 Simulation Results

Though our goal is robust grasping on hardware, evaluation in simulation is useful since we can test grasps on a larger set of objects and execute multiple perturbed grasps to probe robustness on identical object poses and perceptual inputs. We

evaluate all methods on a held-out set of 212 objects: each method generates 5 grasps per object, and each grasp is re-simulated 5 times under small wrist translation and rotation noise (25 evaluations per object), which we average into a per-object success rate.

We first study the importance of learned grasp evaluators by varying the sampling strategy and the evaluation method. For sampling, we use either a diffusion model-based sampler (motivated by Weng et al. [89]) or a trivial baseline that stores a fixed set of ~4000 grasps from the training set and simply draws candidate grasps from this fixed pool at inference time (Section 4.4.3). To ensure the diffusion sampler is a strong and fair baseline, we train it exclusively on high-quality grasps with $y_{\text{PGS}} = 1.0$, roughly 300K grasps (8.45% of the training data), about 10 times the amount used to train existing generative models [89, 88]. For evaluator refinement, we use a BPS evaluator, a NeRF evaluator, or no evaluator (which randomly chooses grasps, equivalent to an untrained evaluator).

Figure 4.14A shows that every method using a learned evaluator outperforms the evaluator-free methods by a large margin, *independent of the choice of sampler or object representation*. A key reason is data efficiency: an evaluator can additionally train on the ~3M "bad" grasps that are byproducts of generating the good ones, whereas a generative sampler learns only from positives.

Comparing the two samplers, the fixed and diffusion samplers perform similarly once paired with an evaluator, even though the fixed sampler is significantly worse without one. This is because, although the average diffusion-sampled grasp is far better than the average fixed grasp, the *best* samples from each are of similar quality. Since only the top-ranked candidates are refined and only the best refined grasp is executed, even a fixed sampler that is a reasonable prior usually supplies at least one good initial guess.

4.5.2 Ablation Study on Dataset Size

We next study how much our dataset's scale contributes to performance. Prior works show a clear trend toward larger multi-fingered grasp datasets for training grasp evaluators (Figure 4.15). In fact, no public dexterous grasping dataset larger than ours is suitable for training the diffusion sampler in our experimental setup (at the time of writing). The only large dataset with visual observations is closed-source [86]. Moreover, existing datasets do not parameterize the grasp execution motion for reproducibility, rarely consider a tabletop setting, and often contain grasps

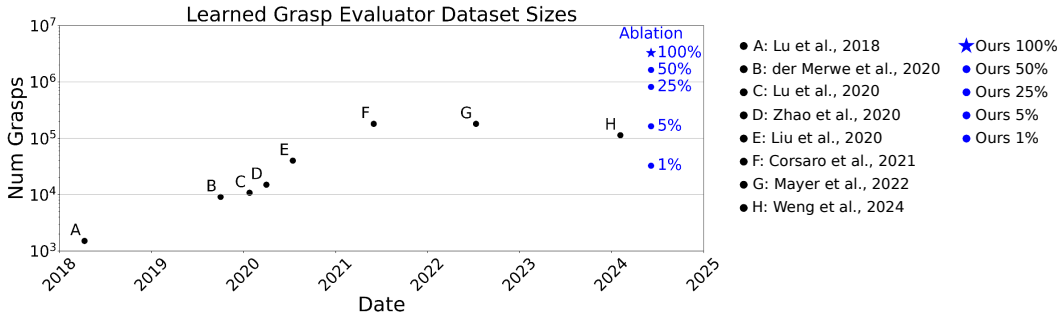


Figure 4.15: **Comparison between learned grasp evaluator dataset sizes in prior works over time.** Our dataset and our ablations shown on the right in blue. The full training set contains 3.26M grasps.

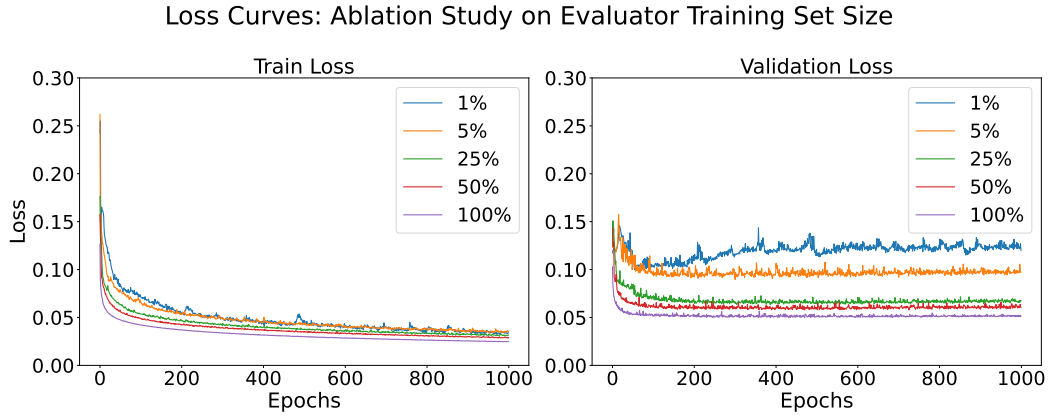


Figure 4.16: **Comparison of loss curves across each model trained on a different fraction of the training dataset and validated on the same validation dataset.** These show a clear correlation between dataset scale and validation performance. The full training set contains 3.26M grasps.

that intersect objects in a non-physical way [47]. To measure the impact of data scale directly, we train four BPS evaluator models with identical architectures and training parameters on different fractions of the training dataset (1%, 5%, 25%, 50%, 100%), all to convergence, and compare them in the same simulation setup with the same fixed diffusion sampler. Figure 4.14B shows a strong correlation between dataset size and grasp planner performance. It is plausible this trend continues with more data, but we leave that to future work.

Figure 4.16 shows the train and validation loss curves of each model. All models reach similar train losses, but their validation losses improve consistently with dataset scale, confirming the correlation.

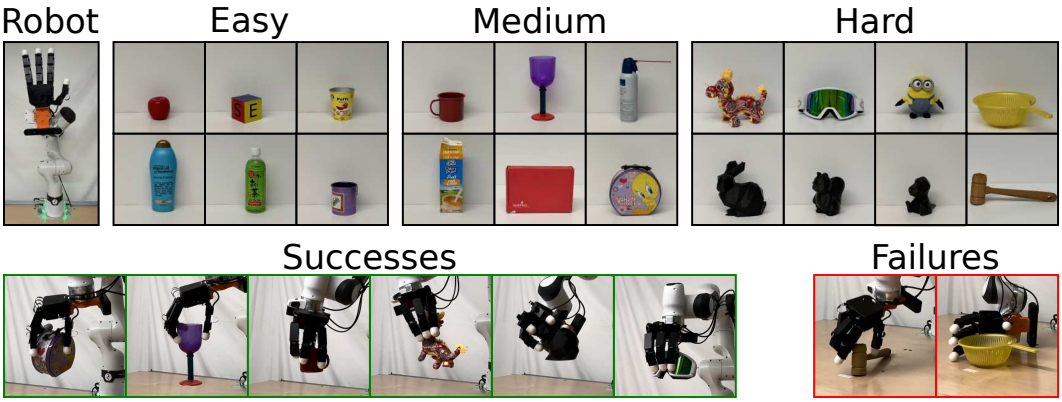


Figure 4.17: **Hardware setup.** Our robot consists of an Allegro hand and ZED 2i camera mounted onto a Franka Research 3. We select 20 objects categorized as easy, medium, or hard based on the complexity of their geometry, the presence of "distractor" geometry, and/or resemblance to objects in the training dataset. We selected these objects to maximize size/shape diversity while providing a clear upper bound on expected performance via the hard objects. The bottom row shows representative successes and failures for the "Fixed/BPS" configuration.

4.5.3 Hardware Experimental Setup

On hardware, we evaluate whether grasp evaluators trained on our dataset improve grasping performance in the real world. We compare samplers with/without the evaluator, with/without a diffusion-based sampler inspired by Weng et al. [89], and also vary the object representation in order to identify the factors driving performance. We also compare to FRoGGeR [73, 94], the analytic method introduced in Part I.

Hardware and protocol. Our hardware setup consists of an Allegro hand mounted onto a Franka Research 3 arm with an eye-in-hand ZED 2i camera. We only use monocular RGB because the objects are too close to the camera for reliable depth readings. For repeatability, we choose a canonical pose for each object that we use for all trials. The arm follows a fixed trajectory to collect 100 NeRF training images for every grasp. For each method, we request 40 grasps from the planner (~GPU memory upper bound), then execute the highest-ranking one according to the method's metric (Diffusion No Evaluator has no metric, so the first grasp is selected). Because FRoGGeR generates grasps serially, we request only 10 grasps or as many as it can generate in 20 seconds. A grasp is considered a failure if the planner fails to return any grasps or if it does not pick the object up 20cm without dropping it. If feasible grasps are generated but the motion generation fails either during planning or execution (i.e., before the grasp), we discard the trial as the

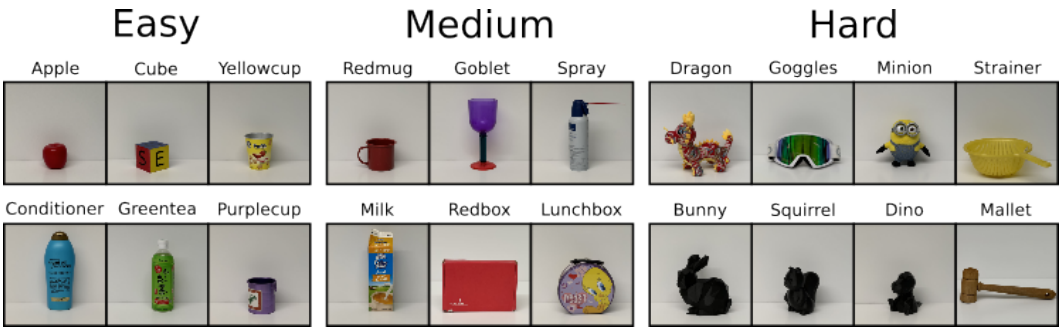


Figure 4.18: **Real-world objects labeled with their names corresponding to Table 4.4.**

failure is not due to the grasp planner itself.

Object selection. For each method, we execute 5 grasps on each of the 20 objects of varying geometry and appearance for a total of 100 grasps per method. Our dataset includes simple objects (like boxes), and adversarial ones (like reflective ski goggles), which we organize into easy, medium, and hard categories (Figure 4.18). We remark that the baselines' reported hardware results were largely limited to convex or nearly-convex objects at low quantity [94, 89]. For a more challenging evaluation, we included more objects with non-trivial visual or geometric properties. Easy objects are those that have high-quality grasps when approached from any or most directions. For example, tall, cylindrical objects can be grasped overhead or from the side. Medium objects are those with more unusual geometry, including "distractors," or objects which are harder to grasp when the hand is arbitrarily oriented. For example, the mug has a handle and the lunchbox is more easily pinched in the thin direction. Lastly, hard objects contain very unusual, non-convex geometry/visual features, or are difficult to grasp without affordances. For example, the goggles are both reflective and transparent, which poses a challenge for object reconstruction algorithms, and the bunny's ears are a large distractor for grasp planners. The bunny, squirrel, and dino were custom printed as hard test objects. The strainer and mallet were chosen to be out-of-distribution objects, providing a ceiling on the performance of our grasping algorithms.

Motion planning. To evaluate a given grasp in the real world, we use a motion planner to move the hand to the corresponding pre-grasp pose. We use cuRobo [119], which performs parallelized collision-free motion generation. In particular, we use cuRobo's graph planner, which uses a Probabilistic Road Map (PRM) to find a collision-free path with a 1mm collision buffer. Object collision avoidance requires an object mesh, which is acquired through the process explained in Section 4.4.2.

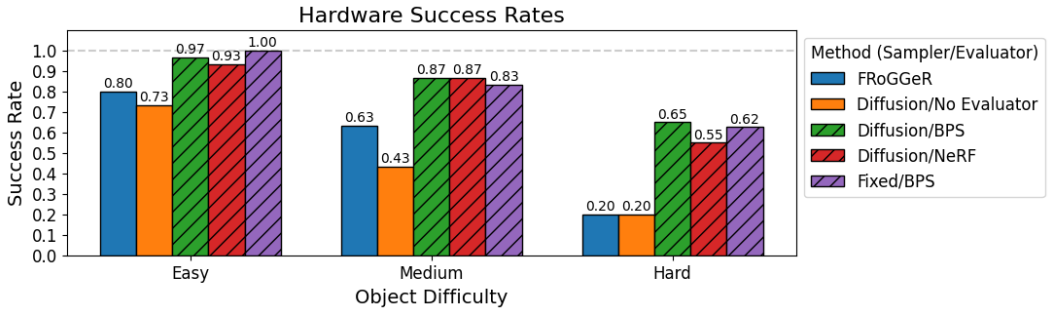


Figure 4.19: **Average grasping success rates on hardware across object difficulties.** We find that the methods that leverage evaluators (hatched) outperform both analytic and generative-modeling-only baselines. Detailed per-object success rates are shown in Section 4.5.5.

We decompose the motion into three stages: (1) start pose to pre-grasp pose, (2) pre-grasp pose to grasp pose, and (3) grasp pose to lift pose. The first stage is the most challenging because the hand needs to find a collision-free path to get very close to the object. To simplify this, we use inverse kinematics to adjust the pre-grasp finger joints, moving them an additional 3cm backward along the fingertip grasp directions. To avoid damaging the cabling of the wrist-mounted camera during execution, we also filter out grasp samples if either (1) the normal direction of the palm is within 60 degrees from the upward direction of the world frame or (2) the direction from the palm to the middle finger deviates by more than 60 degrees from the forward direction of the world frame. Finally, we use `cuRobo` to solve all 40 motion planning problems in parallel and execute the first grasp for which a motion plan is successfully found.

4.5.4 Hardware Results

Figure 4.19 shows the hardware results. We find that all evaluator methods (diagonally-striped bars) outperform the baselines by a considerable margin *independent of the sampler or object representation*, with the worst model in each category achieving 93%, 83%, and 55% success rates respectively. We also observe that all evaluator-based methods perform similarly, which suggests the model performance is not sensitive to the object representation. This may help disentangle representations useful for different downstream tasks from potential negative effects on grasping performance present in other learning-based methods.

Difficulty	Objects	Methods				
		No Evaluator		Evaluator		
		FRoGGeR	Diffusion No Evaluator	Diffusion BPS	Diffusion NeRF	Fixed BPS
Easy	Apple	4/5	3/5	5/5	5/5	5/5
	Cube	3/5	1/5	5/5	4/5	5/5
	Yellowcup	5/5	4/5	5/5	5/5	5/5
	Conditioner	4/5	5/5	5/5	4/5	5/5
	Greentea	3/5	4/5	4/5	5/5	5/5
	Purplecup	5/5	5/5	5/5	5/5	5/5
	TOTAL	24/30 (80%)	22/30 (73%)	29/30 (97%)	28/30 (93%)	**30/30 (100%)**
Medium	Redmug	2/5	4/5	5/5	5/5	5/5
	Goblet	5/5	3/5	5/5	5/5	5/5
	Spray	4/5	0/5	2/5	4/5	3/5
	Milk	1/5	5/5	5/5	5/5	5/5
	Redbox	5/5	1/5	4/5	5/5	3/5
	Lunchbox	2/5	0/5	5/5	2/5	4/5
	TOTAL	19/30 (63%)	13/30 (43%)	**26/30 (87%)**	**26/30 (87%)**	25/30 (83%)
Hard	Dragon	0/5	0/5	2/5	2/5	4/5
	Goggles	0/5	0/5	4/5	2/5	4/5
	Minion	0/5	3/5	5/5	5/5	5/5
	Strainer	0/5	0/5	0/5	0/5	0/5
	Bunny	0/5	1/5	5/5	4/5	3/5
	Squirrel	1/5	2/5	4/5	3/5	4/5
	Dino	2/5	2/5	5/5	5/5	5/5
	Mallet	5/5	0/5	1/5	1/5	0/5
	TOTAL	8/40 (20%)	8/40 (20%)	**26/40 (65%)**	22/40 (55%)	25/40 (62%)
All	TOTAL	51/100 (51%)	43/100 (43%)	**81/100 (81%)**	76/100 (76%)	80/100 (80%)

Table 4.4: **Results of different grasp planning methods on various objects in the real world.** Images of these objects can be found in Figure 4.18.

4.5.5 Detailed Hardware Results

Table 4.4 presents the detailed per-object results of all grasp planning methods in the real world. Independent of object difficulty, the evaluator-based methods clearly improve over the baselines. The out-of-distribution strainer is the hardest case, with no method achieving a single successful grasp.

Qualitatively, we observed that our evaluator-based methods very rarely exhibited *edge-seeking* behavior, wherein the fingers are placed on corners or edges of objects,

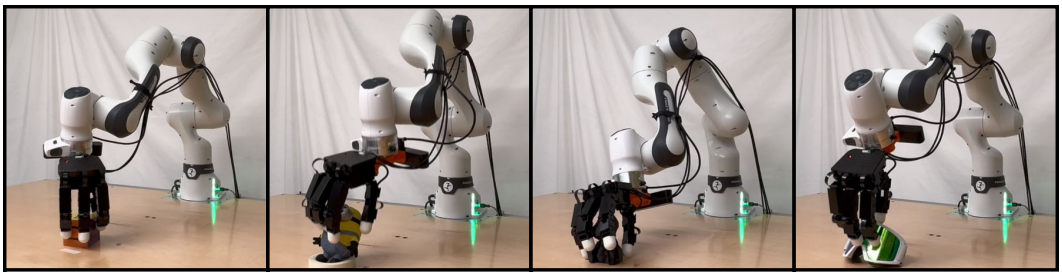


Figure 4.20: **Some examples of failed grasps generated by the diffusion sampler.** Though they failed, the grasps appear qualitatively reasonable.

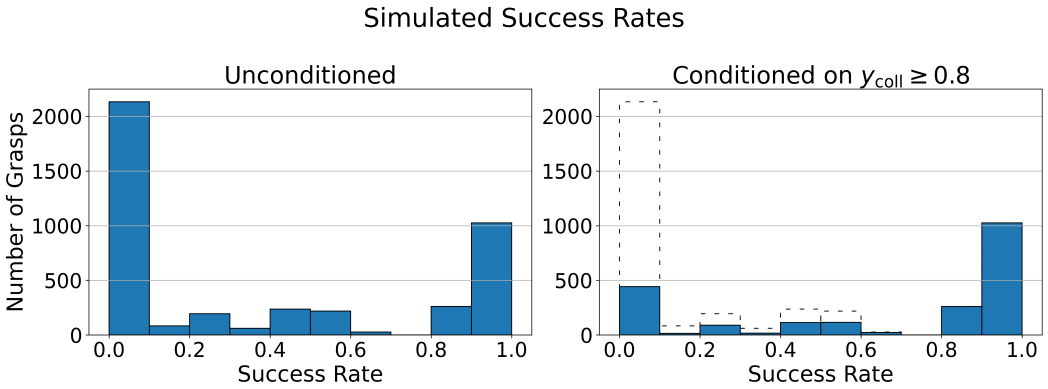


Figure 4.21: **Success rates for all simulated evaluator-based methods.** When only considering grasps with $y_{\text{coll}} \geq 0.8$, the median success rate increases from 37% to 80%. 2100/4240 grasps satisfied $y_{\text{coll}} \geq 0.8$. On the right, the unconditioned distribution is plotted with dashed lines for comparison.

and has been noted as a common failure mode by other works [73, 94]. We suspect this may be a result of our label smoothing, though we do not rigorously test this. However, "distractor" geometries like the bunny ears or spray nozzle frequently caused grasp planning failures, attracting the fingers towards them. FRoGGeR relies on online mesh reconstruction, so objects like the goggles caused issues due to transparency/reflectivity for it. Lastly, many of the diffusion sampler's unrefined grasps appeared "close" to success even when failing, which suggests that generative modeling may be a viable strategy with more data. Some examples are shown in Figure 4.20.

4.5.6 Discussion

At first glance, our simulation success rates appear much lower than our hardware success rates and the simulation results of similar studies [89, 88], but this discrepancy is an artifact of labeling. In the real world we only attempt collision-free

grasps, so we report the conditional probability

$$p(\text{success} \mid \text{no collisions}).$$

In simulation we instead evaluate *all* planned grasps, counting those in collision as failures, and report $p(\text{success})$. Figure 4.21 compares the unconditional success rates with those conditioned on $y_{\text{coll}} \geq 0.8$: the median rises from 37% to 80%, closely matching the 76-81% our evaluator-based methods achieve on hardware (Table 4.4). Our learning problem is also harder than those of prior studies in three ways. We use a tabletop setting (which disallows non-physical fully-caging grasps), greater object diversity, and precision rather than power grasps, which further explains the gap to other reported numbers.

The relative ordering of the fixed and diffusion samplers also differs between simulation and hardware: Fixed+Evaluator slightly outperforms Diffusion+Evaluator in simulation, yet the two are nearly identical on hardware. We attribute this to distribution overlap. In simulation, the train and test objects are drawn from the same distribution, so some test objects resemble training objects, and a few fixed-dataset grasps benefit from mild overfitting. Because no real-world objects appear in the training set, this advantage vanishes on hardware.

4.6 Conclusion

This chapter introduced a large-scale dataset of 3.5M grasps on 4.3K unique objects with RGB images, point clouds, and NeRFs, addressing key challenges for robust sim-to-real transfer of multi-finger grasping. Comprehensive ablation studies highlighted the crucial role of learned grasp evaluators, which significantly outperform existing analytic and generative baselines in both simulation and the real world, and showed that dataset scale is a key driver of real-world performance.

The learned evaluator embodies one view of how to put large-scale data to work, a discriminative one, in which a network scores grasps proposed by a separate sampler and the two are composed at inference. A second view gained rapid popularity around the time this work was completed. Rather than factoring the problem into a sampler and a metric, it treats manipulation more directly, scaling up demonstration data and learning the action-generating model itself. Imitation learning methods such as action-chunking transformers [5] and diffusion policies [6] fit expressive generative policies to demonstrations, and vision-language-action models such as RT-1 [7], RT-2 [8], OpenVLA [9], Octo [10], and π_0 [11] extend this recipe across

large, heterogeneous robot datasets [12, 13, 14]. The results here are an early signal of the same bet. The scale ablation shows that dataset size, more than the particular quantity being optimized, drives real-world performance, and the near-miss grasps produced by our lightly-supervised diffusion sampler suggest that directly generating actions becomes competitive once the data are plentiful enough. The evaluator can thus be read as a precursor to these directions, occupying a middle ground between the analytic grasp metrics of Part I and the scale-driven, directly-learned policies that increasingly dominate manipulation research, already leaning on scale to cross the sim-to-real gap while keeping an explicit success model in the loop.

In contrast to Part I, the approach in this chapter (generating massive simulated experience to compensate for the sim-to-real gap) treats a physics simulator as the approximate model rather than deriving one from first principle mechanics. The simulator's contact dynamics, visual renders, etc. are imperfect, but the scale of data allows learned models to average over errors and generalize to real hardware. The next chapter takes this simulator-as-model idea further by using simulation not for offline data generation, but as a *real-time forward model* for online planning in contact-rich manipulation.

Chapter 5

SIMULATION-BASED PREDICTIVE CONTROL FOR CONTACT-RICH MANIPULATION

5.1 Introduction

The preceding chapter showed that physics simulators, when used as black-box approximate models at sufficient scale, can bridge the sim-to-real gap for grasp synthesis. This chapter pushes the simulator-as-model paradigm further: rather than using simulation to generate offline training data, we use it as a *real-time forward model* for online planning. The key question is whether a simulator's contact dynamics—despite their approximation errors—are accurate enough to serve as a predictive model for closed-loop control of contact-rich manipulation.

For contact-rich tasks, reinforcement learning (RL) has been the dominant paradigm due to its ability to generate real-world robust plans. One well-studied task is in-hand *cube reorientation*, where a hand must rotate a cube to match consecutive goal orientations. Pioneered by OpenAI [27] and extended by others [120, 121], RL policies trained with massively-parallelized, domain-randomized simulations have achieved remarkably robust sim-to-real transfer for cube rotation. But, this *offline* simulation-based approach requires substantial pre-execution computation and is inflexible to changing task specifications.

Leveraging recent advances in real-time parallel simulation (e.g., MJPC) [122], we instead study the *online* approach of sampling-based predictive control (SPC) methods for in-hand manipulation, which continuously replan by simulating parallel rollouts and applying optimal control actions over short time horizons. In contrast with RL, such online planning methods can adjust the task or model without re-training, but may demand expensive online computation. While tools like MJPC have made SPC feasible for many simulated contact-rich tasks, their real-world utility remains largely unproven.

This chapter presents DROP (**D**exterous **R**eorientation via **O**nline **P**lanning), a system architecture (Fig. 5.1) that consists of (i) a simple sampling-based planner and (ii) a vision-based state estimator comprising a keypoint detection model, a pose smoother, and a collision-aware state corrector. The aim is not to design the best cube reorientation policy, but to test the viability of sampling-based strategies

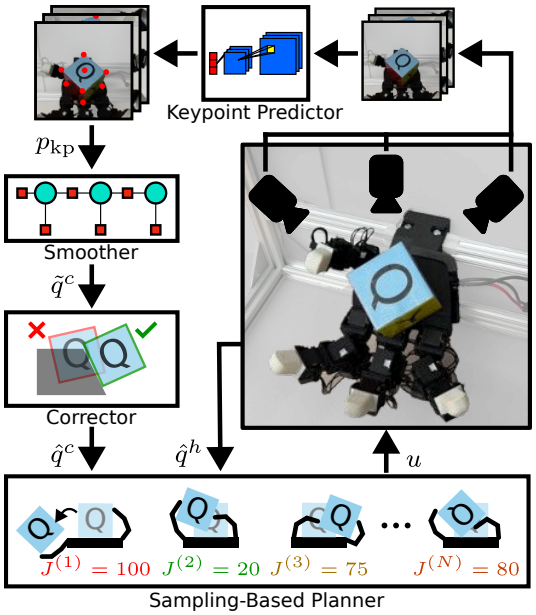


Figure 5.1: **The DROP architecture.** DROP consists of (i) a vision-based cube pose estimator (composed of the Keypoint Predictor, Smoother, and Corrector), and (ii) a sampling-based planner that selects control actions by conducting model-based rollouts and iteratively improving the sampling distribution online based on the costs $J^{(i)}$.

by thoroughly assessing DROP's real-world performance, architectural design, and key factors for robustness. DROP is the first online planning method for cube reorientation on hardware, and its performance is comparable with prior RL-based methods. We also present `judo`, an open-source toolbox for sampling-based MPC that grew out of this work, and show how it enabled further results on cube rotation with fewer computational resources and on a new loco-manipulation platform. For reproducibility, we provide open-source code, estimator weights, and hardware setup[1].

5.2 Background and Preliminaries

5.2.1 Related Work

The first viable methods for in-hand cube reorientation employed *reinforcement learning (RL)*, starting with Dactyl [27], which popularized domain randomization for sim-to-real transfer of RL policies. Subsequent research built on Dactyl with improved domain randomization [120], tactile-only sensing [124, 125], and "palm-down" rotations for various objects [126, 121]. While typically limited to specific

[1]Project website: https://caltech-amber.github.io/drop [123].

robots or object classes, RL's robust hardware performance has established it as the standard for in-hand manipulation.

Others have explored model-based approaches to contact-rich manipulation via *contact-implicit trajectory optimization* (CITO). Originally used in locomotion [127, 128], recent work [129, 122] shows that such planners can generate motions for simple contact-rich tasks like bimanual lifting [130] or real-time planar rolling and sliding [131, 132]. Other approaches have combined CITO with learned signed distance fields [133], global search [134], or smoothed dynamics [135].

Sampling-based planning has a long history in manipulation, starting with methods like RRT, PRM, and STOMP [136, 137, 138]. Recent advances in SPC [16, 139] have enabled simple contact-rich manipulation like pushing, sliding, and pick and place [140, 141], while new tools for real-time parallel simulation of contact-rich tasks [122], have made SPC viable for simulated contact-rich tasks with remarkably simple controllers. Still, little evidence exists for the viability of SPC for *real-world* contact-rich manipulation (concurrently, some works have studied it for locomotion [142]), and overall, model-based methods (like SPC or CITO) have seldom studied tasks as hard as cube reorientation, which exhibits many simultaneous unpredictable contact modes.

Lastly, high-quality *object tracking* is key for most of these methods, especially model-based online search. RL approaches [27, 120] usually use single-object pose estimation networks, but recent computer vision advances suggest that more general tracking pipelines like frame-to-frame point trackers [143, 144, 145] and foundation models for rigid body pose prediction [146] may boost performance.

5.2.2 Mathematical Notation

The notation in this chapter follows the self-contained conventions of Part II (cf. Chapter 4), using standard control conventions in which bold lowercase symbols denote vectors: $\boldsymbol{x}$ is the full system state, $\boldsymbol{q}$ the generalized configuration, $\boldsymbol{v}$ the velocity, and $\boldsymbol{u}$ the control.

We model the continuous-time cube-in-hand dynamics as

$$\dot{\boldsymbol{x}}(t) = f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right), \tag{5.1}$$

where $\boldsymbol{x}$ is the state and $\boldsymbol{u}$ are control inputs. We do not assume access to a closed-form expression for f, but instead to a simulator (MuJoCo [78]) that generates state trajectories $\boldsymbol{x}(t)$ given an initial state $\boldsymbol{x}_0$ and a control sequence $\boldsymbol{u}(t)$.

The state $\boldsymbol{x}$ consists of the generalized positions and velocities of both the cube $(\boldsymbol{q}^c, \boldsymbol{v}^c)$ and hand $(\boldsymbol{q}^h, \boldsymbol{v}^h)$:

$$\boldsymbol{x} = [\boldsymbol{q}, \boldsymbol{v}] = [\boldsymbol{q}^c, \boldsymbol{q}^h, \boldsymbol{v}^c, \boldsymbol{v}^h]. \tag{5.2}$$

The control actions $\boldsymbol{u}$ are position setpoints for the hand, which are tracked by a lower-level PD controller internal to the model f. Following [122], we represent a control trajectory $\boldsymbol{u}(t)$ on $t \in [0, T]$ with K spline knots,

$$\mathbf{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_k, \ldots, \boldsymbol{u}_K], \tag{5.3}$$

which constitute the decision variables for online planning. The control objective is encoded by a cost functional

$$J(\mathbf{U}; \boldsymbol{x}_0) = \int_0^T \ell\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \, \mathrm{d}t, \tag{5.4}$$

where querying $J(\mathbf{U}; \boldsymbol{x}_0)$ requires simulating f using an open-loop control signal $\boldsymbol{u}(t)$ from initial condition $\boldsymbol{x}_0$.

Throughout this chapter, we use the following notation to denote "subtraction" of two quaternions $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{H}$:

$$\boldsymbol{a} \ominus \boldsymbol{b} := \operatorname{Log}\left(\boldsymbol{a} \circ \boldsymbol{b}^{-1}\right) \in \mathfrak{so}(3), \tag{5.5}$$

and similarly, when $\boldsymbol{a}, \boldsymbol{b} \in \mathrm{SE}(3)$, $\boldsymbol{a} \ominus \boldsymbol{b} \in \mathfrak{se}(3)$.

5.3 The DROP Algorithm

DROP consists of (i) a sampling-based planner and (ii) a keypoint-based cube pose estimator (see Fig. 5.1). The planner continually updates the control spline $\mathbf{U}$ via SPC. We use MJPC [122] to handle both parallel rollouts and policy updates.

5.3.1 Sampling-Based Planning and Control

Algorithm 1 describes a generic SPC procedure. At time $t \in \mathbb{R}$, the planner receives a state estimate $\hat{\boldsymbol{x}}(t)$ and rolls out a batch of N open-loop control sequences $\mathbf{U}^{(i)}$ drawn from some parametric distribution $\pi_\theta(\mathbf{U})$. Each control trajectory is simulated (in parallel) to obtain a cost $J^{(i)}$, and all costs are jointly used to update the sampling parameters θ. The control input $\boldsymbol{u}(t)$ can be obtained for any time t from the spline parameters $\mathbf{U}$. This allows planning to proceed asynchronously, with the parameters θ updated as quickly as computational limits allow.

Algorithm 1: Sampling-based Predictive Control

Input: θ, N, planner-specific parameters.
while *planning* **do**
  $\boldsymbol{x}_0 \leftarrow \hat{\boldsymbol{x}}(t)$; // estimate curr state
  for $i = 1$ *to* N *;* // multi-threaded
  do
    $\mathbf{U}^{(i)} \sim \pi_\theta(\mathbf{U})$; // sample controls
    $J^{(i)} \leftarrow J\left(\mathbf{U}^{(i)}; \boldsymbol{x}_0\right)$; // eval rollout
  $\theta \leftarrow$ `update_params` $\left(\mathbf{U}^{(1:N)}, J^{(1:N)}\right)$;
  $\boldsymbol{u}(t) \leftarrow$ `get_action`(θ, t) ; // asynchronous

In our experiments, we test two simple planning strategies that both use a diagonal Gaussian distribution $\pi_\theta(\mathbf{U}) = \mathcal{N}\left(\overline{\mathbf{U}}, \boldsymbol{\Sigma}\right)$ with parameters $\theta = \left(\overline{\mathbf{U}}, \boldsymbol{\Sigma}\right)$. In Algorithm 1, `get_action`$(\theta, t)$ is a spline interpolation with knots $\overline{\mathbf{U}}$.

Predictive sampling (PS) repeatedly updates $\overline{\mathbf{U}}$ to be the best sample, fixing $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$. Despite its simplicity, PS has demonstrated surprisingly effective performance on complex robot manipulation and locomotion tasks in simulation [122].

The **cross-entropy method (CEM)** instead fits both $\overline{\mathbf{U}}$ and $\boldsymbol{\Sigma} = \mathrm{diag}(\boldsymbol{s})$ to the sample mean and variance of the M best *elite samples*, where $\boldsymbol{s}$ is a vector of covariances. CEM is only marginally more complex than PS, but has long been used for general gradient-free optimization[147].

5.3.1.1 The Cube Reorientation Problem

As in prior work [120], the task is to use a dexterous hand to rotate a cube to within 0.4 rad of as many goal orientations in a row as possible without dropping. The goals are uniformly randomly sampled over $\mathrm{SO}(3)$. In contrast with [120], to ensure sufficient task difficulty, each new goal must be at least 90° from the prior one.

Mathematically, the DROP cube reorientation problem is expressed as the optimal control problem

$$\begin{aligned} \min_u \int_0^T \left\{ \lambda_{\mathrm{g}} \cdot \ell_{\mathrm{g}}(\boldsymbol{r}^c(t)) + \lambda_{\mathrm{p}} \cdot \ell_{\mathrm{p}}(\boldsymbol{p}^c(t)) \right\} \mathrm{d}t \\ \text{s.t.} \quad \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{aligned} \tag{5.6}$$

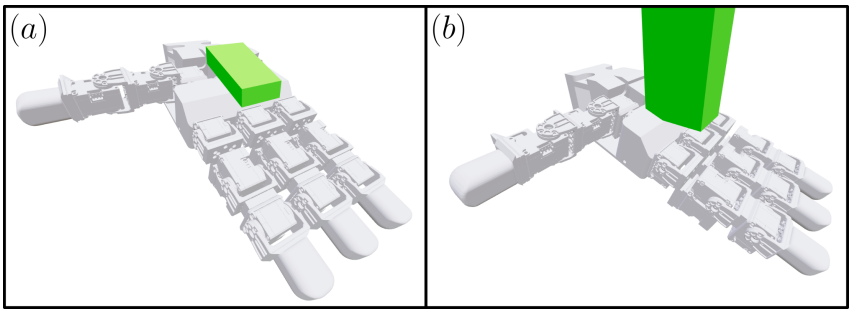


Figure 5.2: The safe regions $\mathcal{S}$ (green) for (a) when the cube's xy coordinates lie in the palm, and (b) when the cube's xy coordinates lie outside the palm.

where $\lambda_{(\cdot)}$ denote weights, the dynamics $f(\boldsymbol{x}, \boldsymbol{u})$ are only available through simulation, and

$$\ell_{\mathrm{g}}(\boldsymbol{r}^c) := \left\| \boldsymbol{r}^c \ominus \boldsymbol{r}^c_{\mathrm{goal}} \right\|_2^2, \tag{5.7}$$

$$\ell_{\mathrm{p}}(\boldsymbol{p}^c) := \mathrm{dist}_{\mathcal{S}}(\boldsymbol{p}^c), \tag{5.8}$$

are running costs. The variables $\boldsymbol{p}^c$ and $\boldsymbol{r}^c$ denote the positional and rotational components of the cube pose $\boldsymbol{q}^c = [\boldsymbol{p}^c, \boldsymbol{r}^c]$, which are extracted from the simulated state $\boldsymbol{x}(t)$. ℓ_{g} penalizes rotational distance from the goal, while ℓ_{p} penalizes the cube leaving a "safe" region $\mathcal{S}$ in Cartesian space.

We prioritize drop reduction by setting λ_{p} high and choosing a conservative region $\mathcal{S}$ (detailed below). In practice, we use a relatively low λ_{g}, which slows down the planner but also amplifies differences in rotation rate across methods, allowing easier quantitative comparison.

We now make the control problem (5.6) concrete by detailing the position cost, the safe region $\mathcal{S}$, the cost weights, and the time discretization.

The position cost term. For the position cost term $\ell_{\mathrm{p}}(\boldsymbol{p}^c)$ in (5.8), the function $\mathrm{dist}_{\mathcal{S}}(\cdot)$ is parameterized by the "safe region" $\mathcal{S}$ in which we would like the cube's center to remain. Specifically, we have

$$\begin{aligned} \mathrm{dist}_{\mathcal{S}}(\boldsymbol{p}^c) &:= d(\|\boldsymbol{p}^c\|_{\mathcal{S}}), \\ d(x) &:= 0.05 \cdot \log\left(1 + \exp\left(\frac{250x}{0.05}\right)\right), \end{aligned} \tag{5.9}$$

where $\|x\|_{\mathcal{S}}$ reports the Euclidean distance between a point x and a set $\mathcal{S}$, and $d(x)$ is a scalar-valued function that applies a very quickly-growing penalty when $x > 0$. In particular, its parameters are chosen such that when $x = 0.01$, $d(x) \approx 1$.

The set $\mathcal{S}$ is defined by two cases. If the cube's x and y coordinates lie within a specified rectangle, then we parameterize $\mathcal{S}$ as a parallelepiped whose x and y faces are axis-aligned, and whose z faces are angled at the same angle as the palm of the hand, $\beta := 20°$. Otherwise, we instead specify a uniform minimum z height denoted z^- and no maximum height. The second case is designed so that if the planner predicts the cube will leave the palm, it prioritizes not dropping it and returning it back to the palm. See Fig. 5.2 for a visualization of these two cases.

The global origin of the system is located in the center of the interface between the hand mount and the bar to which it is affixed. The side length of the cube is $b := 0.07$. Let $(\boldsymbol{p}_x^c, \boldsymbol{p}_y^c, \boldsymbol{p}_z^c)$ denote the position coordinates of the center of the cube. The dimensions associated with the figure are specified in Table 5.1.

Dim	Value
x^-	0.08
x^+	0.14
y^-	-0.02
y^+	0.02
z^-	$\begin{cases} \frac{b}{2\cos(\beta)} - \boldsymbol{p}_x^c \cdot \tan(\beta), & \text{if } \boldsymbol{p}_x^c \in [x^-, x^+],\ \boldsymbol{p}_y^c \in [y^-, y^+] \\ -0.015, & \text{otherwise} \end{cases}$
z^+	$\begin{cases} z^- + 0.035, & \text{if } \boldsymbol{p}_x^c \in [x^-, x^+],\ \boldsymbol{p}_y^c \in [y^-, y^+] \\ +\infty, & \text{otherwise} \end{cases}$

Table 5.1: Dimensions of the set $\mathcal{S}$.

Cost Weights. The weights in (5.6) are $\lambda_{\mathrm{g}} = 1.0, \lambda_{\mathrm{p}} = 2.5$.

Time discretization. While (5.4) is expressed as an integral over the interval $[0, T]$, in practice we numerically compute it by discretizing the interval using a fixed time step Δt:

$$\int_0^T \ell\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)\, \mathrm{d}t \approx \sum_{j=0}^{H} \ell\Big(x(j\Delta t), u(j\Delta t)\Big) \cdot \Delta t, \tag{5.10}$$

where $H = T/\Delta t$.

5.3.1.2 Planner Implementation Details

While Algorithm 1 captures the core of the planner, a few implementation choices were important for its performance in practice.

Warm-starts by time shifting. Because the cost J at some time depends on the current state $\boldsymbol{x}_0 = \hat{\boldsymbol{x}}(t)$, once time elapses, the optimization problem (5.6) changes. Instead of solving the problem from scratch, the solution is *warm started* by applying a time shift to the previous solution (e.g., as in [148]). For the last time step of the warm-started solution, we simply copy the second-to-last spline parameter.

To be precise, let h denote an iteration of the mean control spline knots $\overline{\mathbf{U}}$. Then, applying the shift operation gives

$$\begin{aligned} \overline{\mathbf{U}}[h] &= [\boldsymbol{u}_1[h], \boldsymbol{u}_2[h], \ldots, \boldsymbol{u}_K[h]]\,, \\ \text{shift}\left(\overline{\mathbf{U}}[h]\right) &= [\boldsymbol{u}_2[h], \boldsymbol{u}_3[h], \ldots, \boldsymbol{u}_K[h], \boldsymbol{u}_K[h]]\,, \end{aligned}$$

which the new samples will be centered on (and similarly for any other parameters like sampling variances).

Retaining the best rollout in PS. In predictive sampling (PS), MJPC additionally keeps the current nominal (i.e., the best) control spline as one of the samples when resampling, so that if none of the sampled trajectories improves performance, the nominal policy is not updated. We found that this is critical to the performance of PS.

Minimum standard deviation in CEM. While the sampling variance is fixed in PS, in CEM, a diagonal covariance is repeatedly fit to the top M elite samples. However, in practice, the variance may quickly collapse to very low values, preventing the planner from exploring. Therefore, we specify a minimum standard deviation uniformly over the diagonal entries of $\boldsymbol{\Sigma}$, denoted σ_{min}.

Default Parameters. Table 5.2 lists the values of planner parameters used in experiments unless otherwise stated.

5.3.2 The Pose Estimation Pipeline

Our pose estimator consists of three parts: a keypoint predictor, a fixed-lag smoother, and a collision-aware corrector.

5.3.2.1 (A) Keypoint Predictor

The prediction model. The estimator takes in images $\mathbf{I}_c \in \mathbb{R}^{C\times H\times W}$ from n_c cameras. We first train a vision model g_φ that predicts 8 fixed keypoints on the cube corresponding to its corners, $\boldsymbol{p}_{\text{kp}} = g_\varphi(\mathbf{I}_c)$. The keypoint prediction task is supervised from a training dataset of 686,000 images of a simulated cube rendered

Method	Parameter	Value
All	Spline Order	0
	Plan Horizon	1.0s
	Plan Timestep	0.01s
	Num Spline Knots	4
PS	Num Samples	120
	Sampling Stdev	0.3
CEM	Num Samples	120
	Num Elites	4
iLQR	Num Parallel Line Search Threads	120

Table 5.2: Default planner parameter values used in all experiments.

by `Blender`, which includes ground-truth pixel locations for all keypoints, even those that are outside the frame or occluded. We generate randomized background scenes using `kubric` [149], which spawns the cube along with other random assets in a `pybullet` simulation. Similar datasets are commonly used to train "track-any-point" models [143, 144, 145] which exhibit strong sim-to-real transfer for similar keypoint tracking tasks.

Crucially, we then augment these images with random affine transforms; visual effects like color, shadow, and contrast; and randomized backgrounds. We also found that pruning images where the cube was nearly occluded, or too close for a reliable pose estimate, was essential for good performance. Figure 5.3 shows some of the resulting images.

To train g_φ, we fine-tuned an ImageNet-pretrained `resnet18` using `AdamW` with an MSE loss. The only model adjustments were the number of input channels (depending on RGB or RGBD inputs) and the dimension of the output layer. Full details of the synthetic dataset, the augmentation pipeline, and the training procedure are provided in Secs. 5.8.1 and 5.8.2.

5.3.2.2 (B) Pose Smoother

We use GTSAM [150], a factor graph-based state estimation package, to convert keypoints into a cube pose estimate via fixed-lag smoothing. Given a fixed pinhole camera model with known camera poses and cube size, we derive keypoint measurement factors that relate keypoints $\boldsymbol{p}_{\text{kp}}$ to a cube pose $\boldsymbol{q}^c$. This allows GTSAM to fuse keypoint predictions from an arbitrary number of cameras in real-time to yield

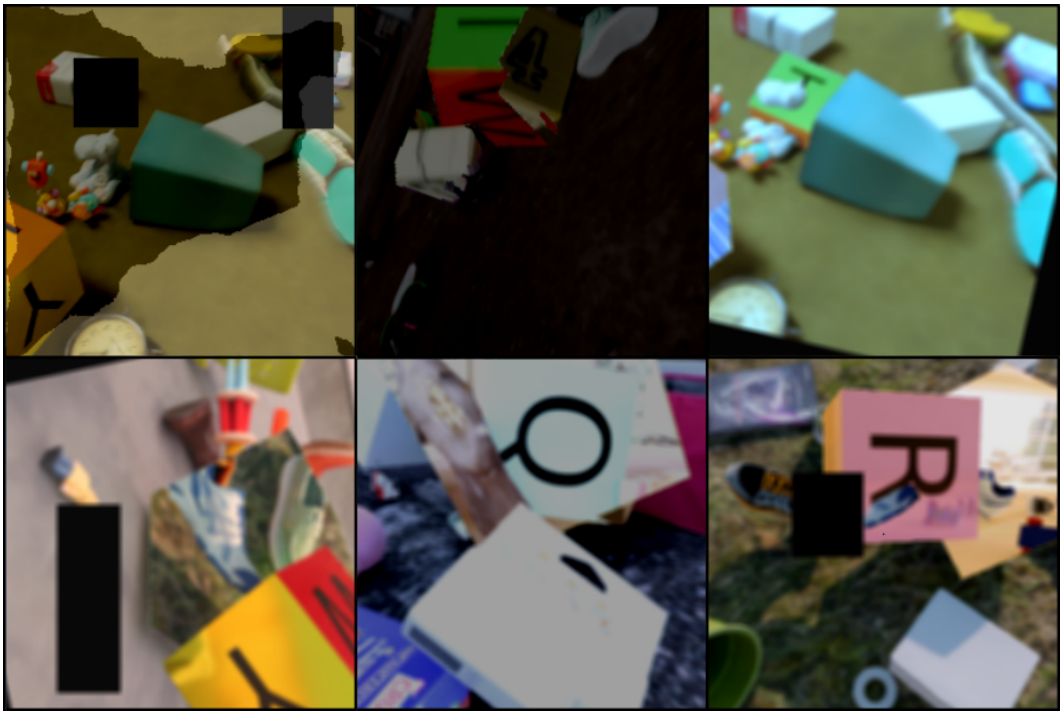


Figure 5.3: **Image augmentations.** To train the keypoint predictor, we augmented simulated images of the cube with random crops, affine transformations, spliced backgrounds, random deletions, and visual adjustments in color, contrast, brightness, and reflectivity.

a smoothed cube pose estimate $\tilde{\boldsymbol{q}}^c$. See our open-source implementation for exact details [123].

In both simulation and hardware, we estimate velocities by numerically differentiating position estimates (drawn from the smoother for the cube and joint encoders for the hand) and applying an exponential moving average filter with parameter $\alpha = 0.1$ to compute a smooth velocity estimate $\tilde{\boldsymbol{v}}$.

Smoother details. We use a factor graph to fuse keypoint predictions into a smoothed cube pose estimate. Here, we describe the structure of that graph along with the implementation details and hyperparameters of the smoother. Readers unfamiliar with factor graphs and incremental smoothing can first consult the primer in Sec. 5.9. For the most fine-grained explanation, please see our open-source code: https://caltech-amber.github.io/drop.

The cube pose estimation factor graph. In DROP, we seek to estimate the cube poses (and velocities) over time. For computational speed, we do not model the full dynamics of the cube-hand system (which gives rise to the need for the corrector later). In particular, the stiff contact dynamics would introduce substantial numerical

challenges when conducting successive linearizations to solve (5.21).

Instead, we use a *constant velocity model* to approximate the cube dynamics. Because the linear and angular velocity of the cube is typically small when supported by the palm, this simple model (accompanied by a sufficiently tuned mistrust of it) provides a coarse but sufficiently-accurate model of the cube's motion.

For a given camera with a known global pose in conjunction with a pinhole camera model, the measurement model computes the expected keypoint locations in the camera's image frame (in pixel coordinates). The "true" measurements come from the keypoint predictor network g_φ.

Sketch of the keypoint projection factor. The keypoint factor (one for each keypoint, indexing suppressed for clarity) for a camera has the form

$$\phi_t^{\text{meas}}(\boldsymbol{q}_t^c; \nu^{\text{cam}}, \boldsymbol{p}_{\text{kp}}^{\text{pix}}, \boldsymbol{p}_{\text{kp}}^c, \boldsymbol{p}^{\text{cam}}, \boldsymbol{r}^{\text{cam}}), \tag{5.11}$$

where $\boldsymbol{q}_t^c$ is the pose of the cube at time t, ν^{cam} is the camera's known intrinsics, $\boldsymbol{p}_{\text{kp}}^{\text{pix}} \in \mathbb{R}^2$ is the measured keypoint locations in pixel coordinates from the predictor g_φ, $\boldsymbol{p}_{\text{kp}}^c \in \mathbb{R}^3$ is the location of the associated keypoint in the cube's local frame, and $(\boldsymbol{p}^{\text{cam}}, \boldsymbol{r}^{\text{cam}}) \in \text{SE}(3)$ is the known fixed pose of the camera in the world frame.

Let the function $\text{proj} : \mathbb{R}^3 \to \mathbb{R}^2$ project a spatial point expressed in the camera frame to a coordinate in pixel space, which is implemented in GTSAM [150]. The measurement residual is then computed as follows:

$$\boldsymbol{p}_{\text{kp}}^{\text{pix}} - \text{proj}(\boldsymbol{p}_{\text{kp}}^c; \nu^{\text{cam}}, \boldsymbol{p}^{\text{cam}}, \boldsymbol{r}^{\text{cam}}), \tag{5.12}$$

where the Jacobians associated with frame transforms and the projection function are also computed by GTSAM. For the exact implementation details, please see our open-source implementation [123]. All smoother parameters and noise models are summarized in Sec. 5.8.3.

5.3.2.3 (C) Collision-Aware Corrector

While the smoother's cube pose estimate $\tilde{\boldsymbol{q}}^c$ is usually accurate within 1cm, it may not always be physically compatible with the hand configuration $\tilde{\boldsymbol{q}}^h$ from the encoders, as the smoother has no knowledge of collision dynamics. Thus, $\tilde{\boldsymbol{q}} = [\tilde{\boldsymbol{q}}^c, \tilde{\boldsymbol{q}}^h]$ often corresponds to non-negligible interpenetration between cube and hand, which (i) leads to inaccurate plans that destabilize the closed-loop system, and

(ii) decreases the planning rate, as stiff MuJoCo models with high interpenetration are poorly-conditioned, requiring more solver iterations.

To remedy this, we adapt the method of [151] by using a *corrector*, which maintains an asynchronous internal model

$$\dot{\hat{\boldsymbol{x}}} = \hat{f}(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) \tag{5.13}$$

with state $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{q}}^c, \hat{\boldsymbol{q}}^h, \hat{\boldsymbol{v}}^c, \hat{\boldsymbol{v}}^h]$.

The corrector receives an un-adjusted estimate $\tilde{\boldsymbol{x}}$ from the smoother and encoders, and computes a "corrective wrench"

$$\boldsymbol{w} = -C_P(\hat{\boldsymbol{q}}^c \ominus \tilde{\boldsymbol{q}}^c) - C_D(\hat{\boldsymbol{v}}^c - \tilde{\boldsymbol{v}}^c) \tag{5.14}$$

that is added to (5.13) as a generalized force. Since the corrector starts in a feasible state, simulating (5.13) results in a corrected estimate $\hat{\boldsymbol{x}}$ that is always physically feasible, but pulled toward the (possibly infeasible) vision-based estimate $\tilde{\boldsymbol{x}}$. We also add a constant corrective force in the direction of gravity to counteract any smoother estimates that may unrealistically pull the cube upwards due to high corrector gains C_P, overwhelming the natural gravitational forces of (5.13).

Parameter	Value
Timestep	0.04s
LEAP Hand k_p	3.0
`solimp`	[0.999, 0.999, 0.001, 0.0001, 1]
`solref`	[0.0001, 1]

Table 5.3: Parameters of the corrector's internal model $\hat{f}$ that differ from the planner's internal model f.

Corrector implementation. As described above, the corrector applies a virtual corrective wrench to its internal estimate of the cube state as shown in (5.14). The corrector gain matrices have 6 diagonal entries corresponding to the virtual forces and torques applied to the cube. We use the values

$$\begin{aligned} C_P &= \text{diag}([1000, 1000, 1000, 3, 3, 3]), \\ C_D &= \text{diag}([1, 1, 1, 0.001, 0.001, 0.001]). \end{aligned} \tag{5.15}$$

The extra gravitational corrective force described above is simply an extra 10N added along the $-z$ axis of the virtual forces applied to the cube.

Additionally, there are a few parameters of the corrector dynamics $\hat{f}$ that differ from the planner's internal model f, which we show in Table 5.3.

5.4 Experiments

We thoroughly evaluate the DROP architecture via hardware and simulation experiments designed to answer the following questions:

1. Can DROP perform robust and dynamic cube reorientation in hardware (Sec. 5.4.2)?
2. How do different components of DROP contribute to its overall performance and reliability (Sec. 5.4.3)?
3. How do modeling and state estimation error affect DROP's performance (Sec. 5.4.4)?

Overall, we find that DROP can reliably perform cube reorientation in hardware, achieving performance comparable with RL-based methods while exhibiting surprising robustness and dexterity.

5.4.1 Hardware Setup and Experimental Details

For all experiments, we perform in-hand rotation of a 3D-printed cube with 7cm side lengths and a fixed-base LEAP hand [90] with palm angled downwards at 20° so that the cube slides off without intervention (see Fig. 5.1 and 5.4). We use a 128-thread Ryzen Threadripper Pro 5995wx CPU to plan rollouts in parallel. To capture images, we use three ZED Mini cameras with RGBD channels and perform keypoint estimation on 256x256 center-cropped images at VGA resolution. The estimator runs at about 90Hz, while the planner frequency fluctuates between 25-50Hz.

For our trials, we compare three planners: PS and CEM (the two sampling-based methods discussed in Sec. 5.3.1), as well as iLQR [152], a gradient-based method. PS and CEM use $N = 120$ rollouts while iLQR devotes 120 threads to parallel line search. CEM uses $M = 4$ elite samples. When tuning hyperparameters (such as cost weights), we prioritize robustness at the expense of rotation speed. For all experiments, we used identical costs, code, and hyperparameters for performing simulation rollouts, state estimation, and communicating with hardware. Unlike previous work [27, 120], we did not observe any significant degradation of our hardware stack (e.g., overheating) during testing. Additional hardware details, including the hand gains, the construction of the physical cube, and manual camera calibration procedures, are recorded in Sec. 5.8.4.

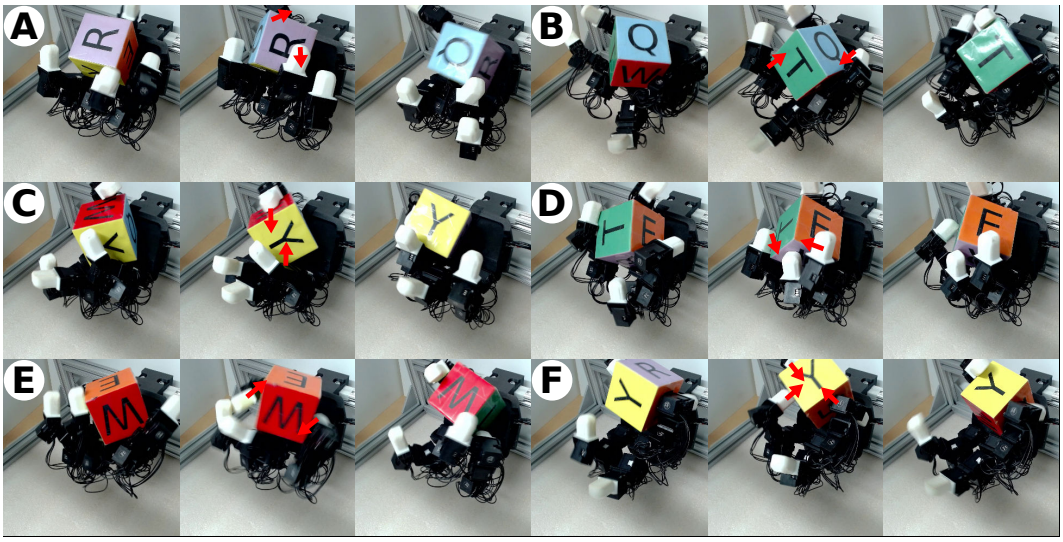


Figure 5.4: **Examples of rotations.** CEM can discover many contact-rich plans for cube reorientation. The red arrows show where forces are primarily applied to achieve rotations. **(A)** The middle finger pushes down on a cube edge while the base of the thumb lifts the opposite corner, rotating the Q face up. **(B)** The ring finger and base of the index finger push on opposite corners to rotate the T face up. **(C)** The thumb pulls down on the W face while the base of the ring finger pushes on the opposite corner to rotate the Y face up. **(D)** The index finger pushes down on the edge of the T face while the ring finger swipes left on the E face to rotate it up. **(E)** The ring finger first swipes inwards, then the thumb quickly follows to pull the W face up. **(F)** The thumb and the ring finger pinch and lift the cube, then the index finger pushes on an edge to rotate the Y face up. The cube is calmly lowered onto the palm.

5.4.2 Main Hardware Results

We begin by presenting results of hardware trials using the full stack shown in Fig. 5.1 and discussed in Sec. 5.3. We study each planner by running 10 trials of the cube reorientation task and analyzing the associated rotation and timing statistics. For examples of interesting rotations, see Fig. 5.4, and for quantitative results, see Table 5.4B. Following prior work, we report statistics assuming the run ends if 80s have elapsed without reaching a goal. Since this cutoff is arbitrary and we used a fairly conservative cost that slows rotation rate, we also report results for the same runs while ignoring the timeout period, ending only when the cube is dropped. For iLQR and PS, this did not change the results.

CEM greatly outperforms PS and iLQR. While iLQR is unable to achieve any rotations and PS only achieves single-digit rotations in hardware, CEM is able to achieve dozens of rotations. Moreover, the rate that it can rotate the cube is also

	Planner	Num Rots (Sorted)	Mean Rots	Median Rots	Mean Rot/s ↑	Median Rot/s ↑
(A) Prior Work	Dactyl [27]	9, 12, 13, 19, 28, 29, 29, 32, 43, 50	26.4 ± 13.4	28.5	Not Reported	Not Reported
	Dextreme [120]	1, 6, 6, 10, 10, 18, 18, 36, 61, 112	27.8 ± 19.0	14.0	Not Reported	Not Reported
(B) Main Results	iLQR	0, 0, 0, 0, 0, 0, 0, 0, 0, 0	0 ± 0	0	N/A	N/A
	PS	0, 0, 1, 1, 1, 2, 2, 3, 4, 5	1.9 ± 1.58	1.5	0.063 ± 0.070	0.033 (0.021, 0.061)
	CEM	2, 2, 11, 11, 17, 22, 27, 34, 39, 48	21.3 ± 14.8	19.5	0.090 ± 0.081	0.062 (0.038, 0.106)
	CEM (No T/O)	2, 10, 11, 11, 22, 39, 48, 53, 59, 81	33.6 ± 24.9	30.5	0.086 ± 0.085	0.063 (0.038, 0.103)
(C) CEM Ablations (No T/O)	RGB Only	1, 1, 12, 13, 20, 37, 52, 79, 94, 128	43.7 ± 41.4	28.5	0.081 ± 0.106	0.053 (0.030, 0.089)
	No Corrector	3, 4, 6, 9, 11, 14, 17, 20, 40, 97	22.1 ± 27.0	12.5	0.065 ± 0.063	0.046 (0.029, 0.080)
	Half Rollouts	2, 7, 17, 21, 21, 27, 29, 37, 44, 65	27.0 ± 17.4	24.0	0.062 ± 0.070	0.040 (0.022, 0.072)

Table 5.4: **Hardware experiments. (A)** Prior works using RL for cube reorientation. Note that [27] performs *axis-aligned* rotations. Best results are shown. **(B)** Among online planners, CEM clearly performs the best. All planners use 120 threads, RGBD images, and the cube pose corrector. We note that when ignoring the 80s timeout imposed in [27, 120], CEM achieves higher mean and median rotation counts than Dactyl and Dextreme. **(C)** Hardware ablations show that using depth images slightly improves rotation rate, but using the corrector and as many threads as possible substantially boosts performance.

nearly 50% faster than PS, suggesting that it can discover and/or execute contact-rich plans much more effectively.

CEM discovers nontrivial contact sequences. CEM finds contact-rich plans that leverage contacts with all parts of the hand. Many rotations are only feasible when the hand first partially rotates the cube using an initial contact sequence, then completes the rotation by gaiting the cube to a different set of contacts. For example, in Fig. 5.4D, the cube is first pushed forward and continually supported by the thumb, allowing the index and ring fingers to then swipe in opposite directions to rotate the cube. Similarly, in Fig. 5.4F, the thumb and ring finger first pinch and lift the entire cube, which allows the index finger to rotate it about the pinched axis before the cube is safely lowered back onto the palm.

Moreover, many discovered plans exploit contact modes that are traditionally challenging to find, like sliding on edges and corners. To execute these motions, the planner employs intuitive strategies like maximizing torque by levering the cube close to a corner or edge. Lastly, our conservative cost function also induced safeguarding behavior, where fingers preemptively blocked the cube from the palm's edges or carefully supported the cube during risky rotations.

The gradient-based iLQR planner is not viable. Corroborating recent work [122, 153], we find that the stiff dynamics of the cube reorientation task prevent gradient-based methods from effectively finding good plans most likely due to poor numerical conditioning, causing jerky, erratic behaviors that do not lead to coherent rotation

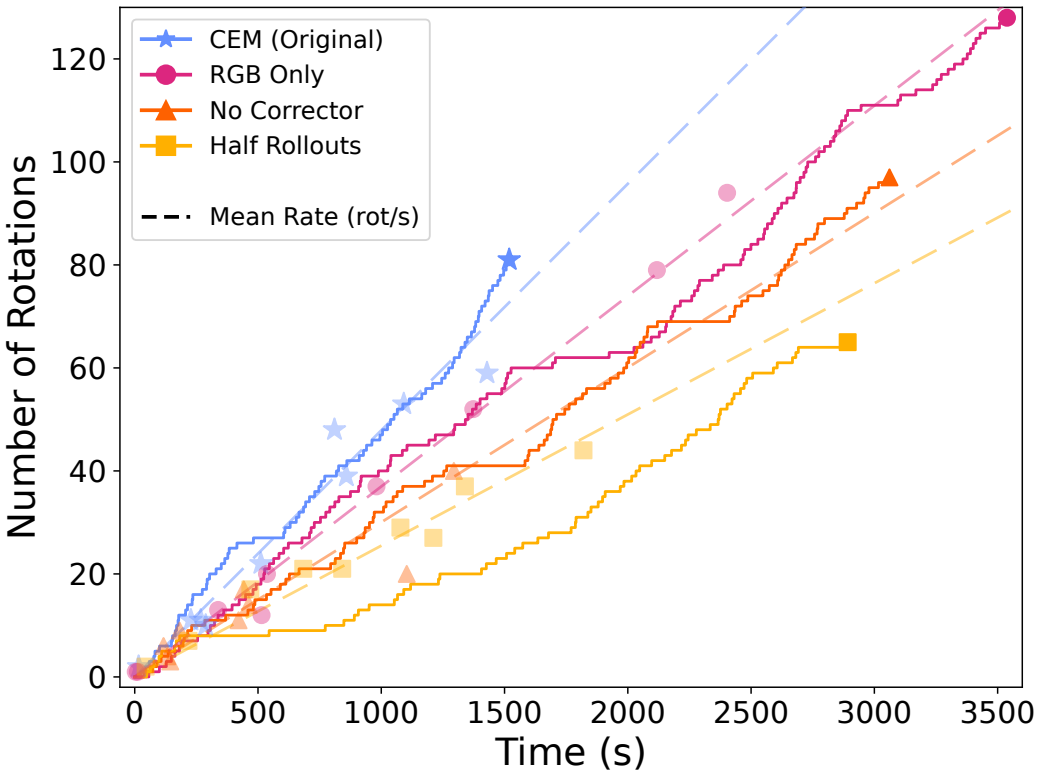


Figure 5.5: **CEM ablation rotation rates.** We use rotation rate as a proxy for planner robustness, as slower rates correspond to "stuck" plans or repeatedly failed moves. Markers are all rotations vs. times for CEM and all ablations. The dashed lines show the mean rotation rates: it is clear that all ablations decrease the rate, which justifies our design of the DROP architecture. The solid lines show individual rotations for each method's longest streak. The long, flat regions correspond to the planner getting "stuck" in local minima.

sequences.

DROP performs comparably to RL. While it is challenging to compare our results to prior RL-based methods like Dactyl [27] or Dextreme [120] due to many factors distinguishing each setup, like hand morphology, cube size, physical properties, camera type, or vision model, Table 5.4A/B shows that our simple CEM planner approaches the performance of RL-based rotation policies (outperforming them when ignoring timeouts) with similar dexterity (see Fig. 5.4).

5.4.3 Hardware Ablations

To understand the impact of key design choices in the DROP architecture, we conducted a series of single-variable ablations with the CEM planner on hardware (Table 5.4C).

Depth improves performance, but only marginally. We compared keypoint detection models trained on RGB versus RGBD images. While RGBD slightly improved rotation rates, it did not significantly outperform RGB, which still achieved 128 rotations, the longest sequence ever observed for DROP. Despite this noisy result, RGB's median rotations (28.5) were still slightly lower than RGBD's (30.5), and overall, the relative rotation rates suggest that depth provides a minor improvement in state estimation accuracy.

The corrector is key for performance. We tested DROP when ablating the corrector, passing raw smoother estimates $\tilde{\boldsymbol{q}}$ directly to the planner. This significantly reduced performance, with mean rotations decreasing by 33%, median rotations by 60%, and rotation rate by 25%. Without the corrector, the planner often became trapped in local minima and long periods of inactivity (flat regions of rollouts in Fig. 5.5). This was caused by the exploitation of non-physical forces in rollouts arising from non-physical hand-cube interpenetration, leading to unrealistic predictions.

DROP is sensitive to the number of rollouts. We reduced the number of rollouts from 120 to 60 and elite samples from 4 to 3, resulting in a 20% decrease in mean rotation count and an over 25% reduction in rotation rate. This highlights the importance of variance reduction via sufficiently-high sample quantity in SPC, and suggesting that improving search efficiency could significantly boost performance.

All ablations increased rotation rate variance. This consistent pattern demonstrates that depth measurements, the corrector, and sufficient rollout quantity all contribute significantly to the planner's reliability and consistency, which justifies the design of the DROP architecture.

5.4.4 Simulated Robustness Study

Lastly, we study DROP's robustness to model and estimation errors by conducting controlled trials in simulation, letting us isolate the planner from the estimation pipeline. We compared iLQR, PS, and CEM under various corrupted conditions, simulating system physics with a 2ms timestep while planner threads utilized a separate physics model with a 10ms timestep. Each configuration underwent five trials, ending upon cube drop, 80s timeout, or 150 rotations.

Specifically, we intentionally induce two types of errors that we believe contribute to real-world brittleness: (i) mis-tuning the planners' internal value of the hand K_p gains, and (ii) corrupting pose estimates from the simulation with a 0.1s lag and additive noise (simulated using a bounded random walk to mimic the state-

correlated bias of real perception errors, with implementation details in Sec. 5.8.5). When corrupting K_p, we study two cases: multiplying the true value by 1.25x and 1.5x, as the results were enlightening for comparing different planners. We also study the effect of the estimator and K_p errors together.

CEM is the most robust planner. Table 5.5 shows that CEM is the best planner under all error conditions. For example, while PS achieves a higher mean rotation count under perfect conditions than CEM, when K_p is mildly corrupted up to 1.25x, PS immediately achieves fewer mean rotations than CEM while suffering an over 2x decrease in rotation rate. At 1.5x, PS hardly rotates the cube at all, while CEM achieves 32.2 mean rotations, and even with the most aggressive errors, CEM was able to achieve dozens of rotations. CEM's superiority can be attributed to its strategy of recomputing π_θ from multiple rollouts, in contrast to PS's single-rollout approach. This strategy appears to be key for robustness, providing a plausible explanation for CEM's markedly better performance in hardware.

Rot/s is a good proxy for robustness. Based on empirical observation, we find that low rotation rates are typically caused by the failure to execute precisely-planned motions or the inability of the planner to escape local minima, resulting in the cube being "stuck," which can be attributed to model or estimation error. CEM's superior speed in these simulations supports our assessments of its robustness on hardware.

Error types have distinct failure modes. While incorrect K_p values primarily resulted in timeouts, corrupted estimates typically caused cube drops. This suggests that perfect state estimates allow for safe "caging" even with poor actuation models, but estimation errors during precise maneuvers often cause drops, as the planner underestimates rollout risk.

5.5 Open-Source Tooling for Sampling-Based MPC

The preceding sections demonstrated that sampling-based predictive control can achieve performance comparable to RL for contact-rich manipulation. However, DROP's implementation relied on MJPC [122], a C++ codebase that, while performant, presents barriers to rapid prototyping: modifying tasks or reward functions requires rebuilding C++ code, and extending the system to new hardware platforms demands substantial engineering effort. More broadly, the sampling-based MPC community lacked shared tools for prototyping, benchmarking, and deploying controllers, reducing the *frictionless reproducibility* [154] needed for rapid development.

Planner	Change	Mean Rot/s ↑	Mean Rots	Drops	Timeouts
iLQR	None	N/A	0 ± 0	3	2
PS	None	0.130	122 ± 54.8	1	0
	K_p x1.25	0.060	44.4 ± 26.4	1	4
	K_p x1.5	0.042	1.2 ± 1.17	0	5
	Est.	0.032	3.2 ± 1.72	5	0
	Est., K_p x1.25	0.048	3.4 ± 3.14	5	0
	Est., K_p x1.5	0.015	0.2 ± 0.40	3	2
CEM	None	0.222	95.4 ± 34.8	0	4
	K_p x1.25	0.148	52.8 ± 26.0	0	5
	K_p x1.5	0.100	32.2 ± 26.6	0	5
	Est.	0.120	46.8 ± 24.7	5	0
	Est., K_p x1.25	0.083	73.8 ± 26.4	4	1
	Est., K_p x1.5	0.058	24.8 ± 13.3	2	3

Table 5.5: **Simulated robustness tests.** We intentionally degrade planners to test their robustness by (i) tuning the hand proportional gain K_p too high, and (ii) corrupting the estimator with noise and lag (denoted "Est."). iLQR failed to achieve any rotations even with perfect information. We observe that PS degrades substantially more than CEM in the presence of model and estimation error, which suggests that CEM may transfer well to hardware.

To address this, we developed `judo`[2], an open-source Python library for developing, tuning, and deploying sampling-based MPC algorithms. The design of `judo` is motivated by three core goals:

1. Maximize research velocity. `judo` is implemented in Python, offering a balance of rapid prototyping and performance. Its simple yet extensible interfaces facilitate extension and contribution to the codebase. An integrated GUI (Fig. 5.6) built on `viser` [155] provides real-time visualization and interactive parameter tuning—including live reward function switching, thread count adjustment, and algorithm comparison—significantly shortening the development loop.

2. Provide accurate and fast simulation. `judo` uses MuJoCo as its physics backend, enabling simulation of complex scenes and geometries while leveraging MuJoCo's highly-optimized, multithreaded forward dynamics for online search.

[2]Code available at https://github.com/bdaiinstitute/judo.

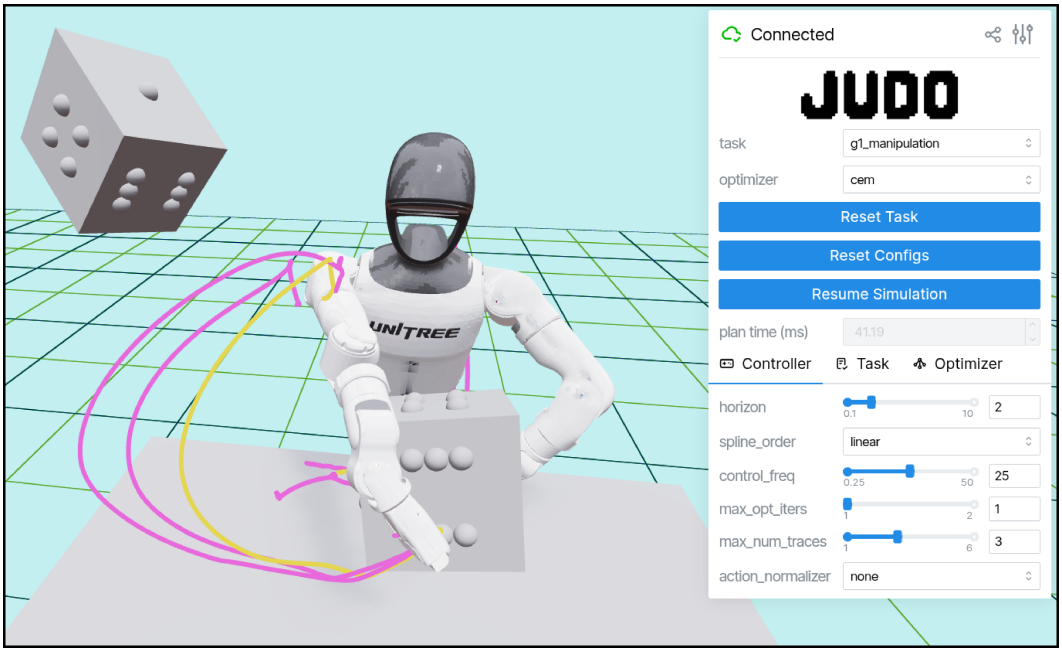


Figure 5.6: The `judo` interface. The GUI is interactive, allowing users to tune parameters in real time. Dropdown menus allow switching between different tasks and controllers.

Crucially, `judo`'s simulation backend is modular, allowing researchers to integrate alternative physics engines and custom dynamics models as needed.

3. Streamline sim-to-real transfer. `judo` separates control logic from physics simulation, running them in asynchronous threads via `dora` [156], a lightweight middleware. This design allows straightforward hardware deployment: researchers implement a hardware interface and substitute it for the simulation component with minimal changes to their controller code. This maximizes code reuse and provides an accurate preview of the control stack's performance prior to deployment.

5.5.1 Controller Interface and Task Suite

At the core of the `judo` API are *tasks* and *optimizers*. The task specifies the reward and system dynamics, while the optimizer specifies the sampling and policy update procedures. `judo` ships with implementations of predictive sampling, CEM, MPPI [16], and CMA-ES, along with a comprehensive suite of benchmark tasks (Fig. 5.7) spanning manipulation, locomotion, and loco-manipulation. Users can benchmark all available algorithms across tasks with standardized evaluation protocols.

`judo` also implements several algorithmic features applicable to all optimizers. *Action space normalization* rescales each control dimension (via min-max or running

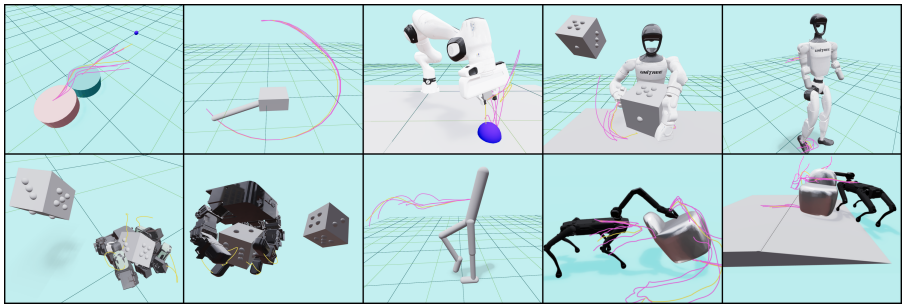


Figure 5.7: The tasks in the `judo` suite. Every task except the cartpole is contact-rich, and there is a wide range of task difficulty, model complexity/morphology, and application area.

statistics) to handle systems with heterogeneous actuator scales. *Noise ramping* applies higher noise to decision variables further in the future, balancing local convergence with global exploration:

$$u_i[k] \sim \mathcal{N}(\mu_i[k], k \cdot \gamma_{\text{ramp}} \cdot \sigma_i[k]), \tag{5.16}$$

where k is the spline index along time and γ_{ramp} is a linear ramping factor. Even this simple schedule substantially improves performance across all optimizers.

5.5.2 Computational Performance

A key concern is whether a Python-based framework can match the speed of C++ tooling like MJPC. Table 5.6 compares control update timings for `judo`, MJPC, and `hydrax` [157] (a JAX-based alternative) on two shared tasks. While the `judo` interface is written in Python, it directly calls MuJoCo's multi-threaded C rollout function, which is by far the computational bottleneck. This eliminates any speed gap: `judo` matches MJPC on the Walker task and slightly outperforms it on the contact-heavy LEAP Cube task (which we attribute to inefficient noise sampling routines in MJPC).

Task	Hydrax (CPU)	Hydrax (GPU)	MJPC	`judo`
Walker	0.1326	0.2190	0.0026	0.0028
LEAP Cube	compile failed	0.6187	0.0324	0.0287

Table 5.6: Average control update timing (s) over 1000 steps. The LEAP Cube task failed to compile in under 1 hour for `hydrax` on CPU.

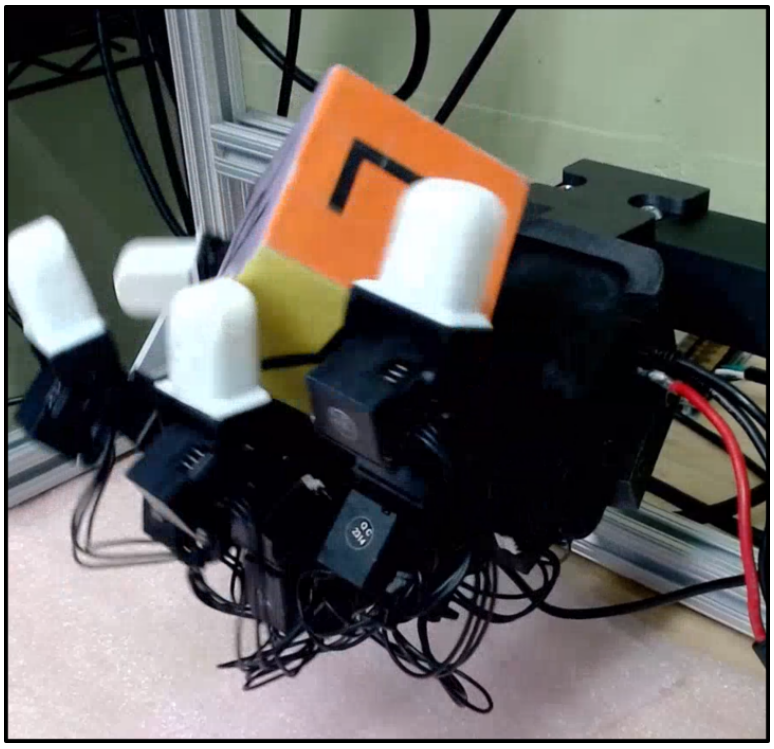


Figure 5.8: Cube rotation setup: a LEAP hand rigidly mounted to a frame rotates a colored cube, whose pose is estimated with a vision-based keypoint tracker [123]. The goal is to reach as many target orientations as possible in sequence without dropping.

5.6 Further Results Enabled by Fast Experimentation

The rapid prototyping afforded by `judo` enabled us to quickly iterate on reward design and deploy controllers on new hardware platforms. We present two case studies illustrating how the tooling led to new results.

5.6.1 Improved Cube Rotation via Tracking-Based Rewards

A key limitation of DROP was its dependence on server-grade hardware: performance degraded sharply when reducing the number of planning threads from 120 to 64, making the method impractical for standard workstations. We hypothesized that this was partly due to the reward structure. The rotational cost ℓ_{g} in (5.6) penalizes distance to a *static* goal orientation. When a complex sequence of intermediate rotations is required, sampling-based rollouts may fail to assign credit to partially-successful trajectories whose value is not realized over a short planning horizon. This problem is exacerbated in low-resource regimes where fewer rollouts lead to higher variance.

To address this, we designed a *tracking-based reward* using `judo`. Instead of

computing orientation error relative to a static target, we continuously generate a short reference trajectory from the cube's current orientation to the goal via spherical linear interpolation at a fixed desired angular velocity. The planner's reward then penalizes deviation from this moving target, transforming a hard global search problem into a sequence of local tracking problems. Designing and tuning this reward—including adjusting weights, optimizer noise, and the planning horizon—took roughly one day from ideation to hardware deployment, enabled by `judo`'s interactive GUI and live parameter tuning.

Results. We directly compared the original static-goal reward from DROP with the tracking-based reward at only 16 planning threads, a 7.5× reduction from the 120 threads used in DROP. Table 5.7 summarizes the results. Over ten trials, the static-goal reward failed to achieve a single rotation, while the tracking-based reward achieved an average of 21.2 consecutive rotations (median 20.0). For context, DROP with 120 threads and the static reward achieved 33.6 mean rotations. This demonstrates that modest reward shaping can substantially improve performance in resource-constrained settings, broadening the practical applicability of sampling-based MPC.

Method	Mean Rots	Median Rots
DROP reward – 120 threads [123]	33.6	30.5
DROP reward – 16 threads	0.0	0.0
Tracking reward – 16 threads	21.2	20.0

Table 5.7: Consecutive cube rotations on hardware. The tracking-based reward, designed using `judo`, recovers strong performance with 7.5× fewer planning threads.

Several aspects of `judo` were critical for this result. The thread count is directly exposed in the GUI, allowing performance to be profiled across different settings without restarting the system. Reward functions can be swapped interactively, enabling immediate comparison of static vs. tracking objectives. Finally, integrating the new task into the existing hardware stack required only lightweight IPC wrappers, since `judo`'s core API is agnostic to the choice of middleware.

5.6.2 Spot Loco-Manipulation

Beyond improving existing tasks, `judo` also enabled prototyping entirely new ones. We consider two loco-manipulation tasks using a Spot robot with a mounted arm: (i) uprighting a fallen chair (Fig. 5.9, top), and (ii) pushing a heavy chair on casters up a ramp (Fig. 5.9, bottom). Both tasks require whole-body contact—leveraging the

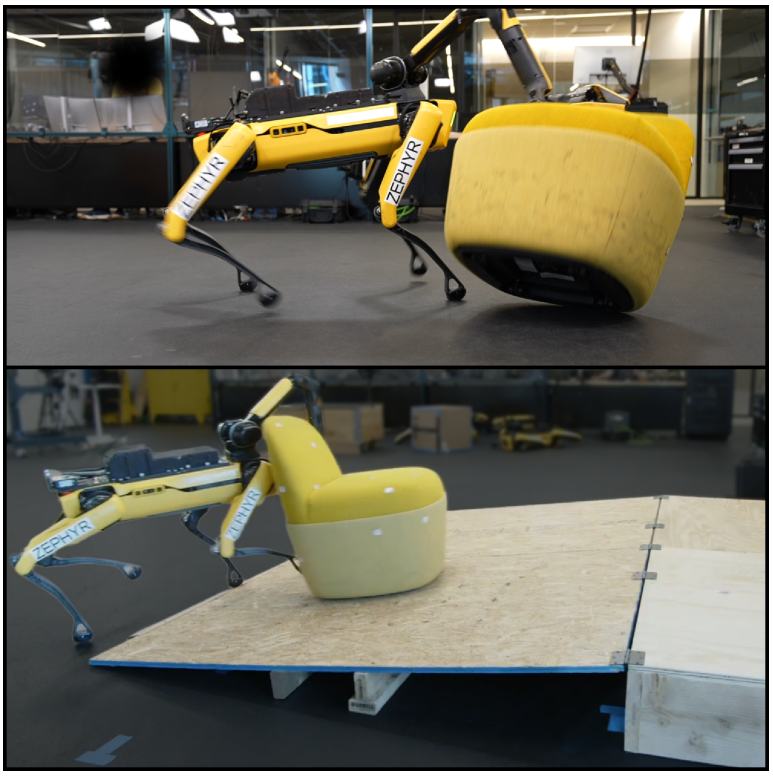


Figure 5.9: A Spot robot uses its whole body to upright a fallen chair (top) and push a chair up a ramp (bottom). A low-level RL policy controls locomotion while `judo` samples high-level base velocity and arm commands for manipulation.

arm, legs, and torso together—to generate sufficient force against gravity, making them representative of loco-manipulation problems with many contact modes.

These tasks illustrate the use of `judo` within a hierarchical control architecture. A low-level RL locomotion policy (trained to track base velocities and arm joint angles) handles balance and walking, while a high-level sampling-based planner commands base velocity and arm positions for manipulation. A multi-term reward incentivizes the robot to stay upright, remain close to the chair, and move it to the desired configuration. Hyperparameters were first selected in simulation and directly deployed on hardware by toggling a single entrypoint flag. `judo`'s interactive GUI enabled real-time parameter tuning directly on the robot, which was crucial for identifying parameters robust to the unmodeled compliance of the chair.

Results. Using CEM with 17 threads and a 3-second rollout horizon, Spot successfully uprighted the chair in 7/10 trials and pushed it up the ramp in 5/10 trials within time limits (Table 5.8). Failures were mostly due to unmodeled compliance in the chair, highlighting sampling-based MPC's sensitivity to the sim-to-real gap.

Nonetheless, in all trials, Spot reliably employed its arms, legs, and body together to manipulate the chair, showcasing the ability of sampling-based MPC to generate highly contact-rich whole-body motions.

Task	Avg Episode Len	Successes	Time Limit
Spot Uprighting	14.1s	7/10	30s
Spot Ramp	93.9s	5/10	120s

Table 5.8: Performance of Spot loco-manipulation tasks on hardware using CEM with 17 planning threads.

This case study demonstrates that `judo` supports layered control architectures (RL locomotion + sampling-based planning), enables online hardware tuning for addressing modeling issues, and provides a sampling-based framework capable of handling the discontinuous contact dynamics inherent in loco-manipulation. The modular API allowed seamless transfer between simulation-only prototyping and hardware deployment.

5.7 Conclusion

This chapter studied the use of physics simulators as approximate models for contact-rich manipulation via sampling-based predictive control. The DROP system demonstrated that a simple CEM planner, paired with a vision-based pose estimator and a collision-aware corrector, can achieve in-hand cube reorientation performance comparable to RL-based methods, all without offline training or domain randomization. The key insight is that a simulator, despite its approximation errors, provides a sufficiently accurate model for online planning when combined with closed-loop feedback. Most surprisingly, this is even true when using approximate contact models that *knowingly* rely on non-physical approximations (like force-at-a-distance) in order to achieve better numerical stability. The resulting efficiency gains allow faster loop closure, which in our experience benefits closed-loop performance more than additional model fidelity, highlighting the importance of the tight coupling between approximate model design and algorithm choice.

The development of `judo` showed that open-source tooling is critical for practical performance: by enabling rapid prototyping, interactive tuning, and straightforward sim-to-real transfer, `judo` led to a tracking-based reward that recovered strong cube rotation performance with 7.5× fewer planning threads, and enabled deployment on a new hardware platform (Spot) for loco-manipulation tasks. These results

underscore that the practical impact of sampling-based MPC depends not only on algorithmic design but also on the quality of the tools available to researchers.

Parts I and II of this thesis have relied on structured physical models—analytic contact mechanics and physics simulators, respectively—as the approximate models driving algorithm design. The next part of this thesis asks what happens when such structure is unavailable to the modeler: given only observed data from a complex or unknown process, with no privileged notion of state, can we learn models suitable for analysis, prediction, and control?

5.8 Appendix: Implementation Details

This appendix records implementation details for DROP's components and experiments: the synthetic dataset and augmentations used to train the keypoint predictor, its training procedure, the smoother parameters, additional hardware details, and the noise model used in the simulated robustness study.

5.8.1 Keypoint Predictor: Dataset and Augmentations

This section details the design decisions behind the synthetic dataset and augmentations used to train the vision model g_φ from Sec. 5.3.2.

The training images were generated per the Movi-F dataset convention from Kubric. The images are sampled from frames of a video generated by simulating a random collection of objects (always including the cube) thrown into the scene from various positions and heights, with random camera motion and motion blur. The non-cube objects are drawn randomly from the Google Scanned Objects dataset, and backgrounds are taken randomly from PolyHaven.

These data were split into train and test datasets by first separating videos into train and test videos then by randomizing the order of all images from these videos. This ensured that very similar frames from the same video did not appear in both the train and test sets. Each video supplied 24 images. In these videos, the cube's size was normalized such that its side length was 2.0m. Thus, when using the keypoint detector model downstream in conjunction with the smoother, the cube size must first be normalized.

Visual/material properties of the cube were randomized using `kubric` by uniform random sampling (see Table 5.9).

The camera pose was sampled such that it was always pointed at the origin, but its

Property	Sampling Range
Roughness	[0.0, 0.3]
Specularity	[0.75, 1.0]
Metallic	[0.25, 0.75]
Index of Refraction	[1.0, 2.0]

Table 5.9: Cube material randomizations.

position was uniformly randomly located in some hemispherical shell with inner radius 7.0m, outer radius 9.0m, and z height exceeding 0.2m.

As mentioned above, images in the dataset where too few or too many pixels corresponded to the cube were removed, as this indicated that the cube was either too occluded or too close to provide an accurate keypoint estimate. Specifically, an image was included in the dataset only if the proportion of cube pixels was in the interval $[0.02, 0.7]$.

Lastly, we note that while the size of the training dataset is fairly large for such a specific task (about 686000 images), it is far fewer than number of images used to train cube pose estimators in other works (e.g., 5 million in [120] and we approximate 76.8 million in [27]). However, it is clear that an improved different approach will need to be used in order to generalize to different objects.

One of the key factors for robust keypoint prediction in the real world was the application of extensive image augmentations during training. The augmentations were randomly sampled and applied during training using a mix of implementations from the open-source image processing package `kornia` and custom augmentations described below. The `kornia` augmentations and parameters are described in Table 5.10. For detailed descriptions of the parameters, please see the `kornia` documentation. Any time the parameter `p` appears, it denotes the probability that the augmentation is applied.

We implemented custom augmentations that applied only to depth images, as well as a custom augmentation for transplanting random backgrounds onto training images. We describe these now and summarize the associated parameters in Table 5.11.

For the depth readings, we (i) uniformly randomly sample a per-image bias (`DepthBias`), (ii) add Gaussian noise (`DepthGaussianNoise`), and (iii) uniformly randomly sample near and far planes centered about some mean value, where everything too close or far is set to 0.0 (`DepthPlane`).

Augmentation	Parameters
RandomAffine	degrees: 90 translate: (0.1, 0.1) scale: (0.9, 1.5) shear: 0.1
RandomErasing1	p: 0.5 scale: (0.02, 0.1) ratio: (2.0, 3.0) same_on_batch: False
RandomErasing2	p: 0.5 scale: (0.02, 0.05) ratio: (0.8, 1.2) same_on_batch: False
RandomPlanckianJitter	mode: blackbody
ColorJiggle	brightness: 0.2 contrast: 0.4 saturation: 0.4 hue: 0.025
RandomGaussianBlur	kernel_size: (5, 5) sigma: (3.0, 8.0) p: 0.5
RandomPlasmaShadow	All defaults

Table 5.10: All RGB image kornia augmentations used during training. Augmentation parameters are defaults if unspecified.

Lastly, we implemented a special augmentation called CustomTransplantation (not to be confused with the RandomTransplantation augmentation in kornia), which requires a batch of RGBD images as well as segmentation masks for the cube. With some probability, each image in the batch (the acceptor) has all of its non-cube pixels replaced by pixels from another image in the batch (the donor) using the following procedure. First, all non-cube pixels in the acceptor are identified, which initializes a background mask. Second, all pixels in the donor that are closer to the camera than the corresponding pixels in the acceptor are added to the background mask. Third, the pixels in the donor mask corresponding to the cube in the donor are removed from the mask (to prevent there from being two cubes in a single image). Fourth, the background mask selects pixels from the donor and replaces the corresponding pixels in the acceptor. Finally, if the resulting transplanted image

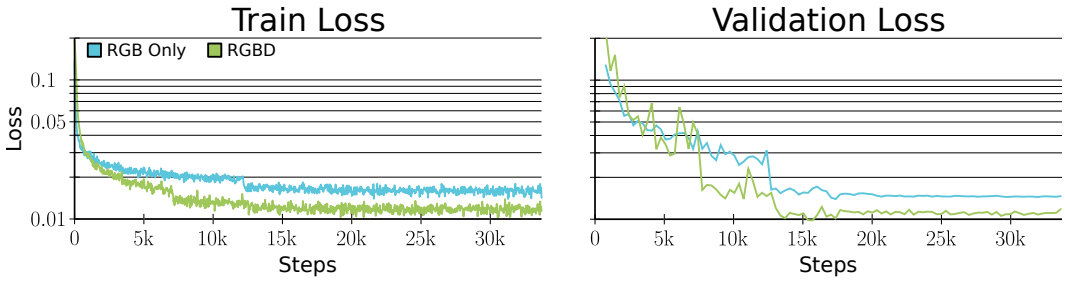


Figure 5.10: Training and validation loss curves for the RGBD and RGB-only vision models. Note that the curves are log scale.

results yields an image that has too large or small of a ratio of visible cube pixels, the transplantation is not applied. This transplantation method allows backgrounds from donors to further obscure the cube in acceptors in random ways, which helps improve robustness.

Augmentation	Parameters
`DepthBias`	`bias_range: (-0.02, 0.02)` `p: 0.5`
`DepthGaussianNoise`	`stdev: 0.005`
`DepthPlane`	`near_mean: 0.1` `near_range: (-0.05, 0.05)` `p_near: 0.5` `far_mean: 0.5` `far_range: (-0.05, 0.05)` `p_far: 0.5`
`CustomTransplantation`	`p: 0.5` `ratio_lb: 0.02` `ratio_ub: 0.7`

Table 5.11: Parameters for custom augmentations.

5.8.2 Keypoint Predictor: Training and Validation

Training hyperparameters are described in Table 5.12. We trained the model using a machine with 8 H100 GPUs for about 8 hours, though we remark that using this many GPUs is not required.

We show the training and validation curves for the RGBD model as well as the RGB-only model used in the ablation study from Sec. 5.4.3 in Fig. 5.10. Overall, the RGBD model seems to perform slightly better, but not by a large margin. Note that the figure is in log scale.

Hyperparameter	Value
batch_size	256
learning_rate (initial)	1e-3
epochs	100
num_workers	4
AMP	True
Loss Function	MSE
Optimizer	AdamW
Scheduler	ReduceLROnPlateau patience: 5 factor: 0.25 min_lr: 1e-6 grad_scaler: True

Table 5.12: Training Hyperparameters.

5.8.3 Smoother Parameters

We summarize all smoother parameters in Table 5.13. All quaternions are reported in wxyz order. Noise parameters are given by standard deviations σ corresponding to diagonal components of the noise model covariance matrices. Rotational noise is represented by Gaussian noise in the tangent space. Huber regularization is applied identically to all noise models.

5.8.4 Additional Hardware Details

While the stack described in Sec. 5.3 is fairly simple, there are a few miscellaneous details that affect performance, which we record here.

LEAP hand gains. The scale of the gains in the simulated model and on hardware are quite different. While in simulation, the planner's LEAP hand gains are $k_p = 1.0$ and $k_v = 0.01$, on hardware, we use $k_p = 75$ and $k_v = 25$. In fact, the hardware gain values are already quite low compared to the values recommended in [90]. This was intentional; by lowering the gains, we lowered the amount of energy imparted on the cube, and allowed the controller to manipulate the cube with a higher degree of control.

The physical cube. The cube itself is made of low-density 3D-printed PLA, weighing 0.108kg. The faces of the cube are simply printed out on regular printer paper and taped onto the cube using reflective, low-friction packing tape. We remark that over time, the frictional properties of the cube may slightly change. Our reported experiments were run using a cube that was used for about 1 week prior, and

Parameter	Value
Cube Pos Prior	$p = [0.1, 0.0, 0.0]$ $r = [1.0, 0.0, 0.0, 0.0]$ $\sigma_p = [0.5, 0.5, 0.5]$ $\sigma_r = [1.0, 1.0, 1.0]$
Cube Vel Prior	$v = [0.0, 0.0, 0.0]$ $\omega = [0.0, 0.0, 0.0]$ $\sigma_v = [0.01, 0.01, 0.01]$ $\sigma_\omega = [0.2, 0.2, 0.2]$
Dynamics Pos Noise	$\sigma_p = [0.01, 0.01, 0.01]$ $\sigma_r = [0.2, 0.2, 0.2]$
Dynamics Vel Noise	$\sigma_v = [0.01, 0.01, 0.01]$ $\sigma_\omega = [0.2, 0.2, 0.2]$
Measurement Noise (in pixels)	$\sigma_z = [3.0, 3.0]$
Meas. Robustness Parameter k_{huber}	1.345
Lookback Window	1

Table 5.13: Smoother parameters.

anecdotally, the friction increased slightly over this period, which helped prevent the cube from slipping out of the hand. In the weeks prior to our final experimental trials, older versions of the cube also exhibited peeling tape, torn faces/edges, and other effects that substantially increased the friction of the cube. For the most repeatable results, we would recommend re-taping a cube from scratch every few weeks.

Manual calibration of camera poses. Before running experiments, we found it important to perform some manual adjustments of each camera's ground-truth pose with respect to the world frame. To perform the calibration, we streamed the estimated cube pose from the smoother and projected the associated analytical keypoint locations back onto each camera's image frame using the keypoint factors derived earlier. By adjusting the camera position, we were able to manually align the projected keypoints with the corners of the cube.

Manual adjustments to the smoothed cube pose. We found that errors in the camera poses manifested as nearly-constant translational biases in the smoothed cube pose estimate. To remedy this, we simply manually adjusted estimated cube pose by adding an appropriate offset.

5.8.5 Implementation of the Noise Random Walk

In Sec. 5.4.4, we explained that when corrupting the perception stack, we simulate additive noise using a bounded random walk. The reason for this choice is that in real life, perception errors are highly-correlated with state. For example, if there is a 1cm error in cube keypoint estimates at some cube pose, then we typically observe that the error would remain at similar levels with low noise so long as the cube does not move much from that state. Thus, simply adding 0-mean uncorrelated noise with high variance is not an accurate simulated representation of what we observe in the real world.

Instead, by modeling the noise as a bounded random walk with very low noise, we can approximately model the effects of sustained pose estimation bias. Specifically, let $\nu \in \mathbb{R}^3, \xi \in \mathfrak{so}(3) \cong \mathbb{R}^3$ denote the position and rotational noises respectively. These noise values are iteratively updated as follows, where Δt represents whatever amount of time has elapsed since the last update:

$$\begin{aligned} \nu(t + \Delta t) &= \text{clip}\left(\nu(t) + z_\nu(t), \nu_{\text{lb}}, \nu_{\text{ub}}\right), \\ \xi(t + \Delta t) &= \text{clip}\left(\xi(t) + z_\xi(t), \xi_{\text{lb}}, \xi_{\text{ub}}\right), \end{aligned} \tag{5.17}$$

where $\nu_{\text{lb}}, \nu_{\text{ub}}, \xi_{\text{lb}}, \xi_{\text{ub}}$ are fixed lower and upper bounds for the random walk, $\text{clip}(\cdot)$ clips the values in the first argument elementwise to the provided lower and upper bounds, and $z_\nu \sim \mathcal{N}(0, \mathbf{\Sigma}_\nu), z_\xi \sim \mathcal{N}(0, \mathbf{\Sigma}_\xi)$ are randomly-sampled noises that drive the random walk.

Finally, we update the cube pose $\boldsymbol{q}^c = [\boldsymbol{p}^c, \boldsymbol{r}^c] \in \text{SE}(3)$ as follows:

$$\begin{aligned} \boldsymbol{p}^c(t + \Delta t) &= \boldsymbol{p}^c(t) + \nu(t + \Delta t) \\ \boldsymbol{r}^c(t + \Delta t) &= \boldsymbol{r}^c(t) \circ \text{Exp}\left(\xi(t + \Delta t)\right). \end{aligned} \tag{5.18}$$

The parameters we used are summarized in Table 5.14.

Parameter	Value
ν_{lb}	[-0.01, -0.01, -0.01]
ν_{ub}	[0.01, 0.01, 0.01]
ξ_{lb}	[-0.1, -0.1, -0.1]
ξ_{ub}	[0.1, 0.1, 0.1]
$\mathbf{\Sigma}_\nu$	diag([0.001, 0.001, 0.001])
$\mathbf{\Sigma}_\xi$	diag([0.0001, 0.0001, 0.0001])

Table 5.14: Parameters of the noise random walk.

5.9 Appendix: A Primer on Factor Graphs and Incremental Smoothing

The pose smoother described in Sec. 5.3.2 is built on a factor graph. For completeness, this appendix gives a brief, self-contained overview of the factor graphs and incremental smoothing that underpin DROP's estimation pipeline.

A primer on factor graphs. Factor graphs are a type of probabilistic graphical model that models the relationships between unknown random variables via *factors*, which can be interpreted as (unnormalized) likelihood functions of subsets of all of the unknowns parameterized by measurements, which themselves are functions of the same unknowns. The factor graph defines a factorization of some global likelihood function over all unknowns, which gives us a computationally-convenient framework for performing *maximum a posteriori* (MAP) inference to recover an estimate of the unknowns. For further explanation, we refer the reader to [158].

Let X denote the total set of all unknowns and X_l denote some indexed subset of X. Then, we can represent the factors formally as

$$\phi(X) = \prod_l \phi_l(X_l), \tag{5.19}$$

which models the independence relationships between the unknown variables of interest. Now, let measurements be modeled $z_l = h_l(X_l)$, where h_l is some measurement model. Then, with a Gaussian noise model and the assumption that the factors take the form

$$\phi_l(X_l) \propto \exp\left\{-\frac{1}{2}\left\|h_l(X_l) - z_l\right\|^2_{\mathbf{\Sigma}_l}\right\}, \tag{5.20}$$

the MAP solution for the unknown variables is equivalent to the following nonlinear least squares problem:

$$X^{\mathrm{MAP}} = \arg\min_X \sum_l \left\|h_l(X_l) - z_l\right\|^2_{\mathbf{\Sigma}_l}. \tag{5.21}$$

In practice, the MAP inference problem is solved by numerical methods like the Gauss-Newton or Levenberg-Marquardt algorithms, which converge to some local minimum by solving successive linearizations of (5.21).

To reduce sensitivity to outlier measurements (i.e., keypoint detection failures), we used a *robust Huber error model*.

Incremental smoothing. We are interested in the setting where we receive a stream of incremental information and we seek to update the MAP estimate over time. In

particular, we consider a canonical graphical model of a dynamical system with two types of factors: *dynamics* factors that model the (Markovian) temporal transitions between states ($\phi_t^{\text{dyn}}(X_{t+1}, X_t)$), and *measurement* factors that model the likelihood of observations on those states ($\phi_t^{\text{meas}}(X_t; z_t)$).

Because the graph would rapidly grow in size as time elapses, when incrementally smoothing, it is common to *marginalize* out the effect of older unknowns to retain their information content while removing them from the graph (see [158, Sec. 5.3]). This way, the number of variables remains fixed and the estimation problem stays tractable.

This scheme gives rise to *fixed-lag smoothing*, where some lookback window specifies the number of states in the past to maintain in the graph at each time step. The case where the lookback is 1 corresponds to an iterated extended Kalman filter (iEKF), since marginalization happens after each measurement is received, and the MAP problem is solved through successive linearizations (see [159, Sec 4.3.1]).

Part III

Data-driven Modeling for General Temporal Data

Chapter 6

KALMAN-IMPLICIT KOOPMAN OPERATOR LEARNING

The previous two Parts of this thesis were concerned with the development of models based (either directly or indirectly) on physical laws. In so doing, this allowed us to develop approximations heavily informed by our understanding of the physical world. Underlying the success of such physics-based approaches is the notion of the *state* of a Lagrangian mechanical system, which permits a compact description of the system's behavior for the purpose of kinematic or dynamic modeling.

However, it is often the case that the state of a system is intractable to compute or not well-defined, preventing the use of state-based analytic frameworks like Lagrangian mechanics. For example, a system's state may correspond to a noisy measurement of some complicated partial differential equation; a compressed representation of some arbitrary temporal sequence, as in a loose, incomplete collection of sensors; or some high-dimensional visual observation of the world, as represented by something like natural images (or even videos). In all such cases, we must develop *data-driven* techniques that can inform our understanding of the world in the absence of analytical priors.

Part III of this thesis explores these ideas via the three forthcoming chapters on *representation learning*. In contrast to the previous parts, here the approximation is primarily in the state itself rather than in some, e.g., dynamic model of a system of interest. Accordingly, the scope of applications in this Part is far broader than the manipulation-focused tasks explored in the preceding chapters.

Chapter 6 develops a method that uses differentiable Kalman filtering to *implicitly* learn a high-dimensional Koopman embedding space to capture nonlinear disturbance dynamics of some hydrodynamic PDE. This jointly produces approximation of the state and dynamics, which we use for both prediction and control. Moving towards more generic data, Chapter 7 develops a novel method to quantize arbitrary continuous time series signals into large-scale discrete vocabularies using a novel stochastic compression algorithm. We show that the size of the corresponding vocabulary drives performance for tasks like temporal signal imputation across many datasets. Lastly, Chapter 8 develops a highly data-efficient approach toward learning a *semantic* notion of state, and asks how to produce higher-quality spa-

tiotemporal embeddings for video, a notoriously high-dimensional and challenging data modality.

6.1 Introduction

Dynamical modeling and prediction are fundamental problems in fields like robotics and control, forming the bedrock for techniques like model predictive control, state estimation, and more. However, obtaining accurate analytical models in unstructured settings can be challenging, especially for nonlinear systems that may be subject to high-dimensional and/or chaotic spatiotemporal phenomena like turbulence or surface waves. This motivates a large body of work on fitting predictive models from data [160, 161, 162, 163].

One appealing approach is Koopman operator theory, which models the *linear* evolution of state embeddings[1] under the infinite-dimensional *Koopman operator* $\mathcal{K}$. In this lifted space, nonlinear dynamics become linear, effectively trading nonlinearity for representational complexity [161]. In practice, one builds finite approximations of the Koopman operator by *explicitly* parameterizing embedding functions via black-box neural networks [164, 162], fixed "dictionaries" of basis functions (e.g., polynomials) [165], or hybrid approaches that learn these dictionaries [166] (see Fig. 6.1A). However, selecting embeddings that capture an approximately invariant subspace is difficult; poor choices produce spurious modes or overfitted models and often necessitate multi-objective losses with carefully tuned weights [164, 167, 162]. If the learned embedding does not capture an invariant subspace, the Koopman dynamics are not closed, leading to multi-step prediction error.

This chapter presents an alternative approach: ***Kal****man-****I****mplicit* ***K****oopman* ***O****perator* (KALIKO) Learning. Unlike these explicit approaches, KALIKO reframes the problem of encoding data $\boldsymbol{x}$ into embeddings $\boldsymbol{z}$ as a Bayesian state estimation problem that infers latent states $\boldsymbol{z}_{1:T}$ from data trajectories $\boldsymbol{x}_{1:T}$. Thus, KALIKO *implicitly* represents an encoder via an (extended) Kalman filter, allowing it to circumvent the challenge of parameterizing an embedding function (see Fig. 6.1B). We show that despite its implicit structure, KALIKO learns highly interpretable dynamics consistent with Koopman theory and prior works (e.g., eigenfunctions for orbits, limit-cycles, and equilibria), achieves superior prediction performance over several baselines, and translates its capabilities to a challenging closed-loop

[1] We use the term "embedding" rather than "observable" to avoid confusion with "measurements" and "observations" in Kalman filtering.

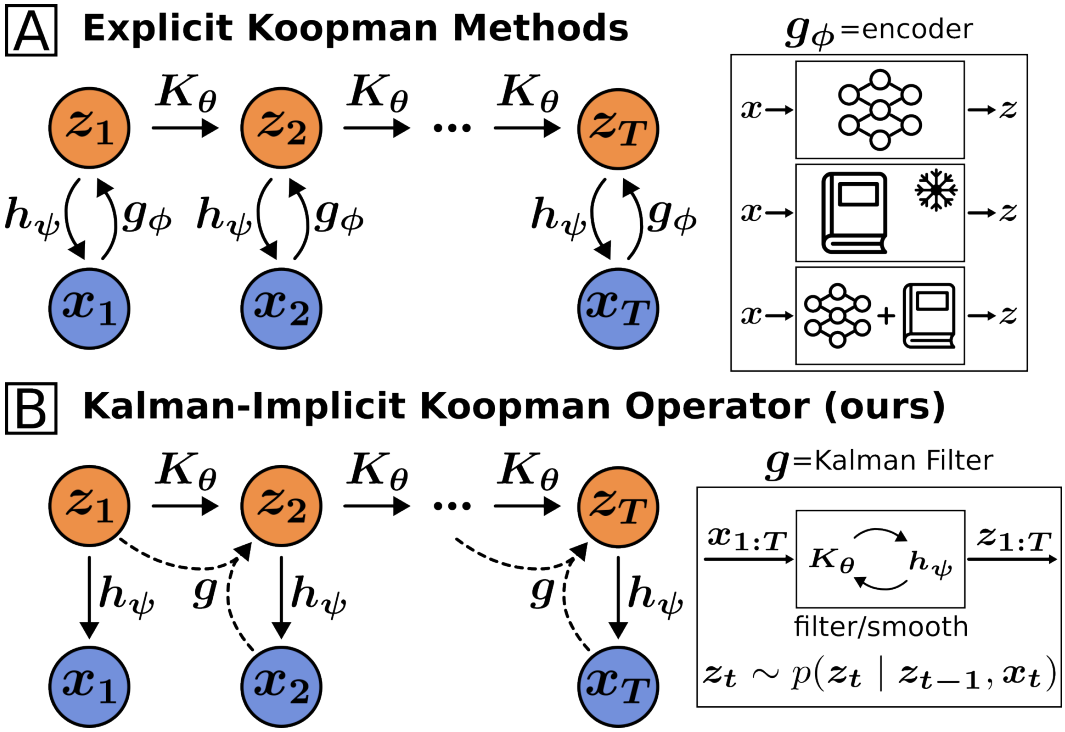


Figure 6.1: **(A)** Most methods for learning the linear Koopman dynamics $\mathbf{K}_\theta$ are *explicit*, parameterizing an encoder $\boldsymbol{g}_\phi$ that maps data $\boldsymbol{x}$ to high-dimensional Koopman embeddings z. Examples include neural networks, fixed dictionaries of basis functions, or learnable dictionaries. **(B)** KALIKO (ours) instead *implicitly* learns Koopman embeddings by letting $\boldsymbol{g}$ be a Kalman filter and smoother, which is composed of only two models: the latent dynamics $\mathbf{K}_\theta$ and decoder $\boldsymbol{h}_\psi$. We show that these implicitly-recovered representations are highly interpretable and yield strong open-loop prediction and closed-loop control performance.

stabilization task.

6.1.1 Related Work

Computing Koopman Embeddings. Throughout this chapter, $\boldsymbol{x} \in \mathcal{X}$ denotes the *data state*, the quantity we actually measure, while $z \in \mathcal{Z}$ denotes the *latent state* of a model, and the dynamics being learned are always the *latent* dynamics $\mathbf{K}$ (all formally defined in Sec. 6.2). Prior methods differ chiefly in how latents are produced from data. A standard data-driven method in the Koopman-related literature is *Dynamic Mode Decomposition* (DMD) [168, 161], which regresses a linear model directly on snapshots of the data,

$$\mathbf{K}_{\text{DMD}} = \arg\min_{\mathbf{K}} \sum_t \|\boldsymbol{x}_{t+1} - \mathbf{K}\boldsymbol{x}_t\|_2^2 .$$

DMD therefore takes the latent state to be the data itself, $\boldsymbol{z}_t = \boldsymbol{x}_t$, and cannot in general realize a nonlinear change of coordinates. *Extended DMD* (EDMD) addresses this by first lifting the data through a *dictionary* of embedding functions $\boldsymbol{g} = (g_1, \ldots, g_m)$, like polynomials, which must be carefully constructed [169, 161]. The latents are defined explicitly as $\boldsymbol{z}_t = \boldsymbol{g}(\boldsymbol{x}_t)$, and the same regression is performed in the lifted space,

$$\mathbf{K}_{\mathrm{EDMD}} = \arg\min_{\mathbf{K}} \sum_t \|\boldsymbol{g}(\boldsymbol{x}_{t+1}) - \mathbf{K}\boldsymbol{g}(\boldsymbol{x}_t)\|_2^2 .$$

Time-delay coordinates provide another practical choice of $\boldsymbol{g}$: Delay-DMD stacks a window of past data into Hankel embeddings $\boldsymbol{z}_t = (\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \ldots)$ to capture approximately Koopman-invariant structure [170]. *Kernel DMD* (KDMD) implements EDMD implicitly, computing only kernel evaluations $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{g}(\boldsymbol{x}_i), \boldsymbol{g}(\boldsymbol{x}_j) \rangle$ between embedded data points, which avoids enumerating the dictionary [162, 171]. Lastly, deep-learning methods parameterize the embedding function (or its dictionary elements) as a neural network $\boldsymbol{g}_\phi$ trained alongside the latent dynamics $\mathbf{K}_\theta$, typically with multi-term objectives of the form

$$\mathcal{L} = \sum_{t,k} \alpha_1 \underbrace{\left\|\boldsymbol{x}_t - \boldsymbol{h}_\psi\left(\boldsymbol{g}_\phi(\boldsymbol{x}_t)\right)\right\|_2^2}_{\text{reconstruction}} + \alpha_2 \underbrace{\left\|\boldsymbol{g}_\phi(\boldsymbol{x}_{t+k}) - \mathbf{K}_\theta^k \boldsymbol{g}_\phi(\boldsymbol{x}_t)\right\|_2^2}_{\text{latent prediction}} + \alpha_3 \underbrace{\left\|\boldsymbol{x}_{t+k} - \boldsymbol{h}_\psi\left(\mathbf{K}_\theta^k \boldsymbol{g}_\phi(\boldsymbol{x}_t)\right)\right\|_2^2}_{\text{data prediction}},$$

where $\boldsymbol{h}_\psi$ is a decoder network, k ranges over prediction horizons, and the weights α_i must carefully balance the competing terms [164, 172, 167, 162, 166].

In every method above, the latent state carries meaning only because it is defined in advance as a fixed function of the data, $\boldsymbol{z}_t := \boldsymbol{g}(\boldsymbol{x}_t)$, whether that function is the identity, a handcrafted dictionary, a delay map, a kernel's implicit feature map, or a trained network. KALIKO departs from this convention entirely: rather than parameterizing an embedding function, it recovers an encoder as Bayesian inference of latents from data trajectories, induced by the learned dynamics and decoder rather than specified in advance (Sec. 6.2).

These diverse approaches share canonical failure modes. For example, even for linear systems, a single spurious basis function can induce severe overfitting [162]. In the data-driven setting, practitioners counter this with heavy regularization [162, 167], complex multi-term losses [164, 173], and/or highly-sensitive weight initializations and hyperparameters [173]. Conversely, convergence of learned dictionaries

is often unclear [167] and poor choices of the above can just as easily trap solutions in bad local minima, leading to underfitting [173, 174].

Kalman Filters and Dynamical Learning. Learnable Kalman filters (KFs) jointly train a latent dynamics model and decoder that maximize the likelihood of data trajectories. The decoder $p(\boldsymbol{x}_t \mid \boldsymbol{z}_t)$ is trained together with the dynamics $p(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1})$ so that new data points $\boldsymbol{x}_t$ accurately update the posterior belief over the latent state $\boldsymbol{z}_t$. Unlike autoencoder-based approaches that learn an encoder/decoder independently of the dynamics, KFs couple representation and prediction by *interleaving* prediction and measurement updates (Sec. 6.2.2), which produces embeddings suitable for both long-horizon prediction and reconstruction.

Prior work has leveraged this insight to learn latent dynamics for image sequences [175, 176] by *compressing* high-dimensional observations into a low-dimensional latent state. In line with Koopman theory, KALIKO instead *lifts* the dimension to find an embedding space where the latent dynamics are approximately linear. Moreover, almost all prior works learn filters for state estimation [177, 178, 179, 180] or simple open-loop prediction tasks [175, 176]. In this chapter, we show that KALIKO's predictions can also be used in a closed-loop control task.

Kalman Filters and Koopman. Some prior works have combined KFs with Koopman methods. Most commonly, KFs are used for latent state estimation *after* learning some linear dynamics [181, 182, 183]. Others have used KFs for spectral analysis [184, 185]. Closest to KALIKO are system identification or operator learning methods, which unlike KALIKO, assume the existence of embeddings *a priori* [186] or train an encoder rather than a decoder [187]. KALIKO differs from all of these methods by using the KF structure to learn the dynamics, not simply applying a KF to some linear dynamics recovered from an unrelated process. Moreover, the main target for learning in KALIKO is the decoder module, which allows the training procedure to cleanly fit the KF formalism.

Time Series Prediction. The machine learning community has used methods like recurrent neural networks [188] to model sequential data and learn dynamical systems [189], especially those that are hard to model, like continuum robot dynamics [190], human motion [191], and video [192]. More recently, transformer-based models have been applied to many phenomena like weather, traffic patterns, electricity usage, etc. [193]. It has been shown that LLM backbones like GPT2 are effective for time series prediction [194], prompting research on time series tokenization [195].

We specifically note the recent work "Koopman-Kalman Enhanced Variational Autoencoder" (K2VAE), which may at first appear similar to KALIKO, but is distinguished by several details. First, K2VAE learns an explicit encoder rather than relying on an implicit embedding scheme. Second, K2VAE uses its KF to drive learning error to 0 with a dynamics $\mathbf{A}$ distinct from $\mathcal{K}$, while KALIKO uses the KF directly on the learned operator. Lastly, KALIKO uses several techniques not present in K2VAE that are crucial for performance (see Sec. 6.3.3 and 6.5.4). We compare to K2VAE in Sec. 6.5.3.

6.1.2 Contributions

Our chief contribution is KALIKO, a method that uses the Kalman filter to *implicitly* learn a Koopman embedding space governed by globally linear latent dynamics. Second, we show that KALIKO's embeddings produce faithful reconstructions, are highly interpretable, and align with theoretical expectations. Lastly, on complex wave data generated by a high-dimensional PDE, KALIKO outperforms several baselines on both an open-loop prediction task and a challenging simulated closed-loop stabilization task where it must stabilize an underactuated manipulator's payload by predicting and compensating for wave disturbances.

6.2 Mathematical Preliminaries

As in Part II, the notation in this part is self-contained and does not carry over the symbols of Parts I and II. Many of the symbols below are fixed by the standard conventions of Koopman operator theory and Kalman filtering, so the same letter is sometimes reused across parts with a different meaning. For instance, $\mathbf{R}$ here denotes the measurement-noise covariance rather than a rotation, and $\mathbf{G}$ the Kalman gain rather than a grasp map. Bold lowercase symbols still denote vectors (e.g., the state $\boldsymbol{x}$), bold uppercase symbols denote matrices (e.g., the latent dynamics $\mathbf{K}$ and covariance $\boldsymbol{\Sigma}$), and calligraphic symbols denote sets and spaces (e.g., the state space $\mathcal{X}$). The lone exception is the infinite-dimensional Koopman operator $\mathcal{K}$, set in bold calligraphic to distinguish it from its finite-dimensional matrix approximation $\mathbf{K}$.

Let $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^n$ be the *data state*, the state of a discrete-time nonlinear system which generates the data of interest,

$$\boldsymbol{x}_{t+1} = \boldsymbol{f}(\boldsymbol{x}_t), \tag{6.1}$$

and $\boldsymbol{z} \in \mathcal{Z} \subseteq \mathbb{R}^m$ be the *latent state* of some corresponding desired linear latent

dynamics

$$z_{t+1} = \mathbf{K}_\theta z_t, \quad \boldsymbol{x}_t = \boldsymbol{h}_\psi(z_t), \tag{6.2}$$

where θ, ψ are learnable parameters and $\boldsymbol{h} : \mathcal{Z} \to \mathcal{X}$ decodes the latent state back to the data domain. We use "encoder" or "embedding function" to refer to any operation $\boldsymbol{g} : \mathcal{X} \to \mathcal{Z}$ mapping data $\boldsymbol{x}$ to latents z.

Given only data trajectories $\boldsymbol{x}_{1:T}$ generated by the unknown dynamics $\boldsymbol{f}$ in (6.1), KALIKO learns the latent linear dynamics $\mathbf{K}_\theta$ and the decoder $\boldsymbol{h}_\psi$ in (6.2) so that latent trajectories evolved under $\mathbf{K}_\theta$ decode to the observed data. Concretely, for each trajectory we infer a latent initial condition z_0 and ask that

$$\boldsymbol{h}_\psi\left(\mathbf{K}_\theta^t z_0\right) \approx \boldsymbol{x}_t, \quad t = 1, \ldots, T.$$

All learning is therefore for the purpose of modeling the unknown latent dynamics. The data state $\boldsymbol{x}$ has meaning fixed by the physical system that generated it, but the latent state z has no inherent meaning before learning, as it is connected to the data only through the learned pair $(\mathbf{K}_\theta, \boldsymbol{h}_\psi)$.

6.2.1 Koopman Theory

We provide a brief overview of Koopman theory, deferring a detailed treatment to [161]. Let $\mathcal{G}(\mathcal{X})$ be a vector space of embedding functions on data $\boldsymbol{x} \in \mathcal{X}$. The *Koopman operator* $\mathcal{K} : \mathcal{G}(\mathcal{X}) \to \mathcal{G}(\mathcal{X})$ acts by composition,

$$(\mathcal{K}\boldsymbol{g})(\boldsymbol{x}) = \boldsymbol{g}\big(\boldsymbol{f}(\boldsymbol{x})\big), \quad \boldsymbol{g} \in \mathcal{G}(\mathcal{X}), \tag{6.3}$$

and is linear even if $\boldsymbol{f}$ is nonlinear. Since $\mathcal{K}$ is generally infinite-dimensional, in practice we seek finite approximations $\mathbf{K} \in \mathbb{R}^{m \times m}$ of $\mathcal{K}$ as in (6.2), and we say $\boldsymbol{g}$ is $\mathcal{K}$*-invariant* if the following closure condition is satisfied:

$$\boldsymbol{g}(\boldsymbol{f}(\boldsymbol{x})) = \mathbf{K}\boldsymbol{g}(\boldsymbol{x}), \quad \forall \boldsymbol{x}. \tag{6.4}$$

That is, the latents z evolve linearly on the range of $\boldsymbol{g}$. When (6.4) fails, $\mathbf{K}$ introduces *spurious modes* that accumulate multi-step prediction error due to the closure residual [162, 165, 161]. Thus, computing and analyzing invariant embeddings is central to Koopman analysis, which we revisit in Sec. 6.4.

6.2.2 Kalman Filtering

We now review the mechanics of Kalman filters (KFs), deferring details to [196]. KFs recursively update a posterior belief $p(z_t \mid z_{t-1}, \boldsymbol{x}_t)$ over the latent state given

new measurements and the previous latent state. We assume Gaussian noise with covariance $\mathbf{Q}$ corrupts the dynamics such that

$$p(z_t \mid z_{t-1}) = \mathcal{N}(\mathbf{K}z_{t-1}, \mathbf{Q}), \tag{6.5}$$

and similarly, that Gaussian noise with covariance $\mathbf{R}$ corrupts the decoder such that

$$p(\boldsymbol{x}_t \mid z_t) = \mathcal{N}(\boldsymbol{h}(z_t), \mathbf{R}). \tag{6.6}$$

Starting from a prior belief over $z_0 \sim \mathcal{N}(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$, the KF iteratively updates its belief by interleaving two updates. First, the *prediction update* propagates the belief at time $t-1$ through the dynamics:

$$\boldsymbol{\mu}_{t|t-1} = \mathbf{K}\boldsymbol{\mu}_{t-1|t-1}, \tag{6.7}$$

$$\boldsymbol{\Sigma}_{t|t-1} = \mathbf{K}\boldsymbol{\Sigma}_{t-1|t-1}\mathbf{K}^\top + \mathbf{Q}. \tag{6.8}$$

Second, the *measurement update* adjusts the posterior by accounting for the likelihood of observing $\boldsymbol{x}_t$:

$$\mathbf{G}_t = \boldsymbol{\Sigma}_{t|t-1}\mathbf{C}_t^\top \left(\mathbf{C}_t\boldsymbol{\Sigma}_{t|t-1}\mathbf{C}_t^\top + \mathbf{R}\right)^{-1}, \tag{6.9}$$

$$\boldsymbol{\mu}_{t|t} = \boldsymbol{\mu}_{t|t-1} + \mathbf{G}_t\left(\boldsymbol{x}_t - \boldsymbol{h}(\boldsymbol{\mu}_{t|t-1})\right), \tag{6.10}$$

$$\boldsymbol{\Sigma}_{t|t} = (\mathbf{I} - \mathbf{G}_t\mathbf{C}_t)\boldsymbol{\Sigma}_{t|t-1}, \tag{6.11}$$

where $\mathbf{C}_t$ is the decoder's Jacobian, $\mathbf{C}_t := \frac{\partial}{\partial z}\boldsymbol{h}(\boldsymbol{\mu}_{t|t-1})$.

The Kalman filter causally updates its belief distribution by observing only past data. However, after observing T data points, it is often useful to update *past* beliefs conditioned on future observations using *Kalman smoothing*. The Rauch-Tung-Striebel smoother caches values from the filtering pass to recursively compute smoothed beliefs from time T to 0:

$$\mathbf{G}_t^s = \boldsymbol{\Sigma}_{t|t}\mathbf{K}^\top \left(\boldsymbol{\Sigma}_{t+1|t}^\top\right)^{-1}, \tag{6.12}$$

$$\boldsymbol{\mu}_{t|T} = \boldsymbol{\mu}_{t|t} + \mathbf{G}_t^s(\boldsymbol{\mu}_{t+1|T} - \boldsymbol{\mu}_{t+1|t}), \tag{6.13}$$

$$\boldsymbol{\Sigma}_{t|T} = \boldsymbol{\Sigma}_{t|t} + \mathbf{G}_t^s(\boldsymbol{\Sigma}_{t+1|T} - \boldsymbol{\Sigma}_{t+1|t})\left(\mathbf{G}_t^s\right)^\top. \tag{6.14}$$

In summary, filtering and smoothing require the dynamics $\mathbf{K}$, decoder $\boldsymbol{h}$, prior belief $(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$, dynamics and measurement covariances $\mathbf{Q}$ and $\mathbf{R}$, and a sequence of data points $\boldsymbol{x}_{1:T}$. In return, they yield filtered and smoothed belief distributions over the corresponding latent states $z_{1:T}$, so we can view the Kalman filter/smoother as an implicit encoder over trajectories,

$$\boldsymbol{g} : \boldsymbol{x}_{1:T} \mapsto z_{1:T}.$$

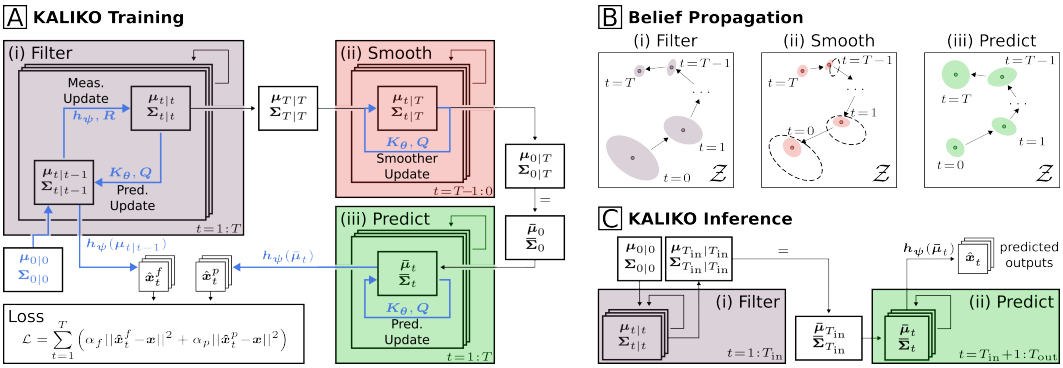


Figure 6.2: **(A) KALIKO Training.** (i) Filter for T steps, outputting a sequence of filtered distributions $\{(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})\}_{t=1}^{T}$ used for the loss. (ii) Run a backward smoother from the final belief $(\boldsymbol{\mu}_{T|T}, \boldsymbol{\Sigma}_{T|T})$, yielding the posterior initial belief $(\boldsymbol{\mu}_{0|T}, \boldsymbol{\Sigma}_{0|T})$. (iii) Roll this belief forward for T steps to get predicted beliefs $\{(\bar{\mu}_t, \bar{\Sigma}_t)\}_{t=1}^{T}$. Filtered and predicted means are decoded to observations and the model is trained end-to-end with a reconstruction loss. Learnable modules/parameters and their operations are highlighted in blue. **(B) Belief Propagation.** The filter's latent belief becomes more certain with more steps. The smoother reuses future evidence to calibrate the initial belief, seeding a strong prediction rollout during training. **(C) KALIKO Inference.** KALIKO filters on T_{in} data points then rolls out its latent dynamics for T_{out} steps. The T_{out} predicted latents are decoded into the final predicted trajectory.

6.3 The Kalman-Implicit Koopman Operator

Our method has two goals: (i) producing high-quality predictions under latent linear dynamics $\mathbf{K}$, and (ii) accurately decoding them with $\boldsymbol{x} = \boldsymbol{h}_\psi(\boldsymbol{z})$. Observe that inaccurate dynamics $\mathbf{K}$ degrade the prediction update (6.7)–(6.8), and an inaccurate decoder $\boldsymbol{h}_\psi$ degrades the measurement update (6.9)–(6.11). Thus, a performant KF requires *both* accurate $\mathbf{K}$ and $\boldsymbol{h}_\psi$. This motivates our method, Kalman-implicit Koopman operator (KALIKO) learning, which trains a KF end-to-end with a single reconstruction objective to jointly achieve both goals. In contrast, autoencoding Koopman methods typically separate prediction and reconstruction [164, 162, 171], which may overemphasize one at the expense of the other.

6.3.1 Training

KALIKO is trained with the replay overshooting method [176], which consists of three stages (see Fig. 6.2A). Given a data trajectory $\boldsymbol{x}_{1:T}$, the *filtering stage* interleaves the prediction and measurement updates in (6.7)–(6.8) and (6.9)–(6.11) respectively to recover the sequences of belief distributions $\{(\boldsymbol{\mu}_{t|t-1}, \boldsymbol{\Sigma}_{t|t-1})\}_{t=1}^{T}$ and $\{(\boldsymbol{\mu}_{t|t}, \boldsymbol{\Sigma}_{t|t})\}_{t=1}^{T}$. The *smoothing stage* then applies (6.12)–(6.14) to these beliefs

in reverse to recover an initial belief at $t = 0$ conditioned on all measurements, $(\boldsymbol{\mu}_{0|T}, \boldsymbol{\Sigma}_{0|T})$. Lastly, the *prediction stage* rolls out T prediction updates (6.7)–(6.8) from the smoothed initial condition *without measurement updates* to recover a sequence of predicted beliefs $\{(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t)\}_{t=1}^{T}$.

KALIKO learns $\mathbf{K}_\theta$ and $\boldsymbol{h}_\psi$ along with the prior $(\boldsymbol{\mu}_{0|0}, \boldsymbol{\Sigma}_{0|0})$ and covariances $(\mathbf{Q}, \mathbf{R})$ end-to-end via the loss

$$\mathcal{L} = \sum_{t=1}^{T} \alpha_f \underbrace{\left\| \boldsymbol{h}_\psi\left(\boldsymbol{\mu}_{t|t-1}\right) - \boldsymbol{x}_t \right\|_2^2}_{\text{filtering}} + \alpha_p \underbrace{\left\| \boldsymbol{h}_\psi\left(\bar{\boldsymbol{\mu}}_t\right) - \boldsymbol{x}_t \right\|_2^2}_{\text{prediction}}, \tag{6.15}$$

where the first term corresponds to the filtered[2] reconstructions of the data and the second to the predicted reconstructions. The coefficients $\alpha_f, \alpha_p \in \mathbb{R}_{>0}$ control the relative weights for filtering and prediction respectively, which we set to 1 in all experiments.

The *filtering loss* is analogous to an autoencoding loss, as it fuses measurement consistency ("decoding") via (6.9)–(6.11) with latent inference via (6.7)–(6.8) ("encoding") [176]. The *prediction loss* ensures accurate open-loop long-horizon predictions, which mirrors inference-time execution. We remark that [176] maximizes the log-likelihood of the data while we found that a simple MSE loss was more computationally-efficient with no performance degradation.

6.3.2 Inference

At inference time, KALIKO uses T_{in} data points to predict the next T_{out} steps. First, the filter consumes $\boldsymbol{x}_{1:T_{\text{in}}}$ to obtain the posterior belief $(\boldsymbol{\mu}_{T_{\text{in}}|T_{\text{in}}}, \boldsymbol{\Sigma}_{T_{\text{in}}|T_{\text{in}}})$. T_{out} prediction steps are then executed from this filtered belief, yielding a sequence of predicted latent beliefs $\{(\bar{\boldsymbol{\mu}}_t, \bar{\boldsymbol{\Sigma}}_t)\}_{t=T_{\text{in}}+1}^{T_{\text{in}}+T_{\text{out}}}$. The predicted means are each decoded to obtain a predicted trajectory: $\hat{\boldsymbol{x}}_t = \boldsymbol{h}_\psi(\bar{\boldsymbol{\mu}}_t)$. See Fig. 6.2C for a visualization.

6.3.3 Additional Implementation Details

Delay Embedding. Inspired by the use of delay embeddings in Koopman methods [170], KALIKO parameterizes $\mathbf{K}_\theta$ as the dynamics on a delay-embedded latent state,

[2] We use the pre-measurement update mean $\boldsymbol{\mu}_{t|t-1}$, as in [176].

Van der Pol	$\begin{cases} \dot{x}_1 = x_1 - x_1^3/3 - x_2 \\ \dot{x}_2 = x_1 \end{cases}$	(6.16)
Pendulum	$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \sin(x_1) \end{cases}$	(6.17)
Duffing	$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 - \delta x_2 - x_1^3 \end{cases}$	(6.18)
Hopf-Bautin (Polar)	$\begin{cases} \dot{r} = re^{(1-2r^2)/2}(r^2 - r^4 - 3/16) \\ \dot{\theta} = 1 \end{cases}$	(6.19)

Table 6.1: Systems used for embedding analysis. We learn discrete-time dynamics despite presenting these systems in continuous time for clarity.

yielding a sparse block-companion matrix form

$$\underbrace{\begin{bmatrix} \boldsymbol{\zeta}_{t+1} \\ \vdots \\ \boldsymbol{\zeta}_{t+n_z-1} \\ \boldsymbol{\zeta}_{t+n_z} \end{bmatrix}}_{z_{t+1}} = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{I} \\ \boldsymbol{\Gamma}_1 & \boldsymbol{\Gamma}_2 & \dots & \dots & \boldsymbol{\Gamma}_{n_z} \end{bmatrix}}_{\mathbf{K}} \underbrace{\begin{bmatrix} \boldsymbol{\zeta}_t \\ \vdots \\ \boldsymbol{\zeta}_{t+n_z-2} \\ \boldsymbol{\zeta}_{t+n_z-1} \end{bmatrix}}_{z_t},$$

where the only learnable parameters are the blocks $\boldsymbol{\Gamma}_1, \dots, \boldsymbol{\Gamma}_{n_z}$, and each $\boldsymbol{\zeta}_t$ is a sub-latent state of dimension d_z, which is stacked into the full delay-embedded latent state $\boldsymbol{z}_t = (\boldsymbol{\zeta}_t, \dots, \boldsymbol{\zeta}_{t+n_z-1}) \in \mathbb{R}^{n_z \cdot d_z}$.

Temporal Chunking. KALIKO temporally chunks the full-resolution data into groups of n_x states, each corresponding to a sub-latent vector as follows:

$$\boldsymbol{\zeta}_t \leftrightarrow (\boldsymbol{\xi}_\tau, \dots, \boldsymbol{\xi}_{\tau+n_x-1}), \tag{6.20}$$

where τ is the original time index satisfying $t = \lfloor \tau / n_x \rfloor$ and $\boldsymbol{\xi}_\tau$ is a raw data point. Thus, $\boldsymbol{h}$ decodes n_z sub-latent vectors into a contiguous sequence of $n_z \cdot n_x$ raw data points:

$$\boldsymbol{h}(\underbrace{\boldsymbol{\zeta}_{t:(t+n_z-1)}}_{z_t}) = \underbrace{\boldsymbol{\xi}_{\tau:(\tau+n_x \cdot n_z-1)}}_{\boldsymbol{x}_t}. \tag{6.21}$$

Temporal chunking greatly speeds up training, as it reduces the number of steps in each training stage by a factor of n_x, until the cost of memory and linear system solves associated with a large data dimension n outweighs the speedup.

Decoder Design. The decoder $\boldsymbol{h}_\psi$ is a convolutional neural network that interleaves depthwise convolutions across the delay dimension (mixing $\boldsymbol{\zeta}$'s) with GELU-activated MLPs, which allows $\boldsymbol{h}$ to mix sub-latent vectors across many time steps

to decode long sequences of raw observations. We found two such blocks sufficient for all experiments.

6.4 Analyzing KALIKO Embeddings

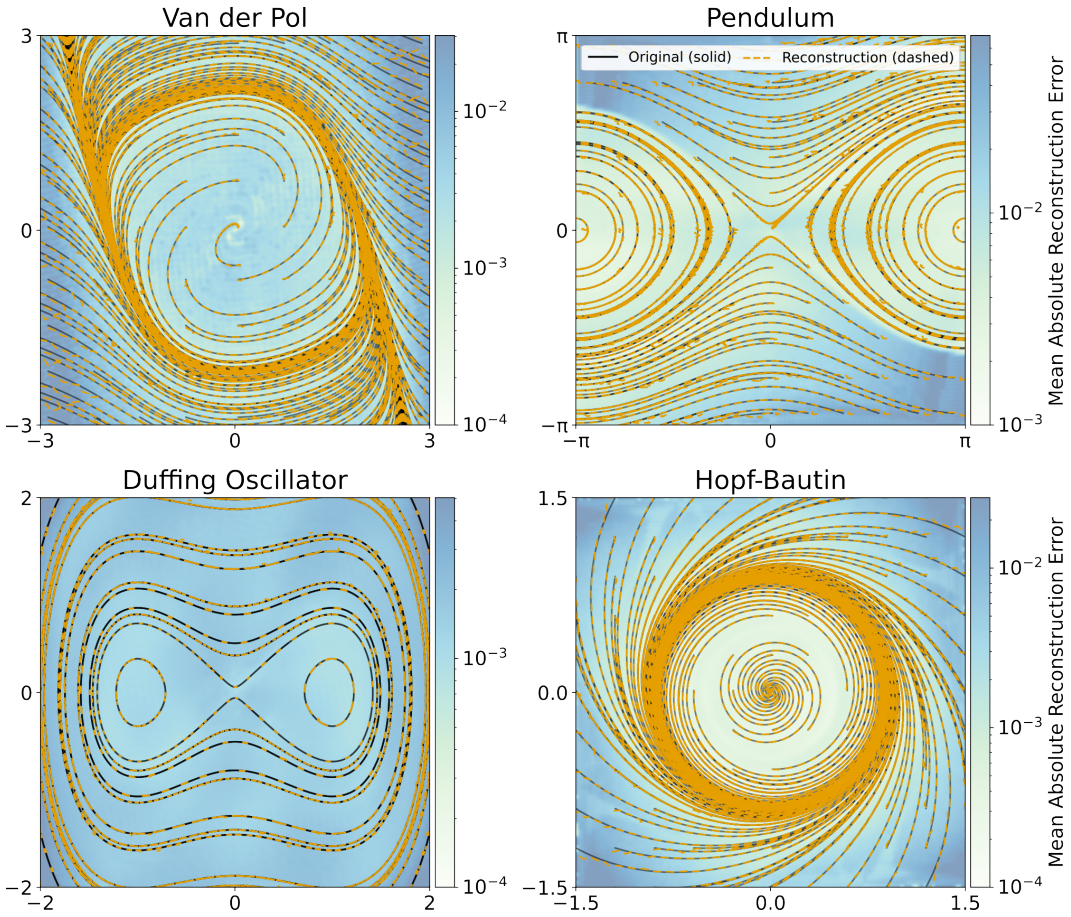


Figure 6.3: **KALIKO Reconstructions.** KALIKO closely reconstructs the vector field of nonlinear systems without an encoder using globally linear latent dynamics. Shown are some ground truth trajectories in black (solid) and their reconstructions in orange (dashed). A denser heatmap of the reconstruction error is shown in blue (darker denotes more error).

6.4.1 Reconstruction Performance

As discussed in Sec. 6.2.2, the Kalman filter/smoother can be viewed as an encoder on sequences of data $\boldsymbol{x}_{1:T}$. We study KALIKO's autoencoding capabilities on four systems for visualization purposes: the Van der Pol oscillator, pendulum, Duffing oscillator, and a two-limit cycle Hopf-Bautin system.

We plot reconstructed trajectories on a heatmap of reconstruction errors (Fig. 6.3).

KALIKO accurately reconstructs the original data, verifying that the Kalman filter and smoother function as an effective encoder $\boldsymbol{g}$. Moreover, all models used identical hyperparameters, demonstrating KALIKO's robustness to features of nonlinear systems like fixed points, limit cycles, and multiple attractors.

6.4.2 Analyzing KALIKO's Eigenfunctions

Sec. 6.2.1 introduced a key property, $\mathcal{K}$-invariance, in (6.4). Observe that the *eigenfunctions* φ of $\mathcal{K}$ satisfy (6.4), since by definition,

$$\lambda\varphi(\boldsymbol{x}_t) = (\mathcal{K}\varphi)\,(\boldsymbol{x}_t), \quad \lambda \in \mathbb{C}, \tag{6.22}$$

thus giving coordinates on which $\mathcal{K}$ acts linearly, making them central objects of study in the literature [161]. A left eigenpair $(\lambda, \boldsymbol{w})$ of the approximation $\mathbf{K}$ of $\mathcal{K}$ (and $\boldsymbol{g}$) implies an eigenfunction $\varphi(\boldsymbol{x}) := \boldsymbol{w}^\top \boldsymbol{g}(\boldsymbol{x})$, following from the following basic sequence of equalities:

$$\begin{aligned}
(\mathcal{K}\varphi)\,(\boldsymbol{x}_t) &= \varphi(\boldsymbol{f}(\boldsymbol{x}_t)) \\
&= \varphi(\boldsymbol{x}_{t+1}) \\
&= \boldsymbol{w}^\top \boldsymbol{g}(\boldsymbol{x}_{t+1}) \\
&= \boldsymbol{w}^\top \mathbf{K}\boldsymbol{g}(\boldsymbol{x}_t) \\
&= \lambda \boldsymbol{w}^\top \boldsymbol{g}(\boldsymbol{x}_t) \\
&= \lambda\varphi(\boldsymbol{x}_t).
\end{aligned}$$

Thus, by examining the eigenfunctions of $\mathbf{K}_\theta$ via eigendecomposition, we can interpret the dynamics captured by KALIKO's implicit encoder (the Kalman filter and smoother). We consider two case studies.

Van der Pol. The Van der Pol (VDP) system (6.16) has a globally-stable limit cycle (Fig. 6.3, top left). After applying KALIKO to learn the VDP dynamics, we identify a complex eigenpair $(\lambda, \boldsymbol{w})$ of the resulting linear system $\mathbf{K}_\theta$ with $|\lambda| \approx 1$, implying a steady-state oscillatory latent mode. We approximate the eigenfunction $\varphi(\boldsymbol{x})$ by evaluating $\boldsymbol{w}^\top \boldsymbol{g}(\boldsymbol{x})$ over latent beliefs in the plane, plotting VDP's limit cycle (green) against the modulus and argument of $\varphi(\boldsymbol{x})$ in Fig. 6.4A/B.

We can now check whether the limit cycle is described by $\varphi(\boldsymbol{x})$. First, we parameterize the (unit-modulus) eigenvalue as $\lambda = e^{i\omega}$ and let $\boldsymbol{z}_0$ be an arbitrary fixed latent element of the corresponding eigenspace, so that $\boldsymbol{z}_t = \mathbf{K}^t \boldsymbol{z}_0 = \lambda^t \boldsymbol{z}_0$. Since norms scale by the modulus of a complex factor and $|e^{it\omega}| = 1$, the latent trajectory must

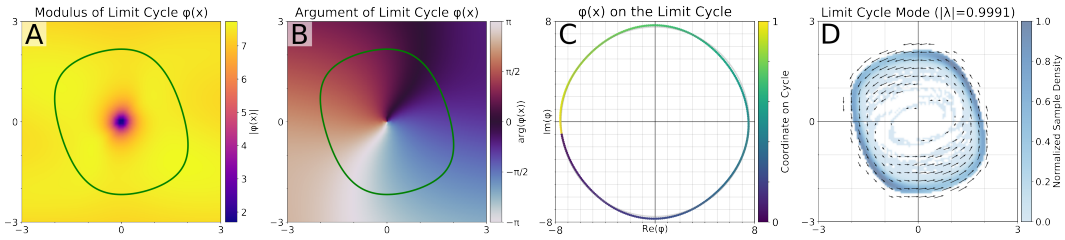


Figure 6.4: **Limit Cycle Eigenfunction for the VDP System.** **(A/B)** Without parameterizing an encoder, KALIKO implicitly recovers an eigenfunction associated with the Van der Pol system's limit cycle (overlaid in green) with $|\lambda| \approx 1$. **(C)** Evaluating $\varphi(x)$ along the limit cycle traces out a nearly perfect circle in the complex plane, showing that KALIKO entirely captures the limit cycle dynamics into this eigenfunction. **(D)** The Koopman mode associated with this eigenvalue corresponds to a vector field that drives trajectories onto the cycle.

have constant modulus:

$$|z_t| = |\lambda^t| \cdot |z_0| = |e^{it\omega}| \cdot |z_0| = |z_0|.$$

Moreover, the argument along the latent trajectory must evolve linearly (mod 2π):

$$\arg(z_t) = \arg(\lambda^t z_0) = t\omega + \arg(z_0) \pmod{2\pi}.$$

Thus, if $\varphi(\boldsymbol{x})$ evaluated on the limit cycle traces out a single revolution of a circle in the complex plane, we can conclude that KALIKO *encodes the limit cycle dynamics entirely in this mode*, which is confirmed in Fig. 6.3C.

We also examine the time-invariant directions in $\mathcal{X}$ given by the associated *right eigenvector* of $\mathbf{K}_\theta$, or the *Koopman mode* [161]. To do so, we encode a set of trajectories, project them onto the limit cycle eigenspace, and decode them, resulting in Fig. 6.4D. The Koopman mode shows the attraction of the limit cycle in a neighborhood around it, verifying that KALIKO has learned VDP's dominant dynamical behavior.

These analyses indicate that KALIKO can stably recover nontrivial behavior of nonlinear systems to very high precision. In particular, if there were spurious modes resulting from this case study, then Fig. 6.4C would appear to be a spiral rather than a clean, closed circle (i.e., the modulus would not be constant, indicating multi-step prediction errors).

Duffing Oscillators. The undamped Duffing oscillator (UDO, $\delta = 0$ in (6.18)) has a saddle at the origin and two neutrally stable wells at $(\pm 1, 0)$. Because the oscillation

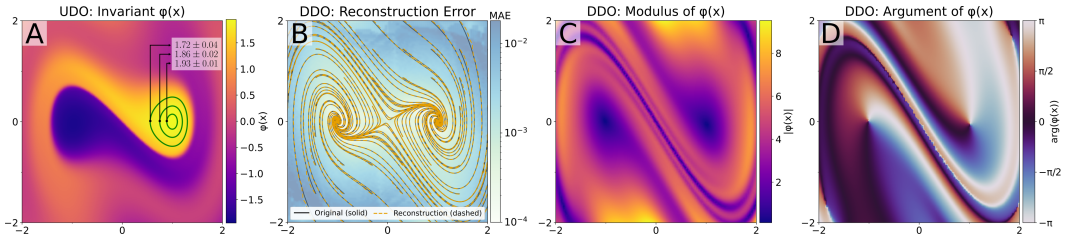


Figure 6.5: **Eigenfunctions for the Undamped and Damped Duffing Oscillator.** **(A)** KALIKO recovers an eigenfunction $\varphi(\boldsymbol{x})$ with $\lambda \approx +1$ capturing the invariance of systems with continuous spectra like the *undamped Duffing oscillator* (UDO). The value of $\varphi(x)$ is nearly constant along each of three distinct cycles (green) with little variation (shown are mean and stdev). Across cycles, the value clearly changes, corresponding to energy invariance. **(B)** Conversely, KALIKO also reconstructs the *damped Duffing oscillator* (DDO) with no adjustments. **(C/D)** KALIKO recovers an eigenfunction capturing the attractive "spiral" behavior about the wells at $(\pm 1, 0)$, reproducing the results from explicit methods in prior work [162, Fig. 5].

frequency depends on energy (i.e., the continuous spectrum case [164]), no finite embedding space can recover a global phase eigenfunction. Instead, invariants (e.g., energy) correspond to $\lambda \approx 1$. Consistent with this, KALIKO recovers an eigenfunction φ that is nearly constant along each closed orbit and varies across energy levels (Fig. 6.5A), capturing UDO's energy invariance.

Adding damping (DDO, $\delta = 1$) eliminates the continuous spectrum, leaving the saddle at the origin and making $(\pm 1, 0)$ asymptotically stable. Here, KALIKO identifies a contractive mode with $|\lambda| < 1$ whose eigenfunction φ tends to 0 at the three equilibria and whose phase clearly depicts the attractive spirals (Fig. 6.5B-D), in line with Koopman analyses in prior work [162, Fig. 5].

The DDO case lets us compare KALIKO directly against the prior well-known *Linearly Recurrent Autoencoder Network* (LRAN) method [162] on the DDO in two variants (Fig. 6.6). When the eigenvalues of the latent dynamics learned by LRAN are fixed to their *known* values in advance, following the "constrained" variant of [162], training only needs to fit the corresponding eigenfunctions, and LRAN centers the recovered phase spirals on the true equilibria. When the latent dynamics are unconstrained, the only option on genuinely unknown systems, the recovered centers drift visibly off the equilibria, exhibiting exactly the failure modes discussed in Sec. 6.1.1. KALIKO faces the same unconstrained setting yet recovers correctly-centered structure with no constraints on its latent dynamics, as shown in Fig. 6.5B-D.

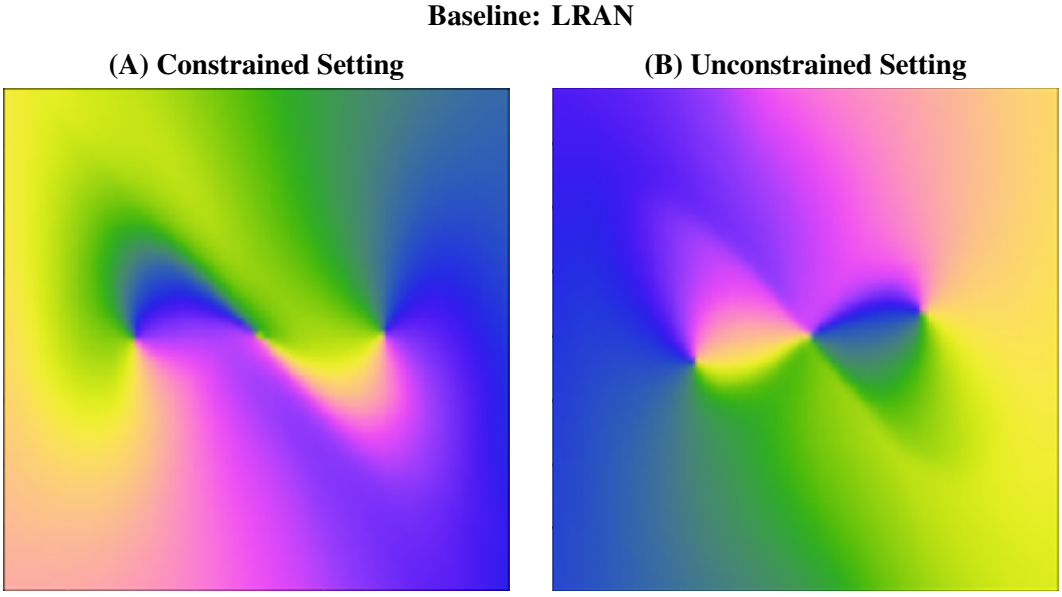


Figure 6.6: **LRAN Eigenfunction Portraits.** Phase of the complex eigenfunction recovered by the baseline LRAN [162, Fig. 5] on the damped Duffing oscillator ($\delta = 1/2$ in (6.18)), whose equilibria at $(\pm 1, 0)$ appear as the centers of the spiraling phase patterns. **(A)** With the eigenvalues of the latent dynamics constrained to the correct values in advance, the centers coincide with the true equilibria (reproduced from [162, Fig. 5]). **(B)** With unconstrained latent dynamics, the centers drift visibly off (reproduced from [162, Fig. 6]). KALIKO requires no such constraints (Fig. 6.5).

Takeaway. Despite using an *implicit* encoder, KALIKO reproduces the expected Koopman structure on these canonical systems (unit-modulus phase on the VDP limit cycle, energy-like invariants for UDO, and contractive modes near DDO equilibria) using *identical hyperparameters* with no system-specific tuning. These results illustrate that KALIKO's implicit encoding does not preclude typical Koopman analysis. Rather, it facilitates it by stably recovering expressive, interpretable representations without sensitive handcrafted features or model tuning.

6.5 Experiments

6.5.1 Simulated Ocean Data

We now study the predictive capacity of KALIKO and its application to a closed-loop stabilization control problem. To do so, we simulate the motion of a barge in various ocean conditions, yielding thousands of trajectories in $SE(3)$. Ocean surface waves are governed by free-surface hydrodynamics, in which an incompressible flow is coupled to nonlinear boundary conditions on a free surface whose shape is itself an unknown of the problem. The state of this PDE, the velocity field over the entire

fluid domain together with the surface elevation, is infinite-dimensional, evolves nonlinearly, and is observed by the barge only through the six degrees of freedom of its pose. These are exactly the high-dimensional spatiotemporal phenomena that motivated this chapter and Part III: no compact analytic state is available to a learner, so the state itself must be approximated from data. To generate data tractably, we adopt the standard linearization of this problem and briefly summarize the resulting pipeline, deferring a detailed treatment of marine hydrodynamics to [24].

Each irregular sea state is drawn from the JONSWAP spectrum [197], an empirical model of wind-driven seas that specifies the power spectral density of the wave elevation,

$$S(\omega) = \frac{\alpha g^2}{\omega^5} \exp\left(-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right) \gamma^{r(\omega)}, \quad r(\omega) = \exp\left(-\frac{(\omega/\omega_p - 1)^2}{2\sigma^2}\right),$$

where g is gravitational acceleration, ω_p is the peak wave frequency, γ is the peak-enhancement factor, σ takes the standard values 0.07 for $\omega \le \omega_p$ and 0.09 otherwise, and α is a nondimensional energy scale determined by the significant wave height H_s and ω_p. For each trajectory, we randomly sample H_s, the peak period $T_p = 2\pi/\omega_p$, γ, and the wave heading. A wave realization is then a superposition of harmonics on a uniform grid of frequencies spanning $[0.25\,\omega_p, 3\,\omega_p]$, where the jth component has amplitude $a_j = \sqrt{2S(\omega_j)\Delta\omega}$ and an independent random phase ϕ_j drawn uniformly from $[0, 2\pi)$.

The barge is a rigid box whose interaction with the waves is modeled with linear potential flow, the linearization of the free-surface problem above about a calm sea and small body motions. At each grid frequency, `capytaine` [198] solves the radiation and diffraction problems with the boundary element method, yielding the added mass $\mathbf{A}(\omega_j)$ and radiation damping $\mathbf{B}(\omega_j)$ matrices and the complex wave excitation force per unit wave amplitude $\hat{\boldsymbol{F}}_{\mathrm{exc}}(\omega_j)$ on the hull. The barge's response per unit wave amplitude (the *response amplitude operator*) $\hat{\boldsymbol{q}}(\omega_j) \in \mathbb{C}^6$ in its six degrees of freedom (surge, sway, heave, roll, pitch, and yaw) then solves the frequency-domain equations of motion,

$$\left[-\omega_j^2\left(\mathbf{M} + \mathbf{A}(\omega_j)\right) + i\omega_j \mathbf{B}(\omega_j) + \mathbf{C}_{\mathrm{hs}}\right] \hat{\boldsymbol{q}}(\omega_j) = \hat{\boldsymbol{F}}_{\mathrm{exc}}(\omega_j),$$

where $\mathbf{M}$ is the barge's rigid-body inertia and $\mathbf{C}_{\mathrm{hs}}$ is its hydrostatic stiffness. Scaling each response by its wave component's amplitude and phase and superposing over the grid recovers the motion time series,

$$\boldsymbol{q}(t) = \sum_j a_j \,\mathrm{Re}\left[\hat{\boldsymbol{q}}(\omega_j)\, e^{i(\omega_j t + \phi_j)}\right],$$

and composing the three rotational degrees of freedom into an orientation yields poses in $SE(3)$. All parameter ranges were chosen such that the computed solutions were numerically stable, yielding realistic solutions. We emphasize that this generative pipeline serves only as a stand-in for field measurements: neither KALIKO nor any baseline observes it, and every learner receives only the resulting pose trajectories.

6.5.2 Baselines and Methodological Details

We compare KALIKO to both Koopman-specific and general time series prediction baselines.

Koopman Baselines. We have 3 Koopman baselines. The *Deep Koopman net* [164] is an autoencoding-based method that has the ability to parameterize continuous eigenvalue spectra. It is trained for long-horizon predictions with a multistep prediction consistency loss along with autoencoding losses, giving a canonical baseline for Koopman autoencoders. The *Koopman-Kalman Variational Autoencoder* (K2VAE) [199] claims state-of-the-art performance in probabilistic time series prediction. K2VAE predicts a "residual" global Koopman operator $\mathbf{K}_{\text{glo}}$ added to a locally-fitted operator derived from DMD, $\mathbf{K}_{\text{loc}}$, yielding the model $\mathbf{K} = \mathbf{K}_{\text{loc}} + \mathbf{K}_{\text{glo}}$. It uses an auxiliary Kalman filter to estimate and drive errors to 0. Since K2VAE assumes that $\mathbf{K}_{\text{loc}}$ is a good prior about which to learn a residual, we additionally implement a simple baseline where we fit a *local DMD* model at each time step on the context T_{in} using delay embeddings as the Koopman embeddings. Several works have explored variations of this idea in an MPC-style context for modeling effects like fluid flows [200, 201]. We used the open-source implementations and settings of the baselines for fair comparison.

Time Series Baselines. We have 2 general time series baselines. *Recurrent neural networks* (RNNs) [188] are classic autoregressive models well-suited for dynamical prediction. RNNs can be viewed as "generalized" filters, making them a natural baseline: they ingest data, update an internal state, and can generate prediction sequences. *PatchTST* [193] is a popular method for general time series modeling, as it requires little domain-specific knowledge to yield reasonable predictions.

Data Preprocessing. For all methods, we map all poses in $SE(3)$ to $\mathfrak{se}(3)$, the tangent space at the identity, using the Log map. This makes the data Euclidean and enables geometrically consistent pose predictions. Second, we normalize the data with a running estimate of the mean and standard deviation along each axis,

which is updated during training. At inference time, we fix these statistics to prevent leakage.

6.5.3 Prediction Performance

First, we evaluate the baselines and KALIKO on a basic *open-loop prediction* task, where each model is supplied an input context of $T_{\text{in}} = 128$ data points and predicts $T_{\text{out}} = 64$ steps. If applicable, methods are trained for 164k steps with a batch size of 32, and we report prediction errors in $\mathfrak{se}(3)$ averaged over all times, dimensions, and batches.

Method	MSE (↓)	MAE (↓)
Deep Koopman	0.0019	0.025
K2VAE	0.0015	0.026
Local DMD	0.0005	0.010
RNN	0.0003	0.010
PatchTST	0.0004	0.011
KALIKO (ours)	**0.0002**	**0.008**

Table 6.2: Performance on open-loop wave prediction.

In Table 6.2, KALIKO outperforms all methods using a global linear latent prediction model. In contrast, the RNN and PatchTST strongly leverage nonlinear models, and K2VAE and local DMD rely heavily on locally-fitted models rather than a global linear operator. Surprisingly, local DMD outperforms both Deep Koopman and K2VAE, which we attribute to the sensitivity of autoencoder-based Koopman methods to their learned embeddings, as noted in Sec. 6.1.1, and K2VAE's noisy predictions (see Fig. 6.7).

6.5.4 Ablations

Delay Length. KALIKO uses a delay length $n_z = 4$ for its latent dynamics. We perform two ablations: letting $n_z = 1$ (no delay) and increasing to $n_z = 6$. We find that the delay is crucial for performance, as the MAE nearly doubles without it. Conversely, using a higher delay does not improve performance, justifying the choice $n_z = 4$.

Decoder Architecture. We replaced KALIKO's convolutional decoder with a simple MLP that performs no depthwise mixing, instead mapping each sub-latent vector ζ_t to a distinct temporal chunk. We find this only slightly harms performance, showing KALIKO's robustness to architectural design decisions.

Learnable Priors. We evaluate KALIKO's performance when fixing the prior to $(\mathbf{0}, \mathbf{I})$. This substantially degrades performance, which we hypothesize is because the scale and anisotropy of the prior are key for allowing KALIKO to learn the most expressive possible embedding space geometry.

Ablation	MSE (↓)	MAE (↓)
$n_z = 4 \rightarrow n_z = 1$	0.0007	0.015
$n_z = 4 \rightarrow n_z = 6$	0.0002	0.008
Conv $\rightarrow$ MLP Decoder	0.0003	0.009
Learnable $\rightarrow$ Fixed Latent Prior	0.0005	0.012
KALIKO (Original)	**0.0002**	**0.008**

Table 6.3: Ablations: effect on KALIKO's prediction performance.

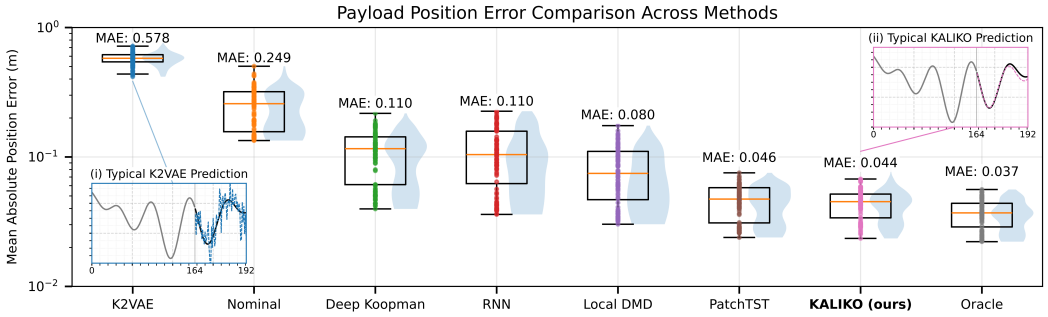


Figure 6.7: **Control Results.** KALIKO outperforms all baselines on the control task with the least variance and near-oracle performance. Insets show one dimension of a hold-out trajectory. Gray denotes input context, black denotes ground truth prediction data. (i) K2VAE's poor performance can be explained by its overly-noisy predictions (blue, dashed). (ii) KALIKO's smooth prediction trajectories enable much more stable closed-loop control (pink, dashed).

6.5.5 Control Performance

We evaluate all methods on a simulated control task: stabilizing a cable-suspended payload with a barge-mounted crane (see Fig. 6.8). The task is challenging because the barge and crane are severely underactuated: the free-body motion is driven by the ocean waves, and the crane's payload height is controllable only via a damped winch. We need not model any control dynamics, since the crane is controlled with sampling-based MPC, which selects controls by choosing high-performance candidates under forward simulations of the system. Thus, the predictive models in Sec. 6.5.3 suffice.

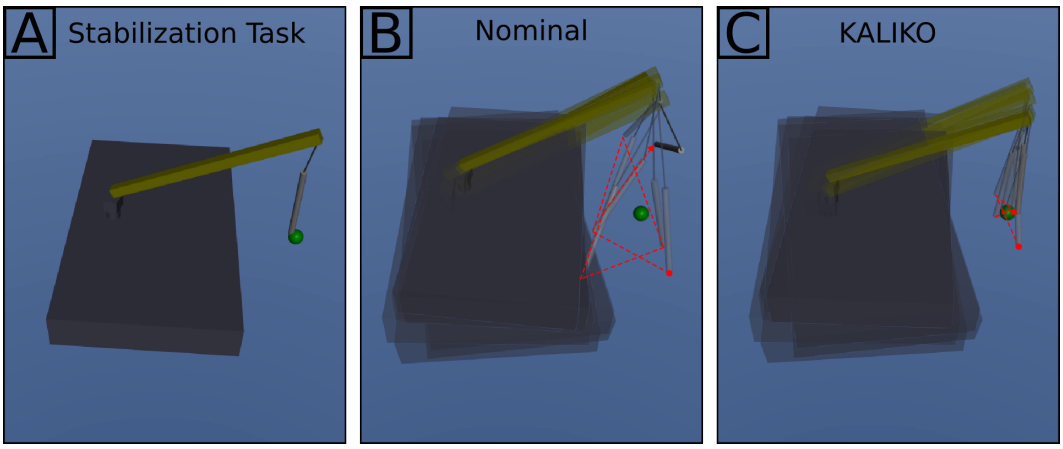


Figure 6.8: **(A)** We aim to stabilize the tip of the payload to a goal location (green) by compensating for motion induced by realistic waves. A crane controls the payload by rotating its base and boom and/or reeling its winch. **(B)** The nominal controller poorly regulates the payload, as seen in this 3 second snapshot (red dashed lines). **(C)** KALIKO stabilizes the payload at near-oracle levels in the presence of the same waves.

We evaluate all methods on 100 random wave simulations, feeding their predictions to an MPPI controller implemented using the open-source toolbox `judo` [202] introduced in Chapter 5. We calibrate performance against two other methods: an **oracle** supplied with ground-truth future barge motion, and a **nominal** stabilizing controller that ignores the wave disturbances. Since the controller is stochastic, the oracle allows us to measure the error induced by its sampling noise. We report mean absolute error in the payload target position (see Fig. 6.7).

KALIKO outperforms all methods with near-oracle performance and the tightest variance, showing that its superior open-loop prediction translates to high-performance closed-loop control. However, prediction performance is not a perfect indicator of control performance, as evidenced by local DMD outperforming the RNN. We attribute this gap to the role of state estimation, which is as crucial as prediction in control. Trained explicitly for filtering, KALIKO maintains an accurate latent belief in closed loop, whereas autoencoding-based Koopman methods and the RNN are trained only for prediction, and their internal states drift. Conversely, local DMD and PatchTST, which are not latent-variable methods, perform well because they operate directly on (patches of) raw data; with reasonably accurate short-horizon predictions and fast enough feedback, the controller can effectively stabilize the system.

6.6 Conclusion

This chapter introduced Kalman-Implicit Koopman Operator (KALIKO) learning, which leverages Kalman filtering to implicitly compute Koopman embeddings and linear latent dynamics without explicitly parameterizing an encoder. KALIKO learns highly interpretable representations, achieves superior open-loop prediction, and outperforms all baselines on a challenging closed-loop control task, with near-oracle performance. In sum, KALIKO turns Bayesian filtering into a practical Koopman encoder, yielding performant embeddings that require no handcrafting or expert domain knowledge.

The experiments in this chapter underscore why the state itself, and not just the dynamics, must be learned. The flow field governing the barge's motion is never measured and cannot be recovered from pose data alone, so there is no physical state to model. KALIKO learns one instead, and the eigenfunction analyses show that a state whose only meaning comes from a fitted operator and decoder can be as structured and interpretable as one derived from physics. This philosophy of modeling the state as a first-class citizen is central to the chapters that follow, which focus less on dynamical modeling and more on core ideas in representation learning.

Chapter 7

BERNOULLI-QUANTIZED TOKENIZATION

7.1 Introduction

The preceding chapters of this thesis have been largely motivated by dynamical prediction and modeling. Even Chapter 6, whose focus was learning a latent—rather than analytic—state, still relied on classical algorithms like Bayesian filtering to learn a dynamical model. The resulting improvement in prediction accuracy motivates a more targeted study of *representation learning*, which is the focus of the remainder of this thesis.

Taking inspiration from large language models, this chapter begins by studying the concept of *tokenization* for time series data, completely divorced from any dynamic interpretation of the data. Tokenization is the process of mapping a corpus of data into a sequence of tokens from a discrete vocabulary, and has been extensively studied as a way to compress text into discrete units that are suitable for learning. While language has natural units of information (alphabets, words, etc.), time series are naturally continuous, so approximating the signal space with a discrete vocabulary is somewhat unintuitive. Nevertheless, for adjacent data modalities like images, this approach has been highly effective [203], with many promising initial studies for time series [195].

A key question we ask is how best to learn *large* token vocabularies for time series data. Existing methods typically use vocabularies that are only hundreds or thousands of tokens large [204, 205, 195], despite the high complexity of general time series data. This bottleneck potentially limits model capacity on real-world time series data because small vocabularies lack the expressiveness to capture diverse temporal patterns, amplitudes, and frequencies across different scales and domains.

The main challenges in scaling up the vocabulary size stem from limitations in traditional vector quantization methods, which require explicitly storing and searching through explicit codebooks during tokenization [206]. As vocabulary sizes grow, the memory requirements for storing these codebooks and the computational cost of nearest-neighbor lookup become prohibitive.

In response to this challenge, this chapter studies the application of *lookup-free*

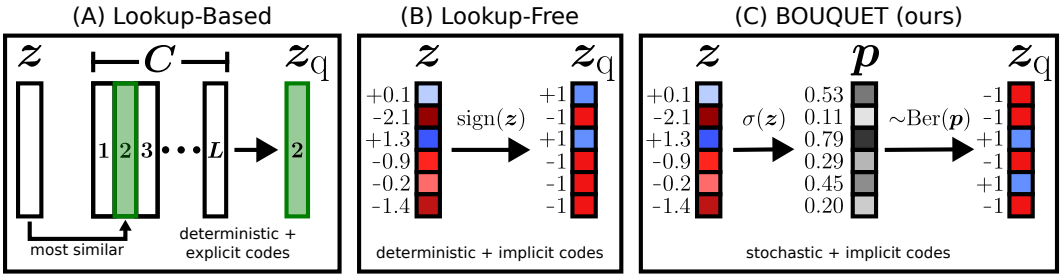


Figure 7.1: Overview of quantization methods. **(A)** Lookup-based quantization explicitly constructs a codebook $\mathbf{C}$ and associates the embedding z with the index of its nearest neighbor in $\mathbf{C}$ (in this example, index 2). **(B)** Lookup-free quantization instead quantizes z to a binary string, which corresponds to a unique index in an implicit codebook of size 2^{d_z}. **(C)** BOUQUET converts z to elementwise Bernoulli probabilities and samples over the implicit codebook. We find that BOUQUET effectively explores the space of codes even without entropy loss terms present in other works, and recovers high-quality representations on a wide variety of data.

quantization (LFQ) [207] methods to time series data. Originally developed for image and video tokenization, LFQ eliminates the need to store and search through a codebook by quantizing continuous embeddings directly into binary strings, allowing the vocabulary size to scale exponentially in embedding dimension with no memory overhead. With no explicit codebook to store, the difficulty shifts to stably learning the implicit codes themselves. Though sometimes effective for time series tokenization, in this chapter we find that LFQ is sensitive to its hyperparameters, and often destabilizes during training.

To address this, we propose *Bernoulli-Quantized Tokenization* (BOUQUET), a novel stochastic lookup-free tokenizer that exhibits better codebook exploration over large vocabularies and stable training with only a single reconstruction loss term. BOUQUET's central idea is to treat quantization as sampling: drawing codes from learned Bernoulli distributions during training explores the implicit codebook naturally and resolves the codebook collapse that plagues deterministic quantizers, with no need for entropy or commitment losses. BOUQUET enables us to achieve remarkable time series reconstructions with minimal compute: all time series models in this chapter use fewer than 250K parameters. Because the implicit codebook costs nothing to store, enlarging the vocabulary is essentially free, and we exploit this to train time series tokenizers with vocabularies of over one billion tokens. Reconstruction performance improves predictably as the vocabulary grows, following a linear trend on a log-log scale that shows no sign of saturating even at this extreme scale. Therefore, by leveraging massive vocabulary sizes, we can effectively reduce

the approximation error between the learned temporal representations and the true data for the purposes of downstream analysis. See Fig. 7.1 for a comparison of quantization methods.

7.1.1 Related Work

Tokenization has been extensively studied as a way to compress text into discrete units that are suitable for learning. This line of research goes back to byte-pair encoding [208], but has seen a resurgence with the emergence of transformer architectures [209], which perform well when operating on data in such a compressed representation. Transformers in the form of large language models [210, 211] along with improved text tokenizers [212, 213] have revolutionized the area of natural language generation.

Researchers have sought to replicate the success of transformer-based foundation models trained on tokenized data by applying tokenization to domains including images [214], audio [215], video [207], and time series [195]. Time series analysis in particular has seen success from adapting language models pretrained on tokenized text to tasks like forecasting, classification, and anomaly detection [216, 217, 218, 194]. This convergence suggests that tokenization may provide a unified framework for processing diverse types of sequential data, though challenges remain in developing efficient tokenization schemes that preserve the relevant temporal patterns and relationships in time series data.

Approaches intended for time series use either codebook-based or classical approaches for tokenization. For example, TOTEM [195] and Wav2Vec 2.0 [219] train VQVAEs to learn a tokenization. Classical approaches include methods from functional analysis like wavelets (WaveToken) [205], quantile binning (Chronos) [204], or symbolic approaches (SAX) [220]. Most methods are evaluated under a specialist regime where the model is trained on data tailored to the task. SAX and TOTEM are unique in that they evaluate multiple independent tasks to see behavior across specialist domains. Seeking similar generalist capabilities to foundation models for text, WaveToken, Chronos, and TOTEM evaluate models with generalist training (with samples from different datasets and tasks) and compare against models trained in the specialist regime. Overall, these models are similar in that their vocabulary sizes are on the order of 1K to 10K.

In contrast to tokenization, other approaches for time series patch the data either in time [221] or in space [222, 26]. This approach is similar to tokenization in that

the possibly continuous time series signal is broken down into a discrete number of representations. The key difference is that the patch embeddings are continuous and dynamically determined by input linear layers, so patching provides little of the compression afforded by tokenization.

7.1.2 Contributions

In summary, the contributions of this chapter are as follows:

1. the first application of LFQ methods for learning time series-specific tokenizations;
2. BOUQUET, a simple, stochastic, and stable lookup-free tokenizer that addresses shortcomings of LFQ;
3. a comprehensive suite of experiments and ablations demonstrating BOUQUET's state-of-the-art and near-state-of-the-art performance on time series imputation and anomaly detection versus other lookup-free quantizers and time series models; and
4. a study of the time and memory costs of scaling token vocabularies, culminating in a scaling law that relates vocabulary size to reconstruction performance for vocabularies of over one billion tokens.

7.2 Preliminaries

The notation of this chapter follows the conventions of Chapter 6, and symbols carry over where concepts coincide: data points are denoted $\boldsymbol{x}$, latent embeddings $\boldsymbol{z}$, and the encoder and decoder $\boldsymbol{g}$ and $\boldsymbol{h}$. As before, bold lowercase symbols denote vectors and functions, bold uppercase symbols denote matrices, and calligraphic symbols denote sets.

Time series are sequences of (possibly multivariate) observations ordered in time, which we denote $\mathbf{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T) \in \mathbb{R}^{T \times d_x}$, where T is the length of the sequence and d_x is the dimensionality of the observations.

Tokenization is the process of mapping a corpus of data into a sequence of tokens from a discrete vocabulary $\mathcal{V} = \{1, \ldots, L\}$. This chapter learns this mapping on continuous data with an autoencoder parameterized by 3 modules:

- an encoder $\boldsymbol{g} : \mathbb{R}^{T \times d_x} \to \mathbb{R}^{T' \times d_z}$ that maps inputs to continuously-embedded sequences of length T';

- a quantizer $\boldsymbol{q} : \mathbb{R}^{T' \times d_z} \to \mathcal{V}^{T'}$ that discretizes the continuous embeddings into codes in $\mathcal{V}$; and
- a decoder $\boldsymbol{h} : \mathcal{V}^{T'} \to \mathbb{R}^{T \times d_x}$ that reconstructs the original data.

7.2.1 Vector Quantized Variational Autoencoders

One approach to tokenization is that of Vector Quantized Variational Autoencoders (VQVAEs) [206], which uniquely map indices in $\mathcal{V}$ to vectors in a learned codebook $\mathbf{C} \in \mathbb{R}^{L \times d_z}$:

$$\boldsymbol{z} = \boldsymbol{g}(\boldsymbol{x}), \tag{7.1a}$$

$$\boldsymbol{z}_{\mathrm{q}} = \boldsymbol{q}(\boldsymbol{z}) \triangleq \underset{\mathbf{C}_i \in \mathbf{C}}{\arg\min} \|\boldsymbol{z} - \mathbf{C}_i\|_2, \tag{7.1b}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{h}(\boldsymbol{z}_{\mathrm{q}}). \tag{7.1c}$$

VQVAEs use straight-through estimation to differentiate through the quantization procedure, and to prevent codebook collapse, rely on specialized heuristic loss terms that are quite sensitive to tune. Many additional algorithmic approaches have been developed to combat codebook collapse, including exponential moving average (EMA) code updates, reinitialization of dead codes, stochastic code usage over a categorical distribution, and affine codebook reparameterizations. For a thorough overview of these methods, we refer the reader to Huh et al. [223].

The critical fact that allows such heuristics to be effective is that each element (an index) of the vocabulary corresponds to some embedding vector in latent space. This affords an enormous degree of freedom when deciding *which* vector a particular index corresponds to. However, because the discrete notion of the index is somewhat independent of the location of the corresponding embedding, controlling properties like coverage in a numerically stable way remains a large challenge in codebook learning.

7.2.2 Lookup-Free Quantization Methods

VQVAEs face scaling limitations, because quantizing $\boldsymbol{z}$ to $\boldsymbol{z}_{\mathrm{q}}$ requires comparing $\boldsymbol{z}$ to every row in $\mathbf{C}$ (see (7.1b)). As vocabulary sizes grow, the memory requirements for storing $\mathbf{C}$ and the computational cost of these comparisons quickly become prohibitive.

Lookup-free quantization (LFQ) [207] instead eliminates the explicit codebook entirely. In contrast with the VQVAE, LFQ directly quantizes $\boldsymbol{z}$ into a binary string

with d_z bits, *implicitly* defining a vocabulary of size 2^{d_z} by simply increasing the encoder's output dimension:

$$\boldsymbol{q}(z) \triangleq z + \text{sg}[\text{sign}(z) - z] \in \{-1, 1\}^{d_z}, \tag{7.2}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator used for straight-through estimation [206]. For example, an embedding vector $(1, 1, -1, 1)$ could be interpreted as the binary string 1101, which can be expressed as the index 13. This explicitly relates indices back to the geometry of the embedding space, ensuring properties like coverage of the latent space of any given codebook size.

However, LFQ has some drawbacks. First, z_q is constrained to be a binary string rather than a learnable code, precluding the use of most of the aforementioned codebook utilization heuristics [223]. To address this, Yu et al. [207] suggest adding the entropy loss terms

$$\mathcal{L}_\text{ent} = \mathbb{E}_z\left[H(\boldsymbol{q}(z))\right] - H(\mathbb{E}_z\left[\boldsymbol{q}(z)\right]), \tag{7.3}$$

which balance the per-sample and average-code entropy. In effect, this encourages the actual usage of the codes to be wide in order to encourage representationl diversity.

Second, while these terms may encourage code usage with careful tuning, LFQ's performance is sensitive to the resulting loss weights (leading to training instabilities in our experiments), and computing the expectations in (7.3) requires an expensive summation over all 2^{d_z} implicit codes, negating much of LFQ's memory savings.

Binary Spherical Quantization (BSQ) [224] addresses these drawbacks with the following quantizer:

$$\boldsymbol{u}_\text{q} = \frac{1}{\sqrt{d_z}}\text{sign}\left(\boldsymbol{u}(z)\right), \quad \boldsymbol{u}(z) = \frac{z}{\|z\|_2}, \tag{7.4a}$$

$$\boldsymbol{u}_{\text{q,ste}} = \boldsymbol{q}(z) \triangleq \boldsymbol{u} + \text{sg}[\boldsymbol{u}_\text{q} - \boldsymbol{u}]. \tag{7.4b}$$

First, due to the spherical projection, BSQ exhibits bounded quantization error, which encourages code exploration and leads to faster convergence with the straight-through estimator. Second, Zhao, Xiong, and Krähenbühl [224] propose a *probabilistic* interpretation of the deterministic computation in (7.4a) by viewing $\boldsymbol{u}_\text{q}$ as a vector of Bernoulli random variables (on the hypersphere). Since the elements of $\boldsymbol{u}_\text{q}$ are conditionally independent given $\boldsymbol{u}$, they define the "distribution"

$$p(\boldsymbol{u}_\text{q} \mid \boldsymbol{u}) = \prod_{i=1}^{d_z} \sigma(2\tau \boldsymbol{u}_i \boldsymbol{u}_{\text{q},i}), \tag{7.5}$$

where σ is the sigmoid and τ is a temperature parameter.

Because $p(\boldsymbol{u}_{\mathrm{q}} \mid \boldsymbol{u})$ is factorized, it allows memory-efficient computation and approximation of the loss terms in (7.3). Specifically, they compute the first term exactly and approximate the second:

$$\mathbb{E}_z\left[H(\boldsymbol{q}(z))\right] = \mathbb{E}_z\left[\sum_{i=1}^{d_z} H(\sigma(2\tau \boldsymbol{u}_i \boldsymbol{u}_{\mathrm{q},i}))\right], \tag{7.6a}$$

$$H(\mathbb{E}_z\left[\boldsymbol{q}(z)\right]) \approx \sum_{i=1}^{d_z} H\left(\mathbb{E}_z\left[\sigma(2\tau \boldsymbol{u}_i \boldsymbol{u}_{\mathrm{q},i})\right]\right). \tag{7.6b}$$

7.3 Bernoulli-Quantized Tokenization

This section introduces our method, Bernoulli-Quantized Tokenization (BOUQUET), which we use to train tokenizers that scale to large vocabulary sizes and effectively explore codes during training without loss weight tuning.

While BSQ simply *interprets* a deterministic quantizer probabilistically, BOUQUET actually treats the binary string generation process *stochastically*, producing $\boldsymbol{z}_{\mathrm{q}}$ during training by sampling binary strings from learned Bernoulli distributions. This empirically encourages sufficient exploration over the implicit codes *without requiring entropy regularization or commitment losses*. Thus, BOUQUET can be trained with just the reconstruction loss $\mathcal{L}_{\mathrm{recon}} = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$, eliminating the challenge of highly-sensitive loss weight tuning.

BOUQUET performs quantization during training via:

$$\boldsymbol{p}(z) = \sigma(\mathrm{BatchNorm}(z)), \tag{7.7a}$$

$$\boldsymbol{b}(z) \sim \mathrm{Ber}(\boldsymbol{p}(z)), \tag{7.7b}$$

$$\boldsymbol{z}_{\mathrm{q,ste}} = \boldsymbol{q}(z) \triangleq 2 \cdot (\boldsymbol{p} + \mathrm{sg}[\boldsymbol{b} - \boldsymbol{p}]) - 1, \tag{7.7c}$$

where σ is the sigmoid and Ber is a vector-valued Bernoulli distribution with support $\{0,1\}^{d_z}$. Here, $\boldsymbol{p}(z)$ are per-bit sampling probabilities that parameterize the distribution $\mathrm{Ber}(\boldsymbol{p})$ from which we sample bit strings $\boldsymbol{b} \in \{0,1\}^{d_z}$. We backpropagate through the quantization using a combination of the reparameterization trick and the straight-through estimator. At inference time, we simply let $\boldsymbol{q}(z) = 2 \cdot \mathbb{1}[\boldsymbol{p}(z) > 0.5] - 1$, where $\mathbb{1}[\cdot]$ is the indicator function.

BatchNorm. The BatchNorm layer plays a critical role in BOUQUET's performance. By centering and scaling the inputs to the sigmoid, the probabilities in

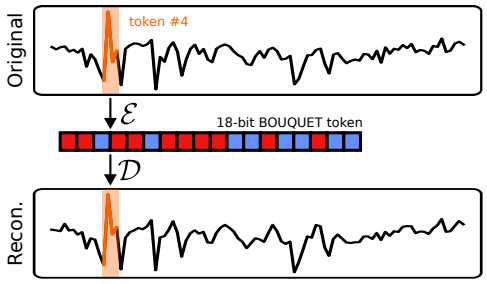


Figure 7.2: One BOUQUET token corresponding to four steps of a time series. Each token represents a temporally compressed sequence with a user-specified temporal compression factor.

(7.7a) remain well-distributed across $[0, 1]$ rather than at the extremes, and specifically, are initialized with values close to 0.5, encouraging code exploration (see Fig. 7.11).

Entropy Loss. Because BOUQUET directly models the Bernoulli distributions generating the predicted bits, if one wishes to use entropy losses, we can substitute the entropy and expected value of the Bernoulli distribution into (7.6). We denote this variant $\text{BOUQUET}_{\text{ENT}}$, and we ablate on it and the BatchNorm in Sec. 7.5.

Time Series. To apply BOUQUET to time series tokenization, we adapt TOTEM, a simple convolutional VQVAE tokenizer proposed by Talukder, Yue, and Gkioxari [195], and change the quantizer to compute (7.7). This yields a (non-causal) tokenizer that performs temporal compression on time series data with a user-specified temporal compression factor. For a visualization of a real token, see Fig. 7.2. Qualitative analyses demonstrating the difference between the internal representations of BOUQUET and BSQ can be found in Sec. 7.6.

7.4 Experiments

Our experiments primarily evaluate three aspects: (i) the efficacy of lookup-free quantizers versus each other, (ii) their overall efficacy versus existing time series methods, and (iii) the computational and performance consequences of scaling the vocabulary. We choose the tasks of time series *imputation* and *anomaly detection*, which can be solved by training a performant autoencoder without any additional models. This allows us to focus the evaluation on the tokenization framework introduced in Sec. 7.2 in isolation from other downstream modeling choices, which keeps our experiments tightly scoped.

For each task, we study two regimes: the *specialist* regime, where a tokenizer is

trained on one dataset and evaluated on a test set from the same distribution, and the *generalist* regime, where it is trained over a union of different datasets and evaluated on each of the same test sets. If a tokenizer has high expressive capacity, then we expect that it will retain performance when generalizing across many datasets (which are often also from different domains).

We follow the standard practice of normalizing each dataset by subtracting the mean and dividing by the standard deviation over all times [225, 194, 195]. For consistency, we download and use the processed data supplied by Talukder, Yue, and Gkioxari [195]. For anomaly detection, we additionally tokenize our data in an instance normalized space [226], where we further subtract the mean and divide by the standard deviation of each individual time series.

Training Details. All tokenizers use a temporal compression factor of 4 time steps, as in TOTEM [195]. The headline imputation and anomaly detection tables report results with vocabularies of 2^{24} codes, and the vocabulary sweep in Sec. 7.4.4 varies the size from 2^{10} to 2^{30}. All time series models are trained on a single GPU. The result tables in this chapter abbreviate dataset names for compactness, and Table 7.1 maps each abbreviation to its full name.

7.4.1 Imputation

Task Definition and Metrics. The goal of time series imputation is to consider a corrupted (i.e., masked) time series $\mathbf{X}_{\mathrm{m}} \in \mathbb{R}^{T \times d_x}$ and impute the missing values to reconstruct the original data $\mathbf{X} \in \mathbb{R}^{T \times d_x}$. We measure performance by computing the MSE and MAE of the predicted reconstruction $\hat{\mathbf{X}}$ versus this data, where both metrics are reported in the normalized space as in prior work [225, 194, 195].

Datasets. We evaluated our models on six datasets: Weather (W), Electricity (E), ETTm1 (m1), ETTm2 (m2), ETTh1 (h1), and ETTh2 (h2) [194], where these last datasets correspond to different traffic flow patterns at either minute or hour granularity. For each dataset, we evaluate performance on 4 distinct masking levels: 12.5%, 25%, 37.5%, and 50%.

Baselines. For visual clarity, Fig. 7.3 only shows the results for BOUQUET, BSQ, TOTEM [195], and GPT2 [194]. We emphasize that the scope of our comparison is broad: the detailed results for the specialist regime are shown in Table 7.3, which compares 18 different methods plus BOUQUET's entropy variant. Of these, 3 are lookup-free methods, 4 are methods claiming SoTA results in 2024, and the remainder are standard time series baselines (see Table 7.2 for a summary of all

Table 7.1: Dataset abbreviations and their full names.

Abbr.	Dataset	Abbr.	Dataset
Imputation			
W	Weather	E	Electricity
m1	ETTm1	m2	ETTm2
h1	ETTh1	h2	ETTh2
Anomaly Detection			
LSD	1sddb40	BID	BIDMC1
CH5	CHARISfive	CH1	CHARISten
CIM	CIMIS44AirTemperature3	CI5	CIMIS44AirTemperature5
EC2	ECG2	EC3	ECG3
FAN	Fantasia	GP7	GP711MarkerLFM5z4
GP5	GP711MarkerLFM5z5	IB4	InternalBleeding4
IB5	InternalBleeding5	IPD	Italianpowerdemand
LG5	Lab2Cmac011215EPG5	LG6	Lab2Cmac011215EPG6
MD	MesoplodonDensirostris	PD1	PowerDemand1
TKF	TkeepFirstMARS	TKS	TKeepSecondMARS
WA5	WalkingAceleration5	ACG	apneaecg
AC2	apneaecg2	GA1	gait1
GH1	gaitHunt1	EPG	insectEPG2
EP4	insectEPG4	LTS	ltstdbs30791AS
MIT	mit14046longtermecg	P3M	park3m
Q5V	qtdbSel1005V	QII	qtdbSel100MLII
R1	resperation1	SL2	s20101mML2
S49	sddb49	CG1	sel840mECG1
CG2	sel840mECG2	T44	tilt12744mtable
T54	tilt12754table	PB2	tiltAPB2
PB3	tiltAPB3	WEW	weallwalk
SMD	Server Machine	MSL	Mars Rover
SMAP	Soil Moisture	SWAT	Water Treatment
PSM	Pooled Server		

Token-based	Model	Abbr.	Arch.	Trained By
✓	BOUQUET	-	Bernoulli Stochastic Sampler	Us
✓	BSQ	-	Binary Spherical Quantizer	Us
✓	LFQ	-	Lookup Free Quantizer	Us
✓	TOTEM	-	VQVAE	[195]
x	GPT2	-	TF	[195]
x	MOMENT	-	TF	[227]
Multi-task	GPT2	GPT2	TF	[194]
	TimesNet	TiNet	Conv	[225]
	UniTS	UniTS	TF	[228]
	UniTS	UniTS	TF	[229]
Single-task	PatchTST	Patch	TF	[193]
	ETSFormer	ETS	TF	[230]
	Fedformer	FED	TF	[231]
	Non-stationary Trans.	Stat	TF	[232]
	Autoformer	Auto	TF	[26]
	Informer	Inf	TF	[233]
	Reformer	Re	TF	[234]
	LightTS	LiTS	Linear	[235]
	DLinear	DLin	Linear	[236]

Table 7.2: **Imputation Baselines.** In our imputation modeling experiments, we explore a diverse set of approaches across two dimensions: (1) whether they use time series tokens (Tokenizer-Based) versus time series patches (Not Tokenizer-Based), and (2) whether they are designed for multiple tasks (**multi-task**) or a single task (**single-task**).

baselines).

Results. In the specialist experiments, we find that BOUQUET far outperforms the existing state of the art in time series imputation across both MSE and MAE. For a visualization of the best 4 methods, see Fig. 7.3. For the most detailed metric values, see Table 7.3.

Over all datasets, metrics, and masking levels, we report that BOUQUET outperforms all other methods 93.8% of the time. Over all methods in the specialist regime, the only other method that matches the performance of lookup-free quantizers is TOTEM, but only for the MSE metric on the ETTm2 dataset.

In the generalist case, the gap in performance between lookup-free tokenizers and time series-specific methods grows even larger, as shown in Fig. 7.3. Moreover, on several datasets, BOUQUET more clearly begins to outperform BSQ, even though they performed similarly in the specialist regime. Notably, we find that BOUQUET's performance drop when moving from the specialist to the generalist case is minimal. For example, examining the most severe masking level across all datasets, we find that the average drop in reconstruction MSE for BOUQUET is 0.0037, for BSQ is 0.0092, for TOTEM is 0.0122, and for GPT2 is 0.0393. This approximately order of magnitude improvement in specialist-to-generalist is typical when comparing BOUQUET to the classical time series methods in Table 7.3. The complete generalist results are reported in Table 7.4.

We also note that the variant $\text{BOUQUET}_{\text{ENT}}$, which uses the BSQ entropy loss

		Tokenizer-Based																			
	Model	BOUQUET		BOUQUET$_{ENT}$		BSQ		LFQ		TOTEM		Not Tokenizer-Based									
												MOMENT*		UniTS		UTSD		GPT2		TiNet	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
W	12.5%	**0.006**	**0.027**	**0.006**	**0.027**	0.007	0.032	0.615	0.593	0.028	0.046	0.033	0.073	0.025	0.041	0.024	0.040	0.026	0.049	0.025	0.045
	25%	**0.009**	**0.030**	**0.009**	**0.030**	0.010	0.040	0.615	0.593	0.029	0.047	0.036	0.078	0.026	0.044	0.026	0.043	0.028	0.052	0.029	0.052
	37.5%	**0.012**	**0.033**	0.013	**0.033**	0.014	0.040	0.615	0.593	0.031	0.048	0.034	0.075	0.027	0.045	0.030	0.047	0.033	0.060	0.031	0.057
	50%	**0.017**	**0.037**	**0.017**	**0.037**	0.018	0.044	0.615	0.593	0.033	0.052	0.035	0.075	0.029	0.049	0.033	0.052	0.037	0.065	0.034	0.062
E	12.5%	0.015	0.077	**0.014**	**0.073**	0.017	0.086	1.012	0.829	0.054	0.154	0.095	0.211	0.031	0.112	0.043	0.129	0.080	0.194	0.085	0.202
	25%	0.024	0.092	**0.023**	**0.090**	0.027	0.100	1.012	0.829	0.059	0.160	0.093	0.211	0.035	0.119	0.049	0.142	0.087	0.203	0.089	0.206
	37.5%	**0.035**	0.112	**0.035**	**0.109**	0.039	0.120	1.012	0.829	0.067	0.169	0.094	0.211	0.040	0.128	0.056	0.151	0.094	0.211	0.094	0.213
	50%	0.051	0.135	0.051	**0.134**	0.056	0.143	1.012	0.829	0.079	0.183	0.092	0.210	**0.046**	0.138	0.065	0.165	0.101	0.220	0.100	0.221
m1	12.5%	**0.007**	**0.045**	0.009	0.056	**0.007**	0.047	1.109	0.800	0.049	0.125	0.069	0.170	0.015	0.079	0.019	0.077	0.017	0.085	0.019	0.092
	25%	**0.011**	**0.054**	0.013	0.064	**0.011**	0.056	1.109	0.800	0.052	0.128	0.071	0.169	0.017	0.082	0.022	0.092	0.022	0.096	0.023	0.101
	37.5%	**0.016**	**0.066**	0.017	0.074	**0.016**	0.067	1.109	0.800	0.055	0.132	0.069	0.163	0.019	0.088	0.028	0.110	0.029	0.111	0.029	0.111
	50%	**0.023**	**0.080**	0.024	0.087	**0.023**	0.081	1.109	0.800	0.061	0.139	0.086	0.169	0.024	0.097	0.035	0.117	0.040	0.128	0.036	0.124
m2	12.5%	**0.009**	**0.047**	0.010	0.054	0.010	0.051	3.025	1.333	0.016	0.078	0.032	0.108	-	-	0.018	0.079	0.017	0.076	0.018	0.080
	25%	**0.012**	**0.054**	0.013	0.060	**0.012**	0.057	3.025	1.333	0.017	0.081	0.029	0.105	-	-	0.019	0.082	0.020	0.080	0.020	0.085
	37.5%	**0.015**	**0.062**	0.016	0.067	0.016	0.065	3.025	1.333	0.018	0.084	0.032	0.109	-	-	0.021	0.085	0.022	0.087	0.023	0.091
	50%	**0.019**	**0.071**	0.021	0.077	0.020	0.074	3.025	1.333	0.020	0.088	0.031	0.110	-	-	0.024	0.094	0.025	0.095	0.026	0.098
h1	12.5%	0.030	**0.095**	0.033	0.097	**0.029**	0.096	1.117	0.805	0.119	0.212	0.160	0.239	0.032	0.118	0.040	0.137	0.043	0.140	0.057	0.159
	25%	0.041	**0.111**	0.043	0.113	0.040	0.112	1.117	0.805	0.127	0.220	0.142	0.238	**0.036**	0.126	0.053	0.155	0.054	0.156	0.069	0.178
	37.5%	0.055	**0.130**	0.057	0.132	0.054	0.131	1.117	0.805	0.138	0.230	0.121	0.228	**0.047**	0.142	0.070	0.175	0.072	0.180	0.084	0.196
	50%	0.077	**0.156**	0.080	0.158	0.076	0.157	1.117	0.805	0.157	0.247	0.132	0.231	**0.060**	0.160	0.093	0.202	0.107	0.216	0.102	0.215
h2	12.5%	0.022	0.077	**0.021**	0.075	**0.021**	**0.074**	3.048	1.336	0.040	0.129	0.051	0.150	-	-	0.040	0.124	0.039	0.125	0.040	0.130
	25%	0.028	0.087	**0.027**	0.086	0.028	**0.085**	3.048	1.336	0.041	0.131	0.079	0.177	-	-	0.043	0.131	0.044	0.135	0.046	0.141
	37.5%	**0.035**	**0.098**	**0.035**	**0.098**	**0.035**	**0.098**	3.048	1.336	0.043	0.136	0.056	0.155	-	-	0.049	0.143	0.051	0.147	0.052	0.151
	50%	**0.044**	**0.113**	**0.044**	**0.113**	**0.044**	**0.113**	3.048	1.336	0.047	0.142	0.056	0.154	-	-	0.053	0.155	0.059	0.158	0.060	0.162
AvgWins		**93.8%**		14.6%		12.5%		0%		8.3%		0%		0%		0%		0%		0%	

		Not Tokenizer-Based (cont.)																	
	Model	Patch		ETS		FED		Stat		Auto		Inf		Re		LiTS		Dlin	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
W	12.5%	0.029	0.049	0.057	0.141	0.041	0.107	0.027	0.051	0.026	0.047	0.037	0.093	0.031	0.076	0.047	0.101	0.039	0.084
	25%	0.031	0.053	0.065	0.155	0.064	0.163	0.029	0.056	0.030	0.054	0.042	0.100	0.035	0.082	0.052	0.111	0.048	0.103
	37.5%	0.035	0.058	0.081	0.180	0.107	0.229	0.033	0.062	0.032	0.060	0.049	0.111	0.040	0.091	0.058	0.121	0.057	0.117
	50%	0.038	0.063	0.102	0.207	0.183	0.312	0.037	0.068	0.037	0.067	0.053	0.114	0.046	0.099	0.065	0.133	0.066	0.134
E	12.5%	0.055	0.160	0.196	0.321	0.107	0.237	0.093	0.210	0.089	0.210	0.218	0.326	0.190	0.308	0.102	0.229	0.092	0.214
	25%	0.065	0.175	0.207	0.332	0.120	0.251	0.097	0.214	0.096	0.220	0.219	0.326	0.197	0.312	0.121	0.252	0.118	0.247
	37.5%	0.076	0.189	0.219	0.344	0.136	0.266	0.102	0.220	0.104	0.229	0.222	0.328	0.203	0.315	0.141	0.273	0.144	0.276
	50%	0.091	0.208	0.235	0.357	0.158	0.284	0.108	0.228	0.113	0.239	0.228	0.331	0.210	0.319	0.160	0.293	0.175	0.305
m1	12.5%	0.041	0.130	0.067	0.188	0.035	0.135	0.026	0.107	0.034	0.124	0.047	0.155	0.032	0.126	0.075	0.180	0.058	0.162
	25%	0.044	0.135	0.096	0.229	0.052	0.166	0.032	0.119	0.046	0.144	0.063	0.180	0.042	0.146	0.093	0.206	0.080	0.193
	37.5%	0.049	0.143	0.133	0.271	0.069	0.191	0.039	0.131	0.057	0.161	0.079	0.200	0.063	0.182	0.113	0.231	0.103	0.219
	50%	0.055	0.151	0.186	0.323	0.089	0.218	0.047	0.145	0.067	0.174	0.093	0.218	0.082	0.208	0.134	0.255	0.132	0.248
m2	12.5%	0.026	0.094	0.108	0.239	0.056	0.159	0.021	0.088	0.023	0.092	0.133	0.270	0.108	0.228	0.034	0.127	0.062	0.166
	25%	0.028	0.099	0.164	0.294	0.080	0.195	0.024	0.096	0.026	0.101	0.135	0.272	0.136	0.262	0.042	0.143	0.085	0.196
	37.5%	0.030	0.104	0.237	0.356	0.110	0.231	0.027	0.103	0.030	0.108	0.155	0.293	0.175	0.300	0.051	0.159	0.106	0.222
	50%	0.034	0.110	0.323	0.421	0.156	0.276	0.030	0.108	0.035	0.119	0.200	0.333	0.211	0.329	0.059	0.174	0.131	0.247
h1	12.5%	0.093	0.201	0.126	0.263	0.070	0.190	0.060	0.165	0.074	0.182	0.114	0.234	0.074	0.194	0.240	0.345	0.151	0.267
	25%	0.107	0.217	0.169	0.304	0.106	0.236	0.080	0.189	0.090	0.203	0.140	0.262	0.102	0.227	0.265	0.364	0.180	0.292
	37.5%	0.120	0.230	0.220	0.347	0.124	0.258	0.102	0.212	0.109	0.222	0.174	0.293	0.135	0.261	0.296	0.382	0.215	0.318
	50%	0.141	0.248	0.293	0.402	0.165	0.299	0.133	0.240	0.137	0.248	0.215	0.325	0.179	0.298	0.334	0.404	0.257	0.347
h2	12.5%	0.057	0.152	0.187	0.319	0.095	0.212	0.042	0.133	0.044	0.138	0.305	0.431	0.163	0.289	0.101	0.231	0.100	0.216
	25%	0.061	0.158	0.279	0.390	0.137	0.258	0.049	0.147	0.050	0.149	0.322	0.444	0.206	0.331	0.115	0.246	0.127	0.247
	37.5%	0.067	0.166	0.400	0.465	0.187	0.304	0.056	0.158	0.060	0.163	0.353	0.462	0.252	0.370	0.126	0.257	0.158	0.276
	50%	0.073	0.174	0.602	0.572	0.232	0.341	0.065	0.170	0.068	0.173	0.369	0.472	0.316	0.419	0.136	0.268	0.183	0.299
AvgWins		0%		0%		0%		0%		0%		0%		0%		0%		0%	

Table 7.3: **Specialist Imputation Results.** All lookup-free tokenizers use vocabularies of 2^{24} codes. Note *MOMENT follows a different evaluation procedure, where instead of randomly masking individual time steps, it masks contiguous chunks of 8 time steps at the given masking ratio. We keep their results in for completeness. Under all conditions in their paper, MOMENT is generally outperformed by GPT-2, so we consider this distinction to have minimal impact on our comparisons. Here we report mean squared error (MSE) and mean absolute error (MAE). We see that BOUQUET is state-of-the-art (SOTA) or near-SOTA for every dataset and threshold. We get an `AvgWins` of 93.8%. `AvgWins` is the total number of tasks a model receives SOTA (ties included) divided by the number of tasks. BOUQUET was SOTA for 45 of the 48 tasks (i.e., 24 examples across two metrics).

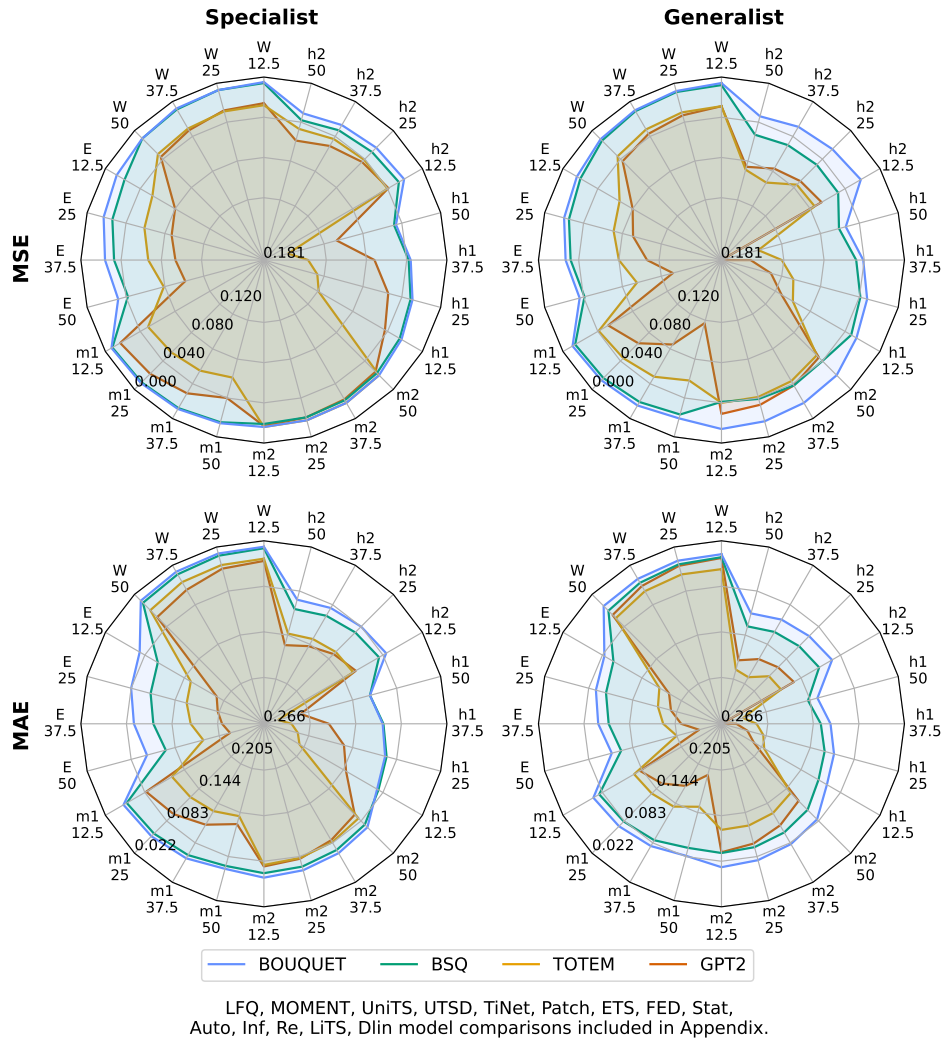


Figure 7.3: **Imputation results.** We visualize four models: our two best lookup-free methods and two high-performing prior works in time series modeling. For almost every dataset and masking level, BOUQUET outperforms all other models in both the specialist and generalist regime. Moreover, BOUQUET suffers the least performance degradation when moving to the generalist regime.

formulation with Bernoulli distribution entropy losses, slightly outperforms BOUQUET in the imputation generalist experiments. This suggests that in some cases, even though BOUQUET can stably train without them, entropy loss terms may improve generalization performance.

Finally, we remark that in imputation, vanilla LFQ achieves poor performance. This is due entirely to poor training stability, and not the inability for LFQ to learn tokenizations on these datasets. When examining the encoder outputs z of our vanilla LFQ tokenizer, we find that they reach extremely high values, e.g., on the

Model		BOUQUET		BOUQUET$_{\text{ENT}}$		BSQ		LFQ		TOTEM		GPT2	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
W	12.5%	0.006	0.040	**0.005**	**0.035**	0.008	0.044	0.601	0.582	0.029	0.060	0.029	0.045
	25%	0.008	0.041	**0.007**	**0.037**	0.009	0.046	0.601	0.582	0.030	0.060	0.033	0.048
	37.5%	0.010	0.043	**0.009**	**0.038**	0.011	0.049	0.601	0.582	0.032	0.062	0.037	0.054
	50%	0.012	0.044	**0.011**	**0.041**	0.014	0.053	0.601	0.582	0.036	0.067	0.043	0.061
E	12.5%	0.016	0.086	**0.014**	**0.078**	0.020	0.100	1.019	0.831	0.065	0.171	0.080	0.186
	25%	0.020	0.093	**0.019**	**0.086**	0.025	0.107	1.019	0.831	0.071	0.179	0.091	0.197
	37.5%	0.027	0.102	**0.025**	**0.096**	0.032	0.116	1.019	0.831	0.080	0.189	0.108	0.213
	50%	0.036	0.115	**0.035**	**0.109**	0.041	0.128	1.019	0.831	0.095	0.205	0.132	0.236
m1	12.5%	0.011	0.069	**0.009**	**0.063**	0.014	0.078	1.115	0.810	0.041	0.132	0.052	0.141
	25%	0.013	0.073	**0.011**	**0.067**	0.016	0.082	1.115	0.810	0.044	0.135	0.065	0.154
	37.5%	0.015	0.078	**0.014**	**0.072**	0.019	0.087	1.115	0.810	0.048	0.139	0.085	0.171
	50%	0.019	0.084	**0.018**	**0.079**	0.023	0.095	1.115	0.810	0.058	0.152	0.117	0.196
m2	12.5%	**0.014**	0.075	0.020	**0.070**	0.041	0.094	2.935	1.312	0.040	0.125	0.029	0.095
	25%	**0.016**	0.078	0.021	**0.073**	0.039	0.096	2.935	1.312	0.041	0.126	0.033	0.101
	37.5%	**0.018**	0.081	0.022	**0.077**	0.038	0.099	2.935	1.312	0.043	0.129	0.038	0.110
	50%	**0.020**	0.086	0.024	**0.082**	0.040	0.104	2.935	1.312	0.048	0.136	0.045	0.121
h1	12.5%	0.027	0.104	**0.022**	**0.093**	0.033	0.117	1.123	0.815	0.100	0.201	0.113	0.217
	25%	0.032	0.111	**0.028**	**0.101**	0.039	0.124	1.123	0.815	0.108	0.209	0.131	0.231
	37.5%	0.041	0.121	**0.036**	**0.111**	0.048	0.134	1.123	0.815	0.122	0.220	0.153	0.247
	50%	0.054	0.134	**0.050**	**0.126**	0.061	0.147	1.123	0.815	0.144	0.237	0.182	0.266
h2	12.5%	**0.022**	0.096	0.024	**0.089**	0.048	0.116	2.958	1.316	0.075	0.175	0.067	0.155
	25%	**0.026**	0.101	0.028	**0.095**	0.048	0.120	2.958	1.316	0.076	0.177	0.071	0.160
	37.5%	**0.029**	0.106	0.031	**0.101**	0.050	0.125	2.958	1.316	0.093	0.195	0.077	0.167
	50%	**0.034**	0.114	0.036	**0.109**	0.053	0.132	2.958	1.316	0.089	0.192	0.086	0.179
AvgWins		16.7%		**83.3%**		0%		0%		0%		0%	

Table 7.4: **Generalist Imputation Results.** We train the models on the full imputation training dataset using random masking with different masking ratios. The models are then evaluated using the fixed masking ratios shown in the table.

order of 10^{17}. Because the magnitude of these values is so high, LFQ struggles to correct itself after destabilizing, leading to almost all inputs being quantized to the same codes, i.e., codebook collapse (hence the identical loss values reported in Table 7.3).

7.4.2 Anomaly Detection

Task Definition and Metrics. The goal of anomaly detection is to train a model over non-anomalous time series $\mathbf{X} \in \mathbb{R}^{T \times d_x}$ such that when given a new time series with anomalous data $\mathbf{X}_{\text{anom}} \in \mathbb{R}^{T_{\text{anom}} \times d_x}$, we can classify which datapoints are anomalous and which are not. We follow the standard practice of point adjustment [237] and report the best adjusted F1 metric as in prior works [237, 225, 194, 227, 195].

Datasets. In this section, we compare to a 42 dataset subset of the UCR Anomaly Archive (the exact subset of which is taken directly from Goswami et al. [227] for comparison), which we refer to as UCR42.

Baselines. On UCR42, we compare the lookup-free tokenizers against Anomaly Transformer [237], DGHL [238], GPT2 [194], TimesNet [225], TOTEM [195], and MOMENT [227] (see Table 7.5 for a summary of all baselines). Table 7.6 summarizes performance in both training regimes.

We remark that MOMENT is not present in the specialist comparison, as it is a

Anomaly Detection Baselines

Type	Model	Abbr.	Arch.
Multi-task	GPT2 [194]	GPT2	TF
	TimesNet [225]	TiNet	Conv
Single-task	Anomaly Trans. [237]	ATran	TF
	PatchTST [193]	Patch	TF
	ETSFormer [230]	ETS	TF
	Fedformer [231]	FED	TF
	Non-stationary Trans. [232]	Stat	TF
	Autoformer [26]	Auto	TF
	Pyraformer [239]	Pyra	TF
	Informer [233]	Inf	TF
	Reformer [234]	Re	TF
	LogTrans [222]	LogTr	TF
	Transformer [209]	Trans	TF
	LightTS [235]	LiTS	Linear
	DLinear [236]	DLin	Linear
Other	DGHL [238]	DGHL	-
	MOMENT-LP [227]	MNT	-

Table 7.5: Anomaly detection baselines.

foundation model pre-trained on a large corpus of 129M datapoints. Similarly, the generalist comparison includes only TOTEM and MOMENT beyond the lookup-free tokenizers, as they are the only methods that report results for this experimental setting in the literature.

Results. For the UCR42 specialist experiments, we find that BOUQUET yields the best result, outperforming the best existing anomaly detection methods 61.9% of the time. Table 7.6 summarizes the mean and median F1 scores and win rates of every model in both regimes. The majority of the best specialist results are due to either BOUQUET or BSQ, demonstrating that lookup-free methods are highly effective at tokenizing "typical" data.

In the generalist regime, we find that $\text{MOMENT}_{\text{LP}}$, a pretrained foundation model that performs linear probing on each dataset, outperforms or matches all other models 50% of the time. This is not surprising, given that $\text{MOMENT}_{\text{LP}}$ has 40M parameters and trains their model for several days. However, our BOUQUET model, which only has 242K parameters and is trained in a matter of hours on one GPU, achieves the next-best win rate at 23.8% of the time.

Unlike in imputation, vanilla LFQ performs moderately well in anomaly detection, achieving a 21.4% win rate, even though the training process is very similar to that of imputation. This supports our observations that vanilla LFQ without regularizers like the BatchNorm layer is very sensitive, motivating the use of methods like BOUQUET. Surprisingly, though BSQ far outperformed LFQ in the specialist case, it only achieved a 7.1% win rate in the generalist setting. This implies that BSQ

Model	n	Mean F1	Median F1	Win Rate
Specialist				
BOUQUET	42	**0.840**	**0.978**	**61.9%**
$\text{BOUQUET}_{\text{ENT}}$	42	0.809	0.963	40.5%
BSQ	42	0.752	0.968	33.3%
LFQ+BN	42	0.749	0.950	31.0%
TOTEM	15	0.877	0.995	19.0%
TimesNet	41	0.570	0.550	16.7%
LFQ	42	0.482	0.385	16.7%
DGHL	41	0.417	0.360	11.9%
LFQ-EC	42	0.502	0.533	9.5%
A. Trans.	41	0.475	0.410	9.5%
GPT2	41	0.449	0.390	9.5%
Generalist				
$\text{MOMENT}_{\text{LP}}$	41	**0.684**	0.740	**50.0%**
BOUQUET	42	0.619	**0.821**	23.8%
LFQ	42	0.540	0.560	21.4%
LFQ-EC	42	0.540	0.560	21.4%
$\text{BOUQUET}_{\text{ENT}}$	42	0.523	0.546	19.0%
TOTEM	42	0.616	0.683	16.7%
BSQ	42	0.515	0.501	7.1%
LFQ+BN	42	0.495	0.534	7.1%

Table 7.6: **UCR42 Anomaly Detection Summary.** Mean and median best adjusted F1 scores and win rates over the 42 UCR42 datasets in the specialist and generalist regimes. A model wins a dataset when it attains the best F1 score there, and ties count as wins for every tied model. Here, n is the number of datasets with a reported score for each model. The mean and median are computed over these n datasets, and missing entries never win. Bold marks the best value in each column. TOTEM reports specialist results on only 15 of the 42 datasets, so its specialist mean and median are not directly comparable to the other models and are left unbolded.

may sometimes struggle in generalization. Overall, across all lookup-free methods, BOUQUET consistently achieves the best performance in all tested settings.

Since TOTEM reports specialist results on only 15 of the 42 UCR42 datasets, a complete head-to-head comparison against it is only possible on that subset. We report these UCR15 results in Sec. 7.8.

7.4.3 Computational Efficiency

We now quantify the computational cost of scaling the vocabulary across tokenizer families: the lookup-free quantizers (BOUQUET, BOUQUET$_{\text{ENT}}$, BSQ, and LFQ, each at both 18 and 24 bits), the VQVAE-based TOTEM, autoencoders built on patching and binning, and the transformer baselines GPT2 and MOMENT. The patching and binning autoencoders share our convolutional backbone and replace the quantizer with input and output linear layers (PatchAE), Chronos-style quantile binning (ChronosAE) [204], and WaveToken-style wavelet binning (WaveTokenAE) [205], which isolates the cost of each quantization scheme. Given a fixed backbone, lookup-free methods (and BOUQUET in particular) incur almost no additional time or memory cost even as the vocabulary grows by orders of magnitude.

Total Time (ms)	Train	Eval
BOUQUET (N≃262K)	6.12±0.45	2.45±0.02
BOUQUET (N≃17M)	6.20±0.35	2.45±0.14
BOUQUET$_{\text{ENT}}$ (N≃262K)	6.59±0.32	2.45±0.10
BOUQUET$_{\text{ENT}}$ (N≃17M)	6.61±0.36	2.45±0.02
BSQ (N≃262K)	6.79±0.32	2.32±0.03
BSQ (N≃17M)	6.70±0.37	2.34±0.01
LFQ (N≃262K)	7.14±0.52	2.29±0.12
LFQ (N≃17M)	7.09±0.53	2.29±0.06
TOTEM (N≃1K, D=64)	5.73±0.39	2.32±0.17
WaveTokenAE (N≃1K, D=64)	8.35±0.54	3.61±0.03
ChronosAE (N≃1K, D=64)	6.58±0.45	2.80±0.09
ChronosAE (N≃2K, D=64)	6.47±0.37	3.14±0.07
ChronosAE (N≃4K, D=64)	6.97±0.18	3.84±0.32
PatchAE	5.41±0.22	1.98±0.03
MOMENT	33.90±0.79	24.80±0.30
GPT2	32.63±0.06	13.56±0.03

Table 7.7: **Timing Comparison.** Mean forward and backward pass times for training and forward pass times for inference, with a one standard deviation interval over 1024 runs. WaveTokenAE incurs a small additional cost for the discrete wavelet transform. The first block contains lookup-free tokenizers with N tokens, the second contains lookup-based tokenization methods, and the third contains non-tokenization methods.

Speed. Table 7.7 reports mean training (forward and backward) and inference (forward) times. Lookup-free tokenizers with roughly 17M tokens run at nearly the same speed as TOTEM with only 1K tokens, as wall-clock time is dominated by

the shared convolutional backbone. The lookup-free tokenizers and TOTEM train within 1.5ms of each other with nearly identical inference times, while the transformer baselines train roughly 5–6× slower with roughly 5–10× slower inference.

Peak VRAM Usage (GB)	Train	Eval
BOUQUET (N≃262K)	0.037	0.031
BOUQUET (N≃17M)	0.037	0.031
$\text{BOUQUET}_{\text{ENT}}$ (N≃262K)	0.038	0.031
$\text{BOUQUET}_{\text{ENT}}$ (N≃17M)	0.038	0.031
BSQ (N≃262K)	0.038	0.031
BSQ (N≃17M)	0.038	0.031
LFQ (N≃262K)	0.053	0.035
LFQ (N≃17M)	0.595	0.407
TOTEM (N≃1K, D=64)	0.054	0.054
WaveTokenAE (N≃1K, D=64)	0.205	0.129
ChronosAE (N≃1K, D=64)	0.204	0.128
ChronosAE (N≃2K, D=64)	0.361	0.215
ChronosAE (N≃4K, D=64)	0.683	0.393
PatchAE	0.081	0.034
MOMENT	1.405	1.400
GPT2	3.590	3.321

Table 7.8: **Peak Memory Comparison.** Peak VRAM usage measured with a batch size of 64. The first block contains lookup-free tokenizers with N tokens, the second contains lookup-based tokenization methods, and the third contains non-tokenization methods.

Memory. Table 7.8 reports peak VRAM usage at a batch size of 64. BOUQUET and BSQ scale from 18 to 24 bits, growing the vocabulary from 262K to nearly 17M codes, with almost no change in memory. Vanilla LFQ's cost instead grows by more than 10×, as its entropy terms require a summation over all codes. The factorized entropy computation of BSQ and $\text{BOUQUET}_{\text{ENT}}$ eliminates this cost, and BOUQUET avoids it entirely by using no entropy terms at all. The remaining comparisons isolate where explicit vocabularies pay their costs. TOTEM shares the same backbone yet uses more peak training VRAM than BOUQUET despite having only 1K codes, reflecting the burden of storing and searching an explicit codebook. The binning-based autoencoders learn an embedding for every bin, so their memory also grows linearly with the number of codes. PatchAE pays a modest cost for the linear layers surrounding its backbone, whereas the BOUQUET quantizer is parameter-free aside from its BatchNorm statistics. Lastly, MOMENT and GPT2

use roughly 40× and 100× the memory of BOUQUET respectively.

Parameters. Table 7.9 compares the sizes of the generalist models. BOUQUET uses roughly 242K parameters, about 100K fewer than TOTEM and roughly two orders of magnitude fewer than GPT2 and MOMENT.

	BOUQUET Params	TOTEM Params	GPT2 Params	MOMENT Params
Imputation	~242,000	~345,000	~60,700,000	~40,000,000
Anomaly Detection	~242,000	~345,000	~46,500,000	~40,000,000

Table 7.9: **Parameter Comparison.** The number of parameters of the BOUQUET, TOTEM, GPT2, and MOMENT generalist models. MOMENT trains on a large background corpus. For all other models, the imputation generalists are trained on the Weather, Electricity, ETTm1, ETTm2, ETTh1, and ETTh2 datasets, and the anomaly detection generalists are trained on the SMD, MSL, SMAP, SWAT, and PSM datasets. BOUQUET employs about 100K fewer parameters than TOTEM, and roughly two orders of magnitude fewer parameters than GPT2 and MOMENT.

7.4.4 Scaling the Vocabulary

Because lookup-free tokenizers pay no time or memory penalty for a larger vocabulary, we can ask a question that is out of reach for explicit-codebook methods: how does performance behave as the vocabulary grows without bound? We sweep the number of bits d_z from 10 to 30 in increments of 2 on the specialist imputation task. The vocabulary consists of all d_z-bit strings, so its size 2^{d_z} grows exponentially in the sweep, from $2^{10} = 1{,}024$ tokens up to $2^{30} \approx 1.07 \times 10^9$. In particular, the vocabulary first exceeds one million tokens at 20 bits and one billion tokens at 30 bits.

Fig. 7.4 shows the resulting reconstruction errors. Across all six datasets and all four masking levels, both MSE and MAE continue to improve as the vocabulary grows, and the improvement is approximately linear on a log-log scale. In other words, the experiments exhibit a scaling law relating vocabulary size to reconstruction performance, and the trend persists beyond one billion tokens with no sign of saturation. To our knowledge, tokenizers at this scale have not previously been trained for time series data, where standard vocabularies are on the order of 1K to 10K codes (Sec. 7.1.1). We view this as the central payoff of the lookup-free paradigm: vocabulary size becomes a knob that reliably buys performance rather than a resource bottleneck.

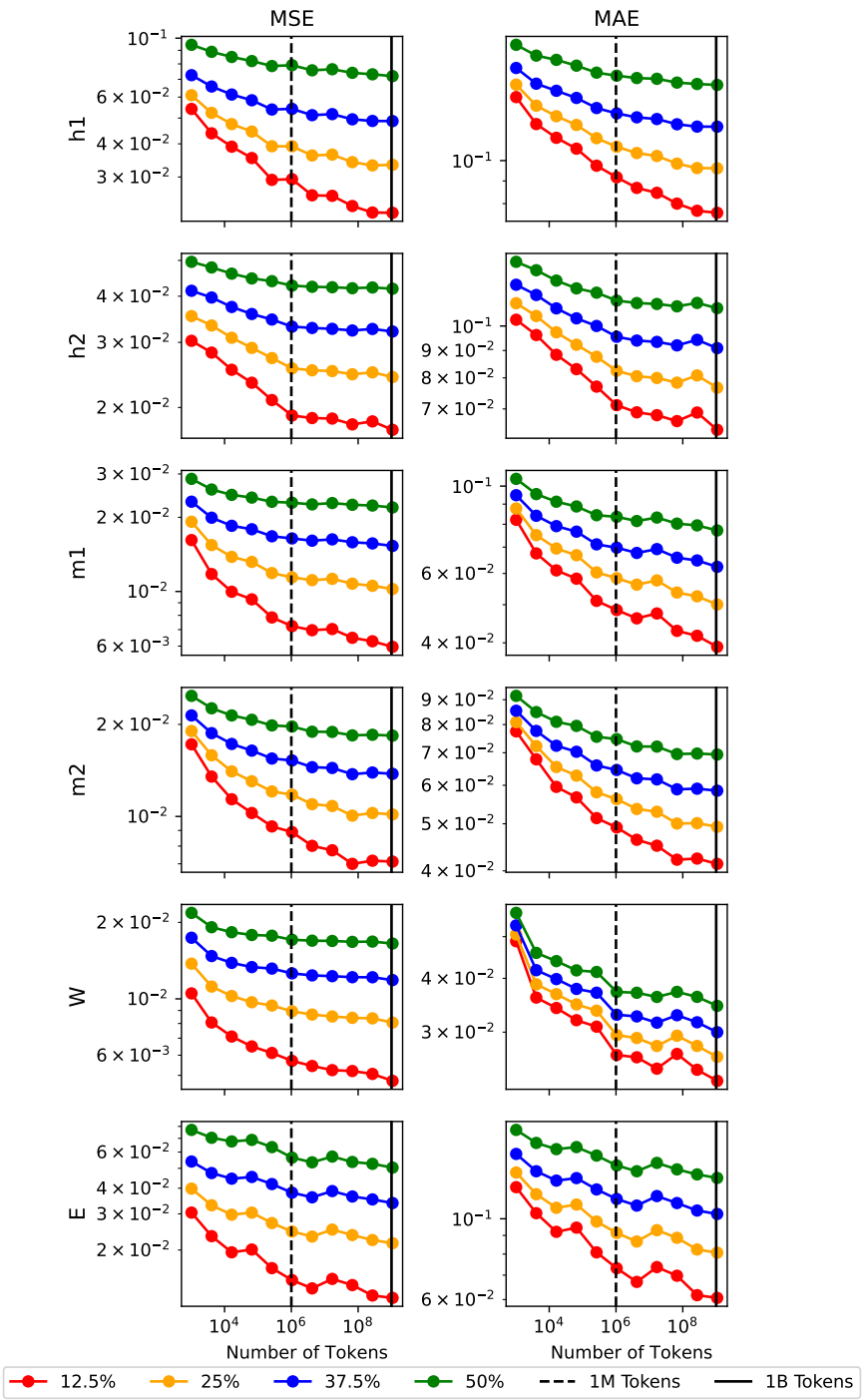


Figure 7.4: **Scaling the Vocabulary Improves Performance.** Specialist imputation error (MSE left, MAE right) as a function of vocabulary size, swept from 2^{10} to 2^{30} tokens across all six datasets (rows) and masking levels (colors), with the 1M and 1B token marks indicated by the dashed and solid vertical lines. Both axes are logarithmic, and the trends are approximately linear on this log-log scale, indicating a scaling law between vocabulary size and reconstruction performance that persists past one billion tokens.

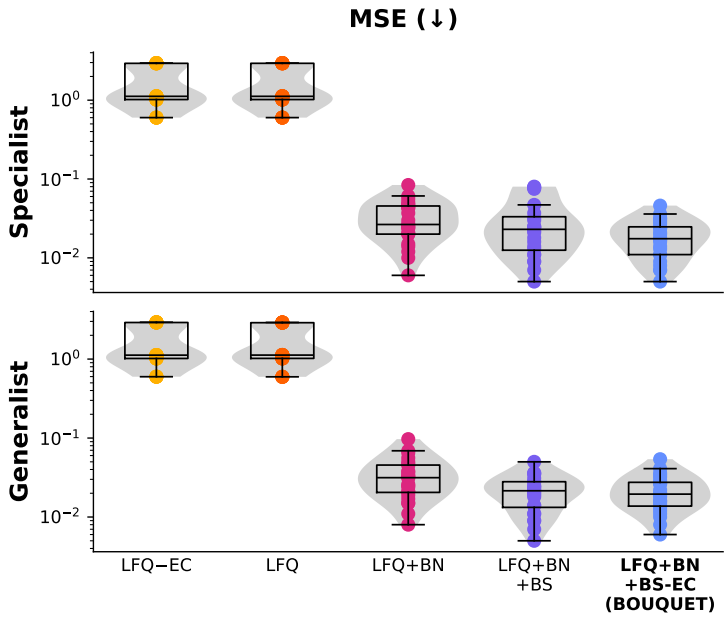


Figure 7.5: Ablation results showing MSE across all imputation datasets, where each dot is an experiment. The ablations evaluate (1) LFQ without entropy/commitment losses (LFQ-EC), (2) vanilla LFQ with entropy/commitment losses, (3) LFQ with batch normalization (LFQ+BN), (4) LFQ with batch normalization and Bernoulli sampling (LFQ+BN+BS), and (5) our final BOUQUET model which removes entropy/commitment losses from LFQ+BN+BS. The top plot shows specialist training results while the bottom shows generalist training.

7.5 Ablations

We conduct several ablation studies to validate the key design choices in BOUQUET: batch normalization, Bernoulli sampling during training, and the removal of entropy and commitment losses. We evaluate these components across both specialist and generalist training regimes for imputation and anomaly detection tasks. We emphasize the wide scope of our ablation experiments: every column in Fig. 7.5 and Fig. 7.6 corresponds to a complete set of the same experiments presented in the previous section on the ablated model, which comprises dozens of evaluations each.

The results in Fig. 7.5 demonstrate the relative impact of each component. Starting with vanilla LFQ (both with and without entropy and commitment losses), we observe consistently high error rates around 10^{0} for the choice of hyperparameters taken from Weber et al. [214]. Adding batch normalization yields a dramatic improvement, reducing error by nearly an order of magnitude to approximately $10^{-1.5}$. This suggests that normalizing the pre-quantized representations plays a crucial role in stabilizing training.

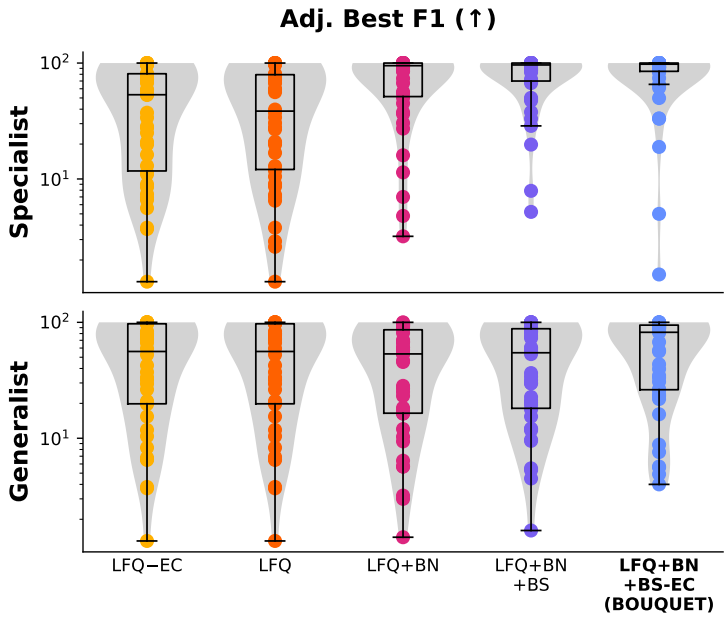


Figure 7.6: Ablation results showing F1 score (higher is better) on the UCR42 anomaly detection dataset. We compare performance across five model variants: (1) LFQ without entropy/commitment losses (LFQ-EC), (2) vanilla LFQ with entropy/commitment losses, (3) LFQ with batch normalization (LFQ+BN), (4) LFQ with batch normalization and Bernoulli sampling (LFQ+BN+BS), and (5) our final BOUQUET model which removes entropy/commitment losses from LFQ+BN+BS. The top plot shows specialist training while the bottom shows generalist training results.

Introducing Bernoulli sampling further improves performance, dropping the error consistently to around 10^{-2}. While this improvement is smaller than that from batch normalization, it notably reduces the variance in performance. Finally, removing the entropy and commitment losses matches the performance when using them in imputation and actually *improves* the median performance in the case of anomaly detection.

The benefits of BOUQUET's design choices become more apparent in the generalist training regime. As shown in our anomaly detection results in Fig. 7.6, LFQ with and without entropy/commitment losses both exhibit high variance in F1 scores. While batch normalization significantly improves specialist performance, the generalist case still shows high variance similar to vanilla LFQ. Adding Bernoulli sampling begins to address this issue, though some poorly performing samples remain. Only after removing the entropy and commitment losses does BOUQUET achieve consistently strong performance across all samples.

A. Specialist In-Domain

Model		LFQ		LFQ-EC		LFQ+BN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
W	12.5%	0.601	0.582	0.601	0.582	**0.006**	**0.034**
	25%	0.601	0.582	0.601	0.582	**0.010**	**0.038**
	37.5%	0.601	0.582	0.601	0.582	**0.014**	**0.042**
	50%	0.601	0.582	0.601	0.582	**0.020**	**0.049**
E	12.5%	1.019	0.831	1.019	0.831	**0.021**	**0.096**
	25%	1.019	0.831	1.019	0.831	**0.030**	**0.11**
	37.5%	1.019	0.831	1.019	0.831	**0.043**	**0.130**
	50%	1.019	0.831	1.019	0.831	**0.043**	**0.130**
m1	12.5%	1.115	0.810	1.115	0.810	**0.012**	**0.063**
	25%	1.115	0.810	1.115	0.810	**0.015**	**0.071**
	37.5%	1.115	0.810	1.115	0.810	**0.020**	**0.080**
	50%	1.115	0.810	1.115	0.810	**0.026**	**0.093**
m2	12.5%	2.935	1.312	2.933	1.311	**0.024**	**0.066**
	25%	2.935	1.312	2.933	1.311	**0.024**	**0.071**
	37.5%	2.935	1.312	2.933	1.311	**0.025**	**0.077**
	50%	2.935	1.312	2.933	1.311	**0.027**	**0.086**
h1	12.5%	1.123	0.815	1.123	0.815	**0.037**	**0.115**
	25%	1.123	0.815	1.123	0.815	**0.047**	**0.128**
	37.5%	1.123	0.815	1.123	0.815	**0.061**	**0.146**
	50%	1.123	0.815	1.123	0.815	**0.084**	**0.170**
h2	12.5%	2.958	1.316	2.959	1.316	**0.045**	**0.102**
	25%	2.958	1.316	2.959	1.316	**0.048**	**0.120**
	37.5%	2.958	1.316	2.959	1.316	**0.051**	**0.119**
	50%	2.958	1.316	2.959	1.316	**0.056**	**0.131**

B. Generalist In-Domain

Model		LFQ		LFQ-EC		LFQ+BN	
Metric		MSE	MAE	MSE	MAE	MSE	MAE
W	12.5%	0.596	0.578	0.598	0.579	**0.008**	**0.042**
	25%	0.596	0.578	0.598	0.579	**0.011**	**0.045**
	37.5%	0.596	0.578	0.598	0.579	**0.015**	**0.050**
	50%	0.596	0.578	0.598	0.579	**0.021**	**0.057**
E	12.5%	1.020	0.831	1.018	0.831	**0.024**	**0.103**
	25%	1.020	0.831	1.018	0.831	**0.033**	**0.118**
	37.5%	1.020	0.831	1.018	0.831	**0.047**	**0.137**
	50%	1.020	0.831	1.018	0.831	**0.065**	**0.160**
m1	12.5%	1.119	0.816	1.117	0.813	**0.015**	**0.078**
	25%	1.119	0.816	1.117	0.813	**0.019**	**0.087**
	37.5%	1.119	0.816	1.117	0.813	**0.025**	**0.097**
	50%	1.119	0.816	1.117	0.813	**0.034**	**0.112**
m2	12.5%	2.890	1.302	2.906	1.305	**0.017**	**0.080**
	25%	2.890	1.302	2.906	1.305	**0.021**	**0.086**
	37.5%	2.890	1.302	2.906	1.305	**0.026**	**0.095**
	50%	2.890	1.302	2.906	1.305	**0.032**	**0.106**
h1	12.5%	1.128	0.821	1.126	0.818	**0.038**	**0.122**
	25%	1.128	0.821	1.126	0.818	**0.050**	**0.138**
	37.5%	1.128	0.821	1.126	0.818	**0.069**	**0.158**
	50%	1.128	0.821	1.126	0.818	**0.097**	**0.185**
h2	12.5%	2.911	1.305	2.928	1.309	**0.031**	**0.109**
	25%	2.911	1.305	2.928	1.309	**0.037**	**0.117**
	37.5%	2.911	1.305	2.928	1.309	**0.045**	**0.129**
	50%	2.911	1.305	2.928	1.309	**0.056**	**0.145**

Table 7.10: **LFQ Imputation Ablations.** We compare LFQ with entropy and commitment loss (LFQ), LFQ without entropy loss (LFQ-EC), and LFQ with a batch norm and entropy loss (LFQ+BN). We see that the batch norm significantly improves performance and results in lower MSE and MAE.

This progression demonstrates that BOUQUET's stochastic training approach naturally encourages code exploration without requiring explicit regularization terms. The combination of batch normalization and Bernoulli sampling provides the stability needed for effective training, while removing unnecessary losses simplifies the optimization problem. These results validate our hypothesis that a simpler training objective combined with built-in exploration via stochastic sampling can outperform more complex deterministic approaches.

The complete numerical results for these ablations are reported in Table 7.10 for imputation and in Tables 7.11 and 7.12 for anomaly detection. For expediency, the anomaly detection ablations are run on the SMD, MSL, SMAP, SWAT, and PSM datasets [194], a widely used suite of multivariate benchmarks covering server, spacecraft, and infrastructure monitoring domains. Because these datasets are standard in prior work and differ in character from the univariate UCR42 archive, they provide a complementary testbed for isolating the effect of each design choice. Fig. 7.7 visualizes the results.

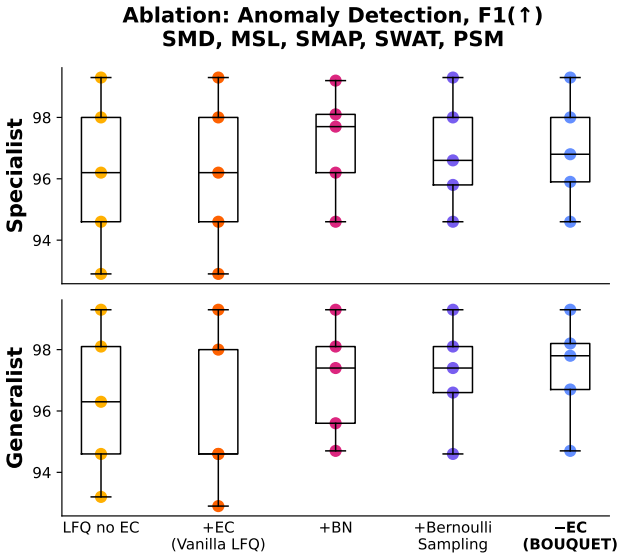


Figure 7.7: **Anomaly detection ablation results** measured by F1 score. Starting from a minimal LFQ configuration without entropy and commitment losses, we reintroduce these losses to recover vanilla LFQ, add batch normalization for a notable performance boost, add Bernoulli sampling for further gains, and finally remove the entropy and commitment losses again to arrive at BOUQUET, which achieves the best results in both the specialist and generalist regimes.

Specialist In-Domain

Model	LFQ	LFQ-EC	LFQ+BN
SMD	0.946	0.946	0.946
MSL	0.962	0.962	0.977
SMAP	0.929	0.929	0.962
SWAT	0.980	0.980	0.981
PSM	0.993	0.993	0.992

Table 7.11: **LFQ Anomaly Detection Ablations (Specialist).** F1 scores of the LFQ, LFQ-EC, and LFQ+BN variants on the SMD, MSL, SMAP, SWAT, and PSM datasets in the specialist in-domain setting.

Generalist In-Domain

Model	LFQ	LFQ-EC	LFQ+BN
SMD	0.946	0.946	0.947
MSL	0.946	0.963	0.974
SMAP	0.929	0.932	0.956
SWAT	0.980	0.981	0.981
PSM	0.993	0.993	0.993

Table 7.12: **LFQ Anomaly Detection Ablations (Generalist).** F1 scores of the LFQ, LFQ-EC, and LFQ+BN variants on the SMD, MSL, SMAP, SWAT, and PSM datasets in the generalist in-domain setting.

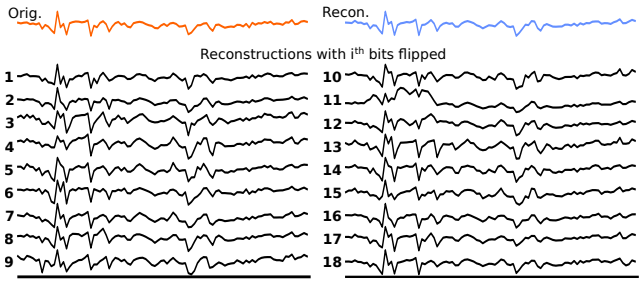


Figure 7.8: Hamming distance in token space is qualitatively correlated with similarity in the time domain. **Top.** A time series and its BOUQUET reconstruction. The series comprises a sequence of 24 tokens, each with 18 bits. **Bottom.** We flip the i^{th} bit for all 24 tokens and all 18 bits. The corrupted reconstructions appear very similar to the pristine one, suggesting BOUQUET preserves structured semantic representations.

7.6 Qualitative Analyses

We now examine BOUQUET's qualitative behavior for insight into its performance. We analyze token reconstruction quality, representative task outputs, the relationship between token distances and semantic similarity, and the evolution of token probabilities during training.

First, Fig. 7.2 demonstrates BOUQUET's ability to perform high-fidelity reconstruction while achieving significant compression. The example shows a 96-step time series compressed into 24 18-bit tokens (a temporal compression factor of 4). While some minor imperfections are visible, as expected with this level of compression, the reconstruction preserves the key features of the original signal. Figs. 7.9 and 7.10 show representative BOUQUET outputs on the two evaluation tasks, imputing masked segments and localizing labeled anomalies on held-out data.

We also conduct an analysis of token corruptions in Fig. 7.8. By flipping individual bits across all tokens in a sequence, we observe that small changes in the token space (1-bit Hamming distance) still correspond to semantically similar reconstructions in the time domain. For instance, flipping the second bit primarily affects the shape of the prominent peak on the left while maintaining the overall shape, suggesting that similar BOUQUET tokens represent similar temporal patterns.

Finally, Fig. 7.11 reveals differences between the token distributions in BOUQUET and BSQ during training. Both methods begin with bit probabilities centered around 0.5, but BOUQUET rapidly pushes probability mass toward 0 and 1, indicating growing confidence in its token assignments. In contrast, BSQ's probabilities remain clustered near 0.5 even after extended training, suggesting that it never

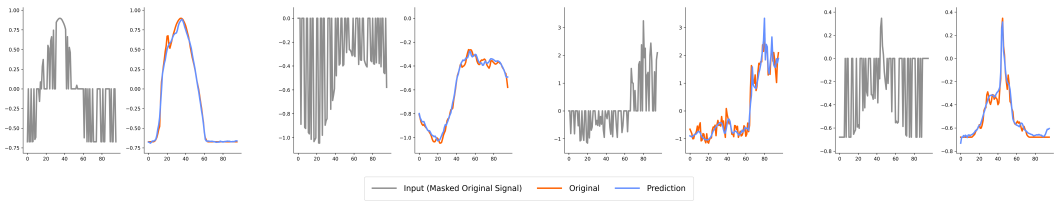


Figure 7.9: **Example BOUQUET Imputation Outputs.** Each pair of panels shows the corrupted input on the left in gray, where masked segments are filled with zeros, and on the right the original signal in orange overlaid with BOUQUET's reconstruction in blue, including its estimates of the missing values.

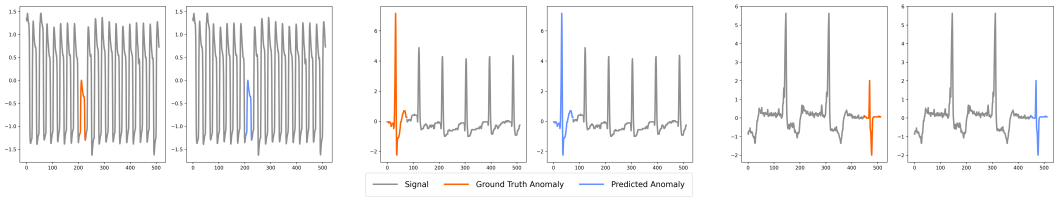


Figure 7.10: **Example BOUQUET Anomaly Detection Outputs.** Each pair of panels shows the same observed time series in gray, with the labeled ground-truth anomaly marked in orange on the left and the model's predicted anomaly marked in blue on the right. The predictions align closely with the labeled regions.

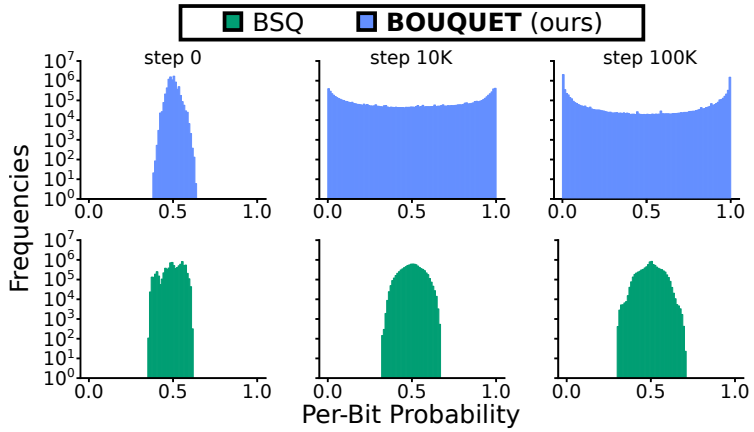


Figure 7.11: Comparison of per-bit probabilities. BSQ keeps most of its pre-quantized bits close to $p = 0.5$, while BOUQUET learns to be more confident as training proceeds, distributing far more mass at $p = 0.0$ and $p = 1.0$. Note that the y-axis is **log scale**.

develops strong preferences for specific tokens. We hypothesize that this ability to commit confidently to tokens while maintaining semantic smoothness in the latent space contributes to BOUQUET's superior performance across our evaluation tasks.

7.7 Conclusion

This chapter studied how to train massive tokenized representations for time series data. We build upon the lookup-free quantization (LFQ) paradigm to develop BOUQUET, a fully stochastic LFQ method that naturally encourages codebook exploration. With no codebook to store, BOUQUET trains vocabularies of hundreds of thousands to over a billion codes at essentially no additional time or memory cost over small-vocabulary baselines, and its reconstruction performance improves without signs of saturation with vocabulary size. BOUQUET also exhibits strong empirical performance, outperforming existing time series representations on most tasks. While our evaluations deliberately isolate the tokenizer through reconstruction-driven tasks, the resulting vocabularies are ultimately intended as inputs to downstream sequence models, in analogy with the text tokenizers used by language models, and we leave this integration to future work.

This chapter demonstrated that even without considering any dynamical structure, we can learn highly expressive tokenized representations for time series data that capture their underlying structure. However, the evaluation of these representations is still quite quantitatively-motivated, with a strong focus on properties like reconstruction performance. For higher-dimensional visual data, such as videos, we may in fact be interested not just in such reconstruction properties, but in the *semantic* properties of the learned representations. Moreover, video data are extremely challenging to acquire, store, and even perform efficient computations on, posing substantial challenges for scaling up any algorithm for extracting such representations. The next and final chapter seeks to address these challenges systematically.

7.8 Appendix: UCR15 Anomaly Detection Results

This section reports supplementary anomaly detection results deferred from Sec. 7.4.2. TOTEM reports specialist results on only 15 of the 42 UCR42 datasets [195], so a complete head-to-head comparison against TOTEM, the most closely related tokenizer to the lookup-free methods, is only possible on this subset, which we refer to as UCR15. Table 7.13 reports the per-dataset specialist results on UCR15. Table 7.14 reports the generalist results on the same subset so that both regimes are compared on identical data, and additionally includes zero-shot MOMENT.

Model	BOUQUET	BOUQUET$_{ENT}$	BSQ	LFQ	TOTEM	GPT2	TiNet	ATran	DGHL
CIMIS44AirTemperature3	96.00	88.90	98.00	11.90	73.80	18.00	47.00	6.00	50.00
GP711MarkerLFM5z4	84.9	54.0	**98.6**	55.9	96.7	48.0	90.0	76.0	31.0
InternalBleeding5	**100.0**	**100.0**	**100.0**	82.5	**100.0**	92.0	**100.0**	94.0	**100.0**
MesoplodonDensirostris	**100.0**	**100.0**	**100.0**	98.2	99.4	**100.0**	**100.0**	**100.0**	79.0
TKeepSecondMARS	**100.0**	**100.0**	**100.0**	22.5	**100.0**	12.0	95.0	83.0	16.0
WalkingAceleration5	**100.0**	**100.0**	**100.0**	92.2	**100.0**	87.0	93.0	99.0	91.0
insectEPG2	**100.0**	**100.0**	**100.0**	2.7	**100.0**	81.0	96.0	12.0	14.0
ltstdbs30791AS	**100.0**	**100.0**	**100.0**	39.8	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
park3m	34.4	41.2	68.6	13.0	67.2	63.0	**93.0**	15.0	20.0
s20101mML2	**100.0**	**100.0**	**100.0**	44.1	**100.0**	5.0	8.0	69.0	15.0
sddb49	99.6	99.8	99.6	27.9	99.8	94.0	**100.0**	89.0	88.0
sel840mECG1	**99.7**	**99.7**	99.1	80.2	99.5	21.0	36.0	16.0	28.0
sel840mECG2	81.8	**88.1**	68.6	83.9	86.8	28.0	21.0	15.0	32.0
tiltAPB2	99.3	92.6	**100.0**	98.3	68.5	83.0	38.0	92.0	36.0
tiltAPB3	**97.6**	57.7	29.0	15.8	23.4	5.0	9.0	17.0	3.0
`AvgWins`	**60.0%**	**60.0%**	**60.0%**	0%	40.0%	13.3%	33.3%	13.3%	13.3%
Avg. Best Adj. F1	**92.9**	88.1	90.8	52.9	87.7	81.7	46.9	55.8	68.4

Table 7.13: **UCR15 Anomaly Detection Specialist Results.** We compare BOUQUET and BOUQUET$_{ENT}$ against BSQ, LFQ, TOTEM, GPT2, TimesNet, Anomaly Transformer, and DGHL, and report the best adjusted F1 score on each dataset. UCR15 is the subset of UCR42 on which TOTEM reports specialist results, permitting a complete comparison against it.

Model	BOUQUET	BOUQUET$_{ENT}$	BSQ	LFQ	TOTEM	MNT-0	MNT-LP
CIMIS44AirTemperature3	32.4	12.7	96.0	11.8	9.8	**100.0**	98.0
GP711MarkerLFM5z4	99.5	**100.0**	99.5	55.5	27.5	69.0	97.0
InternalBleeding5	**100.0**	**100.0**	**100.0**	82.9	99.7	**100.0**	**100.0**
MesoplodonDensirostris	97.3	**100.0**	95.2	99.7	**100.0**	91.0	84.0
TKeepSecondMARS	**100.0**	**100.0**	**100.0**	22.0	31.3	95.0	**100.0**
WalkingAceleration5	**100.0**	72.4	99.2	88.1	81.9	**100.0**	**100.0**
insectEPG2	**89.3**	53.8	69.4	2.3	16.3	11.0	23.0
ltstdbs30791AS	**100.0**	99.5	**100.0**	49.8	**100.0**	**100.0**	**100.0**
park3m	57.9	46.8	56.8	7.1	23.3	56.0	**64.0**
s20101mML2	99.5	99.0	95.7	44.4	**100.0**	65.0	71.0
sddb49	96.2	98.2	97.3	27.8	**100.0**	**100.0**	**100.0**
sel840mECG1	**99.6**	99.3	98.3	80.4	**99.6**	61.0	66.0
sel840mECG2	**95.1**	81.1	89.3	84.3	88.1	36.0	39.0
tiltAPB2	**100.0**	99.6	99.1	98.3	99.9	96.0	98.0
tiltAPB3	19.0	20.9	21.5	15.9	56.3	48.0	**85.0**
`AvgWins`	**53.3%**	26.7%	20.0%	0%	33.3%	33.3%	46.7%

Table 7.14: **UCR15 Anomaly Detection Generalist Results.** We compare BOUQUET and BOUQUET$_{ENT}$ against BSQ, LFQ, TOTEM, and MOMENT (zero-shot MNT-0 and linearly probed MNT-LP), and report the best adjusted F1 score on each dataset. UCR15 is the subset of UCR42 on which TOTEM reports specialist results, used here so that both training regimes are compared on the same datasets.

Chapter 8

SIMMER: A SCALABLE PRETRAINING RECIPE FOR VIDEO-TEXT ENCODERS

[Intentionally Redacted]

BIBLIOGRAPHY

[1] Karl Johan Åström and Richard M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2008. ISBN: 9780691135762.

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 0521833787.

[3] Athanasios C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Advances in Design and Control. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2005. DOI: 10.1137/1.9780898718713.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN: 9780262035613.

[5] Tony Z. Zhao et al. "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware". In: *Robotics: Science and Systems (RSS)*. 2023.

[6] Cheng Chi et al. "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *The International Journal of Robotics Research* (2024).

[7] Anthony Brohan et al. "RT-1: Robotics Transformer for Real-World Control at Scale". In: *Robotics: Science and Systems (RSS)*. 2023.

[8] Anthony Brohan et al. "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control". In: *Conference on Robot Learning (CoRL)*. 2023.

[9] Moo Jin Kim et al. "OpenVLA: An Open-Source Vision-Language-Action Model". In: *Conference on Robot Learning (CoRL)*. 2024.

[10] Octo Model Team et al. "Octo: An Open-Source Generalist Robot Policy". In: *Robotics: Science and Systems (RSS)*. 2024.

[11] Kevin Black et al. "π_0: A Vision-Language-Action Flow Model for General Robot Control". In: *arXiv preprint arXiv:2410.24164* (2024).

[12] Open X-Embodiment Collaboration. "Open X-Embodiment: Robotic Learning Datasets and RT-X Models". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2024.

[13] Homer Walke et al. "BridgeData V2: A Dataset for Robot Learning at Scale". In: *Conference on Robot Learning (CoRL)*. 2023.

[14] Alexander Khazatsky et al. "DROID: A Large-Scale In-the-Wild Robot Manipulation Dataset". In: *Robotics: Science and Systems (RSS)*. 2024.

[15] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd. Nob Hill Publishing, 2017.

[16] Grady Williams et al. “Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving”. In: *IEEE Transactions on Robotics* 34.6 (2018), pp. 1603–1622. DOI: 10.1109/TRO.2018.2865891.

[17] Viktor Makoviychuk et al. *Isaac Gym: High Performance GPU-Based Physics Simulation for Robot Learning*. 2021. arXiv: 2108.10470 [cs.RO].

[18] Anusha Nagabandi et al. *Deep Dynamics Models for Learning Dexterous Manipulation*. 2019. arXiv: 1909.11652 [cs.RO].

[19] Ashish Kumar et al. *RMA: Rapid Motor Adaptation for Legged Robots*. 2021. arXiv: 2107.04034 [cs.LG].

[20] Xuezhi Wang et al. “Self-Consistency Improves Chain of Thought Reasoning in Language Models”. In: *International Conference on Learning Representations (ICLR)*. 2023.

[21] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[22] Michael Urbakh et al. “The nonlinear nature of friction”. In: *Nature* 430 (2004), pp. 525–528. DOI: 10.1038/nature02750.

[23] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[24] John Nicholas Newman. *Marine Hydrodynamics*. Cambridge, MA: MIT Press, 1977.

[25] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[26] Haixu Wu et al. “Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021.

[27] OpenAI et al. “Learning Dexterous In-Hand Manipulation”. In: *CoRR* abs/1808.00177 (2018). arXiv: 1808.00177.

[28] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (1960), pp. 35–45. DOI: 10.1115/1.3662552.

[29] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics*. 3rd. Addison-Wesley, 2002. ISBN: 0201657023.

[30] Jeong Joon Park et al. “DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 165–174.

[31] Ben Mildenhall et al. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *CoRR* abs/2003.08934 (2020). arXiv: 2003.08934. URL: https://arxiv.org/abs/2003.08934.

[32] C. Ferrari and J. Canny. “Planning optimal grasps”. In: *Proceedings 1992 IEEE International Conference on Robotics and Automation*. 1992, 2290–2295 vol.3. DOI: 10.1109/ROBOT.1992.219918.

[33] Hideo Hanafusa and Haruhiko Asada. “Stable Prehension by a Robot Hand with Elastic Fingers”. In: *Proceedings of the 7th International Symposium on Industrial Robots*. 1977, pp. 361–368.

[34] J. Kenneth Salisbury and John J. Craig. “Articulated Hands: Force Control and Kinematic Issues”. In: *The International Journal of Robotics Research* 1.1 (1982), pp. 4–17.

[35] A.T. Miller and P.K. Allen. “Graspit! A versatile simulator for robotic grasping”. In: *IEEE Robotics and Automation Magazine* 11.4 (2004), pp. 110–122. DOI: 10.1109/MRA.2004.1371616.

[36] D. G. Kirkpatrick, B. Mishra, and C. K. Yap. “Quantitative Steinitz’s Theorems with Applications to Multifingered Grasping”. In: *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*. STOC ’90. Association for Computing Machinery, 1990, pp. 341–351.

[37] Elon Rimon and Joel Burdick. *The Mechanics of Robot Grasping*. Cambridge University Press, 2019. DOI: 10.1017/9781108552011.

[38] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. “Leveraging big data for grasp planning”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4304–4311. DOI: 10.1109/ICRA.2015.7139793.

[39] Z. Li and S.S. Sastry. “Task-oriented optimal grasping by multifingered robot hands”. In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 32–44. DOI: 10.1109/56.769.

[40] Laura Downs et al. “Google Scanned Objects: A High-Quality Dataset of 3D Scanned Household Items”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 2553–2560.

[41] Berk Calli et al. “The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research”. In: *2015 International Conference on Advanced Robotics (ICAR)*. July 2015, pp. 510–517.

[42] Ravi Balasubramanian et al. “Human-guided grasp measures improve grasp robustness on physical robot”. In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 2294–2301. DOI: 10.1109/ROBOT.2010.5509855.

[43] Dylan Turpin et al. “Grasp’D: Differentiable Contact-Rich Grasp Synthesis for Multi-Fingered Hands”. In: *Computer Vision – ECCV 2022*. 2022, pp. 201–221.

[44] Jeffrey Mahler et al. “Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics”. In: *ArXiv* abs/1703.09312 (2017).

[45] Douglas Morrison, Peter Corke, and J. Leitner. “Closing the Loop for Robotic Grasping: A Real-time, Generative Grasp Synthesis Approach”. In: *ArXiv* abs/1804.05172 (2018).

[46] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. “ACRONYM: A Large-Scale Grasp Dataset Based on Simulation”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 6222–6227. DOI: 10.1109/ICRA48506.2021.9560844.

[47] Ruicheng Wang et al. “DexGraspNet: A Large-Scale Robotic Dexterous Grasp Dataset for General Objects Based on Simulation”. In: *ArXiv* abs/2210.02697 (2022).

[48] Rhys Newbury et al. *Deep Learning Approaches to Grasp Synthesis: A Review*. 2022. arXiv: 2207.02556 [cs.RO].

[49] Máximo A. Roa and Raúl Suárez. “Grasp quality measures: review and performance”. In: *Autonomous Robots* 38 (2014), pp. 65–88.

[50] Tengyu Liu et al. “Synthesizing Diverse and Physically Stable Grasps With Arbitrary Hand Structures Using Differentiable Force Closure Estimator”. In: *IEEE Robotics and Automation Letters* 7 (2021), pp. 470–477.

[51] Min Liu et al. “New Formulation of Mixed-Integer Conic Programming for Globally Optimal Grasp Planning”. In: *IEEE Robotics and Automation Letters* 5 (2019), pp. 4663–4670.

[52] Xiangyang Zhu and Jun Wang. “Synthesis of force-closure grasps on 3-D objects based on the Q distance”. In: *IEEE Trans. Robotics Autom.* 19 (2003), pp. 669–679.

[53] Hongkai Dai, Anirudha Majumdar, and Russ Tedrake. “Synthesis and Optimization of Force Closure Grasps via Sequential Semidefinite Programming”. In: *International Symposium of Robotics Research*. 2015.

[54] Min Liu et al. “Deep Differentiable Grasp Planner for High-DOF Grippers”. In: *ArXiv* abs/2002.01530 (2020).

[55] Albert Wu, Michelle Guo, and C. Karen Liu. “Learning Diverse and Physically Feasible Dexterous Grasps with Generative Model and Bilevel Optimization”. In: *ArXiv* abs/2207.00195 (2022).

[56] Shuo Liu and Stefano Carpin. “A fast algorithm for grasp quality evaluation using the object wrench space”. In: *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. 2015, pp. 558–563. DOI: 10.1109/CoASE.2015.7294138.

[57] Brandon Amos and J. Zico Kolter. "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *International Conference on Machine Learning*. 2017.

[58] Arne Brøndsted. *An Introduction to Convex Polytopes*. Graduate Texts in Mathematics. New York: Springer-Verlag, 1982. ISBN: 038790722X.

[59] A. Agrawal et al. "Differentiable Convex Optimization Layers". In: *Advances in Neural Information Processing Systems*. 2019.

[60] Daniel De Wolf and Yves Smeers. "Generalized derivatives of the optimal value of a linear program with respect to matrix coefficients". In: *European Journal of Operational Research* 291.2 (2021), pp. 491–496.

[61] Russ Tedrake and the Drake Development Team. *Drake: Model-based design and verification for robotics*. 2019. URL: https://drake.mit.edu.

[62] Khaled Mamou and Faouzi Ghorbel. "A simple and efficient approach for 3D mesh approximate convex decomposition". In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. 2009, pp. 3501–3504. DOI: 10.1109/ICIP.2009.5414068.

[63] Steven G. Johnson. *The NLopt nonlinear-optimization package*. 2011. URL: http://ab-initio.mit.edu/nlopt.

[64] Dieter Kraft. "A software package for sequential quadratic programming". In: *Forschungsbericht Deutsche Forschungs und Versuchsanstalt fur Luft und Raumfahrt* (1988).

[65] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. "Poisson Surface Reconstruction". In: *Symposium on Geometry Processing*. Ed. by Alla Sheffer and Konrad Polthier. The Eurographics Association, 2006. ISBN: 3-905673-24-X. DOI: 10.2312/SGP/SGP06/061-070.

[66] Jonathan Weisz and Peter K. Allen. "Pose error robust grasping from contact wrench space metrics". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 557–562. DOI: 10.1109/ICRA.2012.6224697.

[67] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. "Gaussian process implicit surfaces for shape estimation and grasping". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 2845–2850. DOI: 10.1109/ICRA.2011.5980395.

[68] Jeffrey Mahler et al. "GP-GPIS-OPT: Grasp planning with shape uncertainty using Gaussian process implicit surfaces and Sequential Convex Programming". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4919–4926. DOI: 10.1109/ICRA.2015.7139882.

[69] Cristiana de Farias et al. "Simultaneous Tactile Exploration and Grasp Refinement for Unknown Objects". In: *IEEE Robotics and Automation Letters* (2021). DOI: 10.1109/LRA.2021.3063074.

[70] Miao Li et al. “Dexterous grasping under shape uncertainty”. In: *Robotics and Autonomous Systems* 75 (2016), pp. 352–364. DOI: 10.1016/j.robot.2015.09.008.

[71] Qingkai Lu et al. “Planning Multi-Fingered Grasps as Probabilistic Inference in a Learned Deep Network”. In: *International Symposium of Robotics Research*. 2018.

[72] Qingkai Lu et al. “Multifingered Grasp Planning via Inference in Deep Neural Networks: Outperforming Sampling by Learning Differentiable Models”. In: *IEEE Robotics and Automation Magazine* 27.2 (2020), pp. 55–65. DOI: 10.1109/MRA.2020.2976322.

[73] Albert H. Li, Preston Culbertson, Joel W. Burdick, and Aaron D. Ames. “FRoGGeR: Fast Robust Grasp Generation via the Min-Weight Metric”. In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 6809–6816. DOI: 10.1109/IROS55552.2023.10341806.

[74] Naoki Hayashi, Kohei Segawa, and Shigemasa Takai. “2D Voronoi Coverage Control with Gaussian Density Functions by Line Integration”. In: *SICE Journal of Control, Measurement, and System Integration* 10.2 (2017), pp. 110–116. DOI: 10.9746/jcmsi.10.110.

[75] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.4.1. 2018. URL: http://github.com/google/jax.

[76] Pablo Gómez, Håvard Hem Toftevaag, and Gabriele Meoni. “torchquad: Numerical Integration in Arbitrary Dimensions with PyTorch”. In: *Journal of Open Source Software* 6.64 (2021), p. 3439. DOI: 10.21105/joss.03439.

[77] B. O’Neill. *Elementary Differential Geometry, Revised 2nd Edition*. Elsevier Science, 2006. ISBN: 9780120887354.

[78] Emanuel Todorov, Tom Erez, and Yuval Tassa. “Mujoco: A physics engine for model-based control”. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2012, pp. 5026–5033.

[79] Josh Tobin et al. “Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[80] L. Shao et al. “UniGrasp: Learning a Unified Model to Grasp With Multifingered Robotic Hands”. In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 2286–2293. DOI: 10.1109/LRA.2020.2969946.

[81] Sergey Levine et al. “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection”. In: *The International Journal of Robotics Research* 37.4-5 (2018), pp. 421–436. DOI: 10.1177/0278364917710318.

[82] Chaozheng Wu et al. *Grasp Proposal Networks: An End-to-End Solution for Visual Learning of Robotic Grasps*. 2020. arXiv: 2009.12606 [cs.CV].

[83] Shaochen Wang, Zhou Zhangli, and Zhen Kan. “When Transformer Meets Robotic Grasping: Exploits Context for Efficient Grasp Detection”. In: *IEEE Robotics and Automation Letters* 7 (2022), pp. 1–8. DOI: 10.1109/LRA.2022.3187261.

[84] Yinzhen Xu et al. *UniDexGrasp: Universal Robotic Dexterous Grasping via Learning Diverse Proposal Generation and Goal-Conditioned Policy*. 2023. arXiv: 2303.00938 [cs.RO].

[85] Dylan Turpin et al. *Grasp’D: Differentiable Contact-rich Grasp Synthesis for Multi-fingered Hands*. 2022. arXiv: 2208.12250 [cs.RO].

[86] Dylan Turpin et al. *Fast-Grasp’D: Dexterous Multi-finger Grasp Generation Through Differentiable Simulation*. 2023. arXiv: 2306.08132 [cs.RO].

[87] Maria Attarian et al. *Geometry Matching for Multi-Embodiment Grasping*. 2023. arXiv: 2312.03864 [cs.RO].

[88] Vincent Mayer et al. “FFHNet: Generating Multi-Fingered Robotic Grasps for Unknown Objects in Real-time”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 762–769. DOI: 10.1109/ICRA46639.2022.9811666.

[89] Zehang Weng et al. *DexDiffuser: Generating Dexterous Grasps with Diffusion Models*. 2024. arXiv: 2402.02989 [cs.RO].

[90] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. *LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning*. 2023. arXiv: 2309.06440 [cs.RO].

[91] Puhao Li et al. “GenDexGrasp: Generalizable Dexterous Grasping”. In: *arXiv preprint arXiv:2210.00722* (2022).

[92] C. Borst, M. Fischer, and G. Hirzinger. “Grasping the dice by dicing the grasp”. In: *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*. Vol. 4. 2003, pp. 3692–3697. DOI: 10.1109/IROS.2003.1249729.

[93] Matei Ciocarlie, Corey Goldfeder, and Peter Allen. “Dexterous grasping via eigengrasps: A low-dimensional approach to a high-complexity problem”. In: *Robotics: Science and Systems Manipulation Workshop*. 2007.

[94] Albert H. Li, Preston Culbertson, and Aaron D. Ames. "Toward An Analytic Theory of Intrinsic Robustness for Dexterous Grasping". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 2992–2999. DOI: 10.1109/IROS58592.2024.10802185.

[95] Jeannette Bohg et al. "Data-driven grasp synthesis—A survey". In: *IEEE Transactions on Robotics* 30.2 (2014), pp. 289–309. DOI: 10.1109/TRO.2013.2289018.

[96] Carlos Rubert et al. "Predicting grasp success in the real world—A study of quality metrics and human assessment". In: *Robotics and Autonomous Systems* 121 (2019), p. 103274. DOI: 10.1016/j.robot.2019.103274.

[97] S. El-Khoury and A. Sahbani. "Handling Objects by Their Handles". In: *IROS-2008 Workshop on Grasp and Task Learning by Imitation*. 2008.

[98] Sergey Prokudin, Christoph Lassner, and Javier Romero. *Efficient Learning on Point Clouds with Basis Point Sets*. 2019. arXiv: 1908.09186 [cs.CV].

[99] Jeffrey Ichnowski et al. "Dex-NeRF: Using a Neural Radiance Field to Grasp Transparent Objects". In: *Conference on Robot Learning (CoRL)*. 2020.

[100] Justin Kerr et al. "Evo-NeRF: Evolving NeRF for Sequential Robot Grasping of Transparent Objects". In: *6th Annual Conference on Robot Learning*. 2022.

[101] Qiyu Dai et al. "GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2023).

[102] Snehal Jauhri, Ishikaa Lunawat, and Georgia Chalvatzaki. *Learning Any-View 6DoF Robotic Grasping in Cluttered Scenes via Neural Surface Rendering*. 2024. arXiv: 2306.07392 [cs.RO].

[103] Adam Rashid et al. "Language Embedded Radiance Fields for Zero-Shot Task-Oriented Grasping". In: *7th Annual Conference on Robot Learning*. 2023.

[104] Lin Yen-Chen et al. "MIRA: Mental Imagery for Robotic Affordances". In: *Conference on Robot Learning (CoRL)*. 2022.

[105] William Shen et al. "Distilled Feature Fields Enable Few-Shot Language-Guided Manipulation". In: *7th Annual Conference on Robot Learning*. 2023.

[106] Martin Sundermeyer et al. "Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes". In: (2021).

[107] Luis Felipe Casas Murrilo et al. *MultiGripperGrasp: A Dataset for Robotic Grasping from Parallel Jaw Grippers to Dexterous Hands*. 2024. arXiv: 2403.09841 [cs.RO].

[108] Mark Van der Merwe et al. *Learning Continuous 3D Reconstructions for Geometrically Aware Grasping*. 2020. arXiv: 1910.00983 [cs.RO].

[109] Qingkai Lu et al. “Multifingered Grasp Planning via Inference in Deep Neural Networks”. In: *IEEE Robotics and Automation Magazine* 27.2 (2020), pp. 55–65. DOI: 10.1109/MRA.2020.2976322.

[110] J. H. Halton. “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals”. In: *Numerische Mathematik* 2.1 (1960), pp. 84–90. DOI: 10.1007/BF01386213.

[111] Thomas Müller et al. “Instant Neural Graphics Primitives with a Multiresolution Hash Encoding”. In: *ACM Trans. Graph.* 41.4 (2022), 102:1–102:15. DOI: 10.1145/3528223.3530127.

[112] Matthew Tancik et al. “Nerfstudio: A Modular Framework for Neural Radiance Field Development”. In: *ACM SIGGRAPH 2023 Conference Proceedings*. 2023.

[113] Frederik Warburg et al. “Nerfbusters: Removing Ghostly Artifacts from Casually Captured NeRFs”. In: *International Conference on Computer Vision (ICCV)*. 2023.

[114] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. “Open3D: A Modern Library for 3D Data Processing”. In: *arXiv:1801.09847* (2018).

[115] Stéfan van der Walt et al. “scikit-image: image processing in Python”. In: *PeerJ* 2 (2014), e453. DOI: 10.7717/peerj.453.

[116] Dawson-Haggerty et al. *trimesh*. Version 3.2.0. 2019. URL: https://trimsh.org/.

[117] Yi Zhou et al. “On the Continuity of Rotation Representations in Neural Networks”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[118] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising Diffusion Implicit Models”. In: *arXiv:2010.02502* (2020).

[119] Balakumar Sundaralingam et al. “CuRobo: Parallelized Collision-Free Robot Motion Generation”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 8112–8119. DOI: 10.1109/ICRA48891.2023.10160765.

[120] Ankur Handa et al. *DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality*. 2024. arXiv: 2210.13702 [cs.RO]. URL: https://arxiv.org/abs/2210.13702.

[121] Tao Chen et al. “Visual dexterity: In-hand reorientation of novel and complex object shapes”. In: *Science Robotics* 8.84 (Nov. 2023). ISSN: 2470-9476. DOI: 10.1126/scirobotics.adc9244. URL: http://dx.doi.org/10.1126/scirobotics.adc9244.

[122] Taylor Howell et al. *Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo*. 2022. arXiv: 2212.00541 [cs.RO]. URL: https://arxiv.org/abs/2212.00541.

[123] Albert Li. *DROP: Dexterous Reorientation via Online Planning*. https://caltech-amber.github.io/drop/. 2024.

[124] Zhao-Heng Yin et al. "Rotating without Seeing: Towards In-hand Dexterity through Touch". In: *Robotics: Science and Systems* (2023).

[125] Lennart Röstel et al. "Estimator-coupled reinforcement learning for robust purely tactile in-hand manipulation". In: *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. IEEE. 2023, pp. 1–8.

[126] Tao Chen, Jie Xu, and Pulkit Agrawal. *A System for General In-Hand Object Re-Orientation*. 2021. arXiv: 2111.03043 [cs.RO]. URL: https://arxiv.org/abs/2111.03043.

[127] Michael Posa, Cecilia Cantu, and Russ Tedrake. "A direct method for trajectory optimization of rigid bodies through contact". In: *The International Journal of Robotics Research* 33 (2014), pp. 69–81. URL: https://api.semanticscholar.org/CorpusID:6532910.

[128] Neel Doshi et al. "Contact-Implicit Optimization of Locomotion Trajectories for a Quadrupedal Microrobot". In: *Robotics: Science and Systems XIV*. RSS2018. Robotics: Science and Systems Foundation, June 2018. DOI: 10.15607/rss.2018.xiv.041. URL: http://dx.doi.org/10.15607/RSS.2018.XIV.041.

[129] Simon Le Cleac'h et al. *Fast Contact-Implicit Model-Predictive Control*. 2023. arXiv: 2107.05616 [cs.RO]. URL: https://arxiv.org/abs/2107.05616.

[130] Vince Kurtz et al. *Inverse Dynamics Trajectory Optimization for Contact-Implicit Model Predictive Control*. 2023. arXiv: 2309.01813 [cs.RO]. URL: https://arxiv.org/abs/2309.01813.

[131] Alp Aydinoglu and Michael Posa. *Real-Time Multi-Contact Model Predictive Control via ADMM*. 2022. arXiv: 2109.07076 [cs.RO]. URL: https://arxiv.org/abs/2109.07076.

[132] William Yang and Michael Posa. "Dynamic On-Palm Manipulation via Controlled Sliding". In: *Proceedings of Robotics: Science and Systems*. July 2024.

[133] Wen Yang and Wanxin Jin. *ContactSDF: Signed Distance Functions as Multi-Contact Models for Dexterous Manipulation*. 2024. arXiv: 2408.09612 [cs.RO]. URL: https://arxiv.org/abs/2408.09612.

[134] Claire Chen et al. “TrajectoTree: Trajectory Optimization Meets Tree Search for Planning Multi-contact Dexterous Manipulation”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 8262–8268. DOI: 10.1109/IROS51168.2021.9636346.

[135] Tao Pang et al. *Global Planning for Contact-Rich Manipulation via Local Smoothing of Quasi-dynamic Contact Models*. 2023. arXiv: 2206.10787 [cs.RO]. URL: https://arxiv.org/abs/2206.10787.

[136] Steven Lavalle and James Kuffner. “Rapidly-Exploring Random Trees: Progress and Prospects”. In: *Algorithmic and computational robotics: New directions* (Jan. 2000).

[137] L.E. Kavraki et al. “Probabilistic roadmaps for path planning in high-dimensional configuration spaces”. In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[138] Mrinal Kalakrishnan et al. “STOMP: Stochastic trajectory optimization for motion planning”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[139] Julius Jankowski et al. “VP-STO: Via-point-based Stochastic Trajectory Optimization for Reactive Robot Behavior”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 10125–10131. DOI: 10.1109/ICRA48891.2023.10160214.

[140] Corrado Pezzato et al. *Sampling-based Model Predictive Control Leveraging Parallelizable Physics Simulations*. 2023. arXiv: 2307.09105 [cs.RO].

[141] Mohak Bhardwaj et al. “STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation”. In: *5th Annual Conference on Robot Learning*. 2021.

[142] Juan Alvarez-Padilla et al. *Real-Time Whole-Body Control of Legged Robots with Model-Predictive Path Integral Control*. 2024. arXiv: 2409.10469 [cs.RO]. URL: https://arxiv.org/abs/2409.10469.

[143] Carl Doersch et al. *BootsTAP: Bootstrapped Training for Tracking-Any-Point*. 2024. arXiv: 2402.00847 [cs.CV]. URL: https://arxiv.org/abs/2402.00847.

[144] Nikita Karaev et al. “CoTracker: It is Better to Track Together”. In: 2023.

[145] Yuxi Xiao et al. “SpatialTracker: Tracking Any 2D Pixels in 3D Space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024.

[146] Bowen Wen et al. “FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects”. In: *CVPR*. 2024.

[147] Reuven Rubinstein. “The Cross-Entropy Method for Combinatorial and Continuous Optimization”. In: *Methodology and Computing in Applied Probability* 1.2 (1999), pp. 127–190.

[148] Jacob Sacks et al. *Deep Model Predictive Optimization*. 2023. arXiv: 2310.04590 [cs.RO]. URL: https://arxiv.org/abs/2310.04590.

[149] Klaus Greff et al. “Kubric: a scalable dataset generator”. In: (2022).

[150] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541. URL: https://github.com/borglab/gtsam.

[151] Jad Abou-Chakra et al. *Physically Embodied Gaussian Splatting: A Realtime Correctable World Model for Robotics*. 2024. arXiv: 2406.10788 [cs.RO]. URL: https://arxiv.org/abs/2406.10788.

[152] Weiwei Li and Emanuel Todorov. “ITERATIVE LINEAR QUADRATIC REGULATOR DESIGN FOR NONLINEAR BIOLOGICAL MOVEMENT SYSTEMS”. In: *Proceedings of the First International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,* INSTICC. SciTePress, 2004, pp. 222–229. ISBN: 972-8865-12-0. DOI: 10.5220/0001143902220229.

[153] H. J. Terry Suh et al. *Do Differentiable Simulators Give Better Policy Gradients?* 2022. arXiv: 2202.00817 [cs.LG]. URL: https://arxiv.org/abs/2202.00817.

[154] David Donoho. *Data Science at the Singularity*. 2023. arXiv: 2310.00865 [stat.OT]. URL: https://arxiv.org/abs/2310.00865.

[155] The NerfStudio Team. *Viser: A real-time web visualizer for 3D content*. https://github.com/nerfstudio-project/viser. 2023.

[156] Haixuan Xavier Tao and Philipp Oppermann. *DORA: Dataflow-Oriented Robotic Architecture*. https://github.com/dora-rs/dora. 2025.

[157] Vince Kurtz. *Hydrax: Sampling-based model predictive control on GPU with JAX and MuJoCo MJX*. https://github.com/vincekurtz/hydrax. 2024.

[158] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception*. Vol. 6. 1-2. Foundations and Trends in Robotics, 2017, pp. 1–139. DOI: 10.1561/2300000043.

[159] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2024.

[160] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. “Machine learning for fluid mechanics”. In: *Annual review of fluid mechanics* 52.1 (2020), pp. 477–508.

[161] Steven L. Brunton et al. “Modern Koopman Theory for Dynamical Systems”. In: *SIAM Review* 64.2 (2022), pp. 229–340. DOI: 10.1137/21M1401243. eprint: https://doi.org/10.1137/21M1401243. URL: https://doi.org/10.1137/21M1401243.

[162] Samuel E. Otto and Clarence W. Rowley. *Linearly-Recurrent Autoencoder Networks for Learning Dynamics*. 2019. arXiv: 1712.01378 [math.DS]. URL: https://arxiv.org/abs/1712.01378.

[163] Lu Shi et al. *Koopman Operators in Robot Learning*. 2025. arXiv: 2408.04200 [cs.RO]. URL: https://arxiv.org/abs/2408.04200.

[164] Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. “Deep learning for universal linear embeddings of nonlinear dynamics”. In: *Nature Communications* 9.1 (Nov. 2018). ISSN: 2041-1723. DOI: 10.1038/s41467-018-07210-0. URL: http://dx.doi.org/10.1038/s41467-018-07210-0.

[165] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. “Discovering governing equations from data by sparse identification of nonlinear dynamical systems”. In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937. DOI: 10.1073/pnas.1517384113. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1517384113. URL: https://www.pnas.org/doi/abs/10.1073/pnas.1517384113.

[166] Yuhong Jin, Lei Hou, and Shun Zhong. “Extended Dynamic Mode Decomposition with Invertible Dictionary Learning”. In: *Neural Networks* 173 (2024), p. 106177. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2024.106177. URL: https://www.sciencedirect.com/science/article/pii/S0893608024001011.

[167] Enoch Yeung, Soumya Kundu, and Nathan Hodas. “Learning Deep Neural Network Representations for Koopman Operators of Nonlinear Dynamical Systems”. In: *2019 American Control Conference (ACC)*. 2019, pp. 4832–4839. DOI: 10.23919/ACC.2019.8815339.

[168] Clarence W. Rowley et al. “Spectral analysis of nonlinear flows”. In: *Journal of Fluid Mechanics* 641 (2009), pp. 115–127. DOI: 10.1017/S0022112009992059.

[169] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. “A data–driven approximation of the koopman operator: Extending dynamic mode decomposition”. In: *Journal of Nonlinear Science* 25.6 (2015), pp. 1307–1346.

[170] Steven L. Brunton et al. “Chaos as an intermittently forced linear system”. In: *Nature Communications* 8.1 (May 2017). ISSN: 2041-1723. DOI: 10.1038/s41467-017-00030-8. URL: http://dx.doi.org/10.1038/s41467-017-00030-8.

[171] Yuhuang Meng, Jianguo Huang, and Yue Qiu. "Koopman operator learning using invertible neural networks". In: *Journal of Computational Physics* 501 (Mar. 2024), p. 112795. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2024.112795. URL: http://dx.doi.org/10.1016/j.jcp.2024.112795.

[172] Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. *Learning Koopman Invariant Subspaces for Dynamic Mode Decomposition*. 2018. arXiv: 1710.04340 [cs.LG]. URL: https://arxiv.org/abs/1710.04340.

[173] Jack W. Miller et al. *Eigenvalue initialisation and regularisation for Koopman autoencoders*. 2022. arXiv: 2212.12086 [cs.LG]. URL: https://arxiv.org/abs/2212.12086.

[174] Indranil Nayak et al. "Temporally-consistent koopman autoencoders for forecasting dynamical systems". In: *Scientific Reports* 15.1 (July 2025). ISSN: 2045-2322. DOI: 10.1038/s41598-025-05222-7. URL: http://dx.doi.org/10.1038/s41598-025-05222-7.

[175] Marco Fraccaro et al. *A Disentangled Recognition and Nonlinear Dynamics Model for Unsupervised Learning*. 2017. arXiv: 1710.05741 [stat.ML]. URL: https://arxiv.org/abs/1710.05741.

[176] Albert H. Li, Philipp Wu, and Monroe Kennedy. "Replay Overshooting: Learning Stochastic Latent Dynamics with the Extended Kalman Filter". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 852–858. DOI: 10.1109/ICRA48506.2021.9560811.

[177] Tuomas Haarnoja et al. *Backprop KF: Learning Discriminative Deterministic State Estimators*. 2017. arXiv: 1605.07148 [cs.LG]. URL: https://arxiv.org/abs/1605.07148.

[178] Michelle A. Lee et al. *Multimodal Sensor Fusion with Differentiable Filters*. 2020. arXiv: 2010.13021 [cs.RO]. URL: https://arxiv.org/abs/2010.13021.

[179] Alina Kloss, Georg Martius, and Jeannette Bohg. "How to train your differentiable filter". In: *Autonomous Robots* 45.4 (May 2021), pp. 561–578. ISSN: 1573-7527. DOI: 10.1007/s10514-021-09990-9. URL: http://dx.doi.org/10.1007/s10514-021-09990-9.

[180] Adrien Corenflos et al. *Differentiable Particle Filtering via Entropy-Regularized Optimal Transport*. 2021. arXiv: 2102.07850 [stat.ML]. URL: https://arxiv.org/abs/2102.07850.

[181] Amit Surana and Andrzej Banaszuk. "Linear observer synthesis for nonlinear systems using Koopman Operator framework". In: *IFAC-PapersOnLine* 49.18 (2016). 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016, pp. 716–723. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.10.250. URL: https://www.sciencedirect.com/science/article/pii/S2405896316318304.

[182] Lingyun Zeng, S.M.Hadi Sadati, and Christos Bergeles. "Koopman Operator-based Extended Kalman Filter for Cosserat Rod Wrench Estimation". In: *2023 International Symposium on Medical Robotics (ISMR)*. 2023, pp. 1–7. DOI: 10.1109/ISMR57123.2023.10130210.

[183] Peng Huang, Ketong Zheng, and Gerhard Fettweis. "Data-Driven Koopman Operator-Based Error-State Kalman Filter for Enhanced State Estimation of Quadrotors in Agile Flight". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2024, pp. 10983–10989. DOI: 10.1109/IROS58592.2024.10802457.

[184] Ningxin Liu et al. *Estimate of Koopman modes and eigenvalues with Kalman Filter*. 2024. arXiv: 2410.02815 [eess.SY]. URL: https://arxiv.org/abs/2410.02815.

[185] Zhexuan Zeng et al. *Koopman Spectral Analysis from Noisy Measurements based on Bayesian Learning and Kalman Smoothing*. 2025. arXiv: 2410.00703 [eess.SY]. URL: https://arxiv.org/abs/2410.00703.

[186] Guan Xiaoqing et al. "An Online System Identification Algorithm for Spherical Robot Using the Koopman Theory". In: *IEEE Robotics and Automation Letters* PP (Jan. 2025), pp. 1–8. DOI: 10.1109/LRA.2025.3552997.

[187] Zi Cong Guo et al. *Koopman Linearization for Data-Driven Batch State Estimation of Control-Affine Systems*. 2021. arXiv: 2109.07000 [cs.RO]. URL: https://arxiv.org/abs/2109.07000.

[188] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (Mar. 2020), p. 132306. ISSN: 0167-2789. DOI: 10.1016/j.physd.2019.132306. URL: http://dx.doi.org/10.1016/j.physd.2019.132306.

[189] Nima Mohajerin and Steven L. Waslander. *Multi-Step Prediction of Dynamic Systems with Recurrent Neural Networks*. 2018. arXiv: 1806.00526 [cs.NE]. URL: https://arxiv.org/abs/1806.00526.

[190] Abbas Tariverdi et al. "A Recurrent Neural-Network-Based Real-Time Dynamic Model for Soft Continuum Manipulators". In: *Frontiers in Robotics and AI* 8 (Mar. 2021), p. 631303. DOI: 10.3389/frobt.2021.631303.

[191] Jianjing Zhang et al. "Recurrent neural network for motion trajectory prediction in human-robot collaborative assembly". In: *CIRP Annals* 69.1 (2020), pp. 9–12. ISSN: 0007-8506. DOI: https://doi.org/10.1016/j.cirp.2020.04.077. URL: https://www.sciencedirect.com/science/article/pii/S0007850620300998.

[192] Danijar Hafner et al. *Learning Latent Dynamics for Planning from Pixels*. 2019. arXiv: 1811.04551 [cs.LG]. URL: https://arxiv.org/abs/1811.04551.

[193] Yuqi Nie et al. *A Time Series is Worth 64 Words: Long-term Forecasting with Transformers*. 2023. arXiv: 2211.14730 [cs.LG]. URL: https://arxiv.org/abs/2211.14730.

[194] Tian Zhou et al. *One Fits All:Power General Time Series Analysis by Pretrained LM*. 2023. arXiv: 2302.11939 [cs.LG]. URL: https://arxiv.org/abs/2302.11939.

[195] Sabera Talukder, Yisong Yue, and Georgia Gkioxari. *TOTEM: TOkenized Time Series EMbeddings for General Time Series Analysis*. 2025. arXiv: 2402.16412 [cs.LG]. URL: https://arxiv.org/abs/2402.16412.

[196] Simo Särkkä. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, 2013.

[197] Klaus F. Hasselmann et al. "Measurements of wind-wave growth and swell decay during the Joint North Sea Wave Project (JONSWAP)". In: 1973. URL: https://api.semanticscholar.org/CorpusID:122820060.

[198] Matthieu Ancellin and Frédéric Dias. "Capytaine: a Python-based linear potential flow solver". In: *Journal of Open Source Software* 4.36 (Apr. 2019), p. 1341. DOI: 10.21105/joss.01341. URL: https://doi.org/10.21105%2Fjoss.01341.

[199] Xingjian Wu et al. *K^2VAE: A Koopman-Kalman Enhanced Variational AutoEncoder for Probabilistic Time Series Forecasting*. 2025. arXiv: 2505.23017 [cs.LG]. URL: https://arxiv.org/abs/2505.23017.

[200] Anqi Bao et al. "Data-Driven Model Reduction for Coupled Flow and Geomechanics Based on DMD Methods". In: *Fluids* 4.3 (2019). ISSN: 2311-5521. DOI: 10.3390/fluids4030138. URL: https://www.mdpi.com/2311-5521/4/3/138.

[201] Qiuqi Li, Chang Liu, and Yifei Yang. *Localized Dynamic Mode Decomposition with Temporally Adaptive Partitioning*. 2025. arXiv: 2503.13093 [math.NA]. URL: https://arxiv.org/abs/2503.13093.

[202] Albert H. Li, John Zhang, Jan Brüdigam, Brandon Hung, Aaron D. Ames, Jiuguang Wang, Simon Le Cleac'h, and Preston Culbertson. "Judo: A User-Friendly Open-Source Package for Sampling-Based Model Predictive Control". In: *2026 IEEE International Conference on Robotics and Automation (ICRA)*. 2026.

[203] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV]. URL: https://arxiv.org/abs/2010.11929.

[204] Abdul Fatir Ansari et al. "Chronos: Learning the language of time series". In: *arXiv preprint arXiv:2403.07815* (2024).

[205] Luca Masserano et al. “Enhancing Foundation Models for Time Series Forecasting via Wavelet-based Tokenization”. In: *arXiv preprint arXiv:2412.05244* (2024).

[206] Aaron Van Den Oord, Oriol Vinyals, et al. “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30 (2017).

[207] Lijun Yu et al. “Language Model Beats Diffusion–Tokenizer is Key to Visual Generation”. In: *arXiv preprint arXiv:2310.05737* (2023).

[208] Philip Gage. “A new algorithm for data compression”. In: *C Users Journal* 12.2 (1994), pp. 23–38.

[209] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).

[210] Alec Radford et al. “Improving language understanding by generative pre-training”. In: (2018).

[211] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[212] T Kudo. “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing”. In: *arXiv preprint arXiv:1808.06226* (2018).

[213] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971.

[214] Mark Weber et al. *MaskBit: Embedding-free Image Generation via Bit Tokens*. 2024. arXiv: 2409.16211 [cs.CV]. URL: https://arxiv.org/abs/2409.16211.

[215] Hubert Siuzdak. “Vocos: Closing the gap between time-domain and Fourier-based neural vocoders for high-quality audio synthesis”. In: *arXiv preprint arXiv:2306.00814* (2023).

[216] Nate Gruver et al. “Large language models are zero-shot time series forecasters”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[217] Ming Jin et al. “Time-llm: Time series forecasting by reprogramming large language models”. In: *arXiv preprint arXiv:2310.01728* (2023).

[218] Hao Xue and Flora D Salim. “Promptcast: A new prompt-based learning paradigm for time series forecasting”. In: *IEEE Transactions on Knowledge and Data Engineering* (2023).

[219] Alexei Baevski et al. “wav2vec 2.0: A framework for self-supervised learning of speech representations”. In: *Advances in neural information processing systems* 33 (2020), pp. 12449–12460.

[220] Jessica Lin et al. “Experiencing SAX: a novel symbolic representation of time series”. In: *Data Mining and knowledge discovery* 15 (2007), pp. 107–144.

[221] Yong Liu et al. “itransformer: Inverted transformers are effective for time series forecasting”. In: *arXiv preprint arXiv:2310.06625* (2023).

[222] Shiyang Li et al. “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting”. In: *Advances in neural information processing systems* 32 (2019).

[223] Minyoung Huh et al. *Straightening Out the Straight-Through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks*. 2023. arXiv: 2305.08842 [cs.LG]. URL: https://arxiv.org/abs/2305.08842.

[224] Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. *Image and Video Tokenization with Binary Spherical Quantization*. 2024. arXiv: 2406.07548 [cs.CV]. URL: https://arxiv.org/abs/2406.07548.

[225] Haixu Wu et al. “Timesnet: Temporal 2d-variation modeling for general time series analysis”. In: *arXiv preprint arXiv:2210.02186* (2022).

[226] Taesung Kim et al. “Reversible instance normalization for accurate time-series forecasting against distribution shift”. In: *International Conference on Learning Representations*. 2021.

[227] Mononito Goswami et al. “Moment: A family of open time-series foundation models”. In: *arXiv preprint arXiv:2402.03885* (2024).

[228] Shanghua Gao et al. “UniTS: A unified multi-task time series model”. In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024.

[229] Xiangkai Ma et al. “UTSD: Unified Time Series Diffusion Model”. In: *arXiv preprint arXiv:2412.03068* (2024).

[230] Gerald Woo et al. “Etsformer: Exponential smoothing transformers for time-series forecasting”. In: *arXiv preprint arXiv:2202.01381* (2022).

[231] Tian Zhou et al. “Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting”. In: *International Conference on Machine Learning*. PMLR. 2022, pp. 27268–27286.

[232] Yong Liu et al. “Non-stationary transformers: Exploring the stationarity in time series forecasting”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 9881–9893.

[233] Haoyi Zhou et al. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12. 2021, pp. 11106–11115.

[234] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. “Reformer: The efficient transformer”. In: *arXiv preprint arXiv:2001.04451* (2020).

[235] Tianping Zhang et al. “Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures”. In: *arXiv preprint arXiv:2207.01186* (2022).

[236] Ailing Zeng et al. “Are transformers effective for time series forecasting?” In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 37. 9. 2023, pp. 11121–11128.

[237] Jiehui Xu et al. “Anomaly transformer: Time series anomaly detection with association discrepancy”. In: *arXiv preprint arXiv:2110.02642* (2021).

[238] Cristian I Challu et al. “Deep generative model with hierarchical latent factors for time series anomaly detection”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 1643–1654.

[239] Shizhan Liu et al. “Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting”. In: 2022.