Title

Enabling Efficient and Reliable Robot Manipulation Through Optimization, Interactive Perception and Self-Supervised Learning

Author

Avigal, Yahav

Publication Date

2024

Peer reviewed|Thesis/dissertation

Enabling Efficient and Reliable Robot Manipulation Through Optimization, Interactive Perception and Self-Supervised Learning

By

Yahav Avigal

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Jitendra Malik
Professor Angjoo Kanazawa
Professor Dan Halperin

Spring 2024

Enabling Efficient and Reliable Robot Manipulation Through Optimization, Interactive Perception and Self-Supervised Learning

Abstract

Enabling Efficient and Reliable Robot Manipulation Through Optimization, Interactive Perception and Self-Supervised Learning

by

Yahav Avigal

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Science

University of California, Berkeley

Professor Ken Goldberg, Chair

Robot manipulation research is essential for advancing automation technologies, allowing robotic arms to execute complex and precise tasks across various industries. In recent years, several technologies have matured and transformed industries, such as logistics and manufacturing automation, through advancements like robot grasping. To enable the adoption of these emerging capabilities into real-world automation scenarios, these technologies need to be both efficient and reliable. However, achieving a balance between efficiency and reliability is challenging, as improving one often requires compromising the other. As a result, many impressive methods, initially developed for practical applications, struggle to make the transition into industry use.

This thesis explores five research areas within the field of robot manipulation, examining diverse angles and domains such as industrial automation, deformable manipulation, agricultural robotics, and surgical robotics. It proposes strategies to achieve efficient and reliable robot manipulation. By focusing on enhancing both efficiency and reliability, the thesis aims to facilitate the transition of robot manipulation from proof of concept to practical applications. Among the various methods explored, three primary strategies have proven to be particularly effective: constrained optimization, which given a reliable model, applies strict mathematical constraints to find efficient and reliable solutions; interactive perception and self-supervised learning, which are used to improve the efficiency and reliability in situations where there is high uncertainty in the dynamics of the system or a lack a reliable model. The thesis concludes by discussing the key insights gained through this research, reviewing lessons learned, and suggesting potential directions for future research.

To all of my teachers.

Contents

Contents **ii**

1 Introduction **1**
1.1 Efficiency and Reliability in Motion Planning for Object Transport 2
1.2 Efficiency and Reliability in Grasping Partially Observable Rigid Objects . . 2
1.3 Efficiency and Reliability in Bimanual Deformable Manipulation 3
1.4 Efficient and Reliable Autonomous Gardening 3
1.5 Efficient and Reliable Autonomous Suturing 3
1.6 Thesis Contributions 4

I Efficiency and Reliability in Motion Planning for Object Transport **5**

2 DJ-GOMP: Warmstarting Motion Planning Optimization Using Deep Learning **6**
2.1 Introduction 6
2.2 Related Work 8
2.3 Problem Statement 9
2.4 Method 10
2.4.1 Jerk- and Time-optimized Trajectory Generation 10
2.4.2 Deep Learning of Trajectories 15
2.4.3 Fast Pipeline for Trajectory Generation 18
2.5 Experiments 19
2.6 Discussion and Future Work 23

3 GOMP-FIT: Making Fast Object Transport Reliable **25**
3.1 Introduction 25
3.2 Related Work 27
3.3 Problem Statement 29
3.4 Method 29
3.4.1 GOMP Background 29

3.4.2 GOMP-FIT End-Effector Acceleration Constraints . . . 31
3.4.3 Minimization Objective . . . 32
3.5 Experiments . . . 33
3.5.1 Open-Top Container Transport . . . 35
3.5.2 Fragile Object Transport . . . 36
3.5.3 Filled Wineglass Transport . . . 37
3.6 Discussion and Future Work . . . 37

4 GOMP-ST: Making Fast Object Transport with Suction Grasping Reliable 38
4.1 Introduction . . . 38
4.2 Related Work . . . 40
4.3 Problem Statement . . . 42
4.4 Method . . . 42
4.4.1 Background: GOMP-FIT . . . 43
4.4.2 Learned constraints in the SQP . . . 44
4.4.3 Self-supervised data collection and training . . . 45
4.4.4 Analytic model of suction-cup failure for GOMP-FIT baseline . . . 46
4.5 Experiments . . . 48
4.5.1 Ablation studies . . . 50
4.5.2 Results . . . 50
4.6 Discussion and Future Work . . . 51

5 BOMP: Optimized Motion Planning for Bin Picking 53
5.1 Introduction . . . 53
5.2 Related Work . . . 55
5.3 Problem Statement . . . 57
5.4 Method . . . 57
5.4.1 Grasped Box Shape Estimation . . . 57
5.4.2 Suction Grasp Selection . . . 58
5.4.3 Optimization Formulation . . . 59
5.4.4 Collision Checking . . . 60
5.4.5 Deep Learning Warm-start . . . 61
5.4.6 Speeding Up Computation . . . 62
5.5 Experiments . . . 63
5.5.1 Simulated Experiments . . . 64
5.5.2 Physical Experiments . . . 66
5.6 Discussion and Future Work . . . 67

II Efficiency and Reliability in Grasping Partially Observable Rigid Objects **68**

6 AVPLUG: Approach Vector Planning in Cluttered Environments **69**
6.1 Introduction . . . 69
6.2 Related Work . . . 71
6.3 Problem Statement . . . 72
6.3.1 Definitions . . . 73
6.4 Method . . . 73
6.4.1 Updating the Octree . . . 75
6.4.2 Finding Candidate Target Object Locations . . . 75
6.4.3 Finding Candidate Vectors . . . 76
6.4.4 Evaluating Candidate Vectors . . . 76
6.4.5 Finding and Evaluating Visible Grasps . . . 76
6.5 Experiments . . . 77
6.5.1 Simulation Experiments . . . 77
6.5.2 Environments in Simulation . . . 77
6.5.3 GridSearch Baseline . . . 78
6.5.4 Simulation Results . . . 79
6.5.5 Physical Experiments . . . 80
6.5.6 Visibility vs Graspability . . . 82
6.5.7 Failure Cases . . . 82
6.6 Discussion and Future Work . . . 82

7 Grasping Transparent Objects Reliably **84**
7.1 Introduction . . . 84
7.2 Related Work . . . 85
7.3 Problem Statement . . . 86
7.4 Method . . . 87
7.4.1 Preliminary: Training NeRF . . . 87
7.4.2 Recovering Geometry of Transparent Objects . . . 87
7.4.3 Rendering Depth for Grasp Analysis . . . 88
7.4.4 Improving Reconstruction with Light Placement . . . 89
7.5 Experiments . . . 89
7.5.1 Datasets . . . 90
7.5.2 Synthetic Grasping Experiments . . . 90
7.5.3 Physical Grasping Experiments . . . 91
7.5.4 Comparison to RealSense Depth . . . 92
7.5.5 One vs Many Lights . . . 92
7.5.6 Workcell Setup . . . 94
7.6 Discussion and Future Work . . . 94

8 Grasping Transparent Objects Efficiently **96**
8.1 Introduction . . . 96
8.2 Related Work . . . 97
8.3 Problem Statement . . . 99
8.4 Method . . . 100
8.4.1 Evo-NeRF . . . 100
8.4.2 Grasp Planning Network . . . 102
8.5 Experiments . . . 103
8.5.1 Physical Setup . . . 104
8.5.2 Rapid single object retrieval . . . 104
8.5.3 Sequential decluttering . . . 105
8.5.4 Graspability ablation . . . 106
8.5.5 NeRF Depth vs Ground Truth Depth . . . 106
8.6 Discussion and Future Work . . . 107
8.6.1 Limitations and future work . . . 107

III Efficiency and Reliability in Bimanual Deformable Manipulation **108**

9 Using Interactive Perception to Untangle Long Cables **109**
9.1 Introduction . . . 109
9.2 Related Work . . . 111
9.3 Problem Statement . . . 112
9.3.1 Workspace Definition and Assumptions . . . 112
9.3.2 Task Objective and Metrics . . . 112
9.4 Method . . . 114
9.4.1 Approach Overview . . . 114
9.4.2 Uncertainty-Aware Perception Systems . . . 114
9.4.3 Novel Manipulation Primitives for Interactive Perception . . . 116
9.4.4 Sliding and Grasping for Tangle Manipulation 2.0 (SGTM 2.0) Algorithm 118
9.5 Experiments . . . 118
9.5.1 Experimental Setup . . . 118
9.5.2 Results . . . 120
9.5.3 Failure Modes . . . 121
9.5.4 Ablations . . . 121
9.6 Discussion and Future Work . . . 122

10 Learning Efficient Bimanual Folding of Garments **123**
10.1 Introduction . . . 123
10.2 Related Work . . . 125
10.3 Problem Statement . . . 126

10.4 Method . . . 127
10.4.1 Action Primitives . . . 127
10.4.2 BiMaMa-Net for Bimanual Manipulation . . . 128
10.4.3 Reachability Calibration . . . 130
10.4.4 Training for Smoothing . . . 131
10.4.5 Folding Pipeline . . . 132
10.5 Experiments . . . 134
10.5.1 Experimental Setup . . . 134
10.5.2 Sufficiently Smoothed . . . 135
10.5.3 Folds per Hour . . . 136
10.5.4 Generalization to Unseen Garments . . . 136
10.5.5 System Limitations . . . 136
10.6 Discussion and Future Work . . . 137

IV Efficient and Reliable Autonomous Gardening 138

11 AlhphaGarden 139
11.1 Introduction . . . 139
11.2 Related Work . . . 140
11.3 AlphaGardenSim . . . 142
11.4 Modeling . . . 142
11.4.1 Plant Growth Model . . . 142
11.4.2 Life Cycle . . . 145
11.4.3 Water Stress . . . 147
11.4.4 Irrigation . . . 148
11.4.5 Diversity . . . 148
11.5 Experiments . . . 149
11.5.1 Experimental Setup . . . 149
11.5.2 Policies . . . 151
11.5.3 Evaluation . . . 151
11.6 Discussion and Future Work . . . 153

12 Learning Efficient Policies for Polyculture Farming with Optimized Seed Placements 155
12.1 Introduction . . . 155
12.2 Related Work . . . 157
12.3 Plant Phenotyping . . . 157
12.4 Irrigation Model . . . 160
12.5 Growth Analysis . . . 161
12.6 Companion Planting . . . 162
12.7 Pruning and Irrigation Policies . . . 164

12.8 Simulation Experiments . . . 166
12.9 Discussion and Future Work . . . 169

V Efficient and Reliable Autonomous Suturing 170

13 Automating 2D Suture Placement 171
13.1 Introduction . . . 171
13.2 Related Work . . . 172
13.2.1 Needle Path Planning . . . 172
13.2.2 The Suture Planning Problem . . . 173
13.3 Problem Statement . . . 174
13.4 Method . . . 176
13.4.1 Input . . . 176
13.4.2 Optimization . . . 176
13.4.3 Suture regularity constraints and objectives . . . 177
13.4.4 Generalizing the Diamond Force Model . . . 179
13.4.5 Closure and shear force objectives . . . 181
13.4.6 Force closure objective . . . 181
13.4.7 Parameter settings . . . 182
13.4.8 Adjustment . . . 182
13.5 Experiments . . . 182
13.5.1 Synthetic Splines . . . 182
13.5.2 Physical Experiments on Chicken Skin . . . 184
13.6 Discussion and Future Work . . . 184
13.6.1 Limitations . . . 184
13.6.2 Future work . . . 186

14 Autonomous Suture Tail-Shortening 187
14.1 Introduction . . . 187
14.2 Related Work . . . 189
14.3 Problem Statement . . . 190
14.4 Method . . . 191
14.4.1 Module 1: Learned 2D Surgical Thread Detection . . . 191
14.4.2 Module 2: 2D Surgical Thread Tracing . . . 192
14.4.3 Module 3: 3D Surgical Thread Tracing . . . 193
14.4.4 Module 4: 3D Surgical Thread Tracking . . . 194
14.4.5 Module 5: Surgical Suture Tail-Shortening . . . 194
14.5 Experiments . . . 196
14.5.1 Modules 1-3: 2D Thread Detection and 2D & 3D Tracing . . . 196
14.5.2 Module 4: 3D Surgical Thread Tracking . . . 197
14.5.3 Module 5: Surgical Suture Tail-Shortening . . . 198

14.6 Discussion and Future Work 199

VI Conclusion **200**

15 Concluding Remarks **201**
15.1 Lessons Learned 202
15.1.1 Physical Experiments 202
15.1.2 Rejected Papers 202
15.2 Opportunities for Future Work 203
15.2.1 Reactive Motion Planning Around Moving Obstacles 203
15.2.2 Grasping Transparent Objects in Real-Time 203
15.2.3 Suture Planning in Real-Time 203

Bibliography **205**

Acknowledgments

My journey into robotics and artificial intelligence began 10 years ago and continues to this day. It was sparked by a book from Ray Kurzweil, which opened my eyes to the endless possibilities within the field. I never imagined that this fascination would lead to pursuing a PhD in Electrical Engineering and Computer Science at UC Berkeley. I am grateful to Ray Kurzweil for his inspiring work and to Itay Hay for gifting me this transformative book on my birthday.

I owe a great deal of thanks to Tel Aviv University for laying my foundational knowledge in engineering and computer science. Special gratitude goes to Professor Eyal Zisser for your invaluable guidance, advice, and the friendly discussions we shared. I am also thankful to Professor Dan Halperin for introducing me to robotics and specifically to motion planning. Your trust in me to lead a student team and develop a drone project provided a significant learning opportunity, and your introduction to Professor Goldberg was instrumental in shaping my path.

I would like to thank my advisor, Professor Ken Goldberg. First, thank you for your mentorship. Your belief in my potential and your supportive approach allowed me to experiment, learn from my mistakes, and grow academically and personally. I am thankful for the knowledge you imparted on science and research, as well as on art and innovation, and for showing me how well they work together. I am especially grateful for your hospitality, as you welcomed me and my family into your home for Shabbat dinners and various events. Your genuine care for our well-being and your commitment to supporting my long-term ambitions have been invaluable.

I am immensely appreciative of UC Berkeley and the EECS department for broadening my academic horizons. The enthusiastic and supportive environment fostered by the professors, GSIs, and classmates was pivotal in my educational journey, continually motivating me to enhance my research and expand my perspectives.

Teamwork has always been my preference, and I must acknowledge all my collaborators for their crucial roles in my research. To the AlphaGarden team, thank you for countless hours of planting, irrigating, pruning, paper writing, and then starting the process all over again. To the GOMP team, thank you for consistently tackling the most challenging experiments, whether it involved extensive hours collecting beads from the lab's floor between runs or pushing the air compressor to its mechanical limits. I would like to thank Professor Jeff Ichnowski for showing me the fun and exciting parts of scientific research and for being by my side in the trenches, especially when things did not unfold as expected.

I thank my many remaining co-authors and collaborators, including Vishal Satish, Yi Liu, Max Cao, Zach Tam, Karthik Dharmarajan, Ethan Qiu, Raven Huang, Harry Zhang, Mike Danielczuk, Justin Kerr, Max Fu, Matt Tancik, Angjoo Kanazawa, Kaushik Shivakumar, Vainavi Viswanath, Anrui Gu, Lars Berscheid, Jensen Gao, William Wong, Kevin Li, Grady Pierroz, Fang Shuo Deng, Mark Theis, Mark Presten, Anna Deza, Sebastian Oehme, Jackson Chui, Paul Shao, Atsunobu Kotani, Satvik Sharma, Rishi Parikh, Sandeep Mukherjee, Varun Kamat, Viraj Ramakrishnan, Yashish Mohnot, Harshika Jalan, Julia Isaac, Vincent Schorp,

Will Panitch and Aviv Adler. Thank you for all of the insightful discussions, the long hour running physical experiments at the lab, filming robot demos, and crafting creative "figure-one"s. Lastly, I would like to thank all of my friends at AUTOLab and BAIR.

Lastly, I want to express my deepest gratitude to the most important people in my life – my family. I am who I am today thanks to you.

Chapter 1

Introduction

Robot manipulation, defined as the process whereby "an agent moves things other than itself through selective contact" [1], remains a significant area of research within the field of robotics. The transition of robotic manipulation research into the automation that is adopted in commercial sectors is notably slower. This discrepancy results from the much higher demands of industry for efficiency and reliability.

The difference between automation and robotics is insightfully encapsulated in a statement by Professor Raja Chatila, former President of the IEEE Robotics and Automation Society and current President of the IEEE Global Society on Ethics of Autonomous and Intelligent Systems: "One robot on Mars is robotics, ten robots on Mars is automation." This contrast raises an important question regarding the methods by which we can bridge the gap between robotics, which focuses on proof-of-concept, and automation, which emphasizes cost-effectiveness and reliability in operational contexts.

This thesis investigates five challenging research topics within the field of robot manipulation, with a particular focus on enhancing both efficiency and reliability as a means to facilitate the transition from a proof of concept to industrial applications. In Part I, we will use nonconvex constrained optimization to speed up the execution time and model distillation to speed up the computation of robot motions for object transport, and incorporate an additional constraint learned through self-supervised learning to increase the system's reliability. In Part II, we will use active perception and optimization to improve the reliability of perception systems for motion planning and grasping of transparent or partially occluded objects. In Part III, we will use interactive perception and self-supervised learning for efficient and reliable manipulation of deformable objects with a bimanual robot. In Part IV, we will present an efficient simulator for polyculture farming. We will then use constrained optimization, and leverage the simulator's computational efficiency to develop optimized seed placements, as well as efficient and reliable irrigation and pruning policies. In Part V, we will use nonconvex constrained optimization, interactive perception and self-supervised learning to develop reliable autonomous suturing procedures.

A central theme throughout this work is the delicate balance between efficiency and reliability; Striving for increased efficiency often entails a compromise on system reliability,

and vice versa.

The case studies explored in this dissertation all strive to increase both efficiency and reliability. It is my hope that insights from these case studies can be applied to future projects. Three primary strategies have emerged as particularly effective: constrained optimization, interactive and active perception, and self-supervised learning.

1.1 Efficiency and Reliability in Motion Planning for Object Transport

In tasks like pick-and-place, where actions are performed repetitively, even a marginal improvement can lead to a significant productivity increase. Chapters 2–5 propose methods to optimize motion computation and execution time using advances in motion planning. Motion planning aims to generate paths from a start configuration to a goal configuration while avoiding obstacles. Sampling-based planners [2–4] can find sub-optimal solutions quickly, while optimization-based planners can find optimized solutions [5–8] but are often slower in computation. In Chapter 2, we build on Ichnowski et al. [9], using a nonconvex sequential quadratic program to compute time-optimized trajectories, and model distillation to speed up the calculation by 300×. In some cases, e.g., open-top containers or suction grasping, accelerating too fast can lead to increased content spill, product damage or drops. In Chapters 3 and 4, we formulate additional constraints to increase the system's robustness while optimizing the trajectory and maintaining a balance between efficiency and reliability.

1.2 Efficiency and Reliability in Grasping Partially Observable Rigid Objects

While optimized motion planning on its own can lead to a substantial increase in productivity, it assumes full knowledge of the location of obstacles, information typically acquired using RGBD sensors. However, these sensors face challenges in accurately perceiving objects under certain conditions, for example, occlusions. Leading grasp planners often struggle to identify effective grasps on partially occluded objects [10]. Chapter 6 proposes a method to increase the reliability of grasping using active perception, by executing a series of movements with a wrist-mounted camera to find a collision-free approach towards a viable grasp around occlusions. Another limitation of RGBD sensors is their unreliability in perceiving transparent and translucent objects. Chapters 7 and 8 propose using optimization, leveraging the ability of Neural Radiance Fields (NeRFs) to represent non-Lambertian effects, such as specularities and reflections, while reconstructing the geometry of the scene for grasping transparent objects [11].

1.3 Efficiency and Reliability in Bimanual Deformable Manipulation

Even given efficient motion planning and robust perception, manipulation of deformable objects in 1D [12–14] and 2D [15–17] remains a challenge, due to their large configuration space and complex dynamics. A large body of research exists on single-arm deformable manipulation [16, 18]. However, A dual-arm system extends the workspace, allows for increased payload and for more complex behaviors than a single-arm system [19–22]. Chapters 9 and 10 propose to use dual-arm systems to increase the efficiency and reliability of 1D (cables) and 2D (cloth) deformable manipulation. As accurately simulating the dynamics of deformable objects remains a challenging problem, these chapters propose methods for developing reliable deformable manipulation policies using a physical robot. Chapter 9 uses interactive perception for untangling long cables, given high uncertainty in the state of the cable. Chapter 10 uses self-supervised learning to learn a sequence of efficient and reliable motion primitives for garment smoothing and folding.

1.4 Efficient and Reliable Autonomous Gardening

Unlike cable untangling and garment folding, it is difficult to develop an autonomous farming policy for planting, irrigating, and pruning crops exclusively on a physical robot due to nature's long time-constants. A single experiment could take months or even years. Chapter 11 introduces a fast, first-order simulator that incorporates parameterized individual plant growth models and inter-plant dynamics to simulate competition over resources between plants in close proximity. Yet, we found that the policies developed with the simulator were unstable, stemming from a balance between preserving plant diversity and boosting yield, along with a reliance on initial seed placement. Chapter 12 proposes a method to optimize the seed placement to increase the policy's stability and reliability, and leverage the simulator's computational efficiency to learn a look-ahead policy that increases the policy's performance.

1.5 Efficient and Reliable Autonomous Suturing

In contrast to the repetitive tasks described in Part I, in suturing, we don't repeat motions over and over and execution time is not the main objective. The quality of the suture is important and depends on various factors, such as the suture placement and the forces applied to the wound in the process. Existing works place evenly spaced sutures on simple wound shapes, e.g., straight lines or circular wounds [23]. However, when dealing with more complex wound shapes, planning and evaluating an effective set of sutures may require a more detailed model of how the sutures hold the wound together. Chapter 13 proposes to combine surgical heuristics with a model of the forces imparted by the sutures to ensure that the entire

wound is held sufficiently closed by the sutures. This optimization problem is nonconvex, and similarly to GOMP [9] and Chapters 2– 5, we solve it using a sequential quadratic program. Once the suture placement has been planned, the robot repeats suturing motions, including needle insertion and tail-shortening. The latter is particularly challenging due to the thin and flexible nature of suturing thread, as well as its tendency for self-intersections and partial occlusions. Similarly to SGTM 2.0 for long cable untangling in Chapter 9, Chapter 14 uses interactive perception to increase the reliability of the thread manipulation.

1.6 Thesis Contributions

The primary contributions of this thesis are:

- We frame the objectives of increasing the efficiency and reliability of manipulation systems as constrained optimization problems, as shown in Chapters 2 to 5, 7, 12 and 13.
- We propose a procedure for warm starting an optimizing motion planner with an approximation from a deep neural network to reduce the planning time and improve the efficiency as described in Chapter 2.
- We formalize, tune, and learn acceleration constraints and 3D reconstruction objectives to improve the success rate and the reliability of motion planning and grasping as described in Chapters 3, 4, and 8. We develop active and interactive perception primitives to increase the system's reliability when it is difficult to estimate its state, as presented in Chapters 6, 9 and 14.
- We use self-supervised learning to develop an efficient and reliable system when it is difficult to model its dynamics, as described in Chapters 4, 10, and 14.
- We introduce an efficient polyculture farming simulator that integrates parameterized models of plant growth, as well as plant interactions, and train a policy to optimize plant coverage and diversity over a short horizon, as presented in Chapters 11 and 12.

Part I

Efficiency and Reliability in Motion Planning for Object Transport

Chapter 2

DJ-GOMP: Warmstarting Motion Planning Optimization Using Deep Learning

In some tasks, e.g., pick and place, we repeat motions again and again, and even a small improvement can lead to a significant productivity increase. GOMP [9] leverages a degree of freedom (DOF) in the grasp pose to speed up the motion execution time, but the computation is slow. To increase the efficiency, DJ-GOMP uses a neural network to predict an initialization trajectory to warmstart the optimization, speeding up calculation time by up to 300×.

2.1 Introduction

In e-commerce warehouses, faster pick-and-place pipelines can address labor shortages, lower costs, and increase revenue. However, despite advances in grasp planning (e.g., Mahler et al. [24]), the planning and executing of robot motion remains a bottleneck. To address this, Ichnowski et al. introduced a grasp-optimized motion planner (GOMP) [9] that computes a time-optimal motion plan subject to joint velocity and acceleration limits and allows for degrees of freedom in the pick and place frames (see Fig. 2.1). The motions that GOMP produces are fast and smooth; however, by not taking into account the motion's jerk (change in acceleration), the robot arm will often rapidly accelerate at the beginning of each motion and rapidly decelerate at the end. In the context of continuous pick-and-place operations in a warehouse, these high-jerk motions could result in wear on the robot's motors and reduce the overall service life of a robot. In this chapter, we introduce jerk limits and find that the resulting sequential quadratic program (SQP) and its underlying quadratic program (QP) require computation on the order of tens of seconds which is not practical for speeding up the overall pick-and-place pipeline. We then present DJ-GOMP which uses a deep neural network to learn trajectories that warm start computation, yielding a reduction

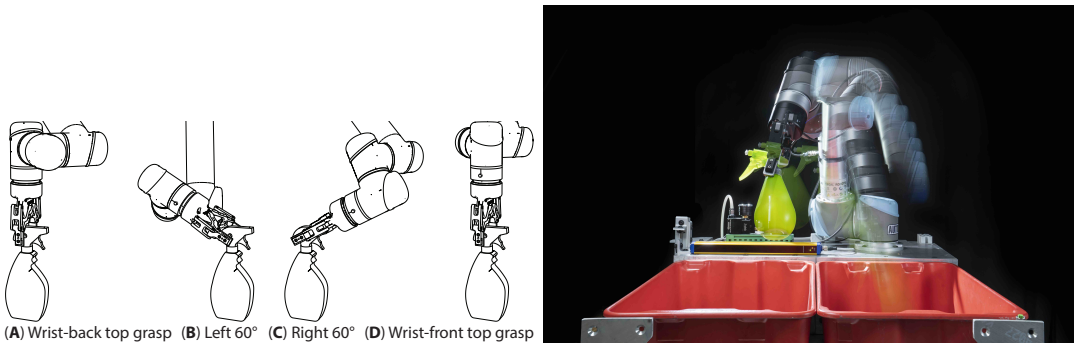


Figure 2.1: **Grasp-optimized motion planning degrees of freedom.** The optimized motion planning allows for degrees of freedom to be added to the pick and or place frames. **Left:** In (A), grasp analysis produces a top-down grasp. Since the analysis is based on two contact points, the motion planner allows for rotation about the grasp contact points shown as rotations in (B) and (C). Similarly, reversing the contact points, visible in (D) as a different arm pose, will still be valid according to grasp analysis. DJ-GOMP computes an optimal rotation for pick and place frames that minimizes time and jerk of the motion. **Right:** Using the additional degree(s) of freedom on the grasp, the motion planner then uses a neural network to warm start an optimization process to quickly compute a time-optimized jerk-limited motion.

in computation times from 29 s to 80 ms, making it practical for industrial use.

For a given workcell environment, DJ-GOMP speeds up motion planning for a robot and repeated task through a three-phase process. The first phase randomly samples tasks from the distribution of tasks the robot is likely to encounter and generates a time-optimal jerk-minimized motion plan using an SQP. The second phase trains a deep neural network using the data from the first phase to compute time-optimized motion plans for a given task specification (Fig. 2.2). The third phase, used in pick-and-place, uses the deep network from the second phase to generate a motion plan to warm start the SQP from the first phase. By warm starting the SQP from the deep network's output, DJ-GOMP ensures that the motion plan meets the constraints of the robot (something the network cannot guarantee), and greatly accelerates the convergence rate of the SQP (something the SQP cannot do without a good initial approximation).

This chapter describes algorithms and training process of DJ-GOMP. In the results section, we perform experiments on a physical UR5 robot, verifying that the trajectories GOMP generates are executable on a physical robot and result in fast and smooth motion. This chapter provides the following contributions: (1) J-GOMP, an extension of GOMP that computes time-optimized jerk-limited motions for pick-and-place operations; (2) DJ-GOMP, an extension of J-GOMP that uses deep-learning of time-optimized motion plans that empirically speeds up the computation time of the J-GOMP optimization by 2 orders of magnitude (300x); (3) Comparison to optimally-time-parameterized PRM* and TrajOpt motion planners in compute and motion time suggesting that DJ-GOMP computes fast motions quickly;

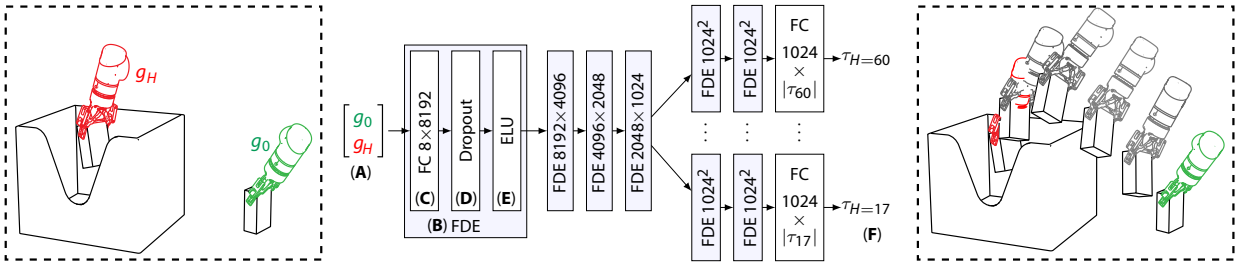


Figure 2.2: **A deep neural network architecture for grasp optimized motion planning** The input is the start and goal grasp frames (A). Each "FDE" block (B) sequences a fully-connected (FC) layer (C), a dropout layer (D), and an exponential linear unit (ELU) layer (E). The output (F) is a trajectory τ_H from the start frame to the goal frame for each of the time steps H supported by the network. A separate network uses one-hot encoding to predict which of the output trajectories is the shortest valid trajectory.

and (4) Experiments in simulation and on a physical UR5 robot suggesting that DJ-GOMP can be practical for reducing jerk to acceptable limits.

2.2 Related Work

Optimization-based motion planners such as CHOMP [7], STOMP [6] and TrajOpt [5] compute motion plans by locally optimizing a trajectory while penalizing collisions or placing barrier functions on collisions [25]. GOMP [9] builds on prior formulations, by taking the mechanical limits of the robot arm and the dynamics between waypoints into consideration, and allowing for a degree of freedom on pick and placement frames. DJ-GOMP further extends by minimizing jerk to avoid joint wear, while significantly reducing computation time. Model distillation, i.e., one model being trained on the output of one or more different models, is a widely used technique to transfer knowledge between models. In many cases, training an ensemble of models improves prediction performance, but is computationally expensive as it significantly increases the required computation resources. The ensemble can be distilled in a compact network [26, 27]. In RL (reinforcement learning), model distillation, also known as *policy distillation*, is used to transfer knowledge from multiple policies into a single multi-task policy [28], and to teach a robot how to solve multiple tasks [29, 30]. In this chapter we use model distillation to improve GOMP's running time. While the repeated optimization executed in GOMP takes up to several minutes, a forward pass in a compact neural network is often executed in milliseconds. We exploit this feature of neural networks to compute similar robot's trajectories faster.

In motion planning and control problems, warm starting the optimization solver with a near-optimal solution can significantly increase the solver's performance while greatly reducing the number of iterations required to reach sufficient optimality [31]. In RL, learning a new task can be warm started by transferring features from old tasks the agent has already

mastered [32]. Memory of motion [33] is another method that use an offline learned policy to warm start a control solver, and was shown to reduce the computation time in locomotion problems, and to increase the performance of nonlinear predictive controllers [34]. With the observation that the bottleneck in our algorithm is the number of iterations required to find the optimal horizon, we use the neural network's output to warm-start J-GOMP with a approximation of the optimal trajectory which results with a faster convergence.

Learning-based methods for motion planning have gained increasing attention in recent years. Motion planning algorithms can require complex cost functions, and learning-based methods, such as learning from demonstrations (e.g., [35–37]) can reduce the amount of hand engineering required. They reduce the hand engineering by leveraging additional knowledge provided by a human demonstrator, often times without the need to explicitly specify the task. A challenge sometimes tackled by learning-based methods is increasing the sampling efficiency in sample-based methods using non-uniform sampling [38], or reinforcement learning [39] for example. Although the "pick-and-place" operation from multiple start to multiple goal configurations (e.g., by allowing the degree of freedom around the grasp axis) is well-suited for sampling-based motion planners, as mentioned in [9], and despite the advances made through learning-based methods, the slow convergence rate of these planners in high dimensions prevents them from producing the required level of improvement in picks-per-hour. Quershi et al. [40] encode a point cloud of the obstacles into a latent space and uses a feed-forward neural network to predict the robot configuration at the next time step given an initial state, goal state, and the obstacles encoding. In doing so, they attempt to solve high-dimensional problems which were previously intractable. In this chapter, we leverage the advantage of having an accurate demonstrator in the form of J-GOMP to generate a large training data set.

2.3 Problem Statement

Let $\mathbf{q} \in \mathcal{C}$ be the complete specification of a robot's n degrees of freedom (e.g., n joint angles of a manipulator arm), where $\mathcal{C} \subseteq \mathbb{R}^n$ is the set of all possible configurations. Let $\mathbf{q}_{\min} \in \mathbb{R}^n$ and $\mathbf{q}_{\max} \in \mathbb{R}^n$ be the (possibly unbounded) lower and upper limits of the robot's configuration. Thus, $\mathbf{q} \in \mathcal{C}$ implies $\mathbf{q}_{\min} \leqslant \mathbf{q} \leqslant \mathbf{q}_{\max}$. Let $\mathcal{C}_{\text{obs}} \subset \mathcal{C}$ be the set of configurations that are in collision with an obstacle, and $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$ be the set of valid configurations.

Let $\mathbf{g}_0 \in SE(3)$ be a grasp frame or *pick* frame produced by grasp analysis, and $\mathbf{g}_H \in SE(3)$ be the goal or *place* frame. Adding degrees of freedom in rotation or translation to these frames, e.g., corresponding to a rotation about the axis defined by parallel-jaw grasp contact points (Fig. 2.1), rotation about the contact normal of a suction-based grasp, or in flexibility of translation, we define a set of starting grasp frames:

$$G_0 = \{\mathbf{g}_i | \mathbf{g}_i = R_{\mathbf{a}}(\theta) g_0 + \mathbf{t}, \theta \in [\theta_{\min}, \theta_{\max}], \mathbf{t} \in [\mathbf{t}_{\min}, \mathbf{t}_{\max}]\},$$

where $R_{\mathbf{a}}(\cdot)$ is a rotation about axis $\mathbf{a}$, $\theta_{\min}$ and $\theta_{\max}$ bound the angle of rotation, and $\mathbf{t}_{\min} \in \mathbb{R}^3$ and $\mathbf{t}_{\max} \in \mathbb{R}^3$ bound the translation degree of freedom. The definition of G_H, the

set of place frames, follows a similar definition, though with potentially different degrees of freedom.

Let $\mathbf{v}_{\max} \in \mathbb{R}^n_+$, $\mathbf{a}_{\max} \in \mathbb{R}^n_+$, and $\mathbf{j}_{\max} \in \mathbb{R}^n_+$, be the velocity, acceleration, and jerk limits of each degree of freedom. Let τ is a continuous sequence of robot configurations composed of $\tau_{\mathbf{q}}(t)$, $\tau_{\mathbf{v}}(t)$, $\tau_{\mathbf{a}}(t)$, and $\tau_{\mathbf{j}}(t)$, the configuration, velocity, acceleration, and jerk at time t respectively. Let $h(\tau)$ is the duration of the trajectory. The objective of DJ-GOMP is to compute:

$$
\begin{aligned}
\underset{\tau}{\text{argmin}} \quad & h(\tau) \\
\text{s.t.} \quad & \tau_{\mathbf{q}}(t) \in [\mathbf{q}_{\min}, \mathbf{q}_{\max}] \cup \mathcal{C}_{\text{free}} && \forall t \in [0, h(\tau)] \\
& \tau_{\mathbf{v}}(t) \in [-\mathbf{v}_{\max}, \mathbf{v}_{\max}] && \forall t \in [0, h(\tau)] \\
& \tau_{\mathbf{a}}(t) \in [-\mathbf{a}_{\max}, \mathbf{a}_{\max}] && \forall t \in [0, h(\tau)] \\
& \tau_{\mathbf{j}}(t) \in [-\mathbf{j}_{\max}, \mathbf{j}_{\max}] && \forall t \in [0, h(\tau)] \\
& p(\tau_{\mathbf{q}}(0)) \in G_0 \\
& p(\tau_{\mathbf{q}}(h(\tau))) \in G_H,
\end{aligned}
$$

where $p : \mathcal{C} \to SE(3)$ is the robot's forward kinematic function to gripper frame. Additionally, should multiple trajectories satisfy the above minimization, DJ-GOMP computes a trajectory that minimizes sum-of-squared jerks over time.

2.4 Method

This section describes the methods in DJ-GOMP. Underlying DJ-GOMP is a jerk- and time-optimizing constrained motion planner based on a sequential quadratic program (SQP). Due to the complexity of solving this SQP, computation time can far exceed the trajectory's execution time. DJ-GOMP uses this SQP on a random set of pick-and-place inputs to generate training data (trajectories) to train a neural network. During pick-and-place operation, DJ-GOMP uses the neural network to compute an approximate trajectory for the given pick and place frames, which it then uses to warm start the SQP.

2.4.1 Jerk- and Time-optimized Trajectory Generation

To generate a jerk- and time-optimized trajectory, DJ-GOMP extends the SQP formulated in GOMP [9]. The solver for this SQP, following the method in TrajOpt [5] and summarized in Alg. 1, starts with a discretized estimate of the trajectory τ as a sequence of H waypoints after the starting configuration, in which each waypoint represents the robot's configuration $\mathbf{q}$, velocity $\mathbf{v}$, acceleration $\mathbf{a}$, and jerk $\mathbf{j}$ at a moment in time. The waypoints are sequentially separated by t_{step} seconds. This discretization is collected into $\mathbf{x}^{(0)}$, where the superscript

represents a refinement iteration. Thus,

$$\mathbf{x}^{(0)} = \left(\mathbf{x}_0^{(0)}, \mathbf{x}_1^{(0)}, \ldots, \mathbf{x}_H^{(0)}\right), \qquad \text{where} \qquad \mathbf{x}_t^{(k)} = \begin{bmatrix} \mathbf{q}_t^{(k)} \\ \mathbf{v}_t^{(k)} \\ \mathbf{a}_t^{(k)} \\ \mathbf{j}_t^{(k)} \end{bmatrix}.$$

The choice of H and t_{step} is application specific, though in physical experiments, we set t_{step} to match (an integer multiple of) the control frequency of the robot, and we set H such that $H \cdot t_{\text{step}}$ is an estimate of the upper bound of the minimum trajectory time for the workspace and task input distribution.

The initial value of $\mathbf{x}^{(0)}$ seeds (or warm starts) the SQP computation. Without the approximation generated by the neural network (e.g., for training data set generation), this trajectory can be initialized to all zeros. In practice, the SQP can converge faster by first computing a trajectory between inverse-kinematic solutions to $\mathbf{g}_0$ and $\mathbf{g}_H$ with only the convex kinematic and dynamic constraints (described below).

In each iteration $k = (0, 1, 2, \ldots, m)$ of the SQP, DJ-GOMP linearizes the non-convex constraints of obstacles and pick and place locations, and solves a quadratic program of the following form:

$$\mathbf{x}^{(k+1)} = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{P}\mathbf{x} + \mathbf{p}^\mathsf{T}\mathbf{x} \tag{2.4.1}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} \leqslant \mathbf{b}, \tag{2.4.2}$$

where $\mathbf{A}$ defines constraints enforcing the trust region, joint limits, and dynamics, and where $\mathbf{P}$ is defined such that $\mathbf{x}^\mathsf{T}\mathbf{P}\mathbf{x}$ is a sum-of-squared jerks. To enforce the linearized non-convex constraints, DJ-GOMP adds constrained non-negative slack variables penalized using appropriate coefficients in $\mathbf{p}$. As DJ-GOMP iterates over the SQP, it increases the penalty term exponentially, terminating on the iteration m at which $\mathbf{x}^{(m)}$ meets the non-convex constraints.

To enforce joint limits and dynamic constraints, Alg. 1 creates a matrix $\mathbf{A}$ and vector $\mathbf{b}$ that enforce the following linear inequalities on joint limits:

$$\begin{aligned} \mathbf{q}_{\min} &\leqslant \mathbf{q}_t \leqslant \mathbf{q}_{\max} \\ -\mathbf{v}_{\max} &\leqslant \mathbf{v}_t \leqslant \mathbf{v}_{\max} \\ -\mathbf{a}_{\max} &\leqslant \mathbf{a}_t \leqslant \mathbf{a}_{\max} \\ -\mathbf{j}_{\max} &\leqslant \mathbf{j}_t \leqslant \mathbf{j}_{\max}, \end{aligned}$$

and the following equalities that enforce dynamic constraints between variables:

$$\begin{aligned} \mathbf{q}_{t+1} &= \mathbf{q}_t + t_{\text{step}}\mathbf{v}_t + \frac{1}{2}t_{\text{step}}^2\mathbf{a}_t + \frac{1}{6}t_{\text{step}}^3\mathbf{j}_t \\ \mathbf{v}_{t+1} &= \mathbf{v}_t + t_{\text{step}}\mathbf{a}_t + \frac{1}{2}t_{\text{step}}^2\mathbf{j}_t \\ \mathbf{a}_{t+1} &= \mathbf{a}_t + t_{\text{step}}\mathbf{j}_t. \end{aligned}$$

Algorithm 1 Jerk-limited Motion Plan

Require: $\mathbf{x}^{(0)}$ is an initial guess of the trajectory, $h+1$ is the number of waypoints in $\mathbf{x}^{(0)}$, t_{step} is the time between each waypoint, $\mathbf{g}_0$ and $\mathbf{g}_H$ are the pick and place frames, $\beta_{\text{shrink}} \in (0,1)$, $\beta_{\text{grow}} > 1$, and $\gamma > 1$

$\mu \leftarrow$ initial penalty multiple
$\epsilon_{\text{trust}} \leftarrow$ initial trust region size
$k \leftarrow 0$
$\mathbf{P}, \mathbf{p}, \mathbf{A}, \mathbf{b} \leftarrow$ linearize $\mathbf{x}^{(0)}$ as a QP
while $\mu < \mu_{\max}$ **do**
  $\mathbf{x}^{(k+1)} \leftarrow \text{argmin}_{\mathbf{x}} \frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{P}\mathbf{x} + \mathbf{p}^{\mathsf{T}}\mathbf{x}$ s.t. $\mathbf{A}\mathbf{x} \leqslant \mathbf{b}$ {warm start with $\mathbf{x}^{(k)}$}
  if sufficient decrease in trajectory cost **then**
    $k \leftarrow k+1$ {accept trajectory}
    $\epsilon_{\text{trust}} \leftarrow \epsilon_{\text{trust}}\beta_{\text{grow}}$ {grow trust region}
    $\mathbf{A}, \mathbf{b} \leftarrow$ update linearization using $\mathbf{x}^{(k)}$
  else
    $\epsilon_{\text{trust}} \leftarrow \epsilon_{\text{trust}}\beta_{\text{shrink}}$ {shrink trust region}
    $\mathbf{b} \leftarrow$ update trust region bounds only
  if $\epsilon_{\text{trust}} < \epsilon_{\text{min\_trust}}$ **then**
    $\mu \leftarrow \gamma\mu$ {increase penalty}
    $\epsilon_{\text{trust}} \leftarrow$ initial trust region size
    $\mathbf{p} \leftarrow$ update penalty in QP to match μ
return $\mathbf{x}^{(k)}$

Additionally, Alg. 1 linearizes non-convex constraints by adding slack variables to implement L_1 penalties. Thus, for a non-convex constraint $g_j(\mathbf{x}) \leqslant c$, the algorithm adds the linearization $\overline{g}_j(\mathbf{x})$ as a constraint in the form:

$$\overline{g}_j(\mathbf{x}) - \mu y_j^+ + \mu y_j^- \leqslant c,$$

where μ is the penalty and the slack variables are constrained such that $y_j^+ \geqslant 0$ and $y_j^- \geqslant 0$.

In the QP, obstacle-avoidance constraints are linearized based on the waypoints of the current iteration's trajectory (Alg. 2). To compute these constraints, the algorithm evaluates the spline

$$\mathbf{q}_{\text{spline}}(s;t) = \mathbf{q}_t + s\mathbf{v}_t + \frac{1}{2}s^2\mathbf{a}_t + \frac{1}{6}s^3\mathbf{j}_t$$

between each pair of waypoints $(\mathbf{x}_t, \mathbf{x}_{t+1})$ against a depth map of obstacles to find the time $s \in [0, t_{\text{step}})$ and corresponding configuration $\hat{\mathbf{q}}_t$ that minimizes signed-distance separation from any obstacle. In this evaluation, a negative signed distance means the configuration is in collision. The algorithm then uses this $\hat{\mathbf{q}}_t$ to computes a separating hyperplane in the form $\mathbf{n}^{\mathsf{T}}\mathbf{q} + d = 0$. The hyperplane is either the top plane of the obstacle it is penetrating, or the plane that separates $\hat{\mathbf{q}}_t$ from the nearest obstacle (see Fig. 2.3). By selecting the top plane of the penetrated obstacle, this pushes the trajectory above the obstacle, which

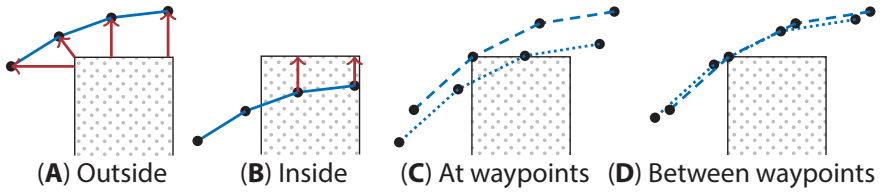


Figure 2.3: **Obstacle constraint linearization**. The constraint linearization process keeps the trajectory away from obstacles by adding constraints based on the Jacobian of the configuration at each waypoint of the accepted trajectory $\mathbf{x}^{(k)}$. In this figure, the obstacle is shown from the side, the robot's path along part of $\mathbf{x}^{(k)}$ is shown in blue, and the constraints' normal projections into Euclidean space are shown in red. For waypoints that are outside the obstacle (A), constraints keep the waypoints from entering the obstacle. For waypoints that are inside the obstacle (B), constraints push the waypoints up out of the obstacle. If the algorithm adds constraints only at waypoints as in (C), the optimization can compute trajectories that collide with obstacles and produce discontinuities between trajectories with small changes to the pick or place frame. These effects are mitigated when obstacles are inflated to account for them, but the discontinuities can lead to poor results when training the neural network. The proposed algorithm adds linearized constraints to account for collision between obstacles, leading to more consistent results shown in (D).

is a specialization of TrajOpt's more general obstacle-avoidance approach that is useful in bin-picking. By selecting the hyperplane of the nearest obstacle, the algorithm keeps the trajectory from entering the obstacle. The linearize constraint for this point is:

$$\mathbf{n}^\mathsf{T}\hat{\mathbf{J}}_t^{(k)}\hat{\mathbf{x}}_t^{(k+1)} \geqslant -d - \mathbf{n}^\mathsf{T}p(\hat{\mathbf{x}}_t^{(k)}) + \mathbf{n}^\mathsf{T}\hat{\mathbf{J}}_t^{(k)}\hat{\mathbf{x}}_t^{(k)},$$

where $\hat{\mathbf{J}}_t$ is the Jacobian of the robot's position at $\hat{\mathbf{q}}_t$. As $\hat{\mathbf{q}}_t$ and $\hat{\mathbf{J}}_t$ are at an interpolated state between optimization variables at $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$, linearizing this constraint requires computing the chain rule for the Jacobian:

$$\hat{\mathbf{J}}_t = \mathbf{J}_p(\hat{\mathbf{q}}_t)\mathbf{J}_q(s),$$

where $\mathbf{J}_p(\hat{\mathbf{q}}_t)$ is the Jacobian of the position at configuration $\mathbf{q}_t$, and $\mathbf{J}_q(s)$ is the Jacobian of the configuration on the spline at s:

$$\mathbf{J}_q(s) = \begin{bmatrix} \frac{\partial p}{\partial q_t} \\ \frac{\partial p}{\partial q_{t+1}} \\ \frac{\partial p}{\partial v_t} \\ \frac{\partial p}{\partial v_{t+1}} \end{bmatrix}^T = \begin{bmatrix} -3\frac{s^2}{t^2} + 2\frac{s^3}{t^3} + 1 \\ 3\frac{s^2}{t^2} - 2\frac{s^3}{t^3} \\ -2\frac{s^2}{t} + \frac{s^3}{t^3} + s \\ \frac{s^3}{t^2} - \frac{s^2}{t} \end{bmatrix}^\mathsf{T}.$$

We observe that linearization at each waypoint will safely avoid obstacles with a sufficient buffer around obstacles (e.g., via a Minkowski difference with obstacles), however, slight variations in pick or place frames can shift the alignment of waypoints to obstacles. This shift of alignment (see Fig. 2.3 (c)), can lead to solutions that vary disproportionately to

Algorithm 2 Linearize Obstacle-Avoidance Constraint

for $t \in [0, H)$ **do**
  $(\mathbf{n}_{\min}, d_{\min}) \leftarrow$ linearize obstacle nearest to $p(\mathbf{q}_t)$
  $\mathbf{q}_{\min} \leftarrow \mathbf{q}_t$
  for all $s \in [0, t_{\text{step}})$ **do** {line search s to desired resolution}
    $\mathbf{q}_s \leftarrow \mathbf{q}_t + s\mathbf{v}_t + \frac{1}{2}s^2\mathbf{a}_t + \frac{1}{6}s^3\mathbf{j}_t$
    $(\mathbf{n}_s, d_s) \leftarrow$ linearize obstacle nearest to $p(\mathbf{q}_s)$
    if $\mathbf{n}_s^\mathsf{T} p(\mathbf{q}_s) + d_s < \mathbf{n}_{\min}^\mathsf{T} p(\mathbf{q}_{\min}) + d_{\min}$ **then** {compare signed distance}
      $(\mathbf{n}_{\min}, d_{\min}, \mathbf{q}_{\min}) \leftarrow (\mathbf{n}_s, d_s, \mathbf{q}_s)$
      $\mathbf{J}_q \leftarrow$ Jacobian of $\mathbf{q}_s$
  $\mathbf{J}_p \leftarrow$ Jacobian of position at $\mathbf{q}_{\min}$
  $\hat{\mathbf{J}}_t \leftarrow \mathbf{J}_p\mathbf{J}_q$
  $b_t \leftarrow -d_{\min} - \mathbf{n}_{\min}^\mathsf{T} p(\mathbf{q}_{\min}) + \mathbf{n}_{\min}^\mathsf{T}\hat{\mathbf{J}}_t\mathbf{x}_t - \mu y_t^+$ {lower bound with slack y_t^+}
  Add $(\mathbf{n}_{\min}^\mathsf{T}\hat{\mathbf{J}}_t\mathbf{x}_t \geqslant b_t)$ and $(y_t^+ \geqslant 0)$ to set of linear constraints in QP

small changes in input. While this may be acceptable in operation, it can lead to data that can be difficult for a neural network to learn.

As with GOMP, DJ-GOMP allows degrees of freedom in rotation and translation to be added to start and goal grasp frames. Adding this degree of freedom allows DJ-GOMP to take a potentially shorter path when an exact pose of the end-effector is unnecessary. For example, a pick point with a parallel-jaw gripper can rotate about the axis defined by antipodal contact points (see Fig. 2.1), and the pick point with a suction gripper can rotate about the normal of its contact plane. Similarly, a task may allow for a place point anywhere within a bounded box. The degrees of freedom about the pick point can be optionally added as constraints that are linearized as:

$$\mathbf{w}_{\min} \leqslant \mathbf{J}_0^{(k)}\mathbf{q}_0^{(k+1)} - (\mathbf{g}_0 - p(\mathbf{q}_0^{(k)})) + \mathbf{J}_0^{(k)}\mathbf{q}_0^{(k)} \leqslant \mathbf{w}_{\min},$$

where, $\mathbf{q}_0^{(k)}$ and $\mathbf{J}_0^{(k)}$ are the configuration and Jacobian of the first waypoint in the accepted trajectory, $\mathbf{q}_0^{(k+1)}$ is one of variables the QP is minimizing, and $\mathbf{w}_{\min} \leqslant \mathbf{w}_{\max}$ defines the twist allowed about the pick point. Applying a similar set of constraints to $\mathbf{g}_H$ allows degrees of freedom in the place frame as well.

The SQP establishes trust regions to constrain the optimized trajectory to be within a box with extents defined by a shrinking trust region size. As convex constraints on dynamics enforce the relationship between configuration, velocity, and acceleration of each waypoint, we observe that trust regions only need to be defined as box bounds around one of the three for each waypoint. In experiments, we established trust regions on configurations.

To find the minimum time-time trajectory, J-GOMP searches for the shortest jerk-minimized trajectory that solves all constraints. This search, shown in Alg. 3, starts with an guess of H, then perform an exponential search for the upper bound, followed by a binary search for the shortest H by repeatedly performing the SQP of Alg. 1. The binary search

Algorithm 3 Time-optimal Motion Plan

Require: $\mathbf{g}_0$ and $\mathbf{g}_H$ are the start and end frames motion frames, $\gamma > 1$ is the search bisection ratio

$H_{\text{upper}} \leftarrow$ fixed or estimated upper limit of maximum time
$H_{\text{lower}} \leftarrow 3$
$v_{\text{upper}} \leftarrow \infty$ {constraint violation}
while $v_{\text{upper}} >$ tolerance **do** {find upper limit}
  $(\mathbf{x}_{\text{upper}}, v_{\text{upper}}) \leftarrow$ call Alg. 1 with cold-start trajectory for H_{upper}
  $H_{\text{upper}} \leftarrow \max(H_{\text{upper}} + 1, \lceil \gamma H_{\text{upper}} \rceil)$
while $H_{\text{lower}} < H_{\text{upper}}$ **do** {search for shortest H}
  $H_{\text{min}} \leftarrow H_{\text{lower}} + \lfloor (H_{\text{upper}} - H_{\text{lower}})/\gamma \rfloor$
  $(\mathbf{x}_{\text{mid}}, v_{\text{mid}}) \leftarrow$ call Alg. 1 with warm-start trajectory $\mathbf{x}_{\text{upper}}$ interpolated to H_{mid}
  if $\mathbf{v}_{\text{mid}} \leqslant$ tolerance **then**
    $(H_{\text{upper}}, \mathbf{x}_{\text{upper}}, v_{\text{upper}}) \leftarrow (H_{\text{mid}}, \mathbf{x}_{\text{mid}}, v_{\text{mid}})$
  else
    $H_{\text{lower}} \leftarrow H_{\text{mid}} + 1$
return $\mathbf{x}_{\text{upper}}$

warm starts each SQP with an interpolation of the trajectory of the current upper bound of H. The search ends when the upper and lower bound of H are the same.

2.4.2 Deep Learning of Trajectories

To speed up motion planning, we add a deep neural network to the pipeline. This neural network treats the trajectory optimization process as a function f_τ to approximate:

$$f_\tau : SE(3) \times SE(3) \to \mathbb{R}^{H^* \times n \times 4},$$

where the arguments to the function are the pick and place frames, and the output is a discretized trajectory of variable length H^* waypoints, each of which has a configuration, velocity, acceleration, and jerk for all n joints of the robot. We assume that the neural network $\tilde{f}_\tau$ can only approximate f_τ, thus $\tilde{f}_\tau(\cdot) = f_\tau(\cdot) + E(\tau)$ for some unknown error distribution $E(\tau)$. As such, the output of $\tilde{f}_\tau$ may not be dynamically or kinematically feasible. To address this potential issue, we use the network's output to warm start a final pass through the SQP. This process can be thought of as polishing the output of the neural network approximation to overcome any errors in the network's output.

In this section, we describe a proposed neural network architecture, its loss function, training and testing dataset generation, and the training process. While we posit that a more general approximation could include the amount of pick and place degrees of freedom as inputs, for brevity we assume that f_τ and its neural-network approximation are parameterized by a preset amount of pick and place degrees of freedom. In practice, it may also be appropriate to train multiple neural networks for different parameterizations of f_τ.

Architecture

The deep neural network architecture we propose is depicted in Fig. 2.2. It consists of an input layer connected through four fully-connected blocks to multiple output blocks. The input layer takes in the concatenated grasp frames $\begin{bmatrix}\mathbf{g}_0^\mathsf{T} & \mathbf{g}_H^\mathsf{T}\end{bmatrix}^\mathsf{T}$. Since the optimal trajectory length H^* can vary, the network has multiple output heads for each of the possible values for H^*. To select which of the outputs to use, we use a separate classification network with 2 fully-connected layers with one-hot encoding trained using a cross-entropy loss.

We refer to the horizon classification and multiple output network as a HYDRA (Horizon Yielding Distillation through Retained Activations) network. The network yields both an optimal horizon length and the trajectory for that horizon. To train this network (detailed below), the trajectory output layers' activation values for horizons not in the training sample are retained using a zero gradient so as to weight the contribution of the layers during backprop to the input layers.

In experiments, a neural network with a single output head was unable to produce a consistent result for predicting varied length horizons. We conjecture that the discontinuity between trajectories of different horizon lengths made it difficult to learn. In contrast, we found that a network was able to accurately learn a function for a single horizon length, but was computationally and space inefficient, as each network should be able to share information about the function between the horizons. This lead to the proposed design in which a single network with multiple output heads shares weights through multiple shared input layers.

Dataset Generation

We propose generating a training dataset by randomly sampling start and end pairs from the likely distribution of tasks. E.g., in a warehouse pick-and-place operation, the pick frames will be constrained to a volume defined by the picking bin, and the place frames will be constrained to a volume defined by the placement or packing bin. For each random input, we generate optimized trajectories for all time horizons from $H_{\max}$ to the optimal H^*. While this process generates more trajectories than necessary, generating each trajectory is efficient since the optimization for a trajectory of size H warm starts from the trajectory of size $H+1$. Overall, this process is efficient, and with parallelization can quickly generate a large training dataset.

This process can also help detect if the analysis of the maximum trajectory duration was incorrect. If all trajectories are significantly shorter than $H_{\max}$, then one may reduce number of output heads. If any trajectory exceeds $H_{\max}$, then the number of output heads can be increased.

We also note that in the case where the initial training data does not match the operational distribution of inputs, the result may be that the neural network produces sub-optimal motions that are substantially kinematically and dynamically infeasible. In this case, the subsequent pass through the optimization (See Sec. 2.4.3) will fix these errors, though with a

longer computation time. We propose, in a manner similar to DAgger [41], that trajectories from operation can be continually added to the training dataset for subsequent training or refinement of the neural network.

Training

To train the network in a way that encourages matching the reference trajectory while keeping the output trajectory kinematically and dynamically feasible, we propose a multipart loss function $\mathcal{L}$. This loss function includes: a weighted sum of mean-squared error (MSE) loss on the trajectory $\mathcal{L}_\mathcal{T}$, a boundary loss $\mathcal{L}_\text{B}$ which enforces the correct start and end positions, and a dynamics loss $\mathcal{L}_\text{dyn}$ that enforces the dynamic feasibility of the trajectory. The MSE loss is the mean of the sum of squared differences of the two vector arguments: $\mathcal{L}_\text{MSE}(\tilde{\mathbf{a}}, \underline{\mathbf{a}}) = \frac{1}{n}\sum_{i=1}^{n}(\tilde{\mathbf{a}}_i - \underline{\mathbf{a}}_i)^2$. The dynamics loss attempts to mimic the convex constraints of the SQP. Given the predicted trajectories $\tilde{X} = (\tilde{\mathbf{x}}^{H_\text{min}}, \ldots, \tilde{\mathbf{x}}^{H_\text{max}})$, where $\tilde{\mathbf{x}}^h = (\tilde{\mathbf{q}}, \tilde{\mathbf{v}}, \tilde{\mathbf{a}}, \tilde{\mathbf{j}})_{t=0}^{h}$ and the ground-truth trajectories from dataset generation $\underline{X} = (\underline{\mathbf{x}}^{H^*}, \ldots, \underline{\mathbf{x}}^{H_\text{max}})$, the loss functions are:

$$
\begin{aligned}
\mathcal{L}_\mathcal{T} &= \alpha_q \mathcal{L}_\text{MSE}(\tilde{\mathbf{q}},\, \underline{\mathbf{q}}) + \alpha_v \mathcal{L}_\text{MSE}(\tilde{\mathbf{v}},\, \underline{\mathbf{v}}) + \alpha_a \mathcal{L}_\text{MSE}(\tilde{\mathbf{a}},\, \underline{\mathbf{a}}) + \alpha_j \mathcal{L}_\text{MSE}(\tilde{\mathbf{j}},\, \underline{\mathbf{j}}) \\
\mathcal{L}_\text{B} &= \mathcal{L}_\text{MSE}(\tilde{\mathbf{q}}_0,\, \underline{\mathbf{q}}_0) + \mathcal{L}_\text{MSE}(\tilde{\mathbf{q}}_H,\, \underline{\mathbf{q}}_H) \\
\mathcal{L}_\text{dyn} &= \frac{1}{h}\sum_{t=0}^{h-1} \left\| \tilde{\mathbf{q}}_t + t_\text{step}\tilde{\mathbf{v}}_t + \frac{1}{2}t_\text{step}^2\tilde{\mathbf{a}}_t + \frac{1}{6}t_\text{step}^3\tilde{\mathbf{j}}_t \;-\; \tilde{\mathbf{q}}_{t+1} \right\|^2 \\
&\quad + \frac{1}{h}\sum_{t=0}^{h-1} \left\| \tilde{\mathbf{v}}_t + t_\text{step}\tilde{\mathbf{a}}_t + \frac{1}{2}t_\text{step}\tilde{\mathbf{j}}_t \;-\; \tilde{\mathbf{v}}_{t+1} \right\|^2 \\
&\quad + \frac{1}{h}\sum_{t=0}^{h-1} \left\| \tilde{\mathbf{a}}_t + t_\text{step}\tilde{\mathbf{j}}_t \;-\; \tilde{\mathbf{a}}_{t+1} \right\|^2 \\
&\quad + \frac{1}{h}\sum_{t=0}^{h-1} \left\| \frac{1}{t_\text{step}}(\underline{\mathbf{j}}_{t+1} - \underline{\mathbf{j}}_t) - \frac{1}{t_\text{step}}(\tilde{\mathbf{j}}_{t+1} - \tilde{\mathbf{j}}_t) \right\|^2 \\
\mathcal{L}^h &= \alpha_\mathcal{T}\mathcal{L}_\mathcal{T}^h + \alpha_\text{B}\mathcal{L}_B^h + \alpha_\text{dyn}\mathcal{L}_\text{dyn}^h,
\end{aligned}
$$

where values of $\alpha_q = 10$, $\alpha_v = 1$, $\alpha_a = 1$, $\alpha_j = 1$, $\alpha_\text{B} = 4{\times}10^3$, and $\alpha_\text{dyn} = 1$ were chosen empirically. This loss is combined into a single loss for the entire network by summing the losses of all horizons while multiplying by an indicator function for the horizons that are valid:

$$
\mathcal{L} = \sum_{h=H_\text{min}}^{H_\text{max}} \mathcal{L}^h \mathbb{1}_{[\underline{H}^*, H_\text{max}]}(h).
$$

As the ground-truth trajectories for horizons shorter than H^* are not defined, we must insure that some finite data is present so that the multiplication an indicator value of 0 results in 0 (and not NaN). Multiplying by indicator function in this way results in a zero gradient for the part of the network that is not included in the trajectory data.

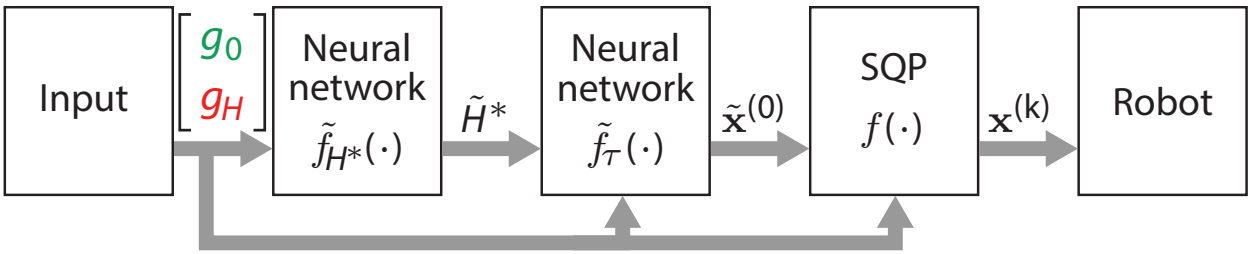


Figure 2.4: **The fast motion planning pipeline**. The pipeline has three phases between input and robot execution. The first phase estimates the trajectory horizon H^* by computing a forward pass of the neural network. The second phase estimates the trajectories for H^* to create an initial trajectory for the sequential quadratic program (SQP) optimization process. The SQP then optimizes the trajectory, insuring that it meets all joint kinematic and dynamic limits so that it can successfully execute on a robot.

During training we observed that the network would often exhibit behavior of co-adaptation in which it would either learn $\mathcal{L}_{\mathcal{T}}$ or $\mathcal{L}_{\text{dyn}}$ but not both. This showed up as a test loss for one going to small values, while the other remained high. To address this problem we introduced dropout layers [42] with dropout probability $p_{\text{drop}} = 0.5$ between each fully-connected layer, shown in Fig. 2.2. We annealed [43] p_{drop} to 0 over the course of the training to reduce the loss further.

2.4.3 Fast Pipeline for Trajectory Generation

The end goal of this proposed motion-planning pipeline is to generate feasible, time-optimized trajectories quickly. The SQP computes feasible, time-optimized trajectories, but is slow when starting from scratch. The HYDRA neural network computes trajectories quickly (e.g., the forward pass on the network in the results section requires $\sim$1 ms to compute), but without guarantees on correctness. In this section we propose combining the properties of the SQP and HYDRA into a pipeline (see Fig. 2.4) to get fast computation of correct trajectories by using a forward pass on the neural network to warm start the SQP.

The first step in the pipeline is to compute $\tilde{H}^*$, an estimate of the optimal time horizon H^*. This requires a single forward-pass through the one-hot classification network. Since predicting horizons shorter than H^* result in an infeasible SQP, it can be beneficial to either compute multiple SQPs around the predicted horizon, or increase the horizon if the difference in the one-hot values for $\tilde{H}^*$ and $\tilde{H}^* + 1$ is within a threshold.

The second step in the pipeline is to compute $\tilde{\mathbf{x}}^{(0)}$, an estimate of the time-optimal trajectory for $\tilde{H}^*$ using a forward pass through the HYDRA network.

The final step is to compute the trajectory using $\tilde{\mathbf{x}}^{(0)}$ to warm start the SQP. In this step, since the warm-start trajectory is close to the final trajectory and generating a smooth training dataset is not a requirement, we can speed up the SQP process by relaxing the termination conditions to the tolerances of the robot and task, e.g., terminating when the

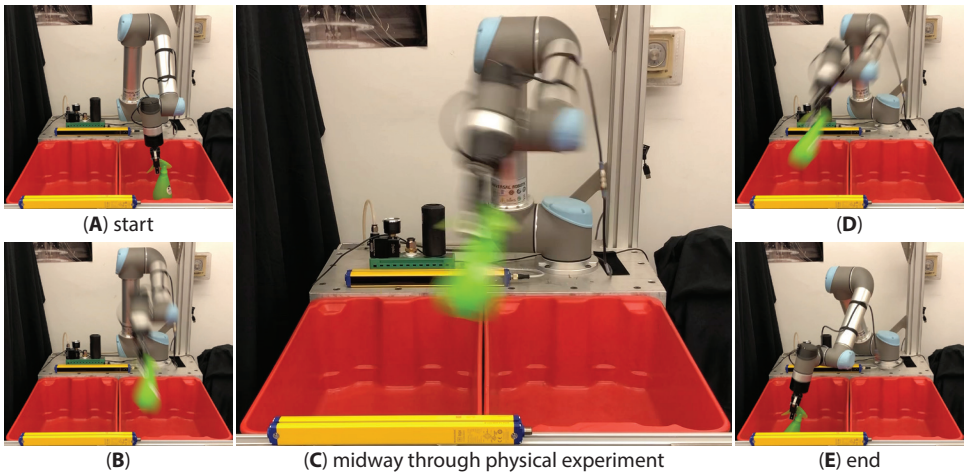


Figure 2.5: **Physical experiment executing jerk-limited motion computed by DJ-GOMP on a UR5.** The motion starts by picking an object from the right bin (A), moves over the divider (B), (C), (D), and ends after placing the object in the left bin (E). Without the jerk limits, the motion takes 448 ms but results in a high jerk at the beginning and end of the motion, which in this case causes the UR5 robot to overshoot its end frame by a few millimeters. With jerk limits, the motion takes 544 ms, reduces wear, and does not overshoot the end frame.

pick point (and other constraints) is within 10^{-3} meters of the target frame, instead of the 10^{-6} meters used in dataset generation.

We observe that symmetry in grippers, such as found in parallel and multi-finger grippers, means that multiple top-down grasps can result in the same contact points (e.g., see Fig. 2.1 (A) and (D)). In this setting, we can use $\tilde{f}_H(\cdot)$ to estimate optimal horizons for all the grasp configurations and quickly select the one with the lowest horizon. Experimentally, we find that breaking ties for optimal horizons using the associated one-hot values, leads to faster trajectory optimization compute times.

2.5 Experiments

We test DJ-GOMP on physical UR5 robot [44] fitted with a Robotiq 2F-85 [45] parallel gripper. In the experiment setup (see Fig. 2.5), the robot must move objects from one fixed bin location to another. We set DJ-GOMP to be constrained according to the specified joint configuration and velocity limits of the UR5. We derived an acceleration limit based on the UR5's documented torque and payload capacity, and we limit the jerk to a multiple of the computed acceleration limit. In practice, we surmise that an operator would define jerk limits by taking into account the desired service life of the robot.

To generate train/test data for the deep neural network, we use all 80 hardware threads of an NVIDIA DGX-1 to compute 100 000 optimized input+trajectory $(\begin{bmatrix}\mathbf{g}_0^\mathsf{T} & \mathbf{g}_H^\mathsf{T}\end{bmatrix}^\mathsf{T}, \mathbf{x}^*)$ pairs,

where $\mathbf{x}^*$ is the discretized trajectory. The J-GOMP optimizer is written in C++ and uses OSQP [46] as the underlying QP solver. The inputs it generates consist of random pick ($\mathbf{t}_0$) and place ($\mathbf{t}_H$) translations drawn uniformly from the pick and place physical space. For each generated translation, we also generate a top-down rotation angle (θ_0 and θ_H) uniformly drawn from $[0, \pi)$. Since a parallel gripper's grasp has an equivalent, though kinematically different (see Fig. 2.1 (a) and (d)), grasp with a 180° rotation, for each translation+rotation grasp we also add its rotation by 180°. Thus for each random $\begin{bmatrix}\mathbf{t}_0^\mathsf{T} & \mathbf{t}_H^\mathsf{T}\end{bmatrix}^\mathsf{T}$ pair, we add 4 grasp frames with rotation (θ_0, θ_H), $(\theta_0+\pi, \theta_H)$, $(\theta_0, \theta_H+\pi)$, $(\theta_0+\pi, \theta_H+\pi)$ and their trajectories.

We train the deep network with the Adadelta [47] optimizer for 50 epochs after initializing the weights using a He Uniform initializer [48]. The network architecture and optimization framework are written in Python using PyTorch. All training and deep network computations are accelerated by GPUs on NVIDIA DGX-1's Tesla V100 SXM2 GPU and Intel Xeon E5-2698 v4 CPUs.

To evaluate the ability of the deep-learning approach of DJ-GOMP to speed up motion planning, we compute 1000 random motion plans both without and with deep-learning-based warm start and plot the results in Fig. 2.6. The median compute time without deep-learning is 29.0 s. Using a network to estimate the optimal time horizon but not the trajectory, can speed up computation significantly, but at a cost of increased failure rate. Using the network to both predict the time horizon and the warm-start trajectory results in a median with deep-learning of 80 ms; when compared to J-GOMP, this shows 2 orders of magnitude improvement, an approximate 300x speedup.

To evaluate the effect on the optimality of the computed trajectories, we compare the sum-of-squared jerks between trajectories generated with the full SQP vs. those generated with a warm-started prediction with the optimal horizon. We observe over 99% of the test trajectories are within 10^{-3} of each other, which is an error value that is within the tolerance bounds we set for the QP optimizer. For a small fraction (less than 1%), we observe the warm-started optimization and the full optimization find different local minima, without clear benefit to either optimization.

As the optimality of the trajectory and the failure rate is dependent on accurately predicting the optimal time horizon of a trajectory, we separately evaluate this prediction. We observe that shorter values of the horizon lead inevitably to SQP failures, while longer values lead to suboptimal trajectories. Since failures are likely to be more problematic than slighty slower trajectories, we propose a simple heuristic to predict longer horizons. When the network predicts a horizon longer than the optimal, we observe that the optimization of trajectories with sub-optimal horizon can be faster than that of the optimal horizon (shown in Fig. 2.6 (B)). This is likely due to the sub-optimal trajectory being less constrained and thus faster to converge. In practice, we propose that using a readily-available multi-core CPU to simultaneous compute multiple SQPs for different horizons around the estimated horizon would be a practical way to address the failures and sub-optimal trajectories. However, if constrained to a single-core computation, using an longer horizon may also be practical as the compute time saved may be more than time saved by using the optimal horizon.

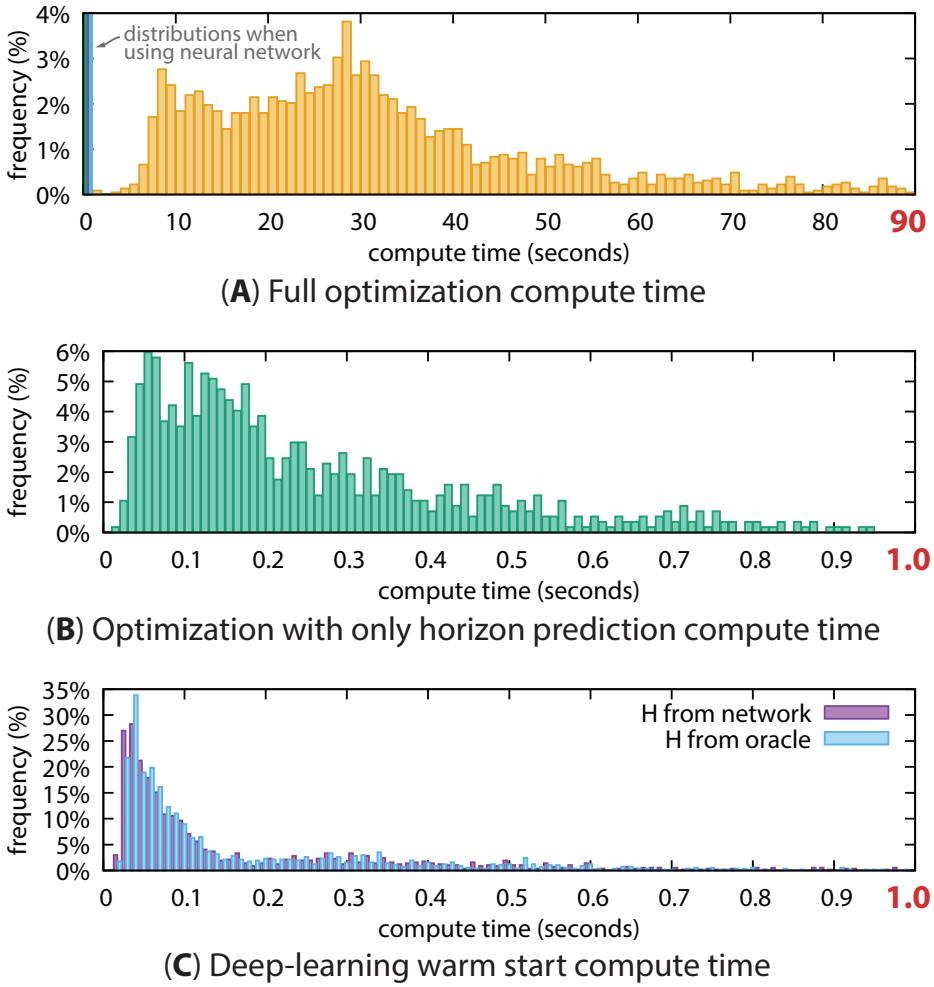


Figure 2.6: **Compute time distribution for 1000 random motion plans.** In these plots the x-axis shows total compute time in seconds for a single optimized trajectory. Plot (A) extends to 90 seconds, while plots (B) and (C) extend to 1 second. The y-axis shows the distribution compute time required. The full optimization process without the deep-learning prediction, shown in the histogram in (A), requires orders of magnitude longer to compute. Using a deep network to predict the optimal time horizon for a trajectory, but not warm-starting the trajectory (B) leads to improvements in compute time, though with increased failures. Using the deep network to compute a trajectory to warm start the optimization (C) further improves the compute time. In (C), the plots include results for both estimated trajectory horizon H, and the exact H from the full optimization to show the effect of misprediction of trajectory length—inexact predictions can result in a faster compute time, since the resulting trajectory is sub-optimal, thus less tightly constrained. The upper limit on the x-axis is shown in red to highlight the difference in scale—plots (B) and (C) are magnified by 2 orders of magnitude.

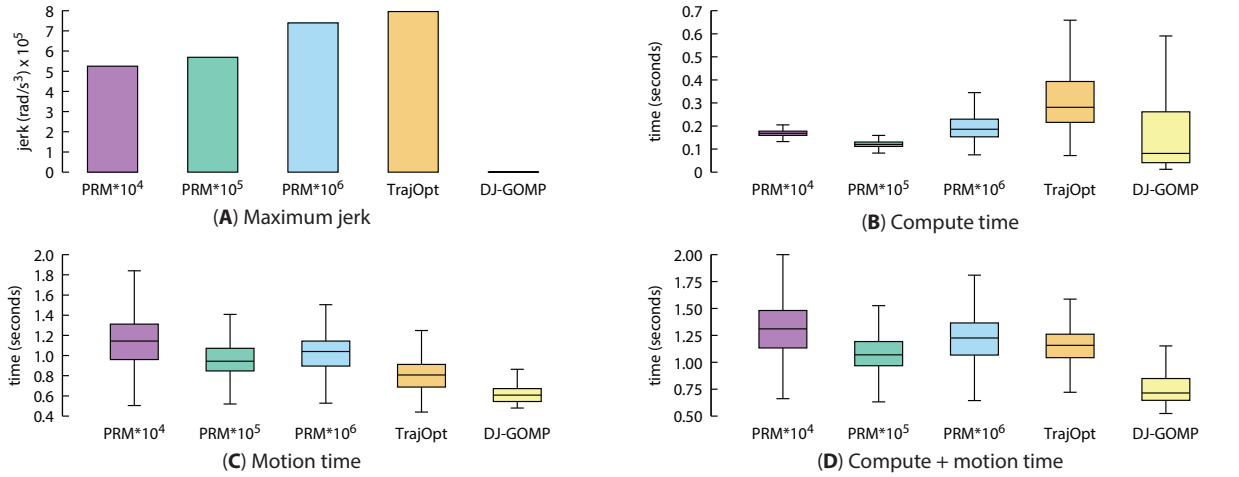


Figure 2.7: **Maximum jerk and timing comparisons for 1000 pick-place pairs computed with PRM*, TrajOpt, and DJ-GOMP.** These graphs compare motion plan (A) jerk, (B) compute time, (C) motion time, and (D) combined compute+motion time. The filled boxes spans the first through third quartile with a horizontal line at the median. The whiskers extend from the minimum to maximum values. Paths computed by Probabilistic Road Map Star (PRM*) [2, 3] and TrajOpt [5] are subsequently optimally time parameterized [49]. The time parameterization does not limit jerk as DJ-GOMP does, which allows for faster but high jerk motions. Even so, as DJ-GOMP directly optimizes the path, unlike PRM* and TrajOpt, DJ-GOMP generates the fastest motions; while its deep-learning-based warm start allows for fast compute and motion times.

To evaluate the effect on failure rate, we record the number of failures with both cold-started and warm-started optimization with the optimal horizon (observing that predicting short horizon is the other source of failures). Cold-started optimizations fail 10.7%, while warm-started optimizations fail 5.7%. These failures occur because the optimizer cannot move the trajectory into a feasible region due to the tight constraints. In experiments, the failure rate went down with additional training data and longer network training, suggesting further improvement is possible.

We compare compute-time and motion-time performance to Probabilistic Road Maps "Star" (PRM*) [2, 3] and TrajOpt [5]. For PRM*, we precompute graphs of 10k, 100k, and 1000k vertices over the workspace in front of the robot. As PRM* is an asymptotically optimal motion planner, graphs with more vertices should produce shorter paths, at the expense of longer graph search time. For TrajOpt we configure the optimization parameters to match that of DJ-GOMP, observing that this improves success rate over the default. Straight-line initialization in TrajOpt fails in this environment due to the bin wall between the start and end configurations—while DJ-GOMP's specialized obstacle model moves the trajectory out of collision, TrajOpt's obstacle model result in linearizations that do not push the trajectory out of collision. We thus initialize TrajOpt with a trajectory above the obstacles in the workspace. Since both PRM* and TrajOpt do not directly produce time-parameterized

trajectories, we use Kunz et al.'s method [49] to compute time-optimal time parameterization. This time-parameterization method first "rounds corners" by adding smooth rounded segments to connect the piecewise linear motion plan from PRM* before computing the optimal timing for each waypoint. Without the rounded corners, the robot would have to stop between each linear segment of the motion plan to avoid an instantaneous infinite acceleration. The radius of the corner rounding is tunable, however rounding corners too much, can result in a motion plan that collides with obstacles. This time parameterization also does *not* minimize or limit jerk, and thus produces high jerk trajectories with peaks in the range 5×10^5 to 8×10^5 rad/s^3 (Fig. 2.7 (a)), and meaning that they should have an advantage in motion time over jerk-limited motions (Fig. 2.8). As a final step, since 180-degree rotated parallel jaw grasps are equivalent, we compute trajectories for each pick and place combination and select the fastest motion. The results for 1000 pick-place pairs are shown in Fig. 2.7. We observe that PRM* has consistent fast compute times, but produces the slowest trajectories. TrajOpt is slower to compute but produces faster trajectories than PRM*. DJ-GOMP, since it directly optimizes for a time-optimal path, produces the fast motions, while the deep-learning horizon prediction and warm start allow it to compute quickly despite complex constraints, and result in the overall fastest combined compute and motion time.

To evaluate whether motion plans that DJ-GOMP generates work on a physical robot, we have a UR5 follow trajectories that DJ-GOMP generates. An example motion is shown in Fig. 2.5. The UR5 controller does not allow the robot to exceed joint limits and issues an automated emergency-stop when it does. The trajectories that DJ-GOMP generates are constrained to the documented limits, and thus do not cause the stop. However, we have observed that, without jerk limits, a high-jerk trajectory can cause the UR5 to overshoot its target and bounce back. With DJ-GOMP's jerk-limited trajectories the UR5 empirically does not overshoot.

2.6 Discussion and Future Work

We presented DJ-GOMP, a deep-learning approach to time-optimal jerk-limited grasp optimized motion planning. DJ-GOMP, like its predecessor GOMP, allows pick and place frames to be considered in the optimization, e.g., when gripper design and grasp analysis mean moving around one or more degrees of freedom will result in the same quality of grasp. To address the long compute time of the optimization process, DJ-GOMP computes an approximation of the optimization using a deep neural network. While the neural network computes trajectories quickly, it does not guarantee that the trajectory is kinematically or dynamically feasible. As a final step, DJ-GOMP performs a constrained optimization warm started with the network's trajectory resulting in fast computation of smooth, jerk-limited, time-optimal motion plans. In future work we will explore expanding DJ-GOMP to additional robots performing more varied tasks that would include increased variation of start and goal configurations and in more complex environments. We will also explore additional

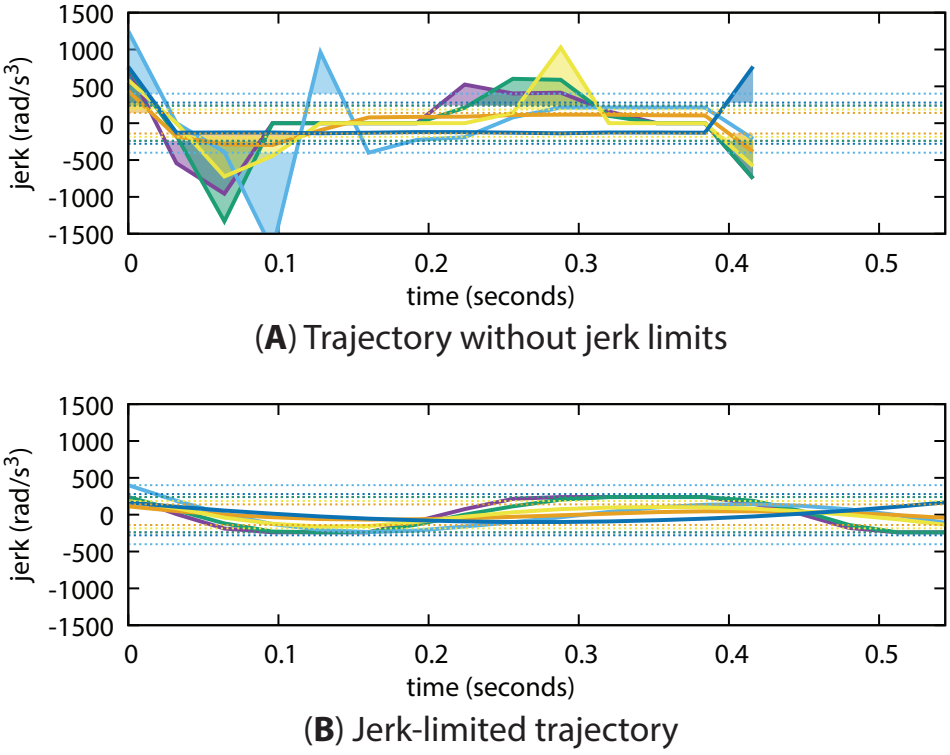


Figure 2.8: **Jerk limit's effect on computed and executed motion.** We plot the jerk (y-axis) of each joint in rad/s^3 over time in milliseconds (x-axis) as computed (A) without jerk limits, and (B) with jerk limits. Without jerk limits, the optimization computes trajectories with large jerks throughout the trajectory (shown in shaded regions). With jerk limits each joint stays within the defined limits (the dotted lines) of the robot.

deep-learning approaches to find better approximations of the optimization process, and thus allow for faster warm starting of the final optimization step of DJ-GOMP.

Chapter 3

GOMP-FIT: Making Fast Object Transport Reliable

The previous chapter discusses how to combine deep learning and optimization to speed up collision-free motions for object transport. However, accelerating too fast with a parallel-jaw gripper can lead to increased drops, content spill and damage the product. GOMP-FIT increases transport reliability of parallel-jaw grippers by introducing end-effector acceleration constraints into the GOMP optimization.

3.1 Introduction

Fast and reliable robot pick-and-place motion is increasingly a bottleneck in automated warehouses. Motions that are too slow, while usually safe, reduce robot throughput, while motions that are too fast can be unreliable, leading to dropped objects due to shearing forces between the object and the gripper that holds it. Prior work, Grasp-Optimized Motion Planning (GOMP [9]), showed that simultaneously optimizing grasp pose and pick-and-place motion could allow for rapid object transport. However, given the torque available to industrial robot arms, the high-speed motions that GOMP generates could lead to product damage and spills. In this chapter, we propose a *fast inertial transport* (FIT) problem and algorithm to address it, in which a robot transports objects at high speeds, banking and limiting motions against inertial forces to safely and reliably place the object.

To enable FIT, we propose *Grasp-Optimized Motion Planning for Fast Inertial Transport* (GOMP-FIT) which computes time-optimized motions while taking into account end-effector and object acceleration constraints imposed by the object being transported. GOMP-FIT incorporates constraints for open-top container and fragile object transport, combinations thereof, and potentially more. For open-top containers, GOMP-FIT aligns inertial accelerations so that the contents remain in the container. For fragile-object transport, GOMP-FIT limits the magnitude of accelerations of the transported object to avoid exceeding a shock threshold. For fragile open-top containers, GOMP-FIT constrains both magnitude and align-

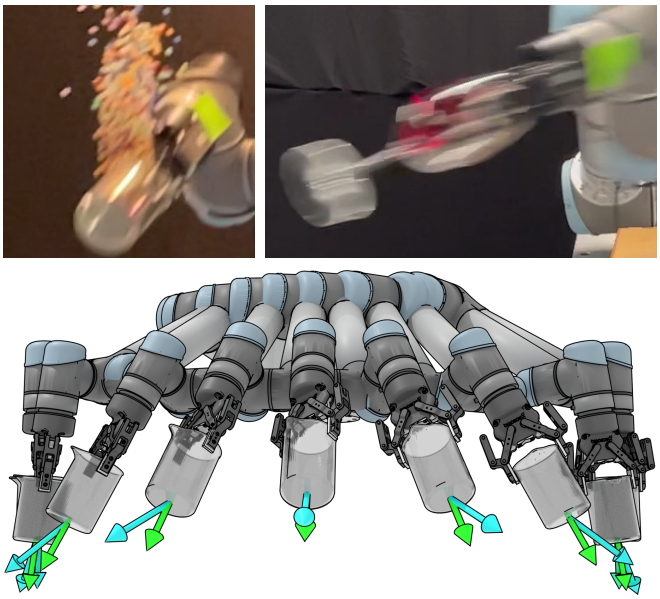


Figure 3.1: **High-speed transport-aware motion planning Top left:** without considering inertial effects, fast transport can lead to spills and damage. **Top right:** GOMP-FIT computes high-speed transport motions for objects such as a filled wineglass, without spilling or breaking fragile objects. **Bottom:** GOMP-FIT constrains the alignment between the end-effector accelerations due to inertial forces and gravity (cyan arrow), against the normal defined by the container (green arrow). Here it is limited to 30°. The optimization rotates the robot's joints to keep within the constraint.

ment of accelerations. As most industrial robot arms do not provide direct access to motor torques, GOMP-FIT instead operates in the joint configuration space, relying on the robot's black-box controller to follow trajectories.

In experiments, a physical UR5 robot transports various objects with end-effector acceleration requirements, including: transporting cups filled to various levels with beads without spilling their contents, transporting an IMU to measure accelerations that a fragile object would experience in transit, and moving a filled wineglass. Experiments suggest that GOMP-FIT can achieve near 100 % success on these tasks while slowing as little as 0 % when there are few obstacles, 30 % when there are high obstacles and 45-degree tolerances, and 50 % when there are 15-degree tolerances and few obstacles compared to GOMP. This chapter provides the following contributions: (1) A formulation of the fast inertial transport motion planning problem; (2) A sequential convex program, including non-convex constraints on accelerations at the end-effector, and a sequential quadratic program to solve it; and (3) Data from experiments on a physical UR5 robot transporting cups filled with beads, a fragile object and a filled wineglass.

3.2 Related Work

Motion planning seeks to produce paths from start to goal while avoiding obstacles. Sampling-based motion planners such as PRM [2], RRG [3], and RRT [4] have favorable properties such as probabilistic completeness and asymptotic optimality, but they may be slow to converge. Such planners also typically require a post-processing step to remove redundant or jerky motion. Optimization-based motion planners such as TrajOpt [5], STOMP [6], CHOMP [7], and KOMO [8] can produce smooth trajectories while fulfilling a set of constraints such as obstacle avoidance through interior point optimization, stochastic gradients, and covariant gradient descent. In prior formulations, the trajectory is discretized into a series of waypoints with a minimization objective such as sum of squares velocity. Motion planners such as GOMP [9] and DJ-GOMP [50] add dynamics constraints and shrinking horizon lengths to find a time-optimized trajectory, and apply jerk limits to avoid damaging the robot. Unlike previous work, this chapter applies additional constraints to the end-effector acceleration to prevent an object held by a robot arm from damage or detachment in high-acceleration trajectories, something that constraints on configurations and their derivatives alone cannot do.

Placing constraints on the end-effector motion path is a requisite part of many problems, including fast inertial transport. Yao and Gupta [51] address the path-planning problem with general end-effector constraints by exploring the task space for feasible end-effector poses through a sampling-based planner. Li et al. [52] propose using RL to compute motion plans for dual-arm manipulators with floating base in space. They propose to take into account a velocity constraint of the end-effector in the planning process to improve the robustness of the algorithm. But these methods do not support acceleration constraints for the fast inertial transport problem we propose.

Dynamic manipulation research uses dynamic properties such as momentum to achieve a task—whether prehensile or non-prehensile. Lynch and Mason [53] exploit centrifugal and Coriolis forces to manipulate objects using low-degree-of-freedom robots. Lynch and Mason [54] also formulate an SQP for dynamic non-prehensile manipulation of objects, such as snatching, throwing, and rolling. They integrate constraints based on a 2D formulation of the problem but do not consider obstacles. Srinivasa et al. [55] propose employing constraints on accelerations at the end point to perform dynamic non-prehensile rotation of an object. Kim et al. [56] propose a method to rapidly computes motions to catch an object in flight. Mucchiani and Yim [57] propose a method to use inertial effects to dynamically sweep up an object and stabilize it using a passive end-effector. In contrast, we propose a 3D prehensile planner for safe and reliable object transport around obstacles.

A promising line of research, especially for objects with unknown or difficult-to-model dynamics, is to employ learning. Zeng et al. [58] propose TossingBot that learns parameters of a pre-defined dynamic motion to throw objects into target bins. Zhang et al. [**zhang2021rotla**] propose a method that learns a sweeping dynamic motion for ropes to hit targets, weave through obstacles, or knock objects down. Wang et al. [59] propose SwingBot, that uses inhand tactile feedback to learn how to swing up previously unseen ob-

jects. These methods rely on a learned model for dynamic manipulation, while we propose using Newtonian physics.

Another promising approach is to combine dynamics considerations and sampling-based planning. Pham et al. [60, 61] proposed admissible velocity propagation (AVP) and a method to perform kinodynamic planning in a reduced dimensionality state space, and Lertkultanon and Pham [62] formulate a ZMP constraint for AVP and integrate bi-directional RRT to non-prehensile object transportation. Unlike the AVP formulation, GOMP-FIT integrates all constraints, including collision, in a single optimization.

To enable real-world interaction between robots and the environments they touch, researchers have looked into optimizing motions with contacts. Posa and Tedrake [63] and Posa et al. [64] propose simultaneously optimizing trajectories and contact points. Hauser [65] proposes a trajectory optimization that considers contact forces and convex time scaling. Luo and Hauser [66] propose integrating feedback and learning to integrate confidence into the optimization, and apply it to the Waiter's Problem of stably transporting objects on a moving platform. While these methods consider contacts, they do not consider obstacle avoidance.

Most closely related to this chapter is research into transport of objects that takes into account inertial effects. Bernheisel and Lynch [67] propose a Waiter's Problem in which object assemblies are stably transported subject to inertial and gravity forces. They focus on multi-part assemblies and assume quasistatic motion in which only the velocity direction matters. In contrast we focus on a containment and acceleration-based inertial effects. Wan et al. [68] demonstrates beverage transport with jerk limits help prevent spills during transport. In contrast, we show that jerk limits alone will not prevent spills when using a manipulator arms. Acharya et al. [69] propose methods to apply minimum-time s-curve trajectories to the Waiter's Problem. To transport objects with unknown mass parameters, Lee and Kim [70] perform online estimation of payload parameters to integrate into the trajectory generation for cooperative aerial manipulators. For fast transport of objects held in a suction grasp, Pham and Pham [71] propose combining RRT with a trajectory time parameterizer that remains within the suction contact stability constraint. In contrast, we propose a single optimization process that incorporates all necessary constraints to perform fast inertial transport.

While the primary intent for GOMP-FIT is not liquid transport, in experiments we show spill-free transport of a liquid. The related problem of slosh-free transport requires constraining accelerations of a container. Chen et al. [72] add liquid transport to the Waiter's problem and use an acceleration filter to orient the end-effector to counter sloshing effects through Cartesian control. Aribowo et al. [73] decouple liquid transport into two steps: computing a translation trajectory, then input shaping to counter sloshing by rotating the end-effector. Yano et al. [74], Reyhanoglu et al. [75], Consolini et al. [76] and Moriello et al. [77] propose using feedback and feed-forward control to avoid sloshing under varying assumptions such as linear actuation and absence of obstacles. In contrast to these works, GOMP-FIT integrates end-effector acceleration constraints into a single time-optimizing method.

3.3 Problem Statement

Let $\mathbf{q} \in \mathcal{C}$ be the complete specification of a robot's degrees of freedom, where $\mathcal{C}$ is the space of all configurations. Let $\mathcal{C}_{\text{obs}} \subset \mathcal{C}$ be the set of configurations that are in an obstacle, and $\mathcal{C}_{\text{free}} = \mathcal{C} \backslash \mathcal{C}_{\text{obs}}$ be the set of configurations that are not in collision. Let $f_k : \mathcal{C} \to SO(3)$ be the forward kinematics function that computes the pose of the object in the robot's end-effector. Let $f_k^{-1} : SE(3) \to \mathcal{C}$ be an inverse kinematic (IK) function that computes the robot's configuration, given a desired location of the end-effector. The IK function may not always have a solution, and may not have a unique solution. Let $\mathbf{x}^T = \begin{bmatrix}\mathbf{q}^T & \dot{\mathbf{q}}^T & \ddot{\mathbf{q}}^T\end{bmatrix} \in \mathcal{X}$ be the dynamic state of the robot at any moment in time, including the first and second derivative of the robot's configuration. Let $f_a : \mathcal{X} \to \mathbb{R}^3$ be the linear acceleration at the end-effector including gravity and the inertial (fictitious) forces: Euler, Coriolis, and centrifugal.

Given a starting grasp $\mathbf{g}_{\text{pick}} \in SE(3)$ and a placement pose $\mathbf{g}_{\text{place}} \in SE(3)$, and constraints on the end-effector accelerations, the objective of GOMP-FIT is to compute a trajectory $\tau : [0, T] \to \mathcal{C}$, where T is the duration, $\tau(t) \in \mathcal{C}_{\text{free}}$, $f(\tau(0)) = \mathbf{g}_{\text{pick}}$, $f(\tau(T)) = \mathbf{g}_{\text{place}}$, and the end-effector acceleration constraints are met at all $\tau(t)$. Furthermore, the objective is to minimize the total trajectory time, subject to the robot's actuation limits. Thus,

$$\begin{aligned} \arg\min_{\tau} \quad & T(\tau) \\ s.t. \quad & f_k(\tau(0)) = \mathbf{g}_{\text{pick}},\ f_k(\tau(T)) = \mathbf{g}_{\text{place}} \\ & \tau(t) \in \mathcal{C}_{\text{free}} \quad \forall t \in [0, T] \\ & \tau(t), \dot{\tau}(t), \ddot{\tau}(t) \in \text{joint limits} \quad \forall t \in [0, T] \\ & f_a(\tau(t)) \in \mathcal{A} \end{aligned}$$

where $T(\tau)$ is the trajectory's duration in seconds (and referenced without parameters equivalently), and $\mathcal{A}$ is the set of problem-specific end-effector acceleration constraints, such as keeping open-top container contents or avoiding excessive shock.

3.4 Method

To compute a high-speed motion that remains within end-effector acceleration constraints, we first discretize the problem, then formulate a non-convex optimization, and finally solve the optimization using a sequence of sequential quadratic programs based on TrajOpt [5] and GOMP [9].

3.4.1 GOMP Background

GOMP-FIT is built on Grasp-Optimized Motion Planning (GOMP). GOMP computes a time-optimized obstacle-avoiding trajectory that incorporates a degree of freedom around pick and place points. This section reviews the GOMP formulation and optimization process.

Trajectory Discretization

To facilitate solving the GOMP-FIT optimization problem, we first formulate a discretization of the trajectory. The discretization serves a second purpose—all industrial robots operate with a fixed control frequency, and can follow trajectories at only that rate or an integer multiple of it. GOMP and GOMP-FIT first define a fixed number of waypoints $0, 1, \ldots, H$, each separated by a fixed time step t_{step}. The time step should be an integer multiple of the robot's control frequency, and H should be sufficient to allow the problem to be solved (more details in Sec. 3.4.1). Each waypoint $\mathbf{x}_t$ includes a configuration $\mathbf{q}_t$ and first and second derivatives $\dot{\mathbf{q}}_t$ and acceleration $\ddot{\mathbf{q}}_t$.

To approximate $\dot{\mathbf{q}}_t$ and $\ddot{\mathbf{q}}_t$, we have the optimization process enforce a *dynamics* constraint between each waypoint in the following form:

$$\mathbf{q}_{t+1} = \mathbf{q}_t + \dot{\mathbf{q}}_t t_{\text{step}} + \left(\frac{1}{3}\ddot{\mathbf{q}}_t + \frac{1}{6}\ddot{\mathbf{q}}_{t+1}\right) t_{\text{step}}^2 \tag{3.4.1}$$

$$\dot{\mathbf{q}}_{t+1} = \dot{\mathbf{q}}_t + \frac{1}{2}\left(\ddot{\mathbf{q}}_t + \ddot{\mathbf{q}}_{t+1}\right) t_{\text{step}}. \tag{3.4.2}$$

This discretization comes from integrating a linear jerk between waypoints.

Sequential Convex Optimization

To solve the discretized problem, we formulate and solve a sequential quadratic program of the following form:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^T P \mathbf{x} + \mathbf{p}^T \mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x} \leqslant \mathbf{b}, \end{aligned}$$

where P is a positive semi-definite matrix for the quadratic costs, $\mathbf{p}$ is a linear cost vector, and the matrix A and vector $\mathbf{b}$ define the linear constraints.

We construct A and $\mathbf{b}$ to include the *convex* constraints: (a) the dynamics from equations 3.4.1 and 3.4.2; (b) the actuation limits, e.g., box bounds of configuration, velocity, and acceleration; and (c) jerk limits from the finite difference of accelerations.

To handle *non-convex* constraints from (i) grasp and placement configurations and degrees of freedom; (ii) obstacle avoidance; and (iii) end-effector acceleration limits, we add linearizations of the constraints to A and $\mathbf{b}$ with slack variables constrained to be positive.

As with GOMP, we establish the trust region around the configurations of the waypoints, and leave the other optimization variables (e.g., $\dot{\mathbf{q}}_t$ and $\ddot{\mathbf{q}}_t$) otherwise untouched.

Linearization of Non-Convex Constraints

To enable obstacle avoidance, grasp-optimization, and now acceleration constraints, GOMP and GOMP-FIT formulates a (non-convex) constraint of the form $g(\mathbf{x}) \leqslant c$, and linearizes

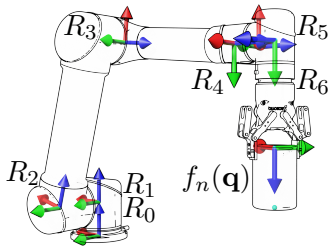


Figure 3.2: **Frames and parameters used in GOMP-FIT computation**. Each set of arrows is a coordinate frame associated with a joint. The recursive Newton-Euler algorithms computes angular velocities and their derivatives, and linear accelerations at each frame R. The translation between frames r. At the end-effector, $f_n(\mathbf{q})$ is the normal we align to inertial forces.

the function around the current iterate $\mathbf{x}^{(k)}$ as:

$$J_g \mathbf{x}^{(k+1)} \leqslant J_g \mathbf{x}^{(k)} - g(\mathbf{x}^{(k)}) - c,$$

where J_g is the Jacobian evaluated at $\mathbf{x}^{(k)}$, and $\mathbf{x}^{(k+1)}$ is the optimization variable for the next iteration. These coefficients are then added to A and $\mathbf{b}$.

Time Optimization

To minimize trajectory time, GOMP and GOMP-FIT repeatedly solve the optimization with a shrinking horizon H until the SQP returns failure, and uses trajectory from the minimum horizon that succeeded.

3.4.2 GOMP-FIT End-Effector Acceleration Constraints

To constrain inertial effects, we first compute the acceleration at the end-point due to gravity, Euler, Coriolis, and centrifugal forces; and then form the constraint. To compute the effect of all joint motions on the end-effector acceleration, we employ the forward pass of the Recursive Newton Euler (RNE) method [78], summarized in Alg. 4. RNE requires the rotation ${}^{i-1}R_i$ between joints $i-1$ and i; the angular velocity vector ω and its derivative $\dot{\omega}$; and the translation between successive joints $\mathbf{r}_i$. We do not use the backward pass of the RNE, as the forward pass is sufficient.

We compute the acceleration at the end-effector $\mathbf{a}_{\text{ee}} = \mathbf{g} - \text{RNE}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, where $\mathbf{g}$ is the gravity vector. We define $f_n(\mathbf{q}) : \mathcal{C} \to \mathbb{R}^3$ as the forward kinematic to the grasp vector (Fig. 3.1)—i.e., normal of a container's surface. We then formulate the non-convex constraints for object transport. For notation convenience, we omit the subscript t, but these constraints apply to all waypoints.

Algorithm 4 Recursive Newton Euler

Input: Configuration and its derivatives $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$
/* Notation: $\mathbf{q}_{[\mathrm{i}]} \in \mathbb{R}$ is i-th joint angle */
$\omega_0, \dot{\omega}_0, \mathbf{a}_0 \leftarrow 0, 0, 0$
for $\mathrm{i} = 1, 2, \ldots,$ number of joints **do**
  $\omega_\mathrm{i} \leftarrow {}^{\mathrm{i}-1}R_\mathrm{i}^T(\omega_{\mathrm{i}-1} + \dot{\mathbf{q}}_{[\mathrm{i}]} z_\mathrm{i})$
  $\dot{\omega}_\mathrm{i} \leftarrow {}^{\mathrm{i}-1}R_\mathrm{i}^T(\dot{\omega}_{\mathrm{i}-1} + \ddot{\mathbf{q}}_{[\mathrm{i}]} z_\mathrm{i} + \dot{\mathbf{q}}_{[\mathrm{i}]} \omega_{\mathrm{i}-1} \times z_\mathrm{i})$
  $\mathbf{a}_\mathrm{i} \leftarrow {}^{\mathrm{i}-1}R_\mathrm{i}^T \mathbf{a}_{\mathrm{i}-1} + \dot{\omega}_\mathrm{i} \times \mathbf{r}_\mathrm{i} + \omega_\mathrm{i} \times (\omega_\mathrm{i} \times \mathbf{r}_\mathrm{i})$
return ${}^0R_N \mathbf{a}_N$

Open-Top Containers

To transport open-top containers, we constrain trajectories to keep the acceleration at the end-effector within a user-defined threshold angle $\theta_{\max}$ of the container normal (Fig. 3.1). This constraint has the form:

$$\cos^{-1}(\ddot{\mathbf{a}}_{\mathrm{ee}} \cdot f_n(\mathbf{q}) / \|\ddot{\mathbf{a}}_{\mathrm{ee}}\|) \leqslant \theta_{\max}. \tag{3.4.3}$$

This can equivalently be expressed as

$$\mathbf{a}_{\mathrm{ee}} \cdot f_n(\mathbf{q}) / \|\mathbf{a}_{\mathrm{ee}}\| \geqslant \cos\theta_{\max}.$$

Fragile Objects

When transporting fragile objects, we constrain trajectories to avoid exceeding a threshold acceleration $a_{\max}$ in any direction. This constraint is:

$$\|\mathbf{a}_{\mathrm{ee}}\| \leqslant a_{\max}. \tag{3.4.4}$$

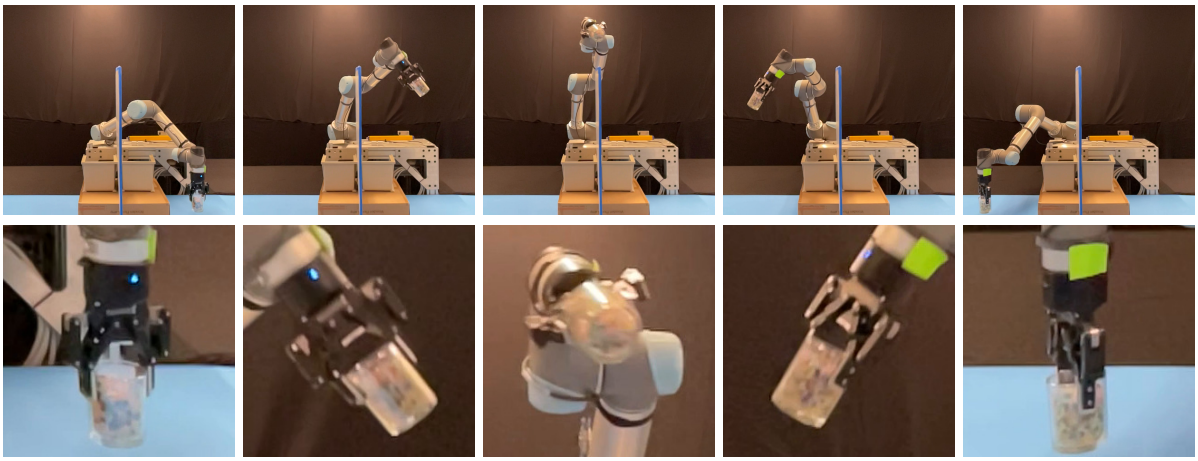
Combining Constraints

When transporting fragile objects in open-top containers, it is possible to include both Eqn. (3.4.3) and Eqn. (3.4.4) as independent constraints.

3.4.3 Minimization Objective

As the constraints on end-effector accelerations and their linearizations depend on the velocity and acceleration of the configuration, we find that adding a minimization objective based on the sum-of-squared velocities or accelerations can work against the linearizations. These objectives tend to "pull" trajectories against the constraints. We thus minimize the sum-of-squared jerks of the trajectory using a finite difference between waypoints:

$$\frac{1}{t_{\mathrm{step}}^2} \sum_{t=0}^{H-1} (\ddot{\mathbf{q}}_{t+1} - \ddot{\mathbf{q}}_t)^2.$$



(a) (b) (c) (d) (e)

Figure 3.3: **Transporting an open-top container over a barrier.** Stills (**top**) and zoom-ins (**bottom**) over the course of a fast inertial transport motion. GOMP-FIT computes a motion that transports an open-top container while avoiding a barrier 0.9 m above the floor, and keeping the end-effector accelerations aligned to a 45° to keep the contents in the container. During this motion, the robot spills 0 beads.

In practice, we find that assigning different weights to different joints benefits motion computation. Specifically, we weigh joints closer to the base higher than joints closer to the end-effector, encouraging those joints to focus more on producing a smooth overall motion with minimal jerk whereas joints near the gripper are encouraged to fulfill other problem-specific constraints such as the start goal constraint.

3.5 Experiments

We experiment with GOMP-FIT on a UR5 physical robot performing a series of tasks in which success is dependent on constraints on the end-effector acceleration. These tasks are: (1) transporting an open-top container without spilling contents, (2) transporting a fragile object without exceeding an acceleration threshold (3) transporting wine held in a wine glass without spilling, or dropping the wineglass.

For baselines, we run both GOMP and J-GOMP. This also forms an ablation in that GOMP-FIT without the acceleration constraints is J-GOMP, and J-GOMP without the jerk limits and minimization is GOMP, as GOMP instead minimizes sum-of-squared accelerations at the joints. Here, both GOMP and J-GOMP include a minor modification so that all baselines share the same dynamics constraints as GOMP-FIT. Additionally, we add baselines of GOMP(+H) and J-GOMP(+H), which are GOMP and J-GOMP but with the same minimum horizon H as computed by GOMP-FIT. These two baselines are to test whether

Tolerance	GOMP	J-GOMP	GOMP(+H)	J-GOMP(+H)	GOMP-FIT
45°	89.7%	46.4%	84.5%	37.1%	**0.0%**
30°	90.0%	48.0%	47.0%	100.0%	**0.0%**
20°	90.3%	50.0%	39.4%	100.0%	**0.0%**
15°	91.2%	54.4%	41.2%	100.0%	**0.0%**

Table 3.1: **Open-top container lost mass.** We measure the percentage of mass lost when performing the transport of an open-top container using various methods. The container starts with a consistent fill level for each tilt/tolerance angle.

Metric	GOMP	J-GOMP	GOMP-FIT 45°	GOMP-FIT 2G
IE	330.9	121.7	**73.6**	82.5
IVE	255.3	61.1	94.5	**18.3**

Table 3.2: **Fragile object transport.** We compute two metrics with respect to the end-effector acceleration vector norm: the integrated error (IE) as the sum of absolute difference from the planned acceleration norm, and the integrated violation of the acceleration constraint (IVE). The end-effector holds a RealSense D435i camera with an IMU to compute the acceleration norm during the motion.

Figure 3.4: **Fill levels used in open-top container transport experiments.** As we use beads (instead of liquids) for safety and ease of cleanup, we fill containers to just before spilling.

it is just the slower speed of the trajectory, or if the additional constraints on end-effector accelerations are necessary for successful completion of the tasks.

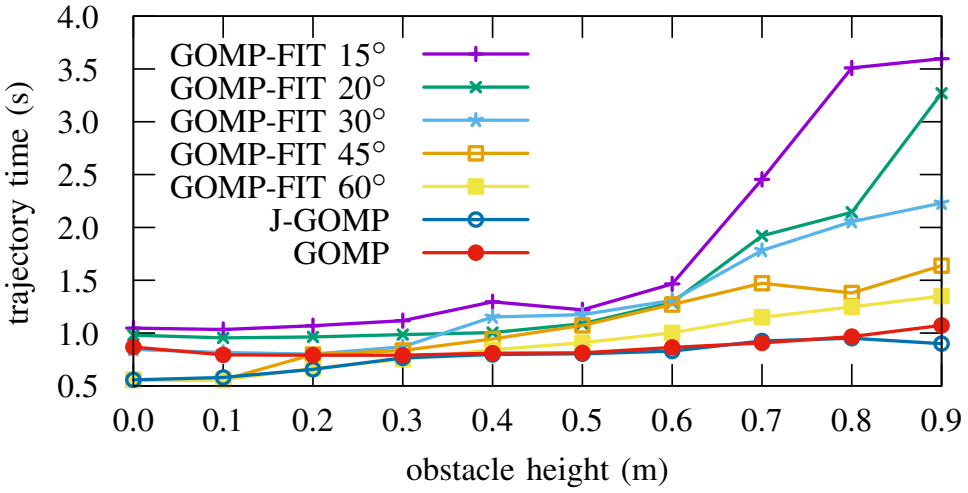


Figure 3.5: **Open-top container transport time vs obstacle height.** GOMP-FIT at various tilt thresholds and baselines compute a transport motion over a wall. Taller obstacles require longer paths and more time to clear.

3.5.1 Open-Top Container Transport

To test if GOMP-FIT can transport an open-top container without spilling the contents, we task a UR5 robot to transport glass containing beads (Fig. 3.3). We vary the fill level based on a tilt-level. We tilt the container 45°, 30°, 20°, and 15°, and fill it so that the beads just barely stay and record the mass (Fig. 3.4). We then have GOMP-FIT compute motions that include the open-top transport constraint (Eqn. 3.4.3) matching the fill level, and execute the motion. We vary this for 3 different start-goal pairs and report the results in Tab. 3.1. From the results we observe that GOMP-FIT does not drop a single bead, while all other planners do. With the (+H) ablations, we observe that running slower is not sufficient for task success, as both baselines spill contents at the same trajectory duration–some of this is attributable to the end-effector taking a longer path to match the trajectory length. Best performing of the baselines is J-GOMP, which, while consistent with prior observations [68] that jerk limits reduce spills, limiting jerk alone is insufficient for fast inertial transport.

We also compare motion time to study how much time is lost due to the open-top constraint, and report the results in Fig. 3.5. The time loss is surprisingly small, as we observe that GOMP-FIT is able to compute motions that tilt against the inertial forces to keep the contents in the container—most of the time lost is in accelerating and decelerating the object at the beginning and end of the motion. As obstacle height increases and angle tolerances tighten, the safe passage takes more time to traverse. With shorter obstacles, GOMP's lack of jerk constraints, requires reducing accelerations limits to prevent protective stops, resulting in worse performance than J-GOMP. Both J-GOMP and GOMP-FIT can run with higher acceleration limits.

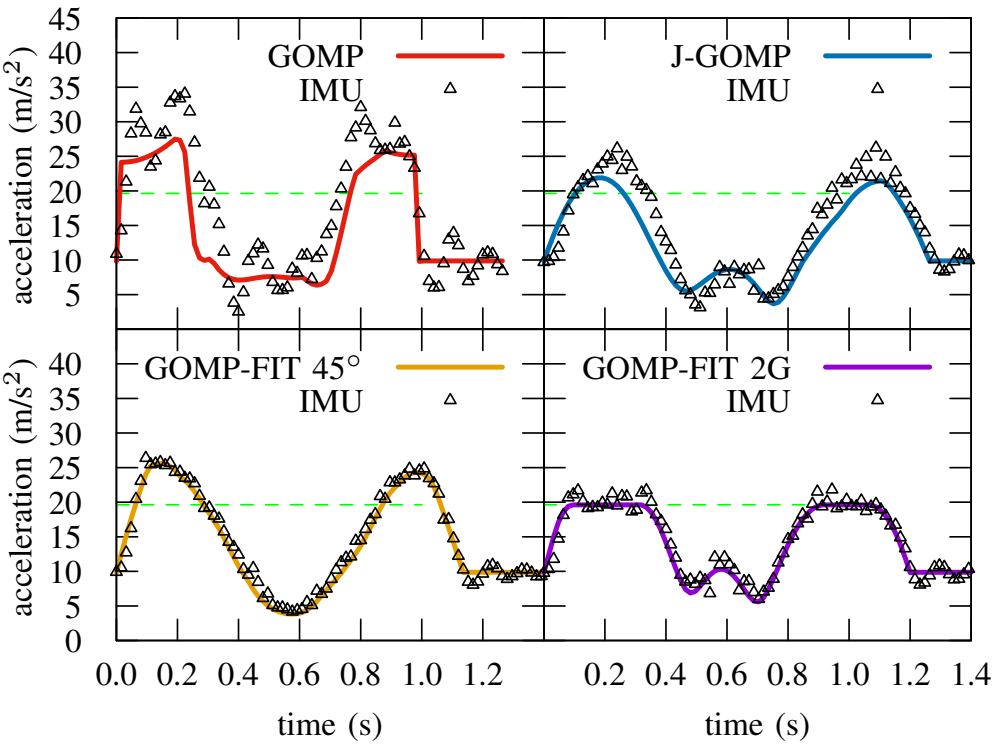


Figure 3.6: **Transported IMU readings vs predicted acceleration magnitude** With an end-effector acceleration limit of 2G (green dashed line), GOMP-FIT 2G produces trajectories with IMU readings within a small error, whereas GOMP and J-GOMP have accelerations far exceeding the limit. GOMP-FIT 45° (without acceleration limits) track well with predictions but exceed the limit.

3.5.2 Fragile Object Transport

Transporting a fragile object may require limiting the end-effector acceleration to avoid damaging or dropping the object. To test if GOMP-FIT can reliably limit the end-effector acceleration, we attach a RealSense D435i camera with an Inertial Measurement Unit (IMU) to the UR5 robot end-effector and record the acceleration norm along the executed trajectories. We compute two metrics: the integrated error (IE) as the sum of absolute difference between the end-effector acceleration norm and the planned acceleration norm:

$$\sum \left(\| A_{\text{IMU}} - A_{\text{traj}} \| \right),$$

and the integrated violation error (IVE) as the difference between the end-effector acceleration norm and the acceleration limit:

$$\sum \max \left(0, \| A_{\text{IMU}} \| - \| A_{\max} \| \right)$$

In experiments we set $A_{\max} = 2\text{G} = 19.74\,\text{m/s}^2$ and compare GOMP-FIT with a 2G acceleration limit, with the baselines GOMP, J-GOMP, and GOMP-FIT with a 45°-alignment constraint. The results are summarized in Tab. 3.2 and a qualitative result is presented in Fig. 3.6. From the measures, we observe that unlike baselines, GOMP-FIT is able to accurately limit accelerations, subject to errors inherent to the sensor and underlying controller.

3.5.3 Filled Wineglass Transport

To test GOMP-FIT for fast inertial transport of a liquid in an open-top container, we compute and execute a motion that transports a glass of wine over a barrier. Due to results of the experiments with the cup containing beads, and the damage (and stains) that would result from spills, we do not attempt any baseline method. This experiment is mostly qualitative, as we do not wish to probe the limits of the method in our current lab setup—however, during these experiments, the robot did not spill a drop. A preview of the accompanying video is in the top-right of Fig. 3.1.

3.6 Discussion and Future Work

We present GOMP-FIT, an optimizing motion planner that solves the fast inertial transport problem of transporting open-top containers, fragile objects, or combinations thereof around obstacles while not spilling, damaging, or losing a grasp due to inertial forces. GOMP-FIT uses a sequential quadratic program to optimize a discretized trajectory with the first and second derivatives of its configuration, and incorporates non-linear constraints for obstacle avoidance, grasp optimization, and on accelerations at the end-effector. In experiments on a physical robot, GOMP-FIT was able to reliably and safely transport objects with only minor slowdown compared to motions from fast planners that did not safely transport objects. Slowed motions of the baseline planners fared little better, suggesting that slowing motions down is not sufficient to achieve reliable inertial transport.

In future work, we will explore addressing the long computation time associated by tuning the optimizer and using deep learning to warm start the computation [50]. The proposed method relies on conservative approximations for joint acceleration limits based on torque limits and transported mass—we hope to tighten up the acceleration limits by transitioning to torque-based limits. In the current formulation, these would be non-convex constraints that would further slow the computation but result in faster motions.

Chapter 4

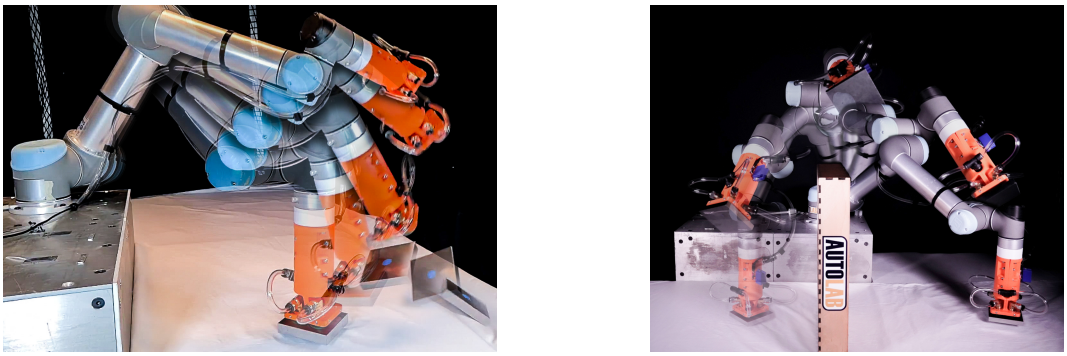
GOMP-ST: Making Fast Object Transport with Suction Grasping Reliable

In contrast to the parallel-jaw grasping discussed in the previous chapter, suction grasping is widely used in industry, but maintaining a suction grasp throughout a high-speed motion requires balancing suction forces against inertial forces while the suction cups deform under strain. GOMP-ST combines self supervised learning with optimization to decrease transport time while increasing reliability and avoiding suction cup failure.

4.1 Introduction

Vacuum suction cup grasping, due to its ability to quickly hold and release a large variety of objects, is a common grasping modality for robots in industrial settings such as warehouses and logistics centers. With the recent rise in demand for robot pick-and-place operations, the speed of object transport is critical. However, suction grasps can fail if the object is transported too quickly. Determining the conditions where suction grasps fail is non-trivial due to the difficult-to-model deformations of suction cups under stress. Existing analytic models make simplifying assumptions, such as rigid suction cups [60] or quasi-static physics [71, 79, 80]. An alternative is to heuristically slow motions when objects are held in suction grasps.

In prior work, the Grasp-Optimized Motion Planner (GOMP [9]) leveraged an unconstrained degree of freedom (DoF) around the grasp axis to optimize pick-and-place motions for parallel jaw grippers. Grasp-Optimized Motion Planning for Fast Inertial Transport (GOMP-FIT [81]) computes time-optimized motions while taking into account end-effector and object acceleration constraints to reduce product damage and spills. However, when executed on robots with suction grippers, the resulting motions can fail due to suction cups detaching.



(a) A fast motion resulting in suction failure (b) A successful transport over obstacle

Figure 4.1: **High-speed motions can cause failure of suction cups.** These multiple-exposure images show motions computed by the time-optimizing motion planners (a) J-GOMP and (b) GOMP-ST. The fast trajectory computed by J-GOMP and inertial forces cause the suction grasp to fail and the grasped block to fall away and to the right. We propose GOMP-ST, an algorithm that incorporates a learned acceleration constraint into a time-optimizing motion planner to avoid such failures.

To address this problem, we propose the Grasp Optimized Motion Planning for Suction Transport (GOMP-ST), an algorithm that computes time-optimized motions by integrating a *learned* suction grasp loss constraint. GOMP-ST first tries varied rapid lifts of a given object with suction grasps to find motions that cause suction failures. It then learns a model from the data based on a history, or sequence, of end-effector accelerations to define a constraint function. At run time, the optimization treats the learned model as a non-linear constraint on the motion by linearizing it and computing a first-order approximation based on its Jacobian. In our implementation, we use a neural network to learn the suction model and use Autograd to obtain the Jacobian.

In experiments with steel rectangular blocks, we learn a model of suction failure on a physical UR5 with a 4-cup vacuum gripper. We then apply the learned model to transport 4 objects of varying mass held in suction between multiple different start and goal pairs and around obstacles. We compare GOMP-ST to GOMP, GOMP-FIT with an analytic model of suction, and ablations of GOMP-ST. We find that GOMP-ST can achieve a near 100 % success rate, for motions that are 16 to 58 % faster.

This chapter provides the following contributions: (1) A novel algorithm, GOMP-ST, Grasp Optimized Motion Planning for Suction Transport, based on: (a) Formulation of a learnable acceleration constraint for suction cup transport; (b) An efficient method for learning the constraint via boundary searching and data augmentation; (c) Integration of the learned constraint into an optimizing motion planner; and (2) Data from experiments with a physical robot comparing GOMP-ST to baselines.

4.2 Related Work

Robot motion planning aims to find safe and efficient robot motions from a start to a goal configuration that avoid obstacles. Sampling-based motion planners, such as PRM [2] and bi-directional RRT [4] have variants that are probabilistically-complete and asymptotically-optimal. Optimization-based motion planners, such as TrajOpt [5], STOMP [6], CHOMP [7], KOMO [8], and ITOMP [82] can compute optimized trajectories iteratively improving paths or interleaving with sampling-based planners [25].

Grasp-Optimized Motion Planning (GOMP) [9] and Deep-Jerk GOMP (DJ-GOMP) [50] leverage a degree-of-freedom in the grasp pose while integrating dynamic and kinematic constraints in a sequential quadratic program that iteratively computes time-optimized pick-and-place motions. GOMP for Fast Inertial Transport (GOMP-FIT) [81] incorporates end-effector acceleration constraints for parallel-jaw grippers by employing the forward pass of the Recursive Newton Euler (RNE) method [78]. In contrast to prior work, GOMP-ST learns a suction-cup constraint to avoid suction grasp failure.

Optimization using constraints based on empirical models is gaining traction. Maragno et al. [83] integrate learned constraints into a mixed-integer optimization using trust regions defined by the convex hull of the training data. In contrast, we propose using domain knowledge to perform data augmentation in a continuous optimization. Kudła et al. [84] propose a learning a decisions tree of constraints and a transform appropriate for mixed integer linear programming. While the decision tree provides flexible conditions for constraints, we instead propose exploiting the nature of the trajectory optimization by regressing on inputs containing the last h time steps. Bartolini et al. [85] and later Lombardi et al. [86] show that neural networks can learn constraints based on difficult-to-model phenomena be integrated into constrained combinatorial optimizations. We employ neural network constraints in continuous non-convex optimization. De Raedt et al. [87] provide a recent survey of constraint acquisition, and Fajemisin et al. [88] provide a recent survey of optimization with constraint learning. These surveys provide a wealth of ideas that could be extended to apply additional constraint learning and learned constraints to optimizing motion planning.

Suction grasping is widely used in industrial settings. Due to the increasing interest in suction grasping for pick-and-place tasks, recent models focus on computing a robust suction grasp. Dex-Net 3.0 [79] introduced a novel suction contact model that quantifies seal formation using a quasi-static spring system, along with a robust version of the model under random disturbing wrenches and perturbations in object pose, gripper pose, and friction. Huh et al. [80] use a learned model and a novel multi-chamber suction cup design to detect failures in the suction seal early to avoid grasp failures. However, these works assume quasi-static physics, an assumption that does not hold for high inertial forces.

Most closely related to this chapter is the work by Pham and Pham [71] which proposes a suction-cup model and identify a contact stability constraint and a pipeline to parameterize time-optimal geometric paths satisfying the constraint. GOMP-ST differs in a few ways. First, their method plans a motion using a sampling-based planner, then time-parameterizes the motion, whereas GOMP-ST integrates planning and time-parameterization into a single

optimization, allowing to explore alternate paths with potentially better timing. Second, their method uses an analytic model that does not include suction cup deformation, whereas GOMP-ST learns a constraint based on suction grasps failures through experimentation.

Dynamic manipulation exploits forces due to accelerations, along with kinematics, static, and quasi-static forces to achieve a task [89]. Lynch and Mason [53] leverage centrifugal and Coriolis forces to allow low degree-of-freedom robots to control objects with more degrees-of-freedom. Lynch and Mason [54] also directly integrate constraints in a sequential quadratic problem to plan robot trajectories that achieve a dynamic task, such as snatching an object, throwing, and rolling, via coupling forces through the non-prehensile contact in an obstacle-free environment. Srinivasa et al. [55] address the problem of rolling a block resting on a flat palm by employing constraints on accelerations at the contact point. Mucchiani and Yim [57] propose a grasping approach that utilizes object inertia for sweeping an object at rest to a goal position by leveraging accelerations and torques from the path as stabilizing forces in the passive end-effector. In this work, we employ constraints on the end-effector accelerations to perform efficient dynamic object transport around obstacles using a suction gripper.

One promising related avenue of research investigates motion planning and optimization that integrates forces present at the end-effector. Hauser [65] investigates including contact forces during the optimization, and Luo and Hauser [66] extend this to include a learned confidence into an optimization. In a similar manner, Bernheisel and Lynch [67] and Acharya et al. [69] both explore different ways to address the *waiter's problem* which requires generating motions of a tray to perform non-prehensile balancing of objects. In contrast to these lines of work, we focus on learning and integrating a constraint to maintain suction contact as a differentiable function, instead of learning the parameters of a model and integrating it into a plan.

When the dynamics are unknown or the existing models are insufficiently accurate, a promising approach is to leverage data-driven methods. Zeng et al. [58] propose TossingBot that learns parameters of a pre-defined dynamic motion to toss objects into target bins using parallel-jaw end-effectors. Wang et al. [59] propose SwingBot, that uses tactile feedback to learn how to dynamically swing up novel objects. In Robots of the Lost Arc [**zhang2021rotla**], a robot computes high-speed motions to induce fixed-end cables to swing over distant obstacles, knock down target objects, and weave between obstacles. Ha and Song [90] propose FlingBot, a self-supervised learning framework to learn dynamic flinging actions for cloth unfolding. Lim et al. [91], uses simulation and physical data to train a model for swinging an object to hit a target. In this work, we integrate a learned a constraint in the solver allowing it to compute fast motions while maintaining the suction seal.

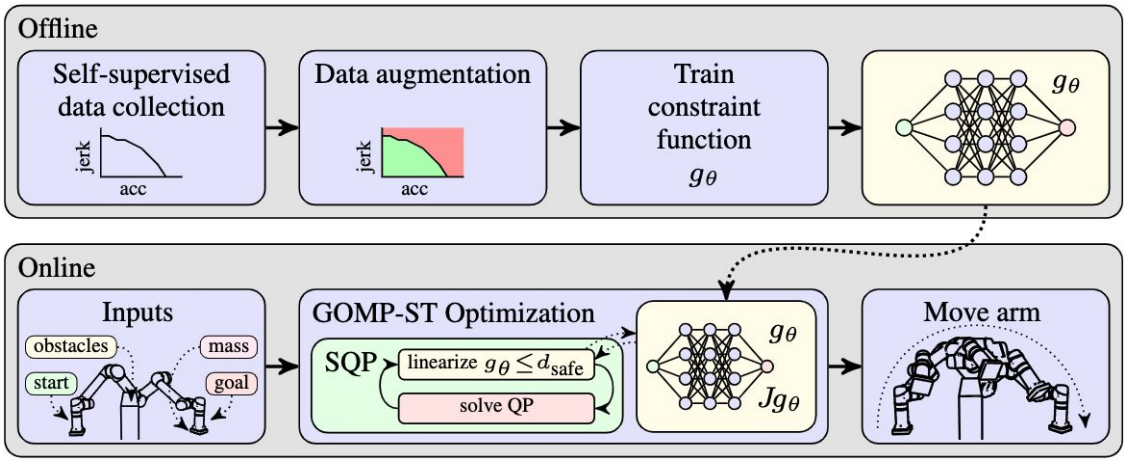


Figure 4.2: **GOMP-ST pipeline.** In an offline process (**top**), GOMP-ST performs self-supervised learning of a constraint function. It first repeatedly grasps objects of known mass to perform a boundary search on a motion profile parameterized by jerk and acceleration limits. Then, during data augmentation, it labels slower motions as grasp successes, and faster motions as grasp failures. Finally, it trains a neural network constraint function g_θ. In the online process (**bottom**), GOMP-ST computes a motion plan for a given problem. The SQP solver repeatedly linearizes the learned suction constraint using a user-specified threshold $d_{\text{safe}} \in [0, 1]$ and the output and Jacobian (via autograd) of the trained neural network.

4.3 Problem Statement

Let $\mathbf{q} \in \mathcal{C}$ be the complete specification of the degrees of freedom for a robot, where $\mathcal{C}$ is the set of all possible configurations. Let $\mathcal{O}$ be the set of obstacles, and $\mathcal{C}_{\text{obs}} \subseteq \mathcal{C}$ be the set of configurations in collision with $\mathcal{O}$. Let $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$ be the set of configurations that are not in collision. Let $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ be the first and second derivatives of the configuration. Let $\mathbf{a}_{\text{ee}} \in \mathbb{R}^3$ be the linear acceleration of the end effector with one or more suction cups holding a known object b, and $\hat{\mathbf{n}} \in \mathbb{R}^3$ be the suction cup normal, as shown in Fig. 4.4

Given a start $\mathbf{q}_0$ and goal $\mathbf{q}_{\text{goal}}$ configuration, the objective of GOMP-ST is to compute a trajectory $\tau = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_H)$, where $\mathbf{x}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t) \in \mathcal{X}$ is the state of the robot at time t, $\mathcal{X}$ is the set of states, such that $\mathbf{q}_t \in \mathcal{C}_{\text{free}}\ \forall t \in [0, H]$, and $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$ are within the box-bounded dynamic limits of the robot, and $\mathbf{q}_H = \mathbf{q}_{\text{goal}}$. The object b will remain attached throughout the motion. As with GOMP, start and goal configurations may also be expressed via forward kinematics and bounds on degrees of freedom.

4.4 Method

GOMP-ST learns a suction cup acceleration constraint via a sequence of physical experiments, then integrates it with a time-optimizing motion planner. See Fig. 4.2 for an overview.

This section starts with background on the prior work, then describes how to learn and integrate the constraint into the motion planner.

4.4.1 Background: GOMP-FIT

The GOMP-FIT [81] algorithm formulates a fast-inertial-transport motion planning as an optimization problem and solves it with sequential quadratic program (SQP) trust-region-based solver. It first discretizes the trajectory into a sequence of $H+1$ waypoints $(\mathbf{x}_0, \dots \mathbf{x}_H)$ that are each separated by a fixed time interval t_{step}. Each waypoint $\mathbf{x}_t$ includes the configuration and its first and second derivatives $\mathbf{x}_t = (\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t)$. The outer loop shrinks H to find a minimum time trajectory. The inner loop solves an SQP where the optimization objective minimizes the sum-of-squared accelerations, the linear constraints keep the motion within the kinematic and dynamics limits of the robot, and the non-linear constraints avoid obstacles and limit linear accelerations experienced at the end-effector. To limit shock, it constrains the magnitude of end-effector acceleration to be below a threshold a_{safe}. To avoid spills, it constrains the angle between the container normal and the acceleration to be below a spill threshold θ_{spill}. The optimization is

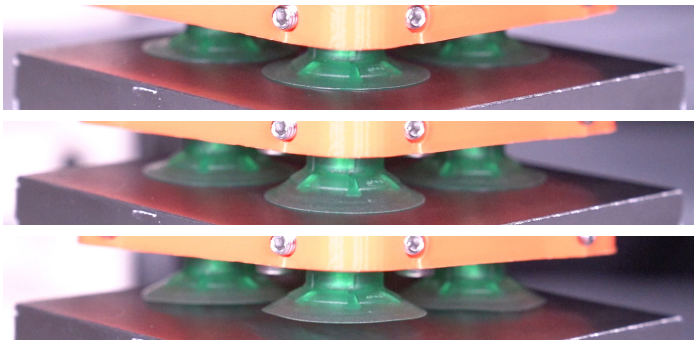
$$
\begin{aligned}
\min_{\mathbf{x}_{[0..H]}} \quad & \frac{1}{2}\sum_{t=0}^{H} \ddot{\mathbf{q}}_t \\
\text{s.t.} \quad & \mathbf{x}_{\min} \leqslant \mathbf{x}_t \leqslant \mathbf{x}_{\max} \quad \forall t \in [0..H] \\
& \mathbf{q}_{t+1} = \mathbf{q}_t + \dot{\mathbf{q}}_t t_{\text{step}} + \frac{1}{2}\ddot{\mathbf{q}}_t t_{\text{step}}^2 \quad \forall t \in [0..H) \\
& \dot{\mathbf{q}}_{t+1} = \dot{\mathbf{q}}_t + \ddot{\mathbf{q}}_t t_{\text{step}} \quad \forall t \in [0..H) \\
& f_{\mathcal{O}}(\mathbf{q}) \geqslant 0 \quad \forall t \in [0..H] \\
& \cos^{-1}(f_a(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \cdot f_{\text{n}}) \leqslant \theta_{\text{spill}} \quad \forall t \in [0..H] \\
& \|f_a(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\| \leqslant a_{\text{safe}} \quad \forall t \in [0..H],
\end{aligned}
$$

where $f_{\mathcal{O}} : \mathcal{C} \to \mathbb{R}$ is the signed distance from robot to set $\mathcal{O}$ (thus implementing the constraint $\mathbf{q}_t \in \mathcal{C}_{\text{free}}$) and $f_a : (\mathcal{C})^3 \to \mathbb{R}^3$ is the linear acceleration at the end-effector computed using the Recursive Newton-Euler (RNE) algorithm. Additionally, GOMP-FIT optionally integrates constraints to optimize the grasp angle and location.

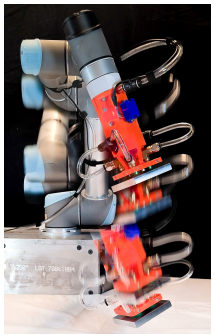
The SQP solver repeatedly linearizes the non-linear constraints (collision, end-effector acceleration, and grasp) to form a quadratic program (QP), and solves the QP, accepting solutions that improve the trajectory. In this optimization, when solving for the $(k+1)$ iterate, a non-linear constraint of the form $g(\mathbf{x}) \leqslant y$ is linearized around the current iterate $\mathbf{x}^{(k)}$ via a first-order approximation using its Jacobian:

$$J\mathbf{x}^{(k+1)} \leqslant y - g(\mathbf{x}^{(k)}) + J\mathbf{x}^{(k)}.$$

GOMP-FIT optimizes the trajectory time by repeatedly solving the SQP with a shrinking horizon H, warm-starting each subsequent SQP solve with an interpolation of the solution



(a) Suction cup deformation before and during suction break.



(b) Vertical lift motion

Figure 4.3: **Suction cup deformation observed with data collection.** (a) These frames from a slow-motion video show the deformation of suction cups as the gripper is rapidly pulled upward. At the beginning of the motion ((a) **top**), the suction cups compress against the grasped surface. As the gripper lifts and starts to break from suction ((a) **middle**), the suction cups deform but still maintain a seal. Continuing to pull away results in a suction grasp failure ((a) **bottom**). (b) During data collection, the robot lifts the mass with a vertical motion while grasping at an angle.

from the prior horizon. The optimization terminates the smallest H the solver detects as feasible. However, when executed on robots with suction grippers, the resulting motions can lead to suction failures.

4.4.2 Learned constraints in the SQP

Maintaining a suction grasp throughout a high-speed motion requires balancing suction forces against inertial forces while the suction cups deform under strain. GOMP-ST defines a series of physical experiments to learn a constraint from real-world data. We model the learned constraint as a function $g_\theta : (\mathcal{X})^h \to [0, 1]$ parameterized by θ, where $(\mathcal{X})^h$ is history of h dynamic states, and a value of 0 indicates the object is held, while a value of 1 indicates a failure. (Notationally, we use h to indicate a history of states, and H to indicate the total trajectory length.)

We integrate this function into the optimization as a non-linear constraint:

$$g_\theta(\cdot) \leqslant d_{\text{safe}},$$

where the argument is a portion of the state from of the trajectory being optimized, and $d_{\text{safe}} \in [0, 1)$ is a tunable failure threshold. To linearize this constraint for the SQP, we use the automatic gradients (Autograd) provided by a neural-network package [92] to compute the Jacobian.

Using slow-motion video capture, we observe that suction cups deform before suction failure (Fig. 4.3), and thus failures are not an instantaneous response to a change in end-effector state. With this observation, we propose that g_θ should be a function of a sequence

(or history) of states of length h. The value of h depends on the geometry and material of the suction cup. In a series of experiments we set h to be long enough to capture the time between deformation start and suction failure that we observe from slow-motion video playback (> 0.1 seconds). Further, to translate the optimization variables from the state of the robot to the state of the end-effector, we utilize RNE function f_a for each state in the history. Thus we formulate the full constraint as:

$$g_\theta(f_a(\mathbf{x}_{t-h+1}), \ldots, f_a(\mathbf{x}_{t-1}), f_a(\mathbf{x}_t)) \leqslant d_{\text{safe}} \quad \forall t \in [1..H).$$

When $\mathbf{x}$ has a negative subscript (i.e., it refers to a state before the start of planning), we replace it with $\mathbf{x}_0$.

4.4.3 Self-supervised data collection and training

To learn g_θ, GOMP-ST implements a self-supervised pipeline that defines a series of experiments to lift objects of known mass and collect data. We attach a pressure sensor to the tube connected to the suction cups, similar to Huh et al. [80]. To minimize delay between pressure changes and pressure readings, we place the sensor close to the suction cups. During each lift, the pipeline records the joint state and pressure sensor readings over time.

To isolate gravity during data collection, the system always lifts against gravity while varying the angle of the suction normal (see Fig. 4.3). As convention, 0° is a top-down grasp and 90° is a grasp in which the suction normal is perpendicular to gravity.

The pipeline computes each lift motion using Ruckig [93] between two points in end-effector (Cartesian) coordinates, and uses an inverse kinematics solver to translate into joint configurations. Ruckig computes straight-line time-optimal motions subject to velocity, acceleration, and jerk limits. During data collection, GOMP-ST varies the motion profile by changing the maximum acceleration $a_{\max}$ and maximum jerk $j_{\max}$ parameters of Ruckig.

The pipeline defines a discretized grid with $a_{\max}$ and $j_{\max}$ axes to fill with values 0 or 1 labels. To reduce data collection time, GOMP-ST performs a boundary search on the motion profiles. It starts with a fixed lower value of $a_{\max}$ and increases $j_{\max}$ until it observes a change in the pressure measurements indicating a suction grasp failure. Afterwards, it iteratively decreases $j_{\max}$ or increases $a_{\max}$ so that it is always exploring the above and below the continuous boundary at which suction fails.

After a suction failure, the automated data collection pipeline takes a top-down image of the scene to find and re-grasp the target object. After confirming the grasp using the pressure measurements, GOMP-ST moves the object to a consistent starting pose before performing the next lift.

The pipeline then trains a multi-layer perceptron with exponential linear units (ELU) activations using the joint and pressure data it collected. We choose ELU for its continuous gradients. Training details are in the experiments section. The pipeline scans the recorded pressure data to find the time at which suction pressure is lost. It then tracks a tunable number of steps h back to create a labeled data point containing h accelerations that led to

the suction failure (labelled as 1). All sequences of h accelerations prior to the suction failure, or in records without suction failures, the pipeline labels as 0. GOMP-ST further perform data augmentation by scaling accelerations leading to failures by $1 + \epsilon$, and non-failures by $1 - \epsilon$, for small positive values of ϵ.

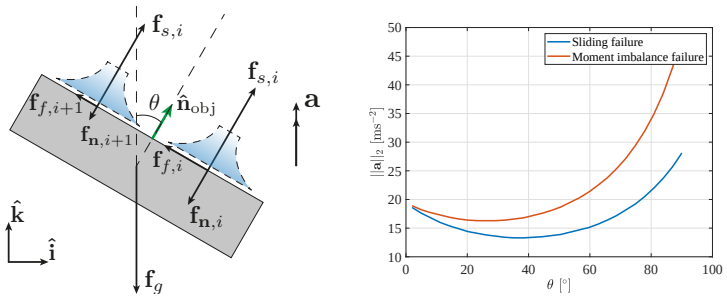
4.4.4 Analytic model of suction-cup failure for GOMP-FIT baseline

As a baseline to the learned model, we compare GOMP-ST to GOMP-FIT where its constraint on the magnitude and direction of the inertial acceleration vector is provided by an analytic model. In previous work, Kolluru et al. [94] and Stuart et al. [95] proposed analytic models for rigid suction cups, but were limited to either single suction cups or symmetrical systems. Our analysis is more closely related to the more general method used by Valencia et al. [96] and Pham and Pham [71] which generalises over multiple suction cups and asymmetric loads, though still with the assumption of rigid bodies. We also choose the rigid-body assumption as opposed to for example the spring model suggested by Mahler et al. [79], since a spring model of suction cups where quasi-static equilibrium is not assumed would require knowledge about the state of the springs. Alternatively, if quasi-static equilibrium is assumed, the state of the springs may be estimated, but the purpose of the additional complexity is defeated from a motion planning perspective as the resulting constraint would be equivalent to that of a rigid-body model.

In this work, the analytic model makes the following assumptions:

1. There are quasi-static conditions in inertial frame, enabling equilibrium analysis.
2. The suction cups are rigid and modelled as point contacts.
3. The transported object is a rigid rectangular cuboid with uniform mass distribution such that its center of mass corresponds to its centroid.
4. There are no air leaks between the suction cups and flat grasping surface, which results in a static and equal suction force across all suction cups.

Consider the free-body diagram in the inertial frame of the object shown in Fig. 4.4, where $\mathbf{f}_{s,i}$ are the suction forces, $\mathbf{f}_{\mathbf{n},i}$ are the contact forces between the suction cups and grasped object, $\mathbf{f}_{f,i}$ are the friction forces, $\mathbf{f}_g$ is the gravitational force, and $\mathbf{a}$ is the acceleration from a balance of forces. $\hat{\mathbf{n}}_{\text{obj}}$ is the unit length (denoted by the hat) normal defining the grasping plane, and $\theta = \cos^{-1}\left[(\mathbf{a} \cdot \hat{\mathbf{n}}_{\text{obj}})/(||\mathbf{a}||_2||\hat{\mathbf{n}}_{\text{obj}}||_2)\right]$ is the angle between the inertial acceleration vector and suction force normal. In the idealised condition, $\mathbf{f}_{s,i} = (p_{\text{atm}} - p_v)A$, where $p_{\text{atm}} - p_v$ is the difference between atmospheric pressure p_{atm} and applied vacuum pressure p_v, and A is the effective area of grip for a single suction cup. Furthermore, under the assumption of dry Coulomb friction, $\mathbf{f}_{f,i} \leqslant \mu \mathbf{f}_{\mathbf{n},i}$, where μ is the static coefficient of friction between the grasped object and suction cups. Finally, $\mathbf{f}_g = m\mathbf{g}$, where m is the object mass, $\mathbf{g}$ is the gravitational acceleration.



(a) Free body diagram (b) Acceleration magnitude profile

Figure 4.4: **Analytic model of suction cup failure** (**a**): A set of suction cups (light blue) hold the gray object, where the suction surface normal is a vector $\hat{\mathbf{n}}$ (not shown) that is always aligned with the vector $\hat{\mathbf{n}}_{\text{obj}}$. The suction forces $\mathbf{f}_{s,i}$ apply the same suction force at each suction contact, while the normal forces $\mathbf{f}_{\mathbf{n},i}$ react unevenly based on the tilt angle θ relative to inertial acceleration and gravity $\mathbf{f}_g$. The green arrow indicates the unit normal direction vector, and all other arrows are force vectors. Under the unimodal model $\theta = 0°$. Under the multimodal model θ can vary. (**b**) shows the acceleration magnitude drop boundaries in the sliding and moment imbalance failure modes for varying angles θ between the inertial acceleration $\mathbf{a}$ and the suction normal $\hat{\mathbf{n}}_{\text{obj}}$. Suction fails for accelerations in the region above the blue curve in the model. We generate this plot by solving the system of equations (4.4.1) – (4.4.6) with a combination of angles and accelerations. For the parameters such as object dimensions and friction coefficient of our system, sliding will always occur before moment imbalance.

We define 2 models, *unimodal* and *multimodal*. The unimodal model assumes $\theta = 0°$, and thus the maximum inertial acceleration before grasp failure is trivially given by $\mathbf{a}_{\text{fail}} = \frac{1}{m}\sum_i \mathbf{f}_{s,i}$. The multimodal model includes multiple failure modes for cases where $\theta \neq 0°$, including suction grasp failure by sliding, force imbalance, or moment imbalance. This model does not include deformation. The analysis uses the equations

$$\sum_i \mathbf{f}_{s,i} + \sum_i \mathbf{f}_{\mathbf{n},i} + \sum_i \mathbf{f}_{f,i} + \mathbf{f}_g = m\mathbf{a} \tag{4.4.1}$$

$$\sum_i \mathbf{r}_i \times \mathbf{f}_{s,i} + \sum_i \mathbf{r}_i \times \mathbf{f}_{\mathbf{n},i} + \sum_i \mathbf{r}_i \times \mathbf{f}_{f,i} = \mathbf{0} \tag{4.4.2}$$

$$\mathbf{f}_{\mathbf{n},i} = -\alpha_i \cdot \hat{\mathbf{n}}_{\text{obj}} \tag{4.4.3}$$

$$\hat{\mathbf{f}}_{f,i} = -\widehat{\mathbf{proj}_{\hat{\mathbf{n}}_{\text{obj}}}(\mathbf{f}_g)} \tag{4.4.4}$$

$$\mathbf{f}_{f,i} = \beta_i \cdot \hat{\mathbf{f}}_{f,i} \tag{4.4.5}$$

$$(\mathbf{f}_{f,i} \cdot \hat{\mathbf{f}}_{f,i}) = \gamma \cdot \mathbf{f}_{\mathbf{n},i} \cdot (-\hat{\mathbf{n}}_{\text{obj}}). \tag{4.4.6}$$

The force (4.4.1) and moment (4.4.2) balance in quasi-static equilibrium, where $\mathbf{r}_i$ is the position vector from the object center of mass to the application of forces point for the ith suction cup. We then require the reaction forces $\mathbf{f}_{\mathbf{n},i}$ to be normal to the grasping plane

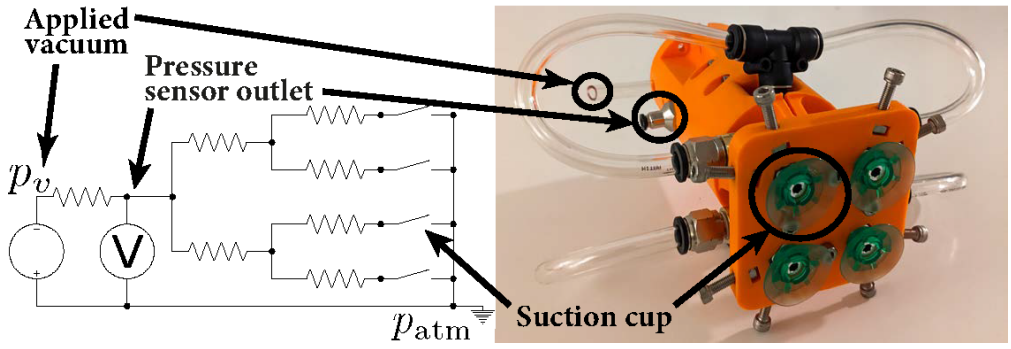


Figure 4.5: **Suction gripper with four suction cups and pressure sensor outlet.** In the custom suction gripper (**right**), the suction is distributed to the four suction cups using branching connectors from the pneumatic tube where a vacuum/negative pressure is applied, as shown in the analogous circuit diagram (**left**). A single pressure sensor is used to measure the applied pressure. In the case of a successful suction grasp, all four switches are open, and so no current/air flows, implying the applied pressure/voltage is equal for all four suction cup regardless of differing resistor values.

(4.4.3), set the direction of the friction forces to be tangent in the grasping plane and opposing the direction of the motion in its absence (4.4.4), (4.4.5), and require that the friction force magnitudes share the same proportionality constant γ to the normal forces (4.4.6).

We solve the system of equations for a range of hypothetical angles θ and accelerations $\mathbf{a}$, and classify each scenario as a failure or a success based on the physical restrictions that $\beta_i \leqslant \mu\alpha_i$ and $\alpha_i \geqslant 0$, where α_i and β_i are the magnitudes of the normal and frictional forces, respectively. The result of this simulation is shown in Fig. 4.4, which shows that the multimodal model converges with the unimodal model when $\theta \to 0^+$, and that in contrast to intuition, the curve is not strictly increasing nor strictly decreasing across the domain. The inputs used to generate the curve use dimensions from our experimental setup, and so the exact axes values do not generalise to other systems. To integrate the model with GOMP-FIT, we approximate the curve using a 4th-order polynomial fit, which we then use to formulate an analytic baseline constraint.

4.5 Experiments

We perform experiments on a physical UR5 with a custom 4-cup vacuum gripper and a set of steel blocks. The gripper has four round 30 mm diameter elastodur flat suction cups driven by a single VacMotion MSV 27 vacuum generator. Multi-cup suction grippers are common in many commercial automated logistics systems due to the increased surface area producing more suction force, and the multiple contact point stabilizing the hold resulting in reduced payload swing. The pressure sensor (Adafruit MPRLS Ported Pressure Sensor Breakout) is fitted in the gripper assembly and attached via USB to the computer that drives the UR5.

Table 4.1: **Object transport success rate and motion time.** We compute 3 pick-and-place motions with 0.8, 0.9, and 1.0 m horizontal separation between pick and place points, and perform each motion 5 times, for a total of 15 trials per algorithm and mass. For algorithms, we use **J-GOMP** as a time-optimized, but not suction-constrained baseline to show a lower bound on time, **GOMP-FIT** with analytic constraints on end-effector acceleration, and **GOMP-ST** with varying history length h, and failure threshold d_{safe}. Masses 1.3, 1.5, and 1.7 were seen at train time; mass 1.6 was not. We highlight multimodal GOMP-FIT and GOMP-ST ($h = 6, d_{\text{safe}} = 0.50$), and compare the relative speedup, observing that only 1 of the 60 trials for GOMP-ST failed, while GOMP-ST speeds up between 16 % and 58 %.

Mass [kg]	**J-GOMP**	**GOMP-FIT**		**GOMP-ST** (h, d_{safe})				**Speedup**
		Unimodal	*Multimodal*	$(1, 0.50)$	$(6, 0.05)$	$(6, 0.50)$	$(6, 0.95)$	
Success Rate								
1.3	0 %	0 %	**100 %**	0 %	**100 %**	**100 %**	0 %	
1.5	0 %	0 %	**100 %**	0 %	**100 %**	**100 %**	66.7 %	
1.6*	0 %	0 %	**100 %**	0 %	**100 %**	93.3 %	**100 %**	
1.7	0 %	33.3 %	**100 %**	0 %	**100 %**	**100 %**	33.3 %	
Motion Time [s]								
1.3	1.355	1.333	**2.304**	1.440	2.637	**1.931**	1.728	**+16 %**
1.5	1.355	1.387	**2.827**	1.525	2.763	**1.984**	1.781	**+30 %**
1.6*	1.355	1.472	**2.827**	1.643	2.795	**1.807**	1.781	**+36 %**
1.7	1.355	1.728	**4.459**	1.173	2.432	**1.856**	1.792	**+58 %**

* mass unseen in training.

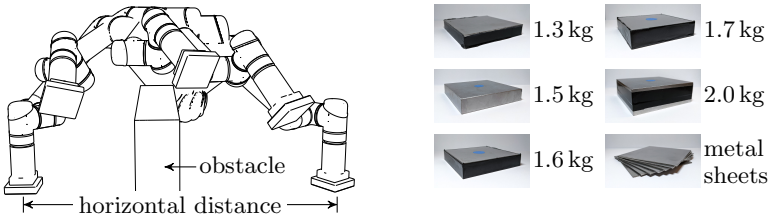


Figure 4.6: **Experimental setup** In experiments, (**left**) we compute trajectories for varying horizontal transport distances, and average the results. Shorter distances require lifting faster, while longer distances can lift more gradually. We train and test on masses (**right**) composed of stacked steel blocks and sheets. We do not train on the 1.6-kg mass to test generalization to unseen masses.

See Fig. 4.1 for a visual of the experimental setup and Fig. 4.5 for a close-up of the suction gripper.

We first perform data collection and train (Sec. 4.4.3) on four different masses, 1.343-, 1.492-, 1.741-, and 2.196-kg steel blocks (Fig. 4.6 right), which we round to 1.3, 1.5, 1.7, and 2 kg hereafter. We then test GOMP-ST and baselines by computing trajectories between varying start and goal positions and around varying obstacles. We automate data collection by using an overhead Intel RealSense 435i to locate the mass after suction failures. We use a relatively heavy mass to lower the end-effector accelerations required for a lost grasp, and

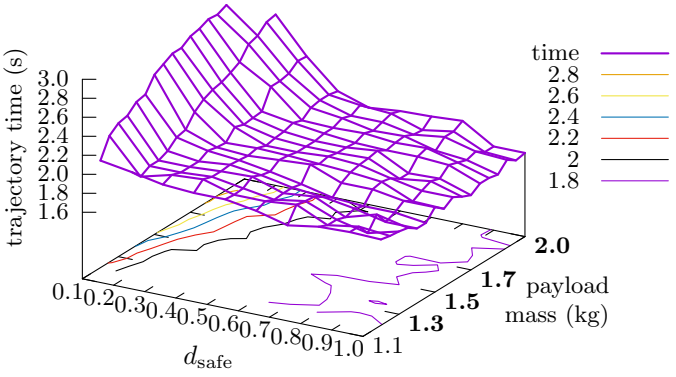


Figure 4.7: **Computed trajectory times for varying d_{safe} and payload mass.** We compute trajectories for a grid of d_{safe} and masses shown on the lower axes, and plot the trajectory times on the vertical axis. With lower d_{safe} or higher mass, the learned suction failure constraint causes the optimizer to generate slower motions. The masses used for training (1.3, 1.5, 1.7, and 2.0 kg) are in bold. The plot suggests that the network generalizes to unseen masses.

for safety as a lost grasp results in the released object having lower kinetic energy E_k, since $E_k = (1/2)mv^2$, and thus do not fly out of the workspace.

We set the automated system to collect training data for each of the four training masses, resulting in 2,367 training trajectories. We perform an 80/20 train/test split on the trajectories, data-augment failures 30× and non-failures 4× in the training set. With the UR5 operating and generating joint data at a 125 Hz, and the pressure sensor operating at 167 Hz, this results in 597,010 training examples and 19,892 test examples.

With 15 trials each, we compare to baselines of J-GOMP, an optimizing motion planner that does not include inertial effects, and GOMP-FIT, an extension of J-GOMP that allows constraints on the linear acceleration at the end-effector. We include 2 GOMP-FIT baselines in which we limit magnitude of end-effector acceleration using the unimodal model and multimodal model.

4.5.1 Ablation studies

We perform two ablation studies on GOMP-ST. First, we compare history length of $h = 1$ to $h = 6$, to study the importance of the motion history relative to single acceleration spikes in the motion. We also compare different values of the failure threshold d_{safe} to study the potential for making a trade-off between speed and reliability.

4.5.2 Results

We vary the horizontal distance between start and goal configurations and compute trajectories for each baseline and variant of GOMP-ST (Fig. 4.6 left). Each trajectory lifts over

an obstacle, thus the shorter horizontal distance (0.8 m) requires more vertical motion, while the longer horizontal distance (1.0 m) results in a period of longer horizontal acceleration. We also vary the masses, using 3 masses that were seen during training (1.3 kg, 1.5 kg and 1.7 kg) and a single mass that was not used during the data collection (1.613 kg, rounded to 1.6 kg). We run each computed trajectory 5 times and report the average success rate and motion time per transported mass as the experiment's result.

In Table 4.1, we show the results of trajectories computed by the baselines and ablations. Here we see that the J-GOMP and the GOMP-FIT unimodal baselines consistently compute motions that lead to suction failures. GOMP-FIT multimodal manages to compute safe motions, however the resulting trajectories are slow. In contrast, GOMP-ST with $h = 6$ and $d = 0.05$ reliably achieves a 100 % success rate, while the baseline method suction grasps fail in nearly all cases. It is also 16 % to 58 % faster than multimodal GOMP-FIT. The ablation of $h = 1$ shows the importance of history in learning the constraint—without it, GOMP-ST consistently computes motions that lead to suction failures.

In the ablation of d_{safe}, we see that lower thresholds result in increased success rate, but reduced speed, while the increased d_{safe} results in faster motions and decreased success rate. In addition, we observe that GOMP-ST's performance when transporting a mass unseen during training ($m = 1.6$ kg) is comparable with its performance on the objects used during the data collection.

We also study trajectories computed with the learned constraint by varying the mass and d_{safe} and plotting the results in Fig. 4.7. The plot suggests that the network is interpolating between training masses.

4.6 Discussion and Future Work

We propose GOMP-ST, an algorithm using learned motion constraints for fast transport of objects held in suction grasps. By learning the constraint from real data, we avoid explicitly modeling difficult-to-model properties such as deformation of the suction cup. We also benefit from the learned model neural-network implementation, as it facilitates automatic generation of gradients needed to linearize the constraint for the solver. Experiments on a physical UR5 suggest that the learned constraint can allow the solver to speed up motions by up to 58 %.

In future work, we will experiment with more complex environments, and include additional inputs to the learned constraint to allow it to adapt to different properties of the grasped object, for example, coefficient of friction and center of mass. We also aim to expand the analytic model to no longer be dependent on quasi-static equilibrium or rigid bodies, and therefore include spring state to the state of the trajectory optimization. While there are compelling reasons to move away from single-suction-cup grippers, they present additional suction failure modes, such as swinging and torquing out of the gripper. Addressing these failure modes may require additional approaches, such as integration with sampling-based optimization methods. Finally, results in modifying the failure threshold suggest that one

could make a trade-off between speed and reliability, but how to beneficially make that trade-off is an open issue.

Chapter 5

BOMP: Optimized Motion Planning for Bin Picking

Previous chapters presented methods to increase the efficiency and reliability of object transport in fixed, known obstacle environments. Yet, objects within these environments may move between picking cycles, potentially necessitating the training of a new model after every cycle. In this chapter we propose BOMP, a motion planning algorithm that extends the training procedure presented in Chapter 2 to increase the motion planner's efficiency in scenarios where obstacles' locations are not known during training.

5.1 Introduction

Robots are increasingly used for package handling and picking in logistics settings. When transporting thousands of packages each day, a reduction of a few tenths of a second in cycle time can significantly increase robot productivity. In deep cluttered bins, contents can shift after each pick, necessitating a strategy to rapidly compute pick-and-place motions each cycle using the latest sensor data (e.g., depth camera image).

Practical approaches include heuristic planning, optimization-based motion planning, and sampling-based motion planning. A common heuristic trajectory, Up-Over-Down, lifts to clear all obstacles, moves horizontally over obstacles to the target location, then lowers. This is easy to implement and has negligible compute time; however, the motion is longer than necessary and fails due to collision when using a long-nosed suction tool common in industry.

Optimization-based methods formulate and solve an optimization problem to find the best or fastest trajectory that avoids collisions. Sampling-based methods randomly sample and connect collision-free waypoints to find a path. The latter two methods yield significantly more successful and faster motions than Up-Over-Down, but at the expense of longer compute times.

Prior work proposed GOMP [9], an optimization-based motion planner incorporating

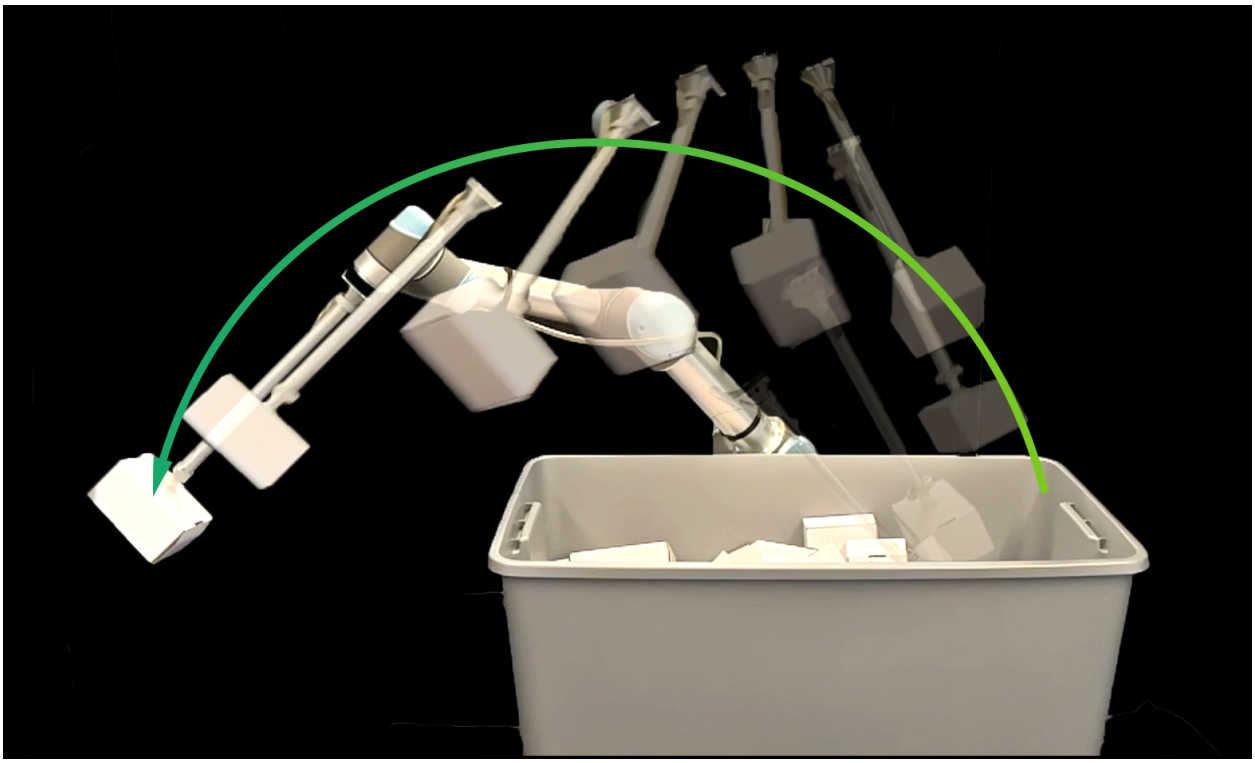


Figure 5.1: **Bin-optimized motion planning.** BOMP executing a time-optimized, jerk-limited, collision-free trajectory moving a box from a bin to a drop-off point. BOMP uses the long-nosed "bluction" tool from Huang, et al. [**huang2022bluction**] to enable the robot to reach all parts of the deep bin. BOMP uses an overhead RGBD camera and an optimizing motion planner to compute the pick-and-place trajectory. In order to speed up the computation, a neural network warm-starts the optimizer. It accepts the obstacle environment, grasped box, and pick points as input, and outputs an initial trajectory.

time-optimization, obstacle avoidance, and grasp optimization. The planner computed fast motions between pick and place points, and the grasp-optimization further sped up motions by allowing pick and place poses to be optimized while retaining the same parallel-jaw grasp contact points. The subsequent DJ-GOMP [50] sped up compute time by using a neural network to warm-start motion planning for time-optimized and jerk-limited trajectories. DJ-GOMP was trained over a distribution of start and end points, but assumes a fixed collision environment. However, boxes often move between picking cycles. In this event, compute time can increase by several orders of magnitude.

In this chapter, to address changing obstacle environments, we propose Bin-Optimized Motion Planning (BOMP). BOMP finds an optimal trajectory while considering collisions between a grasped box and the obstacle environment. We integrate BOMP into an end-to-end bin-picking pipeline, which takes as input an RGBD image of the bin and outputs time-optimal trajectories. BOMP modifies and extends DJ-GOMP by adding the dimensions of a

grasped box and a height map of the environment as inputs to the warm-start neural network, which enables adapting the trajectory due to environment changes. The warm-start neural network is trained to handle a relevant distribution of varying collision obstacles, and predicts a trajectory to warm-start the jerk-limited and time-optimizing trajectory optimization. We generate a dataset of synthetic collision environments in simulation to train the deep neural network.

This chapter makes 4 contributions: (1) BOMP, a time-optimizing, jerk-limited motion planning algorithm for bins where boxes and obstacles are detected via a depth camera; (2) An end-to-end bin-picking pipeline that uses BOMP to iteratively remove the boxes; (3) A deep neural network that learns from 25000 trajectories generated from simulated scenes to accept a height map, grasped box parameters, a number of trajectory segments, and trajectory endpoints as inputs and predicts an initial trajectory to warm-start the planning; and (4) Data from 114 experiments in simulated environments and 15 experiments in physical environments, wherein BOMP generates up to 36 % faster trajectories compared to an Up-Over-Down baseline implemented with an optimal time-parameterizer [49], as well as 58 % faster trajectories compared to Motion Planning Templates [97] (MPT) which implements a parallelized sampling-based motion planner (PRRT*). BOMP successfully generates collision-free trajectories at a 79% rate, which is similar to MPT, and significantly higher than the Up-Over-Down baseline.

5.2 Related Work

Optimization-based motion planners such as CHOMP [7], STOMP [6] and TrajOpt [5] compute motion plans by locally optimizing a trajectory while penalizing collisions or placing barrier functions on collisions [25]. Marcucci et al. [98] take a different approach: they decompose the collision-free space into convex regions and use convex optimization to find a collision-free path. GOMP [9] builds on prior formulations by taking the mechanical limits of the robot arm and the dynamics between waypoints into consideration. It also optimizes over a rotational degree of freedom about the parallel-jaw gripper contact points in each of the pick and placement frames. DJ-GOMP [50] further extends GOMP by minimizing jerk to avoid joint wear while also significantly reducing computation time by warm-starting with the output of a deep neural network.

Prior work has considered model distillation (i.e., one model being trained on the output of one or more different models) for seeding optimization-based motion planners and for achieving more general-purpose models. In many cases, training an ensemble of models improves prediction performance, but is computationally expensive as it significantly increases the required computation resources. The ensemble can be distilled in a compact network [26, 27]. DJ-GOMP used model distillation to improve GOMP's running time. While the repeated optimization executed in GOMP could take up to several minutes to finish, a forward pass in a compact neural network can be executed in milliseconds. DJ-GOMP exploits this feature of neural networks to compute similar robot's trajectories faster for a known col-

lision environment. However, when the collision environment is variable, and only known during runtime, the compute time of DJ-GOMP increases by several orders of magnitude since the neural network warm-start is no longer valid. BOMP addresses this by learning a representation of the collision environment and the grasped box, in addition to robot's trajectories.

Warm-starting an optimizer with a near-optimal solution can significantly increase the solver's performance while greatly reducing the number of iterations required to reach sufficient optimality [31]. In reinforcement learning, learning a new task can be warm-started by transferring features from old tasks the agent has already mastered [32]. Memory of motion [33] is another method that uses an offline learned policy to warm-start a control solver, and was shown to reduce the computation time in locomotion problems, and to increase the performance of nonlinear predictive controllers [34]. In GOMP [9], the optimization slowed by the number of iterations required to find the optimal horizon. To address this, DJ-GOMP [50] uses a neural network's output to warm-start the optimization with an approximation of the optimal trajectory, ultimately resulting with a faster convergence. BOMP uses a similar approach, taking into consideration the grasped box and the current state of the collision environment to predict the optimal collision-free trajectory.

In recent years, researchers have explored the approach of bypassing the optimization step and using purely learning-based methods for motion planning. Motion planning algorithms can require complex cost functions, and learning-based methods, such as learning from demonstrations (e.g., [35–37]) can reduce the amount of hand engineering required. Some learning methods focus on increasing the sampling efficiency in sample-based methods, for example by using non-uniform sampling [38] or reinforcement learning [39]. However, learning-based methods tend to generate less optimal trajectories and often fail to generate new environments.

Although pick-and-place operations are readily addressed by sampling-based motion planners, and despite the advances made via learning-based methods, the non-negligible convergence rate of these planners in high dimensions prevents them from performing well in picks-per-hour.

To address learning in a complex obstacle environment, Quershi et al. [40] encode a point cloud of the obstacles into a latent space and use a feed-forward neural network to predict the robot configuration at the next time step given an initial state, goal state, and the obstacles encoding. In this chapter, we also use the advantage of having an accurate demonstrator in the form of the cold-started optimizer to generate a large training data set, but we do not learn a latent space.

BOMP uses a combination of RGB segmentation data computed using the Segment Anything Model [99] and a depth image. Previous methods such as Dex-Net [10] use only point clouds but do not analyze the RGB image. The segmentation masks generated by SAM [99] and successor models such as FastSAM [100] give valuable information on which points belong to the same object in a scene. We assume that all objects are boxes, and we use this to estimate the pose and dimensions of boxes for grasping. We then pass this information into the optimization-based motion planner.

5.3 Problem Statement

We consider a set of n boxes in a bin, and a six-axis industrial robot arm with a long-nosed suction cup end effector as shown in Figure 5.1.

We consider the problem of transporting the boxes from the bin to a designated dropoff location. This problem is composed of two subproblems that must be solved repeatedly for each box: (1) detecting and selecting a box and a grasp; and (2) achieving this grasp to transport the box with a fast and collision-free trajectory.

An overhead RGBD camera provides an RGB image $\mathbf{y}_{\text{RGB}} \in \mathbb{R}^{h \times w \times 3}$ and a corresponding depth image observation $\mathbf{y}_{\text{D}} \in \mathbb{R}^{h \times w}$ of the bin as viewed from above.

Let $q \in \mathcal{C}$ be the complete specification of the degrees of freedom, or *configuration*, of a robot, where $\mathcal{C}$ is the configuration space. Let $\mathcal{C}_{\text{obs}}$ be the set of robot configurations in which the robot is in collision with the environment, and let $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$ be the free space. Let x_t define a state composed of q_t, $\dot{q}_t$, and $\ddot{q}_t$. Let $\tau = (x_0, \ldots, x_H)$ be a trajectory composed of a sequence of $H + 1$ robot states. Let $Q \subset \mathcal{C}$ be the kinematic limits of the robot, and let $\dot{Q}$, $\ddot{Q}$, and $\dddot{Q}$ be the velocity, acceleration, and jerk limits. We assume these limits are known.

The objective of subproblem (1) is to compute q_0 from $\mathbf{y}_{\text{RGB}}$ and $\mathbf{y}_{\text{D}}$. The objective of subproblem (2) is to compute τ such that it picks the box at q_0 and places it in a desired target location, and $q_t \in \mathcal{C}_{\text{free}} \cap Q$, $\dot{q}_t \in \dot{Q}$, $\ddot{q}_t \in \ddot{Q}$, and $\dddot{q}_t \in \dddot{Q}$ for all $t \in [0, H]$, while minimizing H.

We also assume:

- The bin is in a fixed, known pose.
- All objects in the bin are boxes.
- Boxes are rigid.
- The possible dimensions of all boxes are known in advance.
- Suction-grasped boxes are rigidly attached to the robot end effector.
- The obstacle environment is fixed during each robot motion (thus, we run open-loop motions).
- Each target box is reachable by the robot.

5.4 Method

5.4.1 Grasped Box Shape Estimation

To find a target box, we segment it using SAM [99] on $\mathbf{y}_{\text{RGB}}$ and prompt it with the highest point among the boxes in the bin. We compute this point by first masking out the bin and

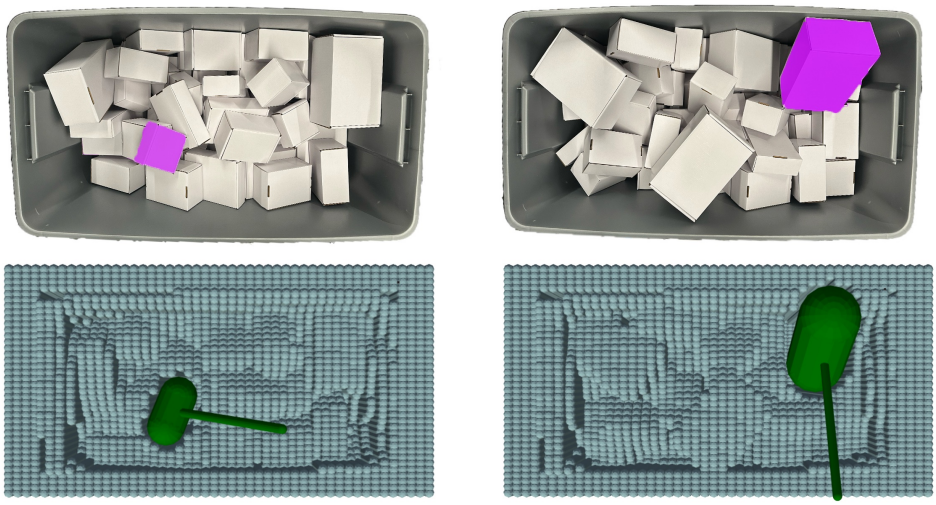


Figure 5.2: **Example scenes and carved height maps**. The top images show bins with target boxes in pink. The bottom images show the capsule-modeled, padded, and carved height maps (blue) and the robot end-effector and grasped box capsule models (green). Note that the bin padding (solid height border) in the height maps prevents carving through the wall in the second scene.

non-workspace pixels. Although this mask is generated by manual SAM prompting, we note that the bin and mask prompt points are fixed between executions. We then segment the depth map with this mask, deproject it to a pointcloud, and select the point with the highest z-coordinate in world space.. Using the highest point heuristically selects the least-occluded box and empties the bin in a top-down order.

To isolate the box and its dimensions, we use the segmentation to mask the depth image $\mathbf{y}_{\mathrm{D}}$ and create a segmented pointcloud. We fit a cuboid to this segmented pointcloud using RANSAC [101], which returns up to 3 orthogonal planes. For each plane, we compute the bounding rectangle of the inliers. We compute potential matches based on knowledge of available box sizes and the face areas and edge lengths of the observed rectangles. We aggregate results using both metrics based on the smallest difference with any potential match. In the case of a tie, we conservatively assume the largest possible box size. Using this estimate of the box dimension, we can compute the box vertex positions from the visible vertices.

5.4.2 Suction Grasp Selection

To determine the initial grasp configuration q_0, we consider grasps at the center of each of the visible faces of the target box. Of these, we only consider those where there exists an IK solution for which the robot would not be in collision with the environment. We then

select the remaining grasp which is closest to a top-down grasp, as these give the robot the most freedom to move in a bin environment. If there is more than one IK solution for a given grasp, we compare the solutions' elbow joint angles and preferentially choose the one where the elbow is concave down (like the pose in Figure 5.1). This lets the robot lift boxes without having to flip its elbow orientation.

5.4.3 Optimization Formulation

The backbone of BOMP's trajectory generation and optimization is derived from the backbone of DJ-GOMP [50]. DJ-GOMP formulates a nonlinear optimization problem and solves it using sequential quadratic programming (SQP) to compute a jerk-limited, obstacle-avoiding, and grasp-optimized motion plan that is within the robot's dynamic limits. In BOMP, we extend the obstacle avoidance to the gripper, grasped box, and more complex collision environment. We summarize the BOMP optimization formulation here:

$$
\begin{aligned}
\underset{q_{[0..H]}}{\text{argmin}} \quad & \frac{1}{2}\sum_{t=0}^{H-1} \| \dddot{q}_t \|_2^2 & \\
\text{s.t.} \quad & q_t \in \mathcal{C}_{\text{free}} & \text{(1)} \\
& q_0 = q_0^d, \quad q_H = q_H^d & \text{(2a), (2b)} \\
& \dot{q}_0 = \dot{q}_H = 0, \quad \ddot{q}_0 = \ddot{q}_H = 0 & \text{(2c), (2d)} \\
& q_{t+1} = q_t + \dot{q}_t t_{\text{step}} + \tfrac{1}{2}\ddot{q}_t t_{\text{step}}^2 + \tfrac{1}{6}\dddot{q}_t t_{\text{step}}^3 & \text{(3a)} \\
& \dot{q}_{t+1} = \dot{q}_t + \ddot{q}_t t_{\text{step}} + \tfrac{1}{2}\dddot{q}_t t_{\text{step}}^2 & \text{(3b)} \\
& \ddot{q}_{t+1} = \ddot{q}_t + \dddot{q}_t t_{\text{step}} & \text{(3c)} \\
& q_t, \dot{q}_t, \ddot{q}_t, \dddot{q}_t \in Q, \dot{Q}, \ddot{Q}, \dddot{Q} & \text{(4)}
\end{aligned}
$$

where $H \in \mathbb{Z}_+$ is the time horizon, or number of waypoints after the start, $t_{\text{step}} \in \mathbb{R}_+$ is the time interval between waypoints, and constraints with subscript $_t$ are for all valid t.

Constraint (1) ensures a collision-free trajectory. Constraints (2a), (2b), (2c), and (2d) fix the trajectory to the desired endpoint configurations, velocities, and accelerations. Constraints (3a), (3b), and (3c) enforce consistent dynamics. Constraint (4) enforces actuation limits.

Aside from the obstacle-avoidance constraints (1), which are non-convex, the remaining constraints are all linear in the decision variables. Therefore, only the obstacle-avoidance constraints must be linearized to form the problem as a locally valid quadratic program (QP).

Following from prior work [9, 50], the solver uses sequential quadratic programming (SQP), but we change how it shrinks time. Instead of reducing H between SQP solves, it keeps H fixed, and reduces t_{step} using an empirically selected upper bound, and a binary search to find the lower bound. (In experiments, we fixed $H = 16$ and chose $t_{\text{step}} = 160\,\text{ms}$ based on a parameter sweep.)

For the first SQP solve, BOMP initializes the solver with a trajectory that is linearly interpolated in joint space between the start and goal configurations. For each subsequent solution (i.e., as t_{step} decreases), BOMP initialize the SQP solver with the trajectory from the previous solve.

5.4.4 Collision Checking

During optimization, BOMP must repeatedly check for collisions between the robot, grasped box, and environment. Due to the long-nosed end-effector, the robot arm is almost always outside of the bin, so we accelerate our collision checking by only checking for collisions the robot's end-effector and the environment.

Collision Model

Using the $\mathbf{y}_{\text{D}}$, we compute a height-map collision environment, and downsample it by max-pooling to improve collision-checking efficiency. (In experiments, we downsample to 30×40.) We use capsules (cylinders with hemispherical end caps) for collision checking because capsule-capsule distance checks have a fast, closed-form solution, roughly 10x faster than the box-box collision checking in Flexible Collision Library [102]. Capsules tightly bound the tall, thin height map pixels, and the tube-like end effector.

We approximate each height-map cell with a vertically-oriented capsule. The cylindrical portion of the capsule extends from some "world bottom" z_0, outside the max reach of the robot, to the top of the height map column, z_{ij}. The set of all environment capsules is $\mathcal{Y}$. We define $\mathcal{R}$ as set of capsules the bounding the robot end effector and grasped box.

Collision Optimization

During the optimization, we check for collisions between the robot end effector, grasped box, and the height map. For efficiency, we only check each capsule in $\mathcal{R}$ against the height map capsules that intersect its axis-aligned bounding box. We define d as the closest distance between capsules in $\mathcal{Y}$ and $\mathcal{R}$ at a given time along the trajectory.

To encourage the trajectory to go over the height map, we define d to be the distance along the up axis (i.e., out of the height map), when the **center axis** of the closest robot-fixed capsule c_R^* intersects the closest environment capsule c_Y^*.

We follow CHOMP [7] in scaling d by the robot's linear speed to disincentivize speeding through obstacles. We compute the robot's linear speed at the collision point by applying the manipulator Jacobian to the joint velocities at that point in the trajectory. We densely sample d uniformly over each trajectory segment between t and $t+1$ and sum the results to get D_t. (In experiments, we sample 50x over each segment based on a parameter sweep to find the minimum value that does not impact on success rate.) Defining $\tilde{D}_t$ as the closest distance using the last QP solve trajectory $\tilde{\tau}$, we define the linearized collision constraints

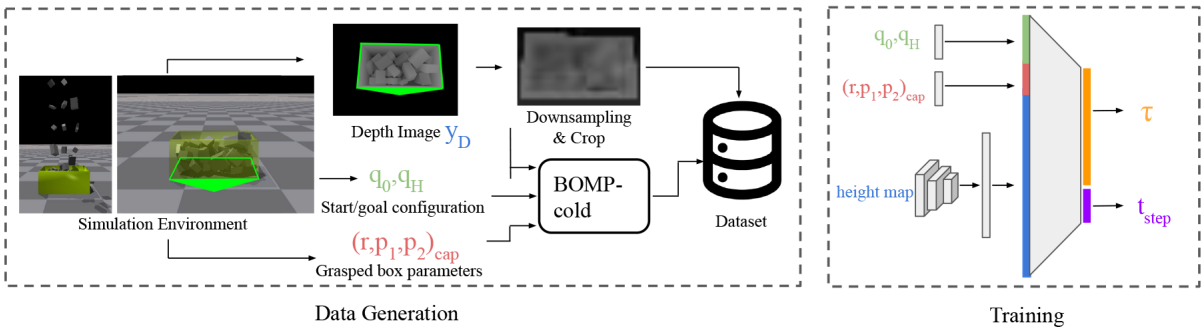


Figure 5.3: **Deep learning warm-start**. To create the training dataset (**left**), we drop boxes randomly in a simulated bin, then generate a downsampled depth map via max pooling. From the topmost box, we compute start and goal configurations and the dimensions of the corresponding box, and pass them to BOMP-cold to generate an optimized trajectory. We save the optimizer inputs with the resulting trajectory in a dataset. The warm-start network (**right**) predicts a trajectory τ and t_{step} given a height map converted from a depth image, start and goal configurations, and the grasped box parameters.

as:

$$\tilde{D}_t + \frac{\partial D_t}{\partial q_t} q_t + \frac{\partial D_t}{\partial q_{t+1}} q_{t+1} + \frac{\partial D_t}{\partial \dot{q}_t} \dot{q}_t + \frac{\partial D_t}{\partial \dot{q}_{t+1}} \dot{q}_{t+1} > 0.$$

We compute the Jacobians using finite differences.

To limit failures from the optimizer stopping with slight (e.g., $< 1\,\text{cm}$) collisions, we inflate the capsules in $\mathcal{R}$ (by $1\,\text{cm}$), and add an equivalent acceptance tolerance. Thus, if the optimizer terminates within the tolerance, the trajectories are verifiably collision-free.

Carving

To avoid "phantom" collisions at q_0, we model the change of the grasped box from being a part of the obstacle environment to being attached to the robot end effector. We *carve* out (remove) any capsule in $\mathcal{Y}$ in the same location as the capsule model for the grasped box. To prevent physically infeasible trajectories due to carving through bin walls, we artificially thicken the bin walls in the height map passed to the optimizer. Figure 5.2 shows carving results.

5.4.5 Deep Learning Warm-start

Computing fast trajectories requires long compute times (roughly 30 seconds). DJ-GOMP showed that a neural network could speed up computation by outputting an initial guess for τ and H (the trajectory and horizon) that could *warm-start* the optimizer and allow it to converge faster. We adapt this technique to reduce the computation time of BOMP. We train

a neural network (Fig. 5.3) that predicts the optimal trajectory τ and the optimal segment duration t_{step} given start and end poses, radius and endpoints of the grasped capsule (in the end-effector frame), and the scene's height map. The neural network passes the height map through two convolutional layers to produce an embedding, which is concatenated with the start pose, end pose, radius, and endpoints of the grasped capsule. This concatenated vector is passed through an encoder to yield a trajectory and t_{step}.

Generating Training Data

To generate training data, we use Isaac Gym [103] to simulate an environment with a rectangular bin that matches the real bin's dimensions and fill it with randomly sampled boxes with dimensions and probability matching our real setup. A virtual depth camera captures overhead depth images of the bin and converts it into a height map. To closely match the data distributions of boxes through the picking cycle, we repeatedly select the topmost box using the same prompting method in real (using the ground truth box shape instead of a SAM estimation) and remove it until the bin is sufficiently empty (more than 19 boxes removed). Before each removal, we record the current scene using the virtual depth camera mentioned above. In experiments, the training set consists of 25,000 simulated scenes and their height maps, grasped boxes, and optimal trajectories generated from a distribution likely to appear at test time.

Warm-starting

Even though the bin contents are different in every scene, we are able to learn a useful warm-start by referencing the height map during training and inference. At test time, we prompt the network with the trajectory endpoints, grasped box parameters, and height map. We warm-start the optimizer with the network's predicted trajectory τ and segment duration t_{step}.

We differentiate from DJ-GOMP, which does not include a grasped box or the height map in the inputs to the neural network. Furthermore, we predict t_{step} given a fixed H; DJ-GOMP did the opposite. This change simplifies the neural network architecture but requires interpolation to run on a robot's fixed-time-interval controllers.

5.4.6 Speeding Up Computation

We make several optimizations to speed up computation. We parallelize the Jacobian calculations for the non-convex obstacle constraints over 12 Intel Core i7-6850K CPUs at 3.60GHz. Like DJ-GOMP [50], we solve only until a collision-free solution is found. Empirically, the additional solve time to find a fully optimal solution does not outweigh the additional computation time cost to generate it.

5.5 Experiments

We compare BOMP with two baselines: (1) an industry-standard Up-Over-Down motion that lifts the grasped box vertically, moves horizontally over the bin wall, then lowers; and (2) an asymptotically optimal sampling-based motion planner from Motion Planning Templates MPT [97] with a subsequent time-parameterization step [49].

We compare the algorithms on compute time, execution time, total time (compute time + execution time), and collision-free trajectory generation rate ("success rate"). In physical experiments, we also report "execution success rate" which considers the percent of all experiments (including those which failed in generation) where the robot successfully transports the box from start to goal.

We note that SAM inference time is not included in our computation time because all methods use the same prompting strategy to compute the initial pose to grasp the object. Therefore we compare only the trajectory computation time in our tables.

Due to reachability constraints associated with our robot and environment, a fully vertical extraction is often not possible. Thus, the Up-Over-Down baseline performs a two-stage "up" motion. The first stage moves the end-effector vertically as far as possible; the second stage moves the box vertically as far as possible while pitching 22.5° back to enable the box to be raised higher when a top-down grasp is no longer kinematically feasible. From most locations in the bin, it is kinematically infeasible for the robot to lift the box higher than the rim of the bin with a top-down grasp.

We use MPT's parallelized RRT* (PRRT* [104]), as it is a fast asymptotically optimal planner. We configure it to optimize for minimum distance traveled in joint space. We run MPT for 1 second (MPT-1), which is similar to BOMP's generation time, and MPT for 10 seconds (MPT-10), which has more time to find and optimize a solution. While PRRT* will eventually find a solution if one exists, it may fail with insufficient planning time. Since the MPT baselines are sampling based, we execute them three times each per scene and average the metrics. We only include successfully computed trajectories in the calculation for the time-based metrics.

We also perform ablations to assess the impact of warm-starting and training the neural network on the height map. We isolate the deep-learning warm-start by including cold-started optimization with an empirically selected heuristic t_{step} of 160 ms, labeled BOMP-t160ms in Table 5.1 and Table 5.2. We also consider the impact of the neural network knowing the height map in BOMP-NH ("no heightmap"). For this ablation, we perform neural network inference *without* an input height map. As in DJ-GOMP [50], since additional SQP solves empirically take more time to compute than they save in execution time, we stop both BOMP and BOMP-t160ms after they first find a feasible trajectory. To compare against optimal execution times, we also show results from the cold-started optimization when run to optimal convergence (labeled BOMP-cold).

For both simulated and physical scenes, we model a UR5 robot reaching into a deep bin (dimensions $1.06 \times 0.562 \times 0.46\,\text{m}^3$) full of boxes. To allow the robot to reach and manipulate boxes deep in the bin, we equip it with the "bluction" tool from Huang, et al. [105] (blade

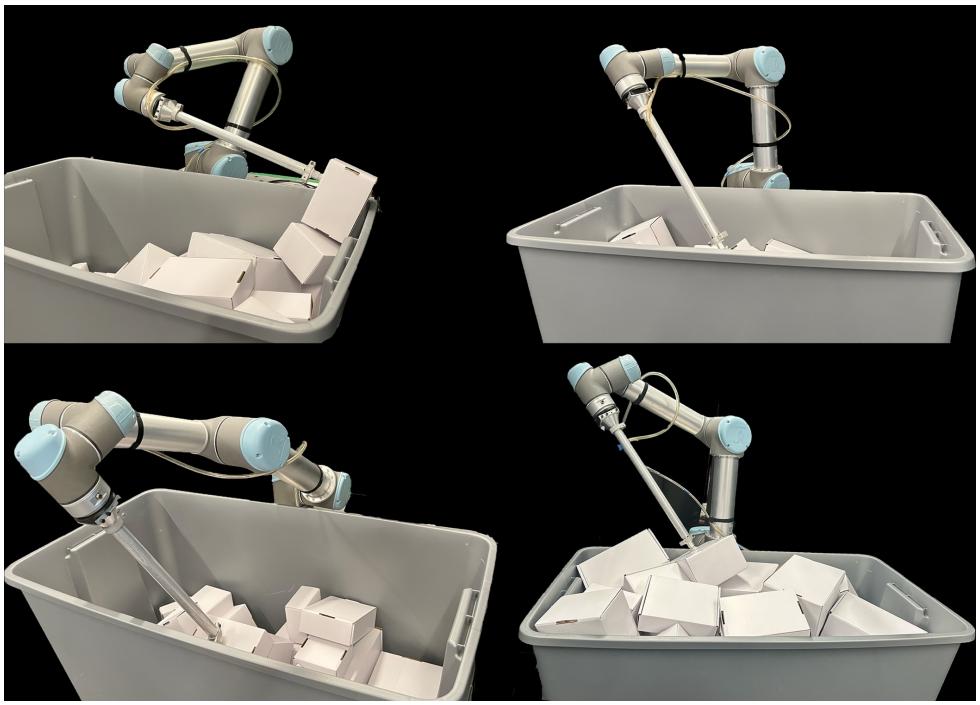


Figure 5.4: **Challenging grasp poses**. In physical experiments, we observe that using the long suction tool to grasp arbitrarily oriented boxes often results in challenging grasp poses such as the ones pictured here. While the industry-standard Up-Over-Down method fails in these cases, BOMP is able to generate fast, jerk-limited, collision-free trajectories.

and camera attachments removed).

We fill the bin with boxes of assorted sizes. For our experiments, we use boxes that are $4\times4\times2\,\text{in}^3$, $6\times4\times3\,\text{in}^3$, $7\times5\times2\,\text{in}^3$, and $9\times6\times3\,\text{in}^3$. We use between 5 and 15 boxes of each type to fill the bin. We use these box sizes because:

- They represent a range of aspect ratios (which directly affects the fidelity of the capsule model).
- The $6\times4\times3\,\text{in}^3$ and $9\times6\times3\,\text{in}^3$ boxes have the $6\times3\,\text{in}^2$ face in common. We demonstrate that our pipeline is robust to this.

5.5.1 Simulated Experiments

We generate trajectories in 114 simulated environments generated using the same process used to generate neural network training data (described in Sec. 5.4.5 and shown in Fig. 5.3). However, these scenes were not previously seen by the neural network.

As the setup can result in kinematically infeasible problems, we exclude generated environments where the initial grasp results in an IK solution that is not in collision. Results are shown over the remaining 96 feasible scenes.

For each simulated environment, we capture a depth image and convert it into a height map and select the topmost box to be removed. In simulation, rather than using SAM [99] to determine box pose and size (Section 5.4.1), we use ground truth box pose and size to inform the solvers of the grasped box. We define the goal at a joint configuration where all in-distribution grasped boxes can be held without collision. For a fair comparison, we use the capsule model (5.4.4) to evaluate collisions for all planners and we carve the height maps as in Fig. 5.2 before solving.

We then use BOMP, baselines, and ablations to plan trajectories that carry the selected box from the given start point to the endpoint. We track computation success rate, compute time, trajectory execution time, and total time (compute time + execution time). Table 5.1 displays the results of these experiments.

We find that MPT-10 and the cold-started BOMP ablations have the highest success rate. With the warm-start, BOMP strikes a balance between fast computation and fast collision-free trajectories. It achieves a similarly high success rate to MPT-10, a fast computation time close to BOMP-t160ms, and a fast execution time close to BOMP-cold.

Only Up-Over-Down achieves a barely faster total time than BOMP due to its negligible computation time, but Up-Over-Down's success rate is also very low. This is primarily due to the kinematic difficulty of safely vertically extracting *arbitrarily oriented* boxes, where the suction normal is generally not aligned to gravity.

BOMP-NH, without a height map of the environment, tends to predict trajectories closer to "average" than BOMP. This means larger (less optimized) values of t_{step} and trajectories that are on average further from optimal. This explains the increase in trajectory execution time and success rate. The higher predicted t_{step} also means that it is less often below the optimal value (i.e., infeasible). This results in the slightly higher success rate. On average, BOMP-NH's computation time is 0.1835 s higher than BOMP, suggesting that the network's knowledge of the obstacles through the height map generates trajectories that are more favorable for warm-starting.

In 93% of failure cases for BOMP-cold and BOMP-t160ms, the target box starts in contact with the bin wall. In these cases, a long tenuous extraction path is necessary to lift the box out of the bin without colliding. The warm-started BOMP and BOMP-NH also occasionally under-predict the optimal segment duration t_{step}, resulting in their slightly lower success rates.

We note that the baselines do *not* generate jerk-limited trajectories, while BOMP and ablations do. In physical experiments (Sec. 5.5.2), we observe that the trajectories generated by MPT are more likely to drop boxes during execution. This particularly happens when the trajectory turns high-jerk "corners" and at the trajectory endpoint.

Algorithm	Success Rate	Exec. Time(s)	Compute Time(s)	Total Time(s)
MPT-1	69.44%	1.941±0.776	1.833±0.044	3.774±0.786
MPT-10	**84.38%**	1.941±1.023	10.872±0.049	12.813±1.042
Up-Over-Down	16.67%	1.689±0.225	**0.837±0.158**	**2.525±0.293**
BOMP-NH	80.21%	1.209±0.146	1.664±2.498	2.872±2.496
BOMP-cold	**84.38%**	**0.982±0.352**	33.696±20.067	34.678±20.002
BOMP-t160ms	**84.38%**	2.560±0.000	1.661±1.144	4.221±1.144
BOMP	78.13%	1.080±0.142	1.481±1.639	2.561±1.649

Table 5.1: **Simulated experiments results.** In 96 feasible simulated environments, we execute 3 trials for the sampling-based methods (MPT-1 and MPT-10) and 1 trial for the deterministic methods. MPT-10 and the cold-started BOMP ablations have the highest success rate. With the warm-start, BOMP achieves a similarly high success rate while also achieving the fastest total time aside from the unreliable Up-Over-Down.

Algorithm	Generation Success Rate	Execution Success Rate	Exec. Time(s)	Compute Time(s)	Total Time(s)
MPT-1	68.89%	53.33%	2.448±1.319	1.747±0.137	4.194±1.334
MPT-10	**93.33%**	66.67%	2.210±1.269	10.792±0.140	13.001±1.300
Up-Over-Down	6.67%	6.67%	1.432±0.000	**0.710**±0.000	**2.142**±0.000
BOMP-cold	86.67%	73.33%	**0.884±0.279**	34.074±14.616	34.958±14.590
BOMP	86.67%	**80.00%**	1.035±0.168	1.583±1.075	2.617±1.067

Table 5.2: **Physical experiments results.** In 15 physical experiments, BOMP successfully executes the most trajectories and the fastest trajectories by total time (except Up-Over-Down, which only successfully executes 1 trajectory). MPT-1 and MPT-10 generate the most theoretically feasible trajectories but their non-smooth trajectories and lack of jerk limits result in several dropped boxes and automatic protective stops during physical execution.

5.5.2 Physical Experiments

We perform similar experiments in 15 real environments (see examples of the physical environment in Fig. 5.2). We use an overhead Intel RealSense D455 camera to capture height maps for collision avoidance and RGB images for segmentation. These images are 480×640 pixels, but we downsample the height map to 30×40 for collision checking (Sec. 5.4.4). Grasped boxes are selected and detected using the full grasp selection pipeline (Section 5.4.1).

As with simulated experiments, we consider only scenes with reachable grasp poses that are not in collision. The results of these experiments are in Table 5.2.

In physical experiments, BOMP successfully executes the most trajectories and achieves the fastest total time (computation + execution) other than Up-Over-Down, which only successfully executes 1 trajectory. While MPT-10 generates the most theoretically feasible trajectories, these often result in dropped boxes or automatic protective stops because its trajectories are not jerk-limited. Specifically, the trajectories computed by the sampling-

based planners often contain sharp turns and sudden changes in speed. BOMP computes a smooth trajectory that better avoids protective stops and drops.

At execution time, the failure cases for BOMP and BOMP-cold are primarily due to the capsule carving (Sec. 5.4.4) over-carving into adjacent obstacles (the capsule model is an overestimate of the actual box volume, so it may enclose other obstacles). When the resulting trajectories are physically executed, these obstacles may dislodge the box from the robot gripper.

Up-Over-Down has an extremely low success rate in this environment due to the complex grasp poses (not simply top down). We use a long suction gripper to be able to access all parts of the bin, but this has a consequence of difficult grasp poses when aligning the gripper to the surface normals of oriented boxes. Figure 5.4 shows some example scenes with complex grasp poses from which BOMP is able to compute a solution, but Up-Over-Down is unable to vertically lift the grasped box.

5.6 Discussion and Future Work

We present BOMP, an optimization-based motion planner integrated into an end-to-end bin picking pipeline. We integrate a grasp and target box identification method using the Segment Anything Model [99], and we warm-start the motion planning optimization on the output of a deep-learning model trained to include obstacles in the form of a height map. We train the network offline on simulated data, then use the trained network online to provide an initial guess of the trajectory to warm-start and speeds up computation.

In 129 experiments in real bin environments and in simulation, we showed that BOMP outperforms heuristic and sampling-based planner baselines in execution time by up to 36% and 58% respectively, while generating jerk-limited trajectories. Furthermore, BOMP achieves the fastest total time (computation time + execution time) among methods with comparable success rate.

In future work, we will address several limitations. We plan to extend beyond known boxes to general unseen grasped objects by using SAM [99] alongside a grasp planner such as Dex-Net 3.0 [79]. We will also speed up SAM prompting by using a smaller model fine-tuned for the bin.

We also plan to improve the fidelity of our capsule modeling without sacrificing too much of its speed. As presented in this chapter, the capsule model overestimates the extents of the grasped box, so it sometimes carves into adjacent obstacles. As a result, the motion planner may compute a solution that would collide in real since not all of the collision environment is present in the carved depth map it uses to plan.

Within these limitations, though, BOMP significantly outperforms baselines in speed while maintaining a comparable or superior success rate.

Part II

Efficiency and Reliability in Grasping Partially Observable Rigid Objects

Chapter 6

AVPLUG: Approach Vector Planning in Cluttered Environments

Leading grasp planners often struggle to identify effective grasps on objects that are partially occluded. AVPLUG enhances grasp reliability by utilizing an octree-based occupancy model and Minkowski sum calculations to determine a collision-free approach vector for grasping.

6.1 Introduction

In many automation tasks, such as extracting a product from a warehouse shelf, removing an ingredient from a refrigerator, or retrieving a tool from a cluttered workbench, desired objects may be hidden behind other objects. This presents a challenge in both locating the target object and finding a grasp for it. To automate such tasks, robots need to first perform visual search, and then robustly grasp and manipulate target objects once found. Although prior work [10, 106, 107] proposed methods for grasping objects in isolation, finding a robust grasp becomes significantly more challenging [108, 109] in a cluttered environment where the target object may be partially or fully occluded.

Mechanical search [110] aims to find a target object in clutter and focuses on clearing a view to the target by pushing or removing occluding objects [110–114]. However, this requires planning and executing multiple collision-free motions of the arm, adding to the overall runtime in the form of both motion planning and execution. Furthermore, the placement of occluding objects is often structured, for example with objects resting on a kitchen counter or supermarket shelf [115]. In such environments, pushing or removing objects may be undesirable. In addition, when objects are in unstable poses, even glancing contacts can lead to accidental toppling, which can damage delicate objects such as glass bottles. In contrast, this work focuses on servoing a wrist-mounted camera with a unicontact suction gripper tool (see Fig. 6.1) to a view from which the target object is fully visible and thus extractable.

Efficiently searching for a view of a target object is related to next-view planning, the

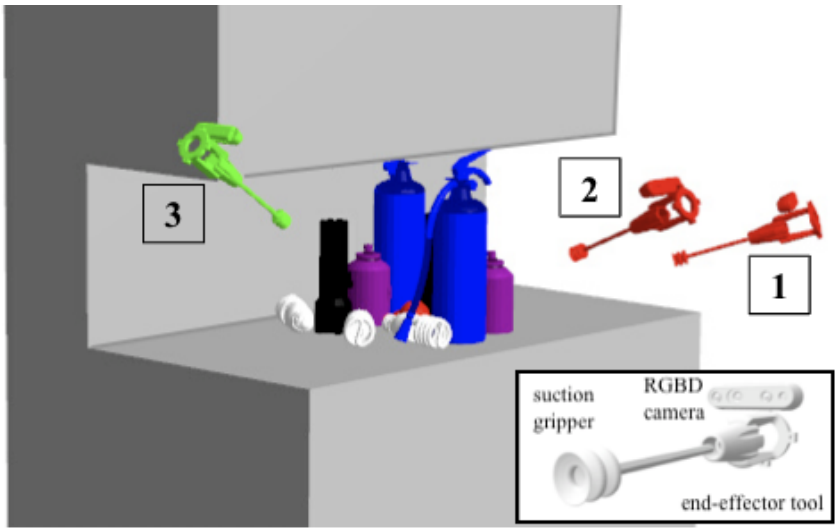


Figure 6.1: AVPLUG searches for an approach vector to grasp the occluded red target object on the worksurface. AVPLUG moves the wrist-mounted tool to a view from which it can find an approach vector for unicontact grasping (successful view shown in green). AVPLUG uses an occupancy map and a Minkowski sum to track previously explored regions of the scene and to find and evaluate candidate vectors. **Inset:** The end effector used by AVPLUG is comprised of an RGBD camera with its optical axis aligned to a unicontact suction gripper.

problem of finding an additional sensor placement to improve scene reconstruction [116]. This topic has a rich history in computer vision [117–119]. Next-view planning requires keeping track of which regions of the scene have already been explored and which have not. For the task of unicontact grasp planning, a full scene reconstruction is computationally expensive and unnecessary; thus, we propose the use of an efficient 3D voxel-based occupancy map (e.g., OctoMap [120]) as it provides the required information and can be computed rapidly.

We focus on unicontact suction grasping. As opposed to parallel-jaw grasping, in which the contact points are rarely visible from the approach vector, unicontact grasp quality is highly correlated with the visible surface normals [79]. Accordingly, we propose that aligning the suction gripper contact axis to the camera optical axis can be well-suited for a unicontact grasp exploration policy, in which finding an unoccluded view of the graspable target surface corresponds to finding an approach vector.

We present Approach Vector Planning for Unicontact Grasping (AVPLUG), an algorithm that leverages an occupancy map based on an octree and Minkowski sum computation to find an approach vector for unicontact grasping of a partially or fully occluded target object in a structured clutter scene, without changing the scene. First, AVPLUG samples potential target object locations from the unknown regions of the occupancy map. It then efficiently casts rays outwards from these candidate locations in order to identify unobstructed candidate vectors. Next, it cross-references these candidate views with a pre-computed expected

grasp quality distribution (which takes into account individual grasp quality as well as uncertainty in target object pose). It moves to the view with the highest expected grasp quality. AVPLUG repeats this process until it finds a collision-free approach vector, or reports failure.

In the case of a fully occluded target object, we encounter an additional challenge, in that there is no clear signal to guide exploration. In order to narrow the search space, we propose to efficiently compute the Minkowski sum [121] between the target object and the region of the occupancy map that we have explored thus far. This constrains the potential locations of the target object on the worksurface.

Experiments in simulation and on a physical Fetch robot suggest that AVPLUG can find an approach vector in up to 20× fewer steps than a baseline policy, even in the presence of dense occlusions and in tight spaces (see Fig. 6.6). This chapter makes three contributions: (1) A formulation of the problem of efficiently finding an approach vector for unicontact grasping a target object in the presence of partial or full occlusions; (2) AVPLUG, an efficient algorithm that uses an octree-based occupancy map and Minkowski sum computation to address the above problem; and (3) Experiments in simulation and on a physical robot comparing AVPLUG with a grid search baseline, which systematically visits views on a discretized grid.

6.2 Related Work

There has been significant prior work on searching for target objects in clutter, however the most common approach is to move or remove occluding objects. For example, Danielczuk et al. [110] defined the mechanical search problem and proposed a pipeline to iteratively search for a partially occluded object through a series of parallel-jaw grasping, suction, and pushing actions. Huang et al. [112] and Danielczuk et al. [111] then extended this work by learning an occupancy distribution to guide the search process to recover the occluded target. Xiao et al. [122] formulate the object search in clutter task as a POMDP and suggest an algorithm that takes into account the robot's current belief to evaluate the success of a manipulation task. Murali et al. [115] leveraged a variational autoencoder [107] to plan 6-DOF parallel-jaw grasps on a partially occluded target object in a cluttered scene, and remove occluding objects if no feasible grasp is found. Boroushaki et al. [123] identify and locate a fully occluded target object using RFID tags. In this work, we instead focus on moving a wrist-mounted camera to find clear approach vectors. We align the optical axis with these approach vectors in order to grasp the target object without affecting the rest of the scene.

In active perception [118, 119, 124, 125], we change the position of the sensor to reveal more of the scene's geometry. This is particularly useful for tasks such as 3D scene reconstruction [126] and mapping [127]. The next-best-view planning problem refers to computing the optimal next view with respect to a chosen goal. In the context of manipulation, a camera mounted on a robot end effector can guide the motion. Kahn et al. [128] model the occluded regions where the target object may be located as a mixture of Gaussians, and en-

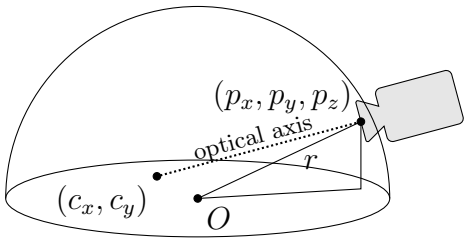


Figure 6.2: States in AVPLUG consist of a camera location (p_x, p_y, p_z) on a sphere V centered on O with radius r, and a focal point (c_x, c_y) (with implicit $c_z = 0$) on the worksurface. (c_x, c_y) represents a potential target location, at which the camera's optical axis points.

courage exploration during the trajectory optimization by penalizing for uncertainty. Other works constrain the action space to top-down (4-DOF) grasps. For example, Morrison et al. [129] propose a top-down grasp planning controller that uses active perception to choose the next-best-view of the camera as it approaches the target object along the z-axis to reveal more robust grasps. Novokovic et al. [130] propose a reinforcement learning based active and interactive perception system from a top-down view to uncover a hidden target cube in a pile of cubes. In contrast, in this work we consider approach directions on a sphere centered on the clutter centroid and consider candidate 5-DOF grasps (unicontact suction grasps have symmetry about the approach vector).

Occupancy maps are 3D representations of the environment that store information about which regions have already been explored and which have not. This information can be used to guide next-best-view planning. Hornung et al. [120] presented OctoMap, an efficient implementation of an octree-based occupancy map. Given a point cloud, OctoMap updates a 3D voxelized representation of the scene with one of three labels per voxel: occupied, empty, or unknown. Santos et al. [131] used an octree alongside a robotic arm and wrist-mounted camera, however they focused on 3D scene reconstruction. Octrees have also been used for grasping a target object in a cluttered scene [113, 114, 132], however in contrast to moving the camera, these works remove occluding objects from the scene to expose the target object.

6.3 Problem Statement

Given:

- An RGBD camera with known intrinsics, mounted in alignment with a vacuum suction cup gripper on a robot arm (see Fig. 6.1 inset).
- A target object of known geometry.
- An environment of unknown objects resting on a planar worksurface, partially or fully occluding the target object.

- A target object detector that returns a binary mask of the target object if it is visible from the RGBD camera.
- A suction grasp planner (Dex-Net 3.0 [79]) that samples candidate suction points on a depth image and returns the point with the highest associated grasp quality value.

Output: an approach vector $\mathbf{v}$ along which a collision-free linear motion can achieve a unicontact grasp of the target object. AVPLUG aims to minimize the number of steps to find such an approach vector, or reports failure.

6.3.1 Definitions

We define the following states, actions and observations:

Worksurface. A worksurface is a planar surface orthogonal to the z-axis which is aligned to gravity. The space reachable by the robot may be bounded from below by the worksurface and from above by a ceiling plane.

Sphere. Let V be a sphere with radius r centered on the origin of the worksurface (see Fig. 6.2).

States ($\mathcal{S}$). Let $\mathbf{s}_t \in \mathcal{S}$ denote a state at timestep t defining the position and orientation of the camera on V. We restrict the camera focal point to within the bounds of the worksurface. The camera can rotate about its placement on V to look at any point on the planar surface, but does not roll around its optical axis. The state space is thus $\mathcal{S} = S^2 \times S^2$, which we represent with a pair of Cartesian coordinates $(\mathbf{p}, \mathbf{c})$, where $\mathbf{p} \in \mathbb{R}^3$ is the location on V, and $\mathbf{c} \in \mathbb{R}^3$ is the point on the worksurface that the camera's optical axis intersects, thus $c_z = 0$ (see Fig. 6.2). Let $\mathbf{v} = \mathbf{c} - \mathbf{p}$ be the approach vector, defined as the direction from the camera to the target.

Actions (A). Let $\mathbf{a}_t = (\Delta p_x, \Delta p_y, \Delta p_z, \Delta c_x, \Delta c_y)$ denote the change from the state $\mathbf{s}_t$ to state $\mathbf{s}_{t+1}$, where $\mathbf{s}_{t+1}$ is restricted to remain on V.

Observations (Ω). Let $\mathbf{y}_t = \mathbb{R}_+^{H \times W \times 4}$ be an $H \times W$ RGBD image taken from state $\mathbf{s}_t$ at timestep t.

6.4 Method

Given an observation $\mathbf{y}_t$, AVPLUG seeks a grasp approach vector aligned within a tolerance angle ψ of the camera optical axis. After detecting and segmenting the target object, AVPLUG samples and evaluates grasps from its visible surface using a provided grasp planner, $\mathcal{G} : \mathbb{R}_+^{H \times W} \to (\mathbb{R}^3 \times S^2, \mathbb{R})$. $\mathcal{G}$ maps grasps parameterized by a 5-DOF pose $g \in \mathbb{R}^3 \times S^2$ to the corresponding grasp quality $q \in [0, 1]$. A higher value of q indicates a more robust grasp. If a termination condition $\mathcal{T}$ is not reached—i.e., there are no visible grasps in $\mathbf{y}_t$ above a certain grasp quality threshold—AVPLUG finds the next approach vector.

Scenes where the target object is fully occluded (e.g., due to inter-object and environmental occlusions) can be particularly challenging, since AVPLUG does not know the target

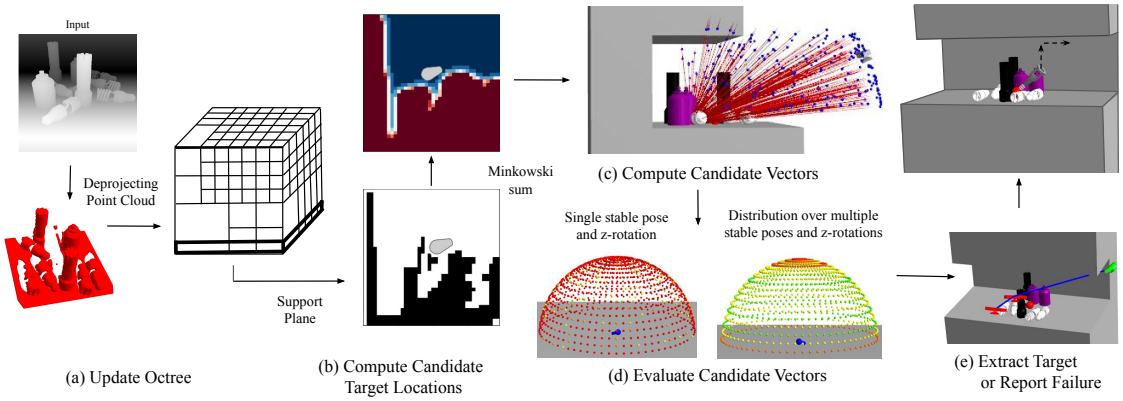


Figure 6.3: **AVPLUG overview.** (a) AVPLUG updates the octree by deprojecting a depth image to a point cloud and inserting it to the octree. (b) AVPLUG queries the part of the octree corresponding to the worksurface in order to identify candidate target locations in the unknown regions (white is unknown, black is occupied). In the figure, the location of the target object is gray, although in practice its location is unknown. To reduce the number of candidate target locations, we compute the Minkowski sum between the convex hull of the known target object and the occupied section of the worksurface. (c) AVPLUG casts rays outwards from candidate target locations to find candidate approach vectors. The corresponding camera positions (restricted to a sphere around the workspace) are marked by blue points. (d) AVPLUG uses a pre-computed expected grasp quality distribution to evaluate each candidate vector. The distribution is computed by averaging ground-truth grasp quality data from the target object model over stable poses and z-axis rotations. The average is weighted according to the relative likelihood of each stable pose. The data is then discretized into bins, with each bin represented by a colored dot in the figure (green represents the highest expected grasp quality, and red represents the lowest). (e) Once the target object is revealed, AVPLUG uses the provided grasp planner $\mathcal{G}$ to find a collision-free approach vector. It then aligns the camera's optical axis with that approach vector, moves linearly along the approach vector to grasp the target object, and extracts the grasped target with an upward motion.

object location. Without full knowledge of object poses and geometries, it is difficult to estimate the location and orientation of the target object, and more difficult still to estimate which views will uncover a graspable surface. We address this with an occupancy map, which we use to compile knowledge from previous views into an estimate of the scene state. This allows the policy to efficiently keep track of unexplored regions of the scene and prioritize them in subsequent steps. The occupancy map used in AVPLUG is based on an octree, $\mathcal{M} : \mathbb{R}^3 \rightarrow \{-1, 0, 1\}$, which maps voxels (minimum-size boxes in the octree) to occupancy values. In this paradigm, -1 represents unknown occupancy, 0 means known to be empty, and 1 means known to be occupied. The resolution of the octree is configurable—higher resolution allows for a more accurate search at the expense of increased processing time.

6.4.1 Updating the Octree

To update AVPLUG's representation of the occupancy map, the depth image in observation $\mathbf{y}_t$ is deprojected to a point cloud using the known camera intrinsics. It is then transformed to a global coordinate frame centered at the center of the worksurface using the known camera extrinsics. This transformed point cloud is inserted into the occupancy map $\mathcal{M}$ (Fig. 6.3(a)).

6.4.2 Finding Candidate Target Object Locations

If the target object is partially visible, AVPLUG approximates the translation of the target object as the center-of-mass of its visible portion. Otherwise, AVPLUG's first priority is finding the target object. With the assumption that all objects rest on a planar worksurface, AVPLUG reduces the computational complexity of the problem by limiting the search for candidate target object locations to the 2D worksurface. We define a 2D occupancy map as the 2D slice of the octree that corresponds to the worksurface, i.e., the portion at $z = 0$ in the global coordinate frame (see Fig. 6.3(b)). We note that AVPLUG does not project the occupancy map onto the worksurface, as this can result in missing a target object that is hidden beneath another object (e.g., a small object hidden below a large bowl). We also observe that there are only occupied and unknown voxels on the worksurface. AVPLUG searches for a set of candidate target object locations $\mathcal{U}$ in the unknown region of the worksurface (Fig. 6.3(b)).

Given the geometry of the target object, and assuming it has a finite set of feasible stable poses on the worksurface, we use a Minkowski sum [121] to estimate an occupancy distribution for the location of the target object. To compute the Minkowski sum, we first generate polygons from both the occupied region and the target object. For the former, we convert the 2D occupancy map to a binary image and find the contours of the occupied components (see Fig. 6.3(B)). If the contours form self-intersecting polygons, we smooth them using erosion and dilation [133]. To create a polygon of the target object, we project the vertices of the known mesh to the worksurface and compute the convex hull of the projected vertices. We then compute the Minkowski sum between the occupied region polygons and the target object polygon. This inflates the occupied region in a way that eliminates unknown points that are less likely to occupy the target object. Since the stable pose of the target object and its rotation about the z axis are unknown, we discretize the rotations into 8 bins and compute a Minkowski sum per stable pose and discretized rotation. We then sum the results and normalize to estimate a distribution for the location of the target object. We then use this distribution to uniformly sample a set of points $\mathcal{U}$ that maximize the likelihood of occupying a portion of the target object.

6.4.3 Finding Candidate Vectors

To find a candidate collision-free approach vector $\mathbf{v}$, AVPLUG casts rays outwards from the approximated target locations. This method draws inspiration from Lozano-Perez et al. [134], who describe an approach to fine motion synthesis by chaining backwards from a known goal toward the current position (Fig. 6.3(c)). If ray i in direction $\mathbf{d}$ does not intersect any occupied voxels, it represents a potentially clear line of sight, and the intersection point $\mathbf{p} \in \mathbb{R}^3$ of the ray with V is computed (see the blue points in Fig. 6.3(c)). There may be few or many candidate vectors, in correspondence with the levels of occlusion in the scene. A valid candidate vector consists of a point on the sphere and the negated ray direction leading to it: $\mathbf{v} = (\mathbf{p}, -\mathbf{d})$. To visit such a view, we move the camera to position $\mathbf{p}$ and align its optical axis with $-\mathbf{d}$.

6.4.4 Evaluating Candidate Vectors

Given the target object geometry and pose we could use $\mathcal{G}$ to compute grasps and check which approach vector aligns with a collision-free candidate vector. However, AVPLUG does not know the target object pose a priori; it only has access to the object geometry and a probability distribution of stable poses. Using this information, AVPLUG computes an expected grasp quality distribution for the target object before viewing any scenes. To compute this distribution, AVPLUG performs a weighted average of grasps and their associated quality, averaging over the known stable poses and all z-axis rotations. Higher likelihood stable poses are given more weight in the average. It then discretizes the data into 5° elevation × 5° azimuth bins (see Fig. 6.3(d)). AVPLUG evaluates each candidate vector based on the score of the bin containing its direction vector $-\mathbf{d}$. This method prioritizes views that align with a larger number of approach vectors (generally corresponding to different stable poses). Such views are more likely to find at least one approach vector that leads to a successful grasp in the given scene.

6.4.5 Finding and Evaluating Visible Grasps

Once the target object is revealed, AVPLUG queries the grasp planner $\mathcal{G}$ for visible grasps $g \in \mathbb{R}^3 \times S^2$. It then sorts the grasps by quality and iterates through the grasps in order to classify them. First, AVPLUG discards any previously seen grasps (from visible from prior views) known to collide or be unreachable. Next, AVPLUG casts a ray in the occupancy map outward from the grasp contact point along the negated approach vector. If there are any collisions with voxels known to be occupied, it classifies this grasp as colliding. Otherwise, AVPLUG declares the grasp is collision-free, then moves to align the camera with the candidate approach vector and attempt the grasp. It caches all remaining visible grasps for later consideration if this grasp fails (e.g., by an undetected collision or obstacles preventing extraction). If none of the visible grasps is collision-free, AVPLUG finds the next view following the steps in Sections 6.4.3, 6.4.4.

6.5 Experiments

To evaluate AVPLUG, we run experiments in simulated and physical environments, and compare to a baseline policy.

6.5.1 Simulation Experiments

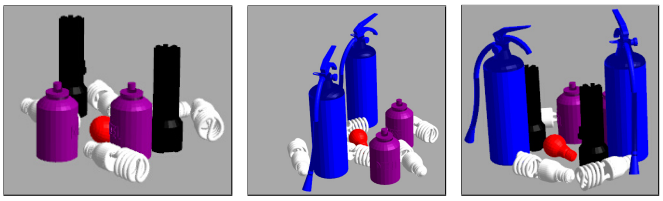
We use $R = 0.01\,\text{m}$ resolution since it empirically allows for a sufficiently accurate Minkowski sum in the fully occluded case and improves grasp collision estimation in the partially occluded case. To implement the octree for the occupancy map, we use the open-source OctoMap [120, 135].

We use ground truth segmentation to generate a binary mask of the target object. Since AVPLUG relies on an external instance segmentation algorithm, in practice, one could use an off-the-shelf object segmentation algorithm such as SD Mask R-CNN [136] with an additional matching phase for classification.

To decide whether the target object is graspable from the current state, we use a grasp planner based on Dex-Net 3.0 [79] as an oracle. Dex-Net 3.0 pre-computes suction grasps and associates quasi-static wrench-resistance quality metrics to a target object mesh, then matches these to pre-computation results at evaluation time.

6.5.2 Environments in Simulation

We first evaluate AVPLUG on two simulated scenes: 1) a tabletop, for which the potential vectors on the sphere V range between elevation angle $\theta \in [0°, 85°]$ and azimuth angle $\varphi \in [-90°, 90°]$, and 2) a counter with a cabinet above that constrains the possible grasp approach directions to a slice of V ranging between elevation angle $\theta \in [55°, 85°]$ and azimuth angle $\varphi \in [-90°, 90°]$. In both cases, we use a sphere V with radius $r = 0.6\,\text{m}$. To generate multiple different levels of occlusion, we use $N = 10$ objects of varying heights. Object models were selected from Thingiverse [137] and YCB [138]. We sample the locations of the objects from a uniform distribution in a bounded $0.4\,\text{m} \times 0.4\,\text{m}$ worksurface, and position each object in a stable pose. We also randomize the initial camera view. We generate 3 tiers of scene complexity (Fig. 6.4) by altering the relative proportions of larger objects; this affects the level of occlusions in the scene. **Tier 1** consists of 2 flashlights, 2 spray bottles and 5 spiral bulbs; **Tier 2** replaces the flashlights from tier 1 with 1.6× higher and 1.7× wider fire extinguishers; and **Tier 3** consists of 2 fire extinguishers, 2 flashlights, 2 spray bottles, and 3 spiral bulbs. For all tiers, the target object is a light bulb, which is easily occluded due to its small size compared to the other objects in the scene. After executing a successful grasp, the target object is extracted with an upwards motion (Fig 6.3(e)).



(a) Tier 1 (b) Tier 2 (c) Tier 3

Figure 6.4: **Difficulty tiers with varying levels of occlusions.** The target is shown in red on a tabletop environment. The difficulty tiers define scenes with increasing levels of complexity due to increased occlusions. (a) Tier 1 includes 2 flashlights, 2 spray bottles, and 5 spiral bulbs. (b) Tier 2 includes 2 fire extinguishers, 2 spray bottles, and 5 spiral bulbs. (c) Tier 3 includes 2 fire extinguishers, 2 flashlights, 2 spray bottles, and 3 spiral bulbs.

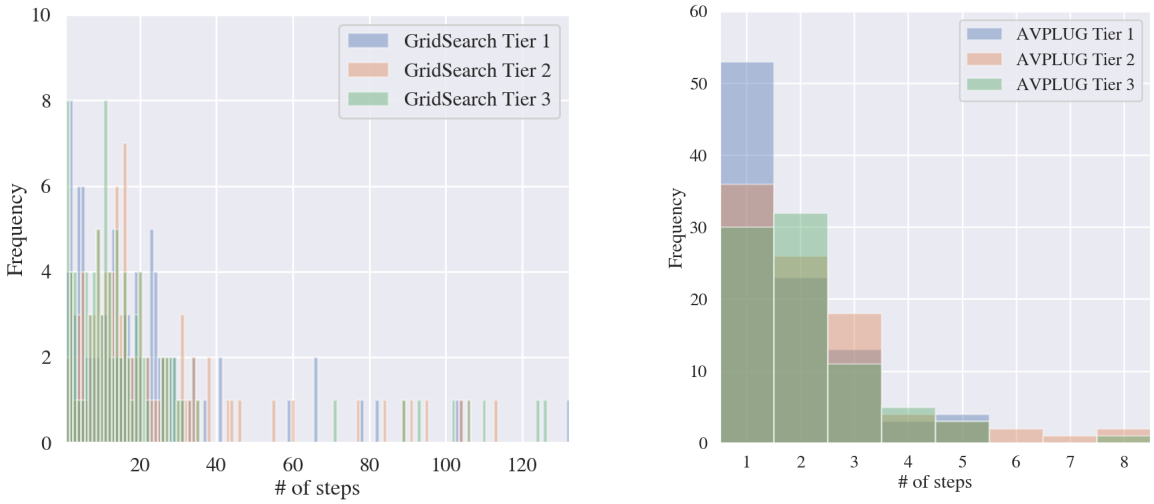


Figure 6.5: Comparisons of steps to completion of successful rollouts in simulation. The scale on the horizontal axis is 15× larger for the *GridSearch* policy.

6.5.3 GridSearch Baseline

We compare AVPLUG to a *GridSearch* baseline. *GridSearch* discretizes the sphere V into 212 fixed-spaced views (the distance between neighboring views is $l = 5°$ in both elevation and azimuth) and systematically visits each view until it finds a view from which it can plan a grasp. This baseline visits all the views to the right of the initial view, moves up to the next row of views once it reaches the maximum azimuth angle defined in 6.5.2, continues the search by moving left until it reaches the next boundary, and so on. Once it reaches the top-most row and cannot move up, it continues the search from the bottom-most row.

Scene	Tier	Policy	# Steps		Distance [m]	
			Median	IQR	Median	IQR
Tabletop	1	GridSearch	16.0	22.2	0.8	1.0
		AVPLUG	**1.0**	**1.0**	**0.6**	**0.4**
	2	GridSearch	20.0	26.0	1.0	1.4
		AVPLUG	**1.0**	**1.0**	**0.7**	**0.6**
	3	GridSearch	16.0	16.0	0.8	0.9
		AVPLUG	**1.0**	**1.0**	**0.7**	**0.8**
Counter	1	GridSearch	14.5	19.0	0.7	0.9
		AVPLUG	**1.0**	**1.0**	0.7	**0.8**
	2	GridSearch	15.5	18.0	**0.7**	**0.9**
		AVPLUG	**2.0**	**2.0**	0.8	1.0
	3	GridSearch	18.0	19.5	0.9	**1.0**
		AVPLUG	**2.0**	**1.0**	0.9	1.2

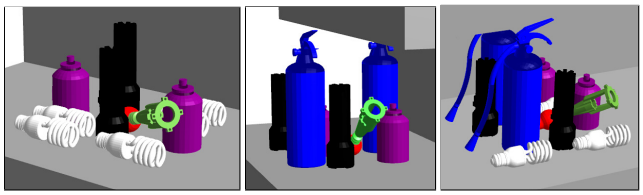
x

Table 6.1: **Simulation Experiments.** Median and interquartile range (IQR) of the number of steps to success and the distance traveled for each policy over 100 rollouts in 2 simulated environments, for successful rollouts. The success rate is 100 % for the *GridSearch* baseline and 94 % to 100 % for AVPLUG. (See Fig. 6.8 for description of failure modes)

GridSearch stops when it finds a view from which it can plan a grasp, or after it has visited all the discretized views.

6.5.4 Simulation Results

We roll out AVPLUG on 100 scenes, until the policy reaches a termination condition $\mathcal{T}$ and it finds a high-quality grasp ($q \geqslant 0.75$) on the target object, or it fails to find a grasp within a maximal number of steps H and the experiment fails. We set $H = 212$ to account for the total number of grid points in the countertop environment; therefore, if a successful approach vector exists, the *GridSearch* baseline will find it. We benchmark the experiments using the following metrics: median and interquartile range (IQR) for number of steps to success, and distance traveled. We use these metrics since the number of steps to success relates to the data acquisition and computation time, and the distance traveled by the robot arm may result in increased travel time and a potential loss in precision. The results are summarized in Table 6.1 and Fig. 6.5, and show that AVPLUG finds an approach vector in up to 20× fewer steps (median) than the baseline. In Fig. 6.5 we observe that the baseline suffers from high variance, as it is sensitive to the initial view—if it starts near a successful approach vector it can terminate quickly, otherwise it may search the grid exhaustively.



(a) (b) (c)

Figure 6.6: **Unicontact grasping in tight spaces.** AVPLUG can find approach vectors for unicontact grasping even in tight spaces due to the high resolution of the occupancy map.

The average computation time of AVPLUG for finding an approach vector is 1.05 s, benchmarked on a server with an Intel Xeon CPU @ 2.20 GHz.

6.5.5 Physical Experiments

We evaluate AVPLUG on physical scenes in the countertop setting using a Fetch mobile robot. To find grasps, we use a planarity-based grasp planner. The grasp planner first samples candidate suction points from a depth image by computing surface normals, then selects only those within 10° of the optical axis. Finally, it ranks these candidates by using a planarity metric: a suction cup-sized ring is projected around the grasp point, and the grasp is scored based on the distance from the ring to the surface depth. This distance is minimized in higher quality grasps [79]. We filter out any grasps that will collide with the scene when approaching or exiting using collision checking between the gripper mesh and the observed point cloud. We consider a grasp successful if it is not in collision and its quality value is above 0.8.

We construct 3 tiers of scenes with matching difficulty to those in simulation. For each tier, we evaluate a single scene, and for each scene we choose 5 random starting views. At each view, we evaluate the baseline once and AVPLUG 3 times, taking the average to account for inherent stochasticity. We use the elevation angle $\theta \in [45°, 75°]$, azimuth angle $\varphi \in [-45°, 45°]$, and radius $r = 0.5\,\text{m}$ for kinematic feasibility. The target object is a red light bulb similar to the target in simulation, and the occluding objects are objects found around the house and lab. We use an HSV color detector to get the binary target segmentation mask. Figure 6.7 shows the experimental setup. Results in Table 6.2 suggest that AVPLUG can consistently find an approach vector in fewer steps (median 2.0) than the baseline (median between 5.0 and 12.0). While the number of search steps taken by baseline policy highly depends on the starting view (with a higher IQR between 3.0 and 10.0), AVPLUG is able to achieve more consistent high performance among random starting views (with a lower IQR of 1.0).

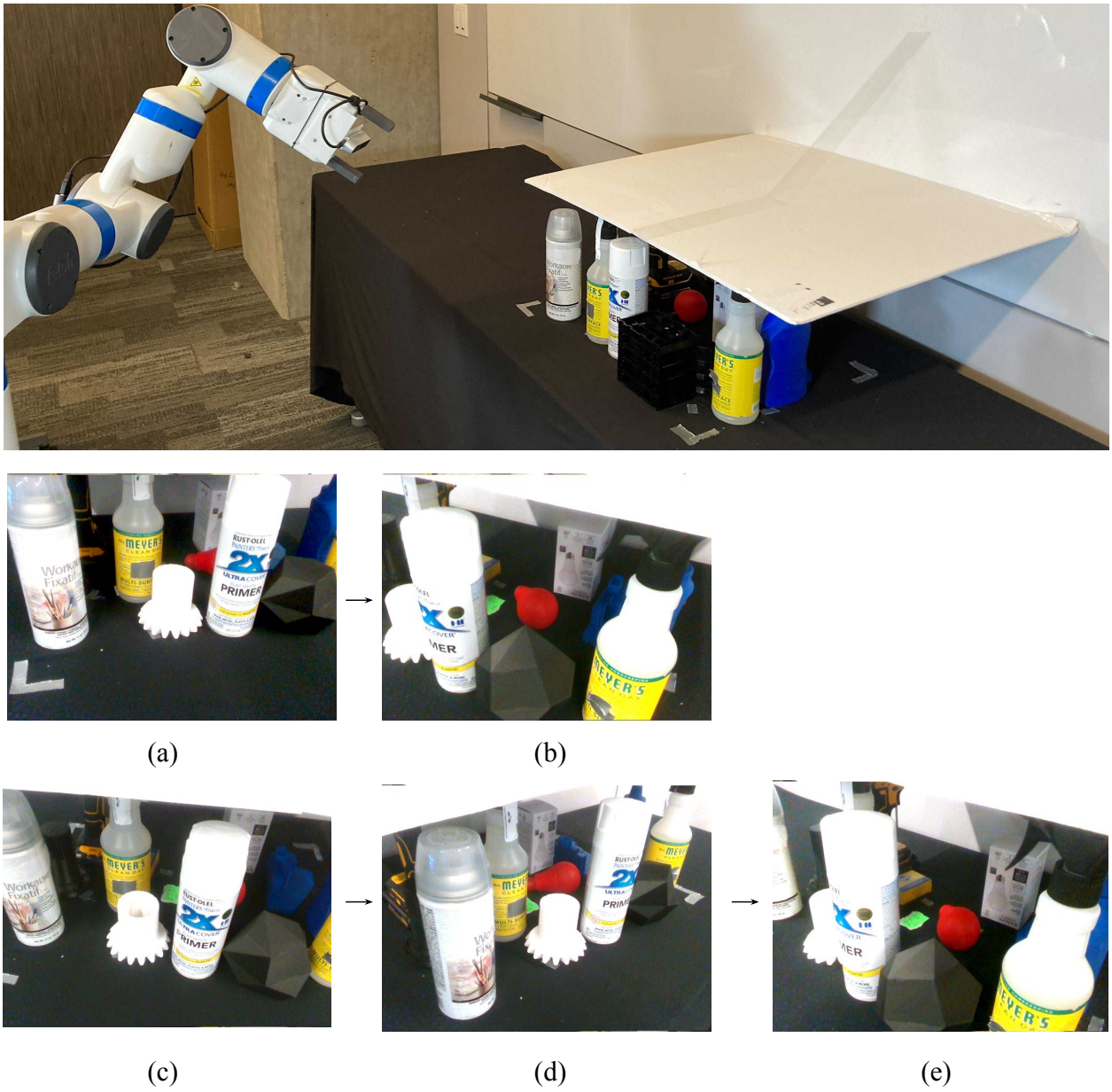


Figure 6.7: **Physical experiments setup. Top:** Physical counter setup with a Fetch mobile manipulator for grasping. **Middle:** In the first experiment starting at (a), the visible part of the target object (in red) is not graspable from the initial position, but is graspable from the next position (b). **Bottom:** In the second experiment starting with view (c), although a successful grasp is found from the second position (d), it leads to a collision between the gripper and the environment. AVPLUG then finds a collision free approach vector on the following step (e).

		# Steps		Distance [m]	
Tier	**Policy**	Median	IQR	Median	IQR
1	GridSearch	5.0	10.0	**0.6**	0.5
	AVPLUG	**2.0**	**1.0**	0.7	**0.4**
2	GridSearch	6.5	9.2	0.6	0.5
	AVPLUG	**2.0**	**1.0**	0.6	**0.3**
3	GridSearch	12.0	3.0	1.1	0.6
	AVPLUG	**2.0**	**1.0**	**0.6**	**0.4**

Table 6.2: **Physical Experiments Results.** Median and interquartile range (IQR) of the number of steps to success and the distance traveled for each policy over 5 rollouts in a physical counter environment. The metrics are reported for successful rollouts. The success rate is 100% for both the *GridSearch* baseline and AVPLUG.

6.5.6 Visibility vs Graspability

An visibility version of AVPLUG evaluated candidate approach vectors according to their *information gain*, defined by the number of voxel labels that changed from unknown to either empty or occupied after aligning the camera optical axis with the corresponding approach vector. In this method, AVPLUG chose an approach vector that maximized the information gain, with the goal of discovering presently hidden graspable surfaces on the target object. One limitation of this approach, however, was that it prioritized distant approach vectors over near and successful ones, since drastic view changes would generally reveal more unobserved parts of the scene. This visibility-based approach is better suited for the purpose of scene reconstruction and mapping than for finding a grasp on an occluded target object—this motivated using known grasp distributions instead.

6.5.7 Failure Cases

Since the occupancy map discretizes the scene into cubic voxels, the occupied section of the octree occasionally extends beyond the true boundaries of the occluding objects (see Fig. 6.8). Furthermore, due to the long and narrow structure of the end effector, valid grasp approach vectors can pass very close to occluding objects. As a result, AVPLUG's grasp evaluation step (Section 6.4.4) may detect collisions when there are none.

6.6 Discussion and Future Work

We present AVPLUG, an algorithm that employs an octree-based occupancy map and Minkowski sum computation to find an approach vector for unicontact grasping. AVPLUG

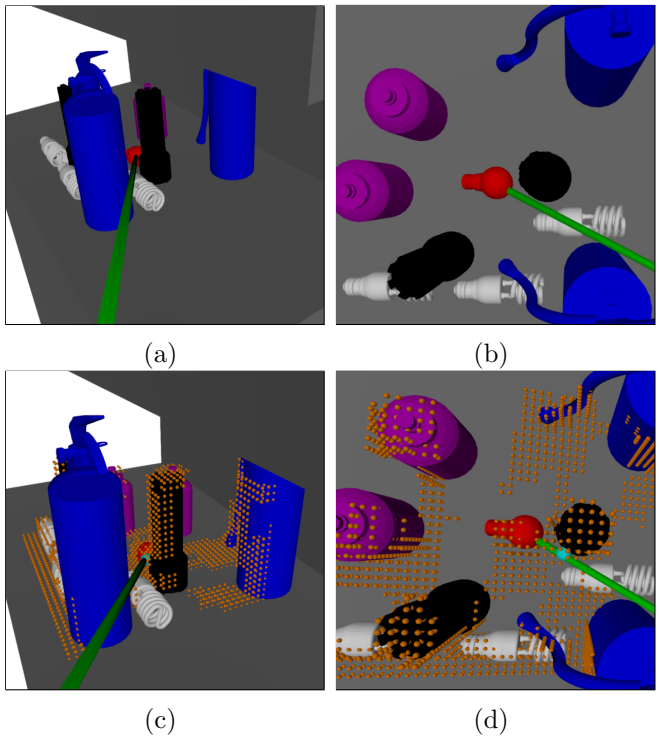


Figure 6.8: **Octree resolution failure case.** AVPLUG queries the grasp planner for available grasps on the target object. The grasp approach axis (green) passes between the flashlight and the fire extinguisher with enough clearance for the long and thin end effector in (a) and (b), therefore the grasp planner declares this to be an accessible grasp. However, when AVPLUG casts this ray through the octree, it finds a collision with the flashlight at the cyan point in (c) and (d). This is because the octree's occupied voxels (denoted by orange points) extend slightly beyond the bounds of the flashlight due to the discretization.

takes advantage of the strong correlation between visibility and graspability in suction grasping by servoing a wrist-mounted camera to find graspable views. It is able to find and extract fully or partially occluded known target objects without the risk of toppling other objects. Experiments in simulation and on a physical robot suggest that AVPLUG can find an approach vector in up to 20× fewer steps compared to a baseline policy, and can extract objects from tight spaces (see Fig. 6.6). In future work, we will utilize shape completion and pose estimation algorithms to reason about the graspable part of the target object. We will also extend this work to a tight shelf environment, from which the object cannot be easily extracted.

Chapter 7

Grasping Transparent Objects Reliably

In this chapter we explore another perception challenge faced by grasp planners that operate on depth images as existing depth cameras have difficulty detecting, localizing, and inferring the geometry of transparent objects. The problem is that depth cameras assume that the surfaces of the objects reflect light uniformly in all directions, but this assumption doesn't hold for transparent objects because of the reflection and refraction. Dex-NeRF uses neural radiance fields (NeRF) to accurately identify and securely grasp transparent objects.

7.1 Introduction

Transparent objects are common in homes, restaurants, retail packaging, labs, gift shops, hospitals, and industrial warehouses. Effectively automating robotic manipulation of transparent objects could have a broad impact, from helping in everyday tasks and performing tasks in hazardous environments. Existing depth cameras assume that surfaces of observed objects reflect light uniformly in all directions, but this assumption does not hold for transparent objects as their appearance varies significantly under different view directions and illumination conditions due to reflection and refraction properties of transparent materials. In this chapter, we propose and demonstrate *Dex-NeRF*, a new method to sense the geometry of transparent objects and allow for robots to interact with them.

Dex-NeRF uses a Neural Radiance Fields (NeRF) as part of a pipeline (Fig. 7.1, right) to compute and execute robot grasps on transparent objects. While NeRF was originally proposed as an alternative for explicit volumetric representations and shown to render novel views of complex scenes realistically [11], it can also reconstruct the scene geometry. In particular, due to the view-dependent nature of the NeRF model, it can learn to represent the geometry associated with transparency accurately. The only input requirement to train a NeRF model is a set of images taken from a camera with known intrinsics (e.g., focal length, distortion) and extrinsics (position and orientation in the world). While the intrinsics can be

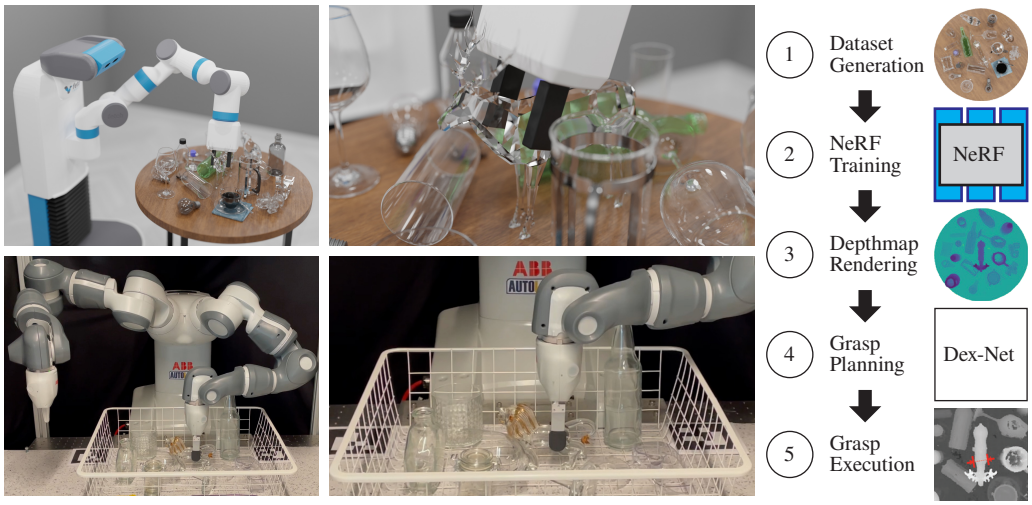


Figure 7.1: **Using NeRF to grasp transparent objects** Given a scene with transparent objects (left column), we the pipeline on the right to compute grasps (middle column). The top row shows Dex-NeRF working in a simulated scene while the bottom row shows it working in a physical scene.

determined from calibration techniques or from the camera itself, determining the extrinsics is often a challenge [139, 140]. However, robots operating in a fixed workcell or with a camera mounted on the manipulator arm, can readily determine camera intrinsics. This makes NeRF a particularly good match for robot manipulators.

In experiments, we show qualitatively and quantitatively that Dex-NeRF can compute high accuracy depth images from photo-realistic synthetic and real scenes, and achieve 90 % or better grasp-success rates on real objects.

his chapter provides the following contributions: (1) Integration of NeRF with robot grasp planning; (2) Transparency-aware depth rendering method for NeRF; (3) experiments on synthetic and real images showing NeRF with Dex-Net generates high-quality grasps; and (4) Synthetic and real image datasets with transparent objects for training NeRF models.

7.2 Related Work

For robots to interact with transparent objects, they must first be able to detect them. The most recent approaches detecting and recognizing transparent objects are data-driven. Lai et al. [141] and Khaing et al. [142] propose using a Convolutional Neural Network (CNN) to detect transparent objects in RGB images. Recently, Xie et al. [143] developed a transformer-based pipeline [144] for transparent object segmentation. Other methods rely on deep-learning models to predict the object pose. Phillips et al. [145] trained a random forest to detect the contours of transparent objects for pose estimation and shape recovery. Xu et al. [146] proposed a two-stage method for estimating the 6-degrees-of-freedom (DOF) pose

of a transparent object from a single RGBD image by replacing the noisy depth values with estimated values and training a DenseFusion-like network structure [147]. Sajjan et al. [148] extend this and incorporate a neural network trained for 3D pose estimation of transparent objects in a robotic picking pipeline. Zhou et al. [149, 150] train a grasp planner directly on raw images from a light-field camera. Zhu et al. [151] used an implicit function to complete missing depth given noisy RGBD observation of transparent objects. However, these data-driven methods rely on large annotated datasets that are hard to curate, whereas Dex-NeRF does not require any prior dataset.

Recently, implicit neural representations have led to significant progress in 3D object shape representation [152–154] and encoding the geometry and appearance of 3D scenes [11, 155]. Mildenhall et al. [11] presented Neural Radiance Fields (NeRF), a neural network whose input is a 3D coordinate with an associated view direction, and output is the volume density and view-dependent emitted radiance. Due to its view-dependent prediction, NeRF can represent non-Lambertian effects such as specularities and reflections, and therefore capture the geometry of transparent objects. However, NeRF is slow to train and has low data efficiency. Yu et al. [156] proposed *Plenoctrees*, mapping coordinates to spherical harmonic coefficients, shifting the view-dependency from the input to the output. In addition, Plenoctrees pre-samples the model into a sparse octree structure, achieving a significant speedup in training over NeRF. Deng et al. [157] proposed JaxNeRF, an efficient JAX implementation of NeRF reduces the training time of a NeRF model from over a day to several hours. Deng et al. [158] add depth supervision to train NeRF 2 to 6× faster given fewer training views. Adamkiewicz et al. [159] proposed an algorithm that uses a NeRF model for robot navigation. In this work, we propose to use NeRF to recover the geometry of transparent objects for the purpose of robotic manipulation.

Traditional robot grasping methods analyze the object shape to identify successful grasp poses [160–162]. Data-driven approaches learn a prior using labeled data [163, 164] or through self-supervision over many trials in a simulated or physical environment [165, 166] and generalize to grasping novel objects with unknown geometry. These approaches rely on RGB and depth sensors to generate an accurate observation of the target object. Additionally, different methods use different inputs, such as depth maps [10, 167, 168], point clouds [146, 169–171], octrees [172], or a truncated signed distance function (TSDF) [173, 174]. In contrast, in this chapter we propose a method to render a high-quality depth map from a NeRF model to then pass to Dex-Net [10] to compute a grasp. While standard depth cameras have gaps in their depth information that needs to be processed out with hole-filling techniques, the depth map rendering from NeRF is directly usable. It is possible that other grasp-planning techniques may be able to plan grasps from NeRF models.

7.3 Problem Statement

We assume an environment with an array of cameras at known fixed locations or that the robot can manipulate a camera (e.g., wrist-mounted) to capture multiple images of the scene.

Given the environment with rigid transparent objects, Dex-NeRF computes a frame for a robot gripper that will result in a stable grasp of a transparent object.

7.4 Method

This section provides a brief background on NeRF, then describes recovering geometry of transparent objects, integrating with grasp analysis, and improving performance with additional lights.

7.4.1 Preliminary: Training NeRF

NeRF [11] learns a neural scene representation that maps a 5D coordinate containing a spatial location (x, y, z) and viewing direction (θ, φ) to the volume density σ and RGB color $\mathbf{c}$. Training NeRF's multilayer perceptron (MLP) requires multiple RGB images of a static scene with their corresponding camera poses and intrinsic parameters. The expected color $C(\mathbf{r})$ of the camera ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ between near and far scene bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{7.4.1}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$ is the probability that the camera ray travels from near bound t_n to point t without hitting any surface. NeRF approximates the expected color $\hat{C}(\mathbf{r})$ as:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \tag{7.4.2}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$ and $\delta_i = t_{i+1} - t_i$ is the distance between consecutive samples on the ray $\mathbf{r}$. The training process minimizes the error between rendered and ground-truth colors.

7.4.2 Recovering Geometry of Transparent Objects

We observe that NeRF does not directly support transparent object effects—it casts a single ray per source image pixel without reflection, splitting, or bouncing. NeRF recovers non-Lambertian effects such as reflections from a specular surface by regressing on view direction and supervising with view-dependent emitted radiance. However, while RGB color $\mathbf{c}$ is view-dependent, the volume density σ is not—meaning NeRF has to learn a non-zero σ to represent any color at that spatial location. The usual result is that the transparent object shows up as a "ghostly" or "blurry" version of the original object in rendered RGB images.

When training, a NeRF model learns a density σ of each spatial location. This density corresponds to the transparency of the point, and serves to help learn how much a spatial

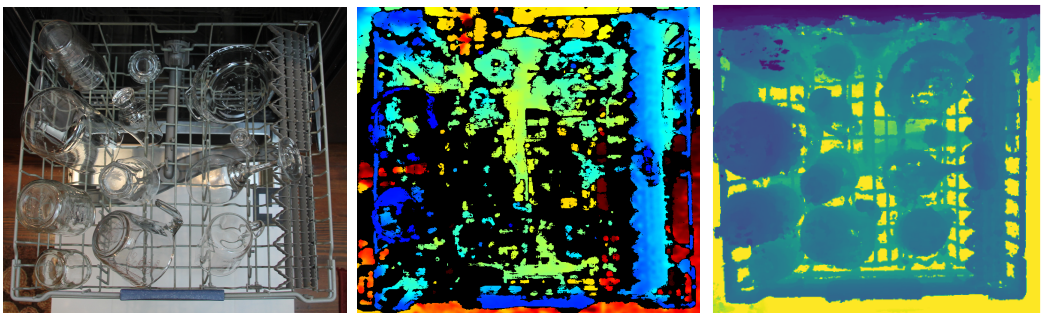


Figure 7.2: **Comparison to RealSense Depth Camera**. We compare the results of the proposed pipeline in a real-world setting against the depth map produced by an Intel RealSense camera. In the left image is the real-world scene, the middle shows the depth image from the RealSense, and the right shows the result of our pipeline. The color scheme in the RealSense image is provided by the RealSense SDK, while the color scheme in the right column is from MatPlotLib. We observe that the RealSense depth camera is unable to recover depth from a large portion of the scene, shown in black. On the other hand, the proposed pipeline, while having a few holes, can recover depth for most of the scene.

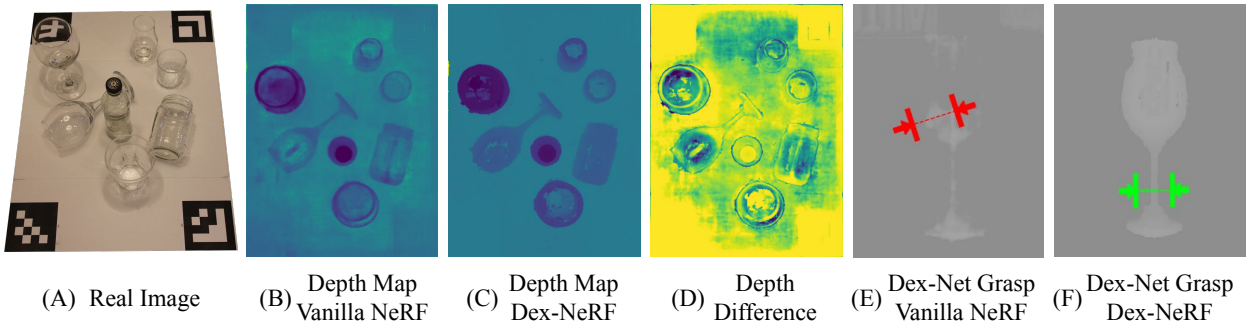


Figure 7.3: **Using NeRF to render depth for grasping transparent objects**. Dex-NeRF uses a transparency-aware depth rendering to render depth maps that can be used for grasp planning. In contrast, Vanilla-NeRF's depth maps are filled with holes and result in poor grasp predictions.

location contributes to the color of a ray cast through it. Although NeRF converts each σ_i to an occupancy probability $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$, where δ_i is the distance between integration times along the ray, thus implicitly giving α_i an upper bound of 1, it does not place a bound on the raw σ value. Dex-NeRF uses the raw value of σ to determine if a point in space is occupied.

7.4.3 Rendering Depth for Grasp Analysis

To compute a grasp from a trained NeRF model, we propose to render a depth image and have Dex-Net use it to plan the grasp. To generate a depth image, we consider two

candidate reconstructions of depth. First, we use the same depth rendering that NeRF uses. This *Vanilla NeRF* reconstruction first converts σ_i to an occupancy probability α_i. It then applies the transformation $w_i = \alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$. To render depth at pixel coordinate $[u, v]$, it computes the sum of sample distances from the camera weighted by the termination probability $D[u, v] = \sum_{i=1}^{N} w_i \delta_i$. When applied on transparent objects, however, this results in noisy depth maps, as shown in Fig. 7.3.

Instead, we consider a second, transparency-aware method that searches for the first sample along the ray for which $\sigma_i > m$, where m is a fixed threshold. The depth is then set to the distance of that sample δ_i. We explore different values for m, and observe that low values result in a noisy depth map while high values create holes in the depth map. In our experiments we set $m = 15$ (see Fig. 7.8).

7.4.4 Improving Reconstruction with Light Placement

For NeRF to learn the geometry of a transparent object, it must be able to "see" it from multiple camera views. If the transparent object is not visible from any views, then it will have no effect on the loss function used in training, and thus not be learned. We thus look for a way to improve visibility of transparent objects to NeRF.

One property that transparent objects share (e.g., glass, clear plastic) is that they are glossy and thus produce specular reflections when the camera view direction is opposite to the surface normal of the incident direction of light. To the NeRF model, a specular reflection viewed from multiple angles will appear as a bright point on a solid surface—e.g., $\mathbf{c} = [1, 1, 1]^T$ and $\sigma > 0$, while from other angles it will appear as $\sigma \leqslant 0$. As σ is view-independent, NeRF learns a σ between fully opaque and fully transparent for such points.

By placing additional lights in the scene, we create more angles from which cameras will see specular reflections from transparent objects—this results in NeRF learning a model that fills holes in the scene. While the number and placement of lights for optimal training is dependent on both the expected object distribution and camera placement, in experiments (Sec. 7.5.5) we show that increasing from 1 light to a 5x5 array of lights improves the quality of the learned geometry.

7.5 Experiments

We experiment in both simulation and on a physical ABB YuMi robot. We generate multiple datasets, where each dataset consists of images and associated camera transforms of one static scene containing one or more transparent objects. We train NeRF models using a modified JaxNeRF [157] implementation on 4 Nvidia V100 GPUs. We use an existing pre-trained Dex-Net model for grasp planning without modification or fine-tuning. We can do this since NeRF models can be rendered to depth maps from arbitrary camera intrinsics and extrinsics, thus we match our NeRF rendering to the Dex-Net model instead of training a new one.

7.5.1 Datasets

As existing NeRF datasets do not include transparent objects, and existing transparent-object-grasping datasets do not include multiple camera angles, we generate new datasets using 3 different methods: synthetic, Cannon EOS 60D camera with a Tamron Di II lens with a locked focal length, and an Intel RealSense.

For synthetic datasets, we use Blender 2.92's physically-based Cycles renderer with path tracing set to 10240 samples per pixel, and max light path bounces set to 1024. We chose theses settings by increasing them until renderings were indistinguishable from the previous setting—finding that lower settings lead to dark regions and smaller specular reflections. For glass materials, we set the index of refraction to 1.45 to match physical glass. We include 8 synthetic datasets of transparent objects: 2 scenes with clutter: light array and single light; 4 singulated objects from Dex-Net: Pipe Connector, Pawn, Turbine Housing, Mount; and 2 household objects: Wineglass upright and Wineglass on side. As these computationally demanding to render due to the high quality settings, we distribute these as part of the contribution.

For the Cannon EOS and RealSense real-world datasets, we place ArUco markers in the scene to aid in camera pose recovery and take photos around the objects using a fixed ISO, f-stop, and focal length. We use bundle adjustment from COLMAP [139, 140] to refine the camera poses and intrinsics to high accuracy. We include 8 physical datasets of transparent objects with a variety of camera poses: table with clutter, Dishwasher, Tape Dispenser, Wineglass on side, Flask, Safety Glasses, Bottle upright, Lion Figurine in clutter. The main difficulty in generating these datasets is calibration and computing high-precision camera poses.

The datasets (at `https://sites.google.com/view/dex-nerf`) differ from prior work in their focus on scenes with transparent objects in a graspable setting, with over 70 camera poses each.

7.5.2 Synthetic Grasping Experiments

We test the ability of Dex-NeRF to generate grasps on the synthetic singulated transparent Dex-Net object datasets. For each dataset, we evaluate the grasp in simulation using a wrench resistance metric measuring the ability of the grasp to resist gravity [79]. Fig. 7.4 shows images of the synthetic objects, Dex-NeRF-generated depth map, and an example sampled grasp for each. To measure the effect of training time on grasp-success rate, we simulate and record grasps over the course of training. In Fig. 7.5, we observe that grasp-success rate improves with training time, but plateaus between 80 % and 98 % success rate at around 50k to 60k iterations. This suggests that there may be a practical fixed iteration limit to obtain high grasp success rates.

We test Dex-NeRF on a scene of a tabletop cluttered with transparent objects. In this experiment, the goal is to grasp a transparent object placed in a stable pose in close proximity to other transparent objects. The challenge is twofold: the depth rendering quality should be

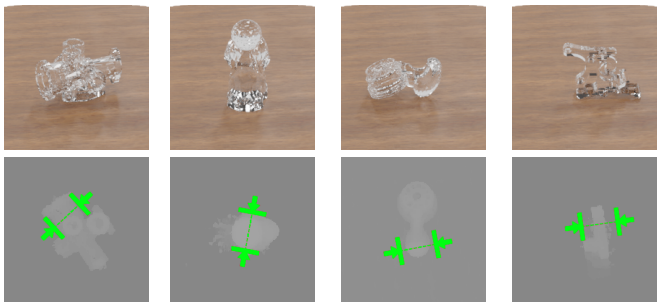


Figure 7.4: **Synthetic singulated objects** used in simulation experiments. **Top row:** image of the object in the training data. **Bottom row:** computed depth map and candidate grasp.

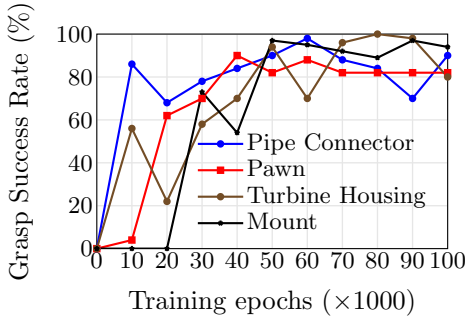


Figure 7.5: **Grasp-success rate vs training epochs.** As opposed to view-synthesis, which requires over 200k epochs, we observe high grasp success rates after 50k to 60k epochs.

sufficient for both grasp planning and collision avoidance. Fig. 7.1 shows the robot and scene in the upper left, and the overhead image, depth, and computed grasp inline in the pipeline, and the final computed grasp with simulated execution is in the upper middle image. The final grasp contact point was accurate to a 2 mm tolerance, suggesting that Dex-NeRF with sufficient images taken from precisely-known camera locations may be practical in highly cluttered environments.

7.5.3 Physical Grasping Experiments

To test the Dex-NeRF in a physical setup, we place transparent singulated objects in front of an ABB YuMi robot, and have the robot perform the computed grasps. We compare to 2 baselines: (1) *PhoXi*, in which a PhoXi camera provides the depth map; and (2) *Vanilla NeRF*, in which we use the original depth rendering from NeRF. The PhoXi camera is normally able to generate high-precision depth maps for non-transparent objects. All methods use the same pre-trained Dex-Net model, and both Vanilla NeRF and Dex-NeRF use the same NeRF model—the only difference is the depth rendering. We test with 6 objects (Fig. 7.6), and compute and execute 10 different grasps for each and record the success rate. A grasp is successful if the robot lifts the object. In Table 7.1, we see that Dex-NeRF gets 90 % and 100 % success rates for all objects, while the baselines get few successful grasps. The PhoXi camera is unable to recover any meaningful geometry which causes Dex-Net predictions to fail. The Vanilla NeRF depth maps often have unpredictable protrusions that result in Dex-Net generating unreliable grasps.

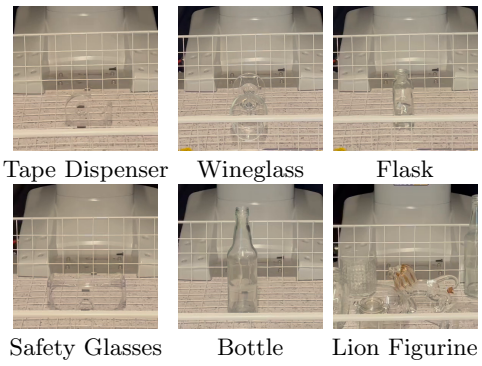


Figure 7.6: **Physical grasps objects.** In the background is the base of the YuMi robot.

Object	PhoXi	Vanilla NeRF	Dex-NeRF
Tape Dispenser	0/10	0/10	**10/10**
Wineglass	0/10	0/10	**9/10**
Flask	0/10	1/10	**9/10**
Safety Glasses	0/10	0/10	**10/10**
Bottle	0/10	**10/10**	**10/10**
Lion Figurine	0/10	3/10	**10/10**

Table 7.1: **Physical grasp success rate.** For each object, we compute a depth map using a PhoXi camera, unmodified Vanilla NeRF, and Dex-NeRF for grasping transparent objects. From the depth map, Dex-Net computes a 10 different grasps, and an ABB YuMi attempts the grasp. Successful grasps lift the object.

7.5.4 Comparison to RealSense Depth

We qualitatively compare the rendered depth map of the proposed pipeline against a readily-available depth sensor on scenes with transparent objects in real-world settings (Fig. 7.2). We select the Intel RealSense as it is common to robotics applications, readily available, and high-performance. The RealSense, like most stereo depth cameras, struggles with transparent objects as they are unable to compute a stereo disparity between pixels from different cameras when the pixels are specular reflections or the color of the object behind the transparent object. The RealSense optionally projects a structured light pattern on the scene to aid in computing depth from textureless surfaces; however, in experiments, we observed no qualitative difference with and without the light pattern emitter enabled. We use a Canon EOS for NeRF, and use a RealSense for a depth image. In this experiment, we observe that the RealSense cannot compute the depth of most transparent objects and often produces regions of unknown depth (shown in black) where transparent objects are. On the other hand, the proposed pipeline produces high-quality depth maps with only a few noisy areas.

7.5.5 One vs Many Lights

We experiment with different light setups to test the effect of specular reflections on the ability of NeRF to recover the geometry of transparent objects. We create two scenes (Fig. 7.7), one with a single bright light source directly above the work surface, and another with an array of 5x5 (25) lights above the work surface. We set the total wattage of the lights in each scene to be the same. Since most lights in the multiple light scene are further away from the work surface than the single light source, the scene appears darker, though more evenly illuminated. The effect of the specular reflections is prominent on the lightbulb in

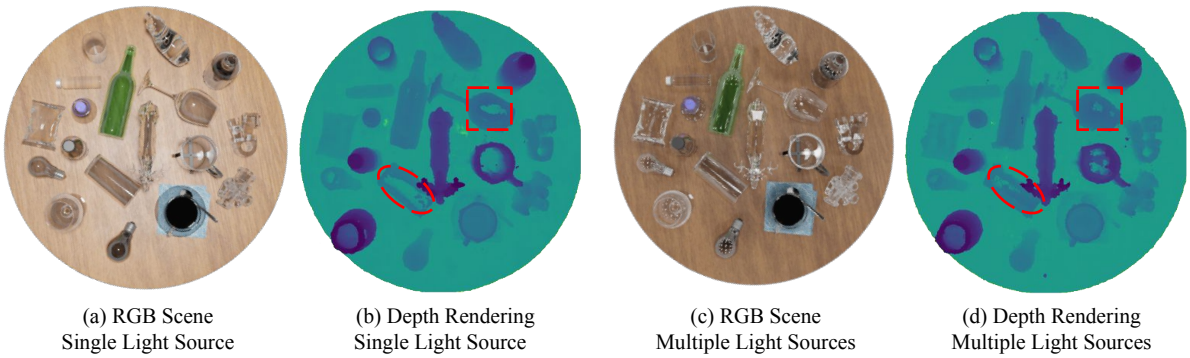


Figure 7.7: More lights mean more specular reflections, and result in better NeRF depth estimation of transparent surfaces. In (a) and (b), we show a scene lit by a single overhead high-intensity light. In (c) and (d) we show the same scene lit by an overhead 5x5 array of lights. The combined light wattage is equal in both scenes. Images (a) and (c) are views of the scene, and (b) and (d) are the corresponding depth images obtained from the pipeline. Two glasses on their sides are missing top surfaces (outlined in dashed red) in (b), while the effect is reduced in (d) due to the additional light sources.

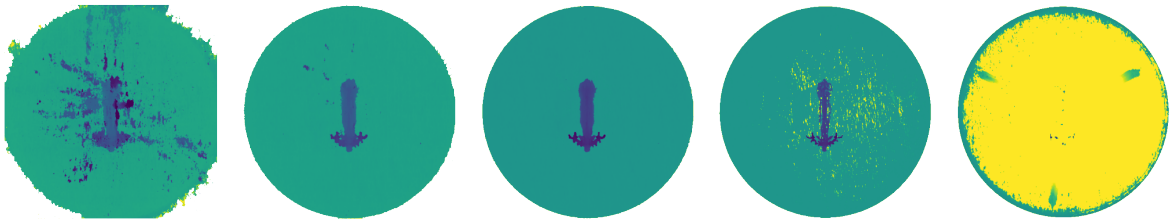


Figure 7.8: **depth rendering using NeRF with different thresholds** Here we show the effect of the threshold value on the depth rendering of an isolated deer figurine. Values too low result in excess noise, while values too high cause parts of the scene to disappear.

the lower part of the image. In the single light source, there is a single specular reflection, while in the multiple light scene, the reflection of the array of lights is visible.

With the same camera setup for both scenes, we train NeRF models with the same number of iterations. We show the depth rendering in Fig. 7.7 and circle a glass and a wineglass on their side. In the single-light source image, the closer surfaces of the glasses are missing, while in the multiple-light source depth image, the glasses are nearly fully recovered. This suggests that additional lights in the scene can help NeRF recover the geometry of transparent objects better.

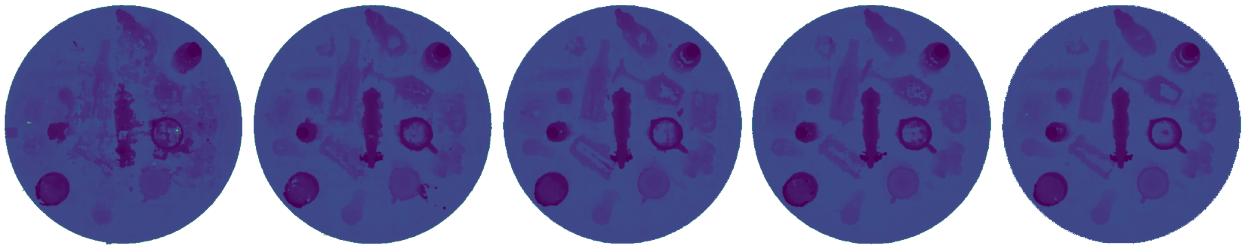


Figure 7.9: **Depth rendering using a grid of overhead cameras.** Using increasing amounts of overhead cameras improves the quality of the depth map and its utility in grasping, however, beyond a certain number of cameras there is a diminishing return.

7.5.6 Workcell Setup

We experiment with a potential setup for a robot workcell in which a grid of overhead cameras captures views of the cluttered scene so that a robot manipulator arm can then perform tasks with transparent objects in the workcell. We propose that a grid of overhead cameras would be practical to setup and would not obstruct manipulator tasks nor operator interventions. The objective is to determine how many overhead cameras would be needed to recover a depth map of sufficient accuracy to perform manipulation tasks.

We place a 2 m by 2 m grid of cameras 1 m above the work surface, and have them all point at the center of the work surface. Each camera has the same intrinsics, and are evenly spaced along the grid. We experiment with grids having 4, 9, 16, 25, 36, and 49 cameras. The environment has the same 5x5 grid of lights as before. For each camera grid, we train JaxNeRF for 50k iterations and compare performance.

After training, we observe increasing peak signal to noise ratios (PSNR) and structural similarity (SSIM) scores with increasing number of cameras. The 2x2 grid of cameras produces a high train-to-test ratio for PSNR, likely indicating overfitting to training data, and results in a depth map without apparent geometry. This ratio decreases with additional cameras. The minimum number of cameras for this proposed setup appears to be around 9 (3x3) as its depth map is usable for grasp planning, while the 5x5 grid shows better PSNR and SSIM and ratio between train and test PSNR, and the 7x7 grid is the best. See Fig. 7.9 for a visual comparison. Additionally, we trained 9x9, 11x11, and 13x13 grids, observing no statistically significant improvement beyond the 7x7 grid.

7.6 Discussion and Future Work

In this work, we showed that NeRF can recover the geometry of transparent objects with sufficient accuracy to compute grasps for robot manipulation tasks. NeRF learns the density of all points in space, which corresponds to how much the view-dependent color of each point

contributes to rays passing through it. With the key observation that specular reflections on transparent objects cause NeRF to learn a non-zero density, we have Dex-NeRF recover the geometry of transparent objects through a combination of additional lights to create specular reflections and thresholding to find transparent points that are visible from some view directions. With the geometry recovered, we pass it to a grasp planner, and show that the recovered geometry is sufficient to compute a grasp, and accurate enough to achieve 90 % and 100 % grasp success rates in physical experiments on an ABB YuMi robot. We created synthetic and real datasets for experiments in transparent geometry recovery, but we believe these datasets may be of interest to researchers interested in extending NeRF capabilities in other ways and thus contribute them as well. Finally, to test if NeRF could be used in a robot workcell, we experimented with grids of cameras facing a worksurface and their ability to recover geometry in potential setup, and showed the increased capabilities and point of diminishing return for additional cameras.

In future work, we hope to address one of the main drawbacks of NeRF—the long training time required to obtain a NeRF model. Many research groups have sped up training time through improved implementations, new algorithms, new network architectures, preconditioned network weights, focused sampling, and more. While these approaches apply to general NeRF training, we plan to exploit features specific to robot scenerios to speed up training, including using depth camera data as additional training data, manipulator-arm-mounted cameras to inspect regions of interest, and visio-spatial foresight to adapt to changes in the environment.

Chapter 8

Grasping Transparent Objects Efficiently

The prior chapter focuses on improving the reliability of grasping transparent objects. However, NeRF costs hours of computation per grasp. Evo-NeRF uses Instant-NGP and trains it concurrently to image capturing to increase its efficiency.

8.1 Introduction

Sequentially grasping transparent objects is critical in industry, pharmaceuticals and households. Sensing these objects is difficult; since camera-based sensors see through transparent objects from most angles, assumptions underlying traditional disparity and structure-from-motion-based methods break. Data driven approaches rely on large synthetic and real-world datasets to address this problem. ClearGrasp [175] trains a CNN to infer local surface normals on transparent objects from RGBD images based on Blender synthetic examples. They show impressive results on 3-5 transparent objects separated by 2cm, but note challenges with open-top containers, partial occlusions in clutter, background distractors, and transparent objects' shadows.

Neural Radiance Fields (NeRFs) [11] are a 3D representation originally designed for novel view synthesis which can reconstruct traditionally challenging-to-model scenes that include transparent objects. Dex-NeRF [176] uses NeRF to grasp transparent objects, but costs hours of computation per grasp. Recent dramatic advancements in NeRF training speed have opened the door for real-time usage [177–179]. We propose Evo-NeRF, a method for rapidly training NeRF for grasping, and RAG-Net, a neural network for robustly computing grasps from NeRF rendered depth images. We apply Evo-NeRF in a purely online setting to sequentially grasp transparent objects in clutter in the time-span of 10s of seconds, as required in dish loading, table clearing, and other household tasks.

To make NeRF practical for robotic grasping, we build on Instant-NGP [179], a fast variant of NeRF. Rather than training on a fixed set of images, we incrementally optimize over a

stream of images as they are captured during a robot motion. Due to NeRF's varying convergence speed on different difficulty scenes, we propose a method to terminate image capture upon achieving sufficient task confidence. We further adapt NeRF to sequential grasping by re-using NeRF weights from grasp to grasp and demonstrate its rapid adaptability to object removal. Since we propose that the robot captures images and trains a NeRF as it moves, motion blur, kinematic limitations, and speed considerations reduce the quality of the recovered geometry and introduce prominent spurious geometry known as *floaters*. We propose adding geometry regularization to the training objective, which improves the recovered geometry, but out-of-the-box grasp planners still struggle to find quality grasps due to remaining artifacts. Dex-NeRF [176] on the other hand, could use an out-of-the-box grasp sampler because it used diverse, high-quality, calibrated, *still* images captured in an offline process.

To mitigate the lower-quality NeRF reconstructions, we propose a novel pipeline to train a grasping network on depth maps directly from NeRFs, which are trained on photorealistic renderings of transparent objects. We find the grasping network transfers well to real-world NeRF reconstructions. This pipeline utilize the training speed of Instant-NGP—without it, the pipeline would be computationally infeasible. Real robot experiments using an actuatable camera to capture images suggest that Evo-NeRF can reconstruct graspable scene geometry rapidly and reliably when combined with RAG-Net, achieving an 89% success rate on single objects within *9.5* seconds of image capturing.

The contributions of this chapter are: (1) novel usage of NeRF in a sequential setting, rapidly evolving the NeRF representation between grasps, (2) improvements in scene geometry reconstruction speed built on existing methods, (3) an approach based on task confidence to efficiently stop image capturing early, (4) a novel training pipeline in simulation to acclimate a grasping planner to NeRF geometry characteristics, (5) a dataset of 8667 Blender rendered scenes of transparent objects with robust grasps, and (6) experimental data suggesting that Evo-NeRF enables rapid grasping on NeRF.

8.2 Related Work

NeRF [11] is a neural-network scene representation that enables photorealistic synthesis of novel views of a scene given a set of images and camera matrices. The representation is a function of location and view angle, and returns a density and view-dependent color. Densities and colors sampled along a camera ray are aggregated using volumetric rendering to produce a pixel color. NeRF is popular in the computer vision and graphics communities with the applications in dynamic scene reconstruction [180, 181], image synthesis [182–184], pose estimation [185–187], and more. Optimizing NeRF to reconstruct a single scene can take hours or days—making it impractical for many robotics applications. Instant-NGP [179] and others [177, 188] speed up NeRF by using voxel feature grids instead of multi-layer perceptions to simplify or remove [178] a computational bottleneck. We build on Instant-NGP [179], which uses a learnable hash encoding and highly optimized CUDA implementation to speed

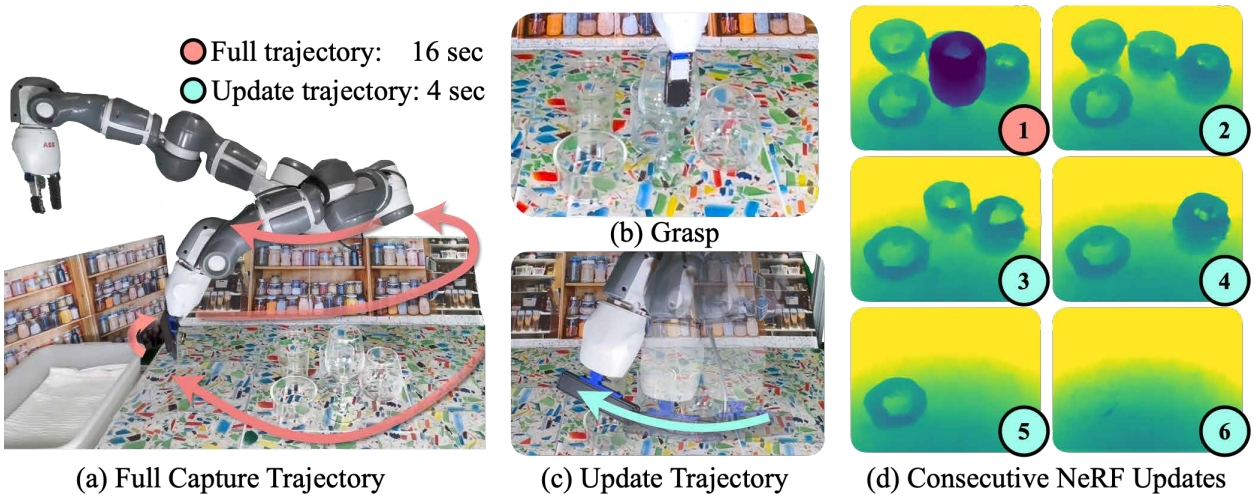
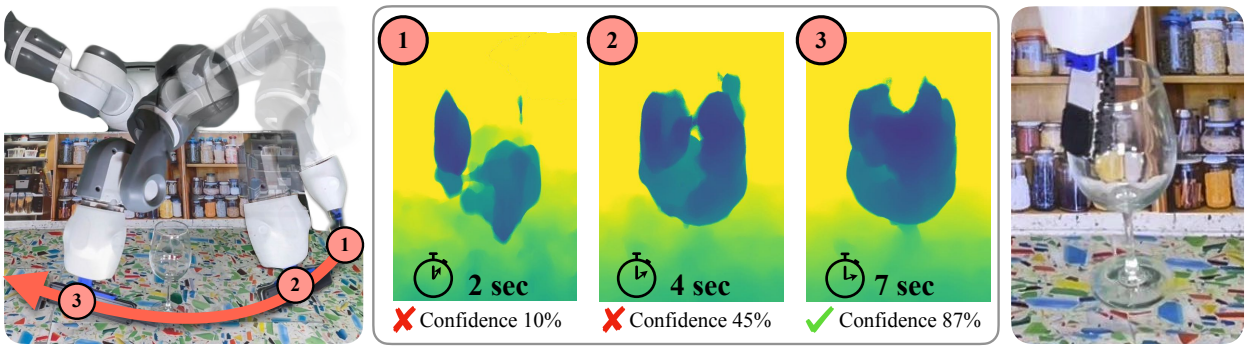


Figure 8.1: **Sequential object removal. (a)** The YuMi moves the camera through a hemisphere trajectory (red arrow) to capture a scene of 5 glass objects, training a NeRF simultaneously. **(b)** The robot immediately plans and executes a grasp from the NeRF after camera capture **(c)** Short camera trajectories are used to evolve the NeRF between grasps **(d)** Evo-NeRF first reconstructs the whole scene (1) with the camera trajectory shown in (a), then progressively updates the scene with small camera captures shown in (c) as objects are removed.

up NeRF training from the order of hours to seconds. Others have also sped up NeRF by reusing computation between scenes by utilizing priors. Existing methods [189–193] use convolutional neural networks (CNNs) to extract image features as input to a shared network that predicts the NeRF. Tancik et al. [194] and Gao et al. [195] speed up NeRF training using meta-learning to initialize network weights to ones that converge faster for likely scenes. In this work, we propose using past reconstructions of a scene as an initialization for the current reconstruction, allowing rapid adaptation to changes in the scene.

Recent research has shown NeRFs to be a promising scene representation for downstream robotics tasks such as navigation and SLAM [196–198] and manipulation [176, 199, 200]. Yen-Chen et al. [185] and Tseng et al. [201] use a trained NeRF model to estimate an object's 6-DOF pose by minimizing the residuals between a rendered image and a given observed image. Driess et al. [202] train a graph neural network to learn a dynamics model in a multi-object scene represented through a NeRF model, while Li et al. [199] condition a NeRF model on a learned latent dynamics model to plan to visual goals in simulated environments. We propose building on advances in NeRF and its applications to robotics to speed up NeRF-based grasping for practical uses.

Most closely related to this chapter are two recent works leveraging NeRFs to manipulate objects that cannot be detected by commodity RGBD sensors. Yen-Chen et al. [200] use a NeRF model offline to train dense object descriptors and manipulate thin and reflective objects. Ichnowski et al. [176] show that manually constructing an offline dataset of a given



(a) Early Stopped Image Capture (b) Online Training of Evo-NeRF (c) Grasp

Figure 8.2: **Evo-NeRF for rapid grasping: (a)** The robot begins capturing images along a hemisphere trajectory (red arrow) **(b)** Evo-NeRF trains a NeRF during arm motion, building graspable geometry of the wineglass. Grasp confidence from RAG-Net builds as NeRF learns geometry, reaching the stopping threshold at (3). **(c)** The robot executes the grasp predicted by RAG-Net at the early stop point.

scene then training NeRF allows off-the-shelf grasp planners [10] to compute successful grasps on transparent objects. ClearGrasp [175] trains a Sim2Real depth prediction network on RGB images, then uses this network in real environments to estimate surface geometry for grasps. This idea has been extended to pointclouds and with more efficient real-world data collection [203, 204]. GraspNeRF [205] explores neural rendering as supervision to train a multi-view feature volume network similar to Kar et al. [206] on photorealistic simulated scenes, which is used for grasping. In contrast, using NeRF directly in real-time does not require a prior on the scene at hand for reconstruction, and has superior performance on thin surfaces, occlusions, and complex backgrounds.

8.3 Problem Statement

Given a set of transparent objects resting on a planar workspace, the objective is for the robot to find, grasp, and remove each object quickly. Objects are placed close to each other (2.5 cm) and the robot has an actuatable camera and a parallel jaw gripper (Fig. 8.1). The focus is on finding robust grasps rapidly, with grasp success measured as transporting an object without dropping.

We assume (1) objects rest in graspable stable poses on a flat surface, (2) objects are in the reachable workspace of the robot with a known forward kinematic model, (3) the camera-to-arm transform is known and stable, and (4) the robot can follow a known obstacle-free trajectory to capture images.

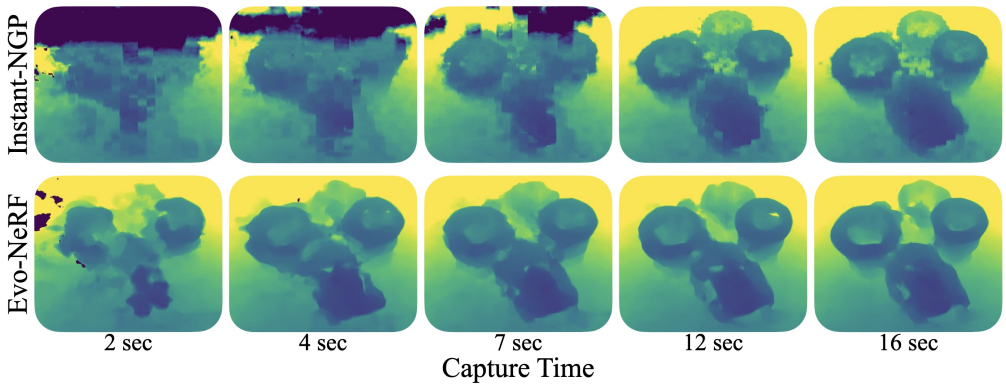


Figure 8.3: **Visual comparison** of Evo-NeRF's training over time vs Instant-NGP on the exact same camera trajectory. Evo-NeRF's geometry regularization improves the convergence of geometry reconstruction, resulting in fewer floaters, smoother surfaces, and ultimately faster grasps.

8.4 Method

To rapidly compute robust grasps, we propose *Evo-NeRF* and *RAG-Net*. Evo-NeRF, or *Evo*lving *NeRF*, builds on Instant-NGP [179], a fast implementation of NeRF, and modifies it to train concurrently to image capturing, to re-use NeRF weights between grasps and to terminate training and image collection early when sufficient task confidence if achieved. RAG-Net, or Radiance-Adjusted Grasp Network, is a network trained to compute grasps from geometry reconstructed from a NeRF.

8.4.1 Evo-NeRF

To shorten the time to get a trained NeRF, we propose Evo-NeRF, a method that pipelines image capture with NeRF training, reuses weights in sequential grasping, adds regularization to counter effects from rapid capture, and includes an early stopping condition to start a grasp when the grasp network has high confidence.

Image capture: The Evo-NeRF method starts with the robot moving a camera around its workspace to capture images. Heuristically, hemispherical captures are ideal for NeRF since they maximally vary the view angles of the scene. The Evo-NeRF capture trajectory sweeps the camera through a discretized hemisphere centered at a location of interest while pointing at the center. First, the camera sweeps around the z-axis to maximize the variance of viewing angle early in the capture sequence. In experiments, we capture images every 3 cm while moving at 20 cm/s. A full capture trajectory takes 16 seconds and includes 80 images with trajectory shown in Fig. 8.1. Though images have motion blur, stopping to take each image is time-consuming and would result in fewer images, yielding lower quality reconstructions. For dataset generation in simulation, we capture 52 images per scene since we prioritize having a large variety of scenes and rendering is time consuming.

Continual NeRF training: NeRF training, even sped up by Instant-NGP, is a bottleneck. We propose continually training NeRF from the moment the first image is captured, and incrementally adding images to the NeRF training dataset as the camera moves to new viewpoints. This effectively pipelines the image-capture and NeRF-training processes, and allows for usable NeRF representations quickly after (and sometimes before) the capture process finishes.

During each capture motion we train NeRF in batches of 48 steps, adding new images between each batch when available. This is akin to other online neural implicit methods like iMAP [196] and NICE-SLAM [207], who also update the image sets between training batches. We compute the camera frame using the forward kinematics and pair it with each image. In practice, this yields pose error around 1 cm, which NeRF accounts for by optimizing the camera extrinsics.

Reusing NeRF weights: In sequential grasping scenarios, scenes often change by only the removal of the last object grasped. To take advantage the information already trained, we use the NeRF network weights from the previous grasp in the subsequent grasp. In implementation, we remove the old images from the training dataset and start capturing and training on images for the next grasp.

Geometry regularization: A well-known artifact of NeRF's volumetric rendering loss are *floaters*, spurious regions of density floating in space. When using NeRF for view synthesis, floaters can go unnoticed, but in grasping, floaters can lead to grasp failures. We apply 2 regularizations which increase the speed and smoothness of geometry reconstruction, visualized in Fig. 8.3.

First, we adapt the total-variation regularization loss (TV-loss) from Plenoxels [208] to discourage floaters and encourage smooth scene geometry. During training, at each step Evo-NeRF sample N random points p_i using rejection sampling to constrain samples to locations with non-trivial density values. Evo-NeRF then queries the density at all 8-connected neighbors n_j at a radius r. The final TV-loss is $L_{\text{tv}} = \sum_{i=1}^{N} \sum_{j=1}^{8} \lambda_{\text{tv}}(\sigma(p_i) - \sigma(n_j^i))^2$, where σ is the raw, pre-activation output from the density network, and λ_{tv} is a loss scaling factor.

Second, sampling along each ray more coarsely during training reduces floaters and quickly acquires meaningful geometry. By training with coarse samples, the NeRF is incentivized to learn a low frequency representation of the scene to minimize reconstruction error.

Efficient perception stopping: In scenes where NeRF is able to recover usable geometry before the full camera trajectory has terminated, Evo-NeRF can terminate the capture phase early to speed task completion. In Sec. 8.5.2 we present experiments showing this by querying grasp confidence of RAG-Net in a closed loop while the robot moves the camera and trains NeRF. When confidence exceeds a threshold, the capture stops early and the robot executes the grasp.

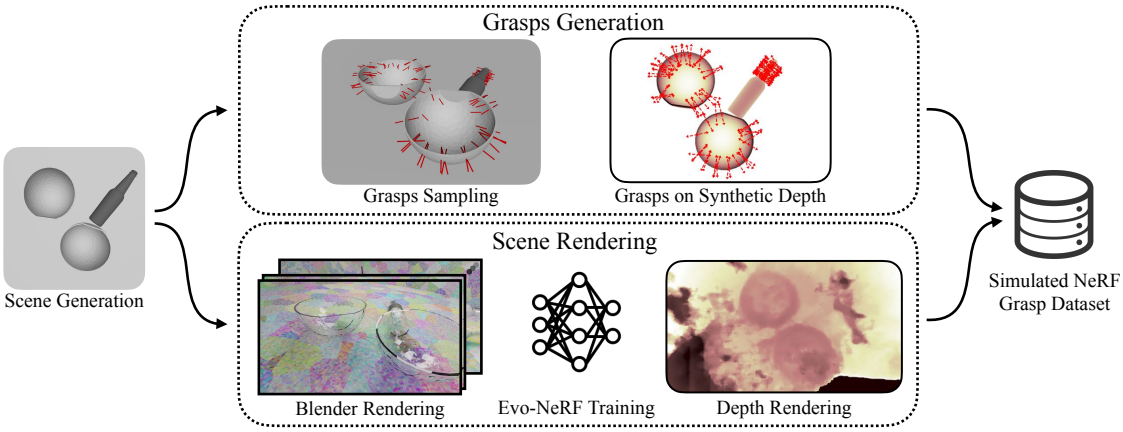


Figure 8.4: **Dataset Generation**. Each scene in the grasping dataset includes a subset of the training objects in simulation (Fig. 8.5). **Top:** Grasp generation samples grasps on the object meshes and projects them to a top-down view. **Bottom:** We render multiple views of each scene using Blender, then train Instant-NGP and render a top-down depth image. We accumulate NeRF depth rendering and projected grasps into a dataset.

8.4.2 Grasp Planning Network

When NeRF is trained to completion with dense camera viewpoints, grasp planners trained on ground truth depth in simulation like Dex-Net [209] produce usable grasps on NeRF-rendered depth. However, in an online setting where viewpoints are limited and NeRF training terminates early, depth images rendered from NeRF appear significantly different from the ground truth depth images in simulation. To mitigate this test-time distribution shift and enable reliable grasping from online NeRFs, we train a network to predict grasps directly on NeRF-rendered depth maps.

Network Architecture: We train a *location* neural networks to predict the center of the grasp location when given a NeRF-rendered image; and we train a *rotation* network to predict the discretied grasp angle when given a cropped patch around the grasp location. We adapt the grasping architecture proposed by Zhu et al. [210], which suggests that an equivariant convolutional neural network learns to perform top-down grasps in fewer samples than standard networks. We train location and rotation networks on a static grasp dataset, in contrast to the online setting in Zhu et al. [210].

Dataset Generation: We generate the training dataset in simulation using 7 object meshes that are representative of the common household transparent objects which are graspable by the YuMi robot, shown in Fig. 8.5a. We model all objects with the same density as glass (2500 kg/m^3). We assemble scenes with labeled grasp qualities by randomly placing objects in stable poses on a planar surface and analytically sampling antipodal grasp closure axes based on mesh surface normals as in Dex-Net [106]. We use a soft point-contact model [211], and evaluate the probability of grasp success using wrench resistance [212], a

(a) Training objects (Blender) (b) In-distribution real objects (c) Out-of-dist. objects (d) Clutter

Figure 8.5: **Training and testing objects. (a)** shows Blender rendering of the 7 objects we use in data generation for computing grasps and rendering in various stable poses. Objects in **(b)** are real objects we considered in-distribution with the training objects. We also test on out-of-distribution objects shown in **(c)**. To test grasping in clutter, we setup various testing scenes with objects in and out of distribution, with examples shown in **(d)**.

common analytic measure for grasp success that is computationally inexpensive (0.02 sec per grasp) and has high precision [213]. We densely sample 1000 collision free grasps for each stable pose and use Blender to render the scenes.

Training: To train RAG-Net, we project sampled grasps onto the depth images and store at each pixel the maximum grasp confidence over all rotations, resulting in confidence heatmaps. We dilate and blur these heatmaps with a 3x3 kernel to smooth the predictions, and randomly augment both the depth images and the confidence heatmaps with translation, shear and scale transformations. To train the rotation network, we sample crops from grasps above 0.7 quality, and use a cross-entropy loss on the output rotation probabilities.

Grasp Planning: To execute a grasp from RAG-Net we render a depth image from NeRF of size 144×256 from the camera pose used during dataset generation, using the ray transmittance truncation of Dex-NeRF [176]. We query the *location* network on this depth image to obtain a heatmap over the image of grasp confidence, then crop a patch of the depth image centered at the argmax of this heatmap. The *rotation* network takes this crop and outputs 8 grasp angle probabilities, and we take the weighted average of the argmax with its neighbors to produce the final grasp angle. We determine grasp depth by analyzing a local deprojected pointcloud from the depth image at the grasp location, and subtracting a static 1.5 cm grasp depth from the highest point.

8.5 Experiments

We evaluate the reliability of Evo-NeRF paired with RAG-Net vs Dex-Net [209], evaluate the speed improvements from early stopping captures and Evo-NeRF's reuse of weights, and ablate aspects of the system including NeRF modifications and training on NeRF depth vs ground-truth depth. We compare to Dex-Net to highlight the improvement in reliability gained from training on NeRF-rendered depth rather than ground-truth depth, and note

	Dex-Net Success	RAG-Net Success	Time
Full Capture	56 %	89 %	16s
Early Stop	11 %	89 %	9.5s

Table 8.1: **Single objects results:** each cell reports the average over 27 different trials. We compare success rates for full capture trajectories vs trajectories which stopped early because of sufficient grasp confidence. Early stopping results in a 41 % speed improvement with no drop in success rate for RAG-Net. Dex-Net struggles to reliably grasp on geometry rendered so early in NeRF training.

that in Dex-NeRF [176], the NeRF model was trained for 1900x longer, with an offline, manually captured set of images with precisely calibrated poses from Colmap [139, 140]. This difference in view quality and training length from rapid capture results in a notable drop in raw Dex-Net grasp robustness because of lower quality reconstructions.

8.5.1 Physical Setup

We evaluate on a physical YuMi robot with a ZED Mini camera. The pose of the ZED relative to the arm holding it is calibrated with a chessboard once before all experiments. We surround the robot with a kitchen-like workspace containing printed images of a countertop and shelves, where test objects are positioned near the center of the workspace. The workspace has 3 LED floodlights positioned across from the robot aiming at the workspace. We use one NVIDIA GeForce RTX 3080 GPU for NeRF training and grasp network inference. We evaluate on 9 different objects, both in distribution and out of distribution with respect to the train set in Fig. 8.5a. We note that in general, RAG-Net performance in simulated scenes is worse than in real scenes because the synthetic dataset contains fewer camera angles than real scenes (52 vs. 80) and more difficult background textures, resulting in more floaters.

8.5.2 Rapid single object retrieval

We apply confidence-based capture early stopping (8.4.1) with a threshold of 70% to execute a grasp as quickly as possible as shown in Fig. 8.2. We place each of the 9 test objects near the center of the workspace, and report grasp success and total time spent capturing images. We repeat each experiment 3 times and compare RAG-Net with Dex-Net [209] and evaluate with and without early capture stopping. Since Dex-Net does not output grasp confidence we use the same stopping point for both networks, as determined by RAG-Net. An experiment is successful if the robot grasps and places the object into the storage bin.

Table 8.1 summarizes the results. Using RAG-Net for early stopping results in a capture time reduction of 41 %, with no drop in reliability. On average, the robot grasps objects

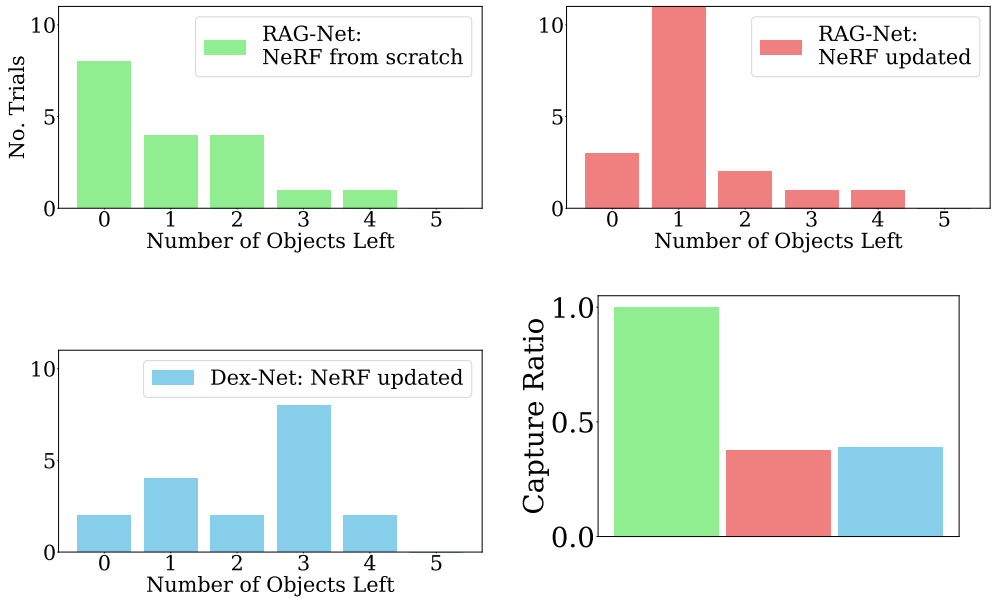


Figure 8.6: **Decluttering results.** Histograms show the number of objects remaining after each trial for RAG-Net **(top)** and Dex-Net **(bottom left)** where lower is better. **Bottom right:** Reusing and updating the NeRF between grasps (red, blue) rather than recapturing the scene (green) reduces capture time by 61% where lower is better.

within 9.5 seconds with an 89% success rate over 54 trials. RAG-Net outperforms Dex-Net in grasp success by 1.6x even with a full capture of the scene as a result of its habituation to NeRF density. RAG-Net's primary failure cases are on out-of-distribution objects, specifically missing grasps on the lightbulb and tape dispenser, likely because the training set has no small profile items. In addition, some grasps failed on the sideways wineglass because it rolled out of the jaws before they closed.

8.5.3 Sequential decluttering

We evaluate on a decluttering task where multiple transparent objects are placed within 2cm in stable poses, and the robot must grasp and place all objects in the bin one by one (Fig. 8.1). We consider three tiers of experiment difficulties with two scenes for each tier, resulting in 6 different scenes (Fig. 8.5). We repeat each scene 3 times and compare against Dex-Net [209]. At the beginning of each experiment, the robot executes a full capture of the scene (Fig. 8.1a). After each consecutive grasp, the robot executes a much smaller capture centered at the grasp location to update the NeRF (Fig. 8.1c). We allow only as many grasp attempts as objects in the scene.

Results are summarized in Fig 8.6, showing the number of remaining objects after each trial and the speedup from updating the NeRF rather than retraining. Evo-NeRF with

	Instant-NGP	Evo-NeRF -TV	Evo-NeRF -Coarse	Evo-NeRF
% Trajectory Used	80.3 %	64.8 %	62.0 %	**52.6 %**

Table 8.2: **Ablations of Evo-NeRF regularizations.** We query Dex-Net continuously through camera capture trajectories and report the percent of the trajectory needed until the highest probability grasp is on an object. We compare vanilla Instant-NGP with Evo-NeRF, as well as ablating TV-loss and coarse ray sampling. Evo-NeRF produces successful grasps the earliest.

RAG-Net clears 72 % of test objects across all tiers while Dex-Net clears 48 % of objects. Evo-NeRF takes 39% of the capture time compared to rebuilding the NeRF from scratch with full capture trajectories after each grasp, while maintaining similar performance (76 %). This suggests Evo-NeRF retains graspable geometry over successive updates, despite their short duration. The primary grasp failure modes for this experiment are the same as in single object experiments, but sometimes the method failed to remove an object if it was moved by more than 2-3cm from accidental contact, which wasn't detected by the smaller deletion captures used between grasps.

8.5.4 Graspability ablation

We ablate the changes made to NeRF speeding geometry graspability. We capture 9 single-object and 3 multi-object scenes, then continuously train NeRF as it captures, using the same static images and holding all other hyperparameters constant. We measure the capture time needed until the highest confidence grasp output from Dex-Net lands on a real object as a proxy for graspability convergence. Table 8.2 shows the percent of the capture trajectory needed, and Fig. 8.3 shows a timelapse of visual qualities over a capture. Results suggest that the proposed method produces graspable geometry faster, with a 32 % reduction in capture time needed to grasp using Dex-Net.

The grasp success labels used in this experiment were manually evaluated, where a human labeled a grasp as successful if its centerpoint lies on graspable geometry. To evaluate its reliability we executed grasps for single-object scenes from Evo-NeRF in the real world. All grasps labeled as successful were in fact successful (9/9), suggesting the metric used is physically reliable.

8.5.5 NeRF Depth vs Ground Truth Depth

This section investigates the distribution shift between training on ground-truth depth and testing on NeRF-rendered depth, to make the argument for training a grasp network directly on NeRF-rendered depth. We compare RAG-Net with two grasp planners: 1) Dex-Net, which is trained on a large dataset with ground-truth depth, and 2) GT-Net, which has the same architecture as RAG-Net but is trained only on ground-truth depth generated in simulation

with pyrender [214]. We test on the held-out test set of NeRF-rendered depth images and report average grasp confidence using the soft-point-contact model and wrench resistance. We calibrate the grasp planners' performance by evaluating GT-Net on ground-truth depth images, a scenario with no distribution shift, and then normalize the results of all planners with respect to this performance.

GT-Net, RAG-Net and Dex-Net achieves 0 %, 42 % and 0.1% success respectively, suggesting a large distribution shift between training on ground-truth depth to testing on NeRF-rendered depth. On low quality grasps with lower than 0.1 wrench resistance, the mean depth estimation error is 2.7cm, compared to 3-5mm for values over 0.1, suggesting a failure reason here is grasping floaters.

8.6 Discussion and Future Work

We introduced Evo-NeRF, a method that rapidly captures and trains NeRFs for practical robotic grasping. While its image capture produces lower-quality reconstructions than prior work, we propose reusing trained weights in sequential grasping, geometry regularization, and continual training to obtain better 3D reconstructions. We further propose a novel training pipeline to train grasp networks on NeRF rendered depth images in simulated environments, which can predict high quality grasps in the physical environment. In experiments, Evo-NeRF and RAG-Net can grasp transparent objects within 10s of seconds with 89 % success on singulated objects.

8.6.1 Limitations and future work

RAG-Net uses rendered depth images, throwing away much of the rich 3D information present in NeRF. Future work should explore 3D grasp planner inputs from NeRF such as density voxel grids, akin to VGN [215]. While hemispherical captures are efficient for reconstructing small workspaces, it may be unsuited to tasks like finding and extracting a target object from a large scene. Though we have shown that NeRF is adaptable to geometry deletion, NeRF still resists adding new geometry because of hash collisions in the positional encoding and the density gradient being pushed towards 0 in empty regions. In our experiments we observed a failure case where an object was toppled over by the grasped object, changing the scene's geometry. Future work in adapting NeRF to changing scenes would greatly improve the practicality of real-time usage. The speed of this method is also unsuitable for industrial applications requiring sub-second cycle times, and is mainly practical for household applications such as tidying which do not have such rapid requirements.

Part III

Efficiency and Reliability in Bimanual Deformable Manipulation

Chapter 9

Using Interactive Perception to Untangle Long Cables

Manipulating one-dimensional deformable objects like ropes or long cables is difficult due to the potential formation of knots and the limited information available from image observations caused by self-occlusions. This chapter explains how we apply Interactive Perception (IP) to reduce perception uncertainty and enhance the system's reliability in untangling long cables.

9.1 Introduction

Long cables, including electrical cords, ropes, and string, are ubiquitous in households and industrial settings [216–218]. These single-dimensional deformable objects can form knots that may restrict functionality or create hazards. Furthermore, as cable length increases, perception and manipulation of these objects become more difficult as the increased amount of free cable (which we refer to as *slack*) can cause the cable to not only fall into unreachable areas of the workspace, but also form complex knots and reach irrecoverable states. Further, retrieving full state information from image observations is especially challenging when slack occludes or falsely resembles true knots.

In our prior work, SGTM 1.0, we approach partial observability with manipulation primitives which attempt to simplify state for perception [219]. However, this approach lacks uncertainty awareness and takes actions that are often overly aggressive or conservative (see 9.2). To address this limitation, this chapter focuses on quantifying uncertainty to enable applying interactive perception [220], which involves physically manipulating objects in a scene to better understand it. By considering perceptual uncertainty, the robot is able to perform targeted actions that clarify the state and take subsequent actions with higher confidence.

This chapter provides the following contributions: (1) Novel perception-based metrics to estimate untangling-specific uncertainty in cable configurations, including tracing, network, and observational uncertainties as described in Section 9.4.2 while reducing reliance on depth

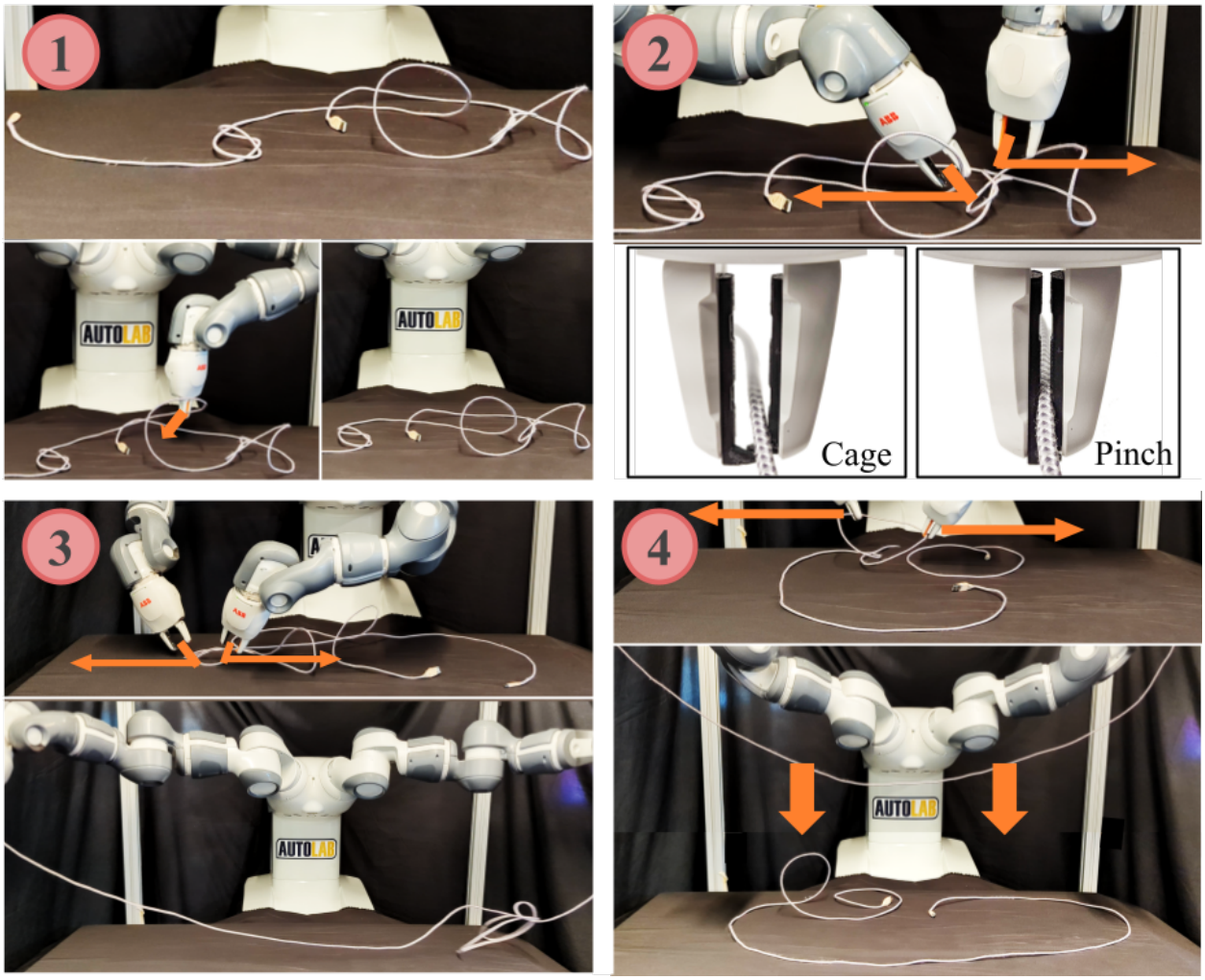


Figure 9.1: **Overview of Sliding and Grasping for Tangle Manipulation 2.0 (SGTM 2.0)**: SGTM 2.0 untangles a long cable with 2 figure-8 knots. (1) The system cannot perceive a clear path to a knot and performs an exposure move, bringing the endpoint cable segment back into the observable workspace. (2) SGTM 2.0 confidently untangles the figure-8 knot using cage-pinch dilation. (3) The system untangles the last figure-8 knot in the scene and does an incremental Reidemeister move. (4) SGTM 2.0 perceives a knot-like region and uses a partial cage-pinch dilation to disambiguate it. After another incremental Reidemeister move, the system terminates confidently having verified that no knots remain.

sensing; (2) Novel primitives, including interactive perception actions, for cable slack management, untangling, and termination described in Section 9.4.3 to reduce the probability of irrecoverable failures; (3) SGTM 2.0, an algorithm using uncertainty quantification to parameterize interactive perception actions for untangling described in Section 9.4.4 (overview in Figure 9.1); and (4) Data from physical experiments suggesting 43% improvement in untangling accuracy and 200% improvement in speed compared to SGTM 1.0, and data

suggesting that interactive perception improves accuracy by 21% in complex cases.

9.2 Related Work

Autonomous deformable object manipulation is an open problem in robotics. Deformable objects have infinite-dimensional configuration spaces, are prone to self-occlusions, and are subject to complex dynamics. An increasingly popular approach to these problems is end-to-end deep learning with imitation learning [221–223] or reinforcement learning [224–226]. Since large-scale physical data collection is difficult, one technique is training in simulation and deploying the learned policy on physical systems [16, 224, 225, 227–233]. However, there the sim-to-real gap remains large due to challenges in modeling deformable objects [227]. An alternative approach is to use self-supervised learning to collect the training data directly on the physical system [226, 234, 235].

In multi-step algorithmic pipelines, perception-based techniques have shown to be effective for deformable object manipulation. Prior work uses dense object descriptors [236] for cable knot tying [230] and fabric smoothing [237], as well as visual dynamics models for non-knotted cables [229, 238] and fabric [16, 17, 229]. However, robust cable state estimation and dynamics estimation remain challenging for self-occluded, knotted configurations. We build on prior keypoint-based work [219, 239, 240] with uncertainty-aware primitives and interactive perception for the task of autonomously untangling long cables.

Early work by Lui and Saxena [241] uses traditional feature extraction to represent a cable's structure as a graph. Recent work [219, 240, 242] use learning-based keypoint detection to parameterize action primitives in an untangling pipeline. Specifically, we improve upon Sliding and Grasping for Tangle Manipulation (SGTM 1.0) [219], an algorithm built on action primitives for autonomous long cable untangling. However, SGTM 1.0 may proceed with untangling despite low confidence in the predicted actions, leading to irrecoverable states. It also relies on shaking, a randomized reset primitive, when progress is difficult. Finally, it also requires a time-consuming physical trace to achieve the necessary confidence for termination. SGTM 2.0 addresses these three issues through *interactive* perception, by taking untangling actions sensitive to uncertainty, making targeted moves to reduce uncertainty instead of generic recovery moves, and terminating only once sufficiently certain across multiple views that the cable is untangled.

In 1984, Goldberg and Bajcsy [243] explored active perception of shape using a robot to actively move a touch sensor to trace object contours. In 1988, Bajcsy [118] defined *active perception* as a search of models and control strategies for perception. Strategies vary according to the sensor and the task goal, including controlling camera parameters [119] and moving a tactile sensor according to haptic input [243]. Recently, Bohg et al. [220] explore the differences between *active* and *interactive* perception, the latter of which specifically exploits environment interactions to simplify and enhance perception to achieve a better understanding of the scene [220, 244]. Within robotic manipulation, several works have focused on improving understanding of the environment through scene interaction. Tsikos

and Bajcsy [245] propose interacting with random heaps of unknown objects through pick and push actions for scene segmentation. Danielczuk et al. [246] present the mechanical search problem, where a robot locates and retrieves an occluded target object from a cluttered bin through a series of targeted parallel jaw grasps, suction grasps, and pushes. Novkovic et al. [244] propose a combination of camera motions with environment interactions to find a target cube hidden in a pile of cubes.

Interactive perception has also been applied to deformable manipulation. Willimon et al. [247] interact with a pile of laundry to isolate and identify individual clothing items. In our work, the robot interacts with the cable to reveal more information about the cable state.

9.3 Problem Statement

As in [219], we consider untangling long ($\sim 3\,\text{m}$) cables from RGB-D image observations. We use a bimanual robot to execute manipulation primitives until the cable reaches a fully untangled state with no knots.

9.3.1 Workspace Definition and Assumptions

The bilateral robot operates in an (x, y, z) Cartesian coordinate frame with two 6-DOF robot arms. The robot is equipped with cage-pinch jaws introduced in [219] to allow for both sliding along and tightly pinching the cable (Figure 9.1(2)). The workspace lies in the xy-plane and the only inputs to the algorithm consist of RGB-D images. The workspace contains a single incompressible electrical cable of length l_c and cross-sectional radius r_c. Cable state $s \in \mathcal{S}$ can be described as a continuous path $c_n(u) : [0, 1] \rightarrow (x, y, z)$ in the workspace, where u indexes the position along the length of the cable. $c_n(0)$ and $c_n(1)$ always refer to the position of the endpoints of the cable. We initialize the cable's state before $n = 0$ with the procedures specified in Section 9.5. One challenge in this problem is that parts of the cable may rest outside the reachable and observable workspace at any point in a rollout (defined as a single experiment aiming to remove all knots in the cable). Moreover, self-occlusions in the cable are possible due to only one overhead camera view. This partial observability motivates the need for actions that reveal more information about the cable state s.

We make the following assumptions: (1) the cable can be segmented from the background via color thresholding; (2) transformations between the camera, workspace, and robot frames are known; and (3) the cable start state contains overhand or figure 8 knots of dense (6-8 cm diameter) or loose (12-14 cm diameter) configurations in series.

9.3.2 Task Objective and Metrics

The goal of the robot is to untangle the cable and terminate at time $t < T_{\max}$, specified in Section 9.5. After each step of a rollout r, a new observation o of the cable state s is taken.

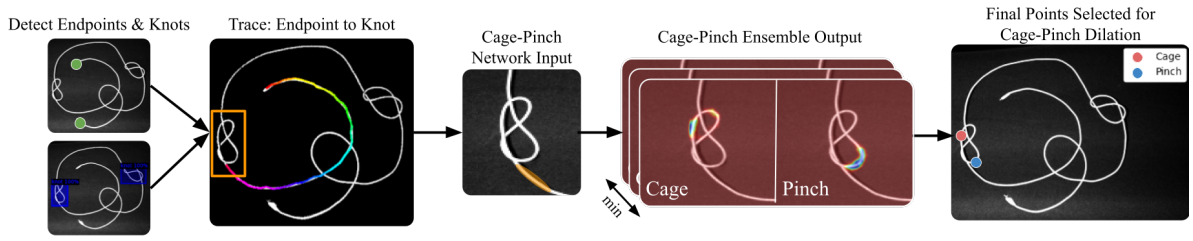


Figure 9.2: **Perception system:** This is the pipeline used to determine which points to cage and pinch for a cage-pinch dilation move, the crucial action for untangling a knot. First, we detect the endpoints and knots. Next, we trace from the endpoint to the first bounding box. If the trace is certain, we run the cage-pinch network ensemble on the cropped knot in the bounding box with the trace tail encoded into one of the channels. We take the pixelwise minimum across the cage-pinch network ensemble outputs, leading to 1 heatmap encoding "worst-case" untangling success probabilities each for the cage and pinch point. We take the argmax of each of the two heatmaps to determine the final points to pinch and cage during the cage-pinch dilation action.

Each primitive constitutes at least one step.

The goal of the robot over the course of each rollout is to untangle the cable and output a termination signal (`DONE`). We use $H_{\texttt{DONE}}$ to represent a step function, with the step occurring when the robot outputs `DONE`. We define an untangled cable as one that has no knots when its endpoints are grasped top-down and extended the maximum feasible distance, with knots defined identically to [219]. We use k_t^r to denote the number of knots in the cable at time t in rollout r and assume the cable is initialized with k_0^r knots.

We use the following metrics to measure performance, where $0 < K \leqslant k_0^r$ refers to the number of knots untangled and R is the total set of rollouts:

1. *Untangling K Success Rate,* the percentage of rollouts that untangle K knots: $\frac{1}{|R|}\sum_{r\in R}\mathbf{1}_{\{\exists t<T_{\max}\ :\ k_t^r\leqslant k_0^r-K\}}$
2. *Untangling Verification Rate*, the percentage of rollouts that untangle all knots and terminate successfully: $\frac{1}{|R|}\sum_{r\in R}\mathbf{1}_{\{\exists t<T_{\max}\ :\ k_t^r=0\ \wedge\ \texttt{DONE}_t\}}$
3. *Average Untangling K Time*, the average time to untangle K knots across all applicable rollouts R_a where this occurs before $T_{\max}$: $\frac{1}{|R_a|}\sum_{r\in R_a}(\min t\ :\ k_t^r \leqslant k_0^r - K)$
4. *Average Untangling Verification Time,* the average time to reach $k_t^r = 0$ and declare termination across all applicable rollouts R_a like above: $\frac{1}{|R_a|}\sum_{r\in R_a}(\min t\ :\ k_t^r = 0 \wedge \texttt{DONE}_t)$

9.4 Method

9.4.1 Approach Overview

Unlike SGTM 1.0, SGTM 2.0 uses interactive perception primitives designed to better manage slack during untangling and to reveal additional information about the cable state $s \in \mathcal{S}$. As s is difficult to estimate from the provided observation $o \in \mathcal{O}$, SGTM 2.0 uses a policy $\pi : \mathcal{O} \rightarrow \mathcal{A}$ built as described in Section 9.4.4 from the components in Sections 9.4.2 and 9.4.3 to directly predict actions to execute. Unlike prior work, SGTM 2.0 includes perception components that lend themselves to probabilistic interpretation and manipulation primitives that are sensitive to perception uncertainty. We note that while the distribution of states the robot encounters may have high variance, SGTM 2.0 is sensitive only to variance that may affect the next untangling action. Practically, this means that for example, even if much of the cable is bunched up and occluded, as long as the path from an endpoint to the first knot is clearly visible, the algorithm can still take an untangling action with high confidence.

9.4.2 Uncertainty-Aware Perception Systems

Endpoint Detection

We train a Faster R-CNN model with a Resnet-50 FPN (Feature Pyramid Network) backbone [248] on 305 hand-labeled examples to detect cable endpoints. We discard all bounding boxes with lower than 99% confidence, achieving an average precision and recall for endpoint detection are 86.7% and 100% respectively.

Knot Detection

To identify all knots in the observable workspace, we use the same architecture as the endpoint model trained on 688 hand-labeled images. The real-world dataset is augmented with flip, contrast, brightness, rotation, saturation, and scale augmentations. We use a 99% detection threshold, which achieves an average precision of 91.3% and an average recall of 95.5%. We analytically filter out misclassified knots by checking if a simple loop fills the bounding box. Because the model's output is dependent on the orientation of the cable, in certain cases, we use multiple observations of the underlying cable state s as described in Section 9.4.3. This allows SGTM 2.0 to be sensitive to what we define as *observational uncertainty*.

Analytic Cable Tracing

The objective of cable tracing is to outline all likely paths of the cable from an overhead image. Given an RGB image and a start pixel (the center of one endpoint), the tracer we introduce in this chapter outputs a set of possible splines. It maintains a set of valid splines and iteratively expands it by exploring candidate successor points, preferring those that do

not deviate sharply from the current spline's trajectory. An example of candidate trace paths is shown in Figure 9.3.

Sliding and Grasping for Tangle Manipulation 2.0 (SGTM 2.0) uses this in 2 scenarios: (1) finding a knot to untangle and (2) achieving robust grasps near cable endpoints. When used for finding a path from endpoint to knot, the tracer terminates once all potential traces intersect the knot within a bounding box. For grasping endpoints, the trace terminates after traveling a fixed distance along the cable. At the termination of tracing, we fit a bounding box B_t to the ending points of all the final traces; if either dimension of B_t is greater than 24 pixels for knot tracing and 12 pixels for endpoint tracing (which requires more precision), the traces diverge and thus `TRACE_UNCERTAIN` is returned; otherwise, `TRACE_CERTAIN` is returned. Note that while traces with very different topologies may be returned, as long as they end near the same point (Figure 9.3, left and center), the untangling-relevant uncertainty remains low. The reason for tracking uncertainty in this phase is that different paths from the endpoint to the knot may lead to different untangling actions, especially with respect to which point to cage and which to pinch.

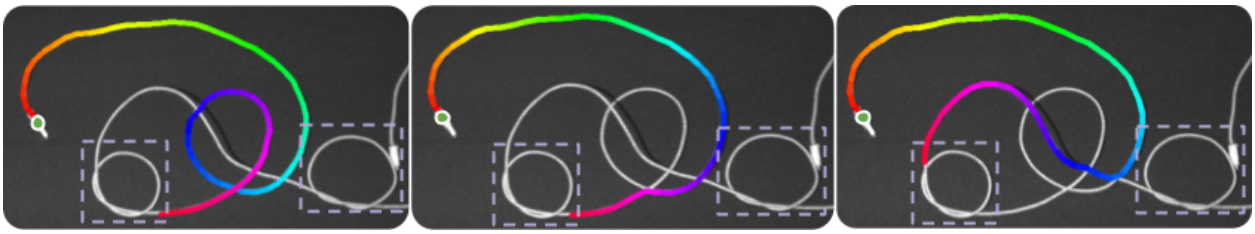


Figure 9.3: **Cable Tracing:** multiple candidate trace paths returned by the tracer to find the closest knot from the top-left endpoint. The tracer finds the correct knot in all cases, but is unclear as to which side of the knot is attached to the free endpoint, and therefore returns `TRACE_UNCERTAIN`.

Cage-Pinch Dilation Point Selection

We use an FCN [249] with a ResNet-34 backbone trained on 568 knot crops to output two heatmaps: one for the cage point and one for the pinch point. We augment this data with random rotations, flips, shear, and synthetically added distractor cables to improve robustness. Approximately 250 of these images are gathered during rollouts to mitigate distribution shift in a style similar to DAgger [250]. The input is a 3-channel image, where one of the channels contains a Gaussian heatmap around the segment of cable entering the image determined by the cable tracing algorithm. This additional input conditions the network and breaks the symmetry between the cage and pinch points. We train the network to predict the cage point as the first graspable point beyond the undercrossing forming the knot, and the pinch point as the place to secure the cable to create an opening for the free end to slide through. Example cage and pinch points and the perception pipeline to obtain the points are shown in Figure 9.2.

The goal of this network is to model the probability $P_p(U_s)$ per grasp pixel p, where U_s corresponds to untangling success on the cropped knot. We train an ensemble of 3 models on the same data but with different initializations. For the cage point, to sample from each of the ensemble heatmaps $h^{\text{cage}} \in H^{\text{cage}}$, we create a new heatmap as such: $h'^{\text{cage}}_{i,j} = \min_{h^{\text{cage}} \in H^{\text{cage}}} h^{\text{cage}}_{i,j}$. The same procedure is used for the pinch point. We return the cage and pinch point as the $\text{argmax}_{i,j}\, h'^{\text{cage}}_{i,j}$ and $\text{argmax}_{i,j}\, h'^{\text{pinch}}_{i,j}$, respectively. If $\max_{i,j} h'^{\text{cage}}_{i,j} \max_{r,s} h'^{\text{pinch}}_{r,s} < \kappa$ where $\kappa = 0.35$ (calibrated empirically), we output `NETWORK_UNCERTAIN`. Otherwise, we output `NETWORK_CERTAIN`.

The above methods help reveal whether the worst-case probability (across our predictive distribution modeled by an ensemble) of untangling success is high enough to proceed with untangling the cable at the specified points. If not, the network is too uncertain in its predicted points to proceed confidently as the next action may instead tighten the knot or lead the cable into an irrecoverable state.

9.4.3 Novel Manipulation Primitives for Interactive Perception

Cage-Pinch Dilation

To untangle an individual knot, the robot leverages the flexibility of the cage-pinch grippers introduced in [219] and depicted in Figure 9.1(2) to cage one point and pinch another point inside the knot and pull apart the arms to a distance determined by the length of the trace from the endpoint to the knot, while moving its wrist joint in a high-frequency sinusoidal motion. A major benefit of cage-pinch actions compared to cage-cage actions from [219] is the ability to better manage slack, preventing accidentally tightening another knot. Following this action, the robot lays the cable down as far forward as kinematically feasible to isolate the newly untangled portion from the remaining cable.

Partial Cage-Pinch Dilation

This primitive is similar to the Cage-Pinch Dilation, but the distance the arms move apart is fixed to 5 cm beyond their starting separation. This is meant to perturb the state and to later retry perception rather than to completely untangle a knot.

Reidemeister Move

In this primitive, the robot uses tracing to find robust grasp points slightly down the cable from the endpoints. Next, the robot moves both arms outward horizontally and up, lifting the cable off the workspace, allowing for loops to fall away. Compared to prior work, we add (1) the vertical component of the action, forming a large "U" with the cable, and (2) the wide lay-out action, which places the cable on the workspace in a "U" shape.

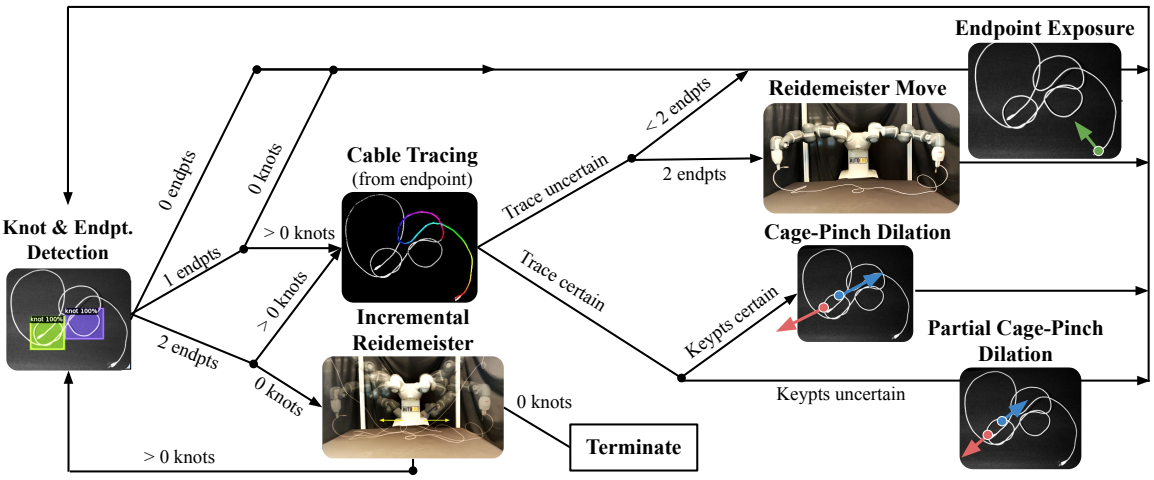


Figure 9.4: **SGTM 2.0 Algorithm:** SGTM 2.0 first detects the number of knots and endpoints in the scene. If the endpoints are not visible, there is no way to verify any knot's relative position to the endpoint. This is necessary because SGTM 2.0 only untangles knots adjacent to an endpoint to avoid knots colliding into each other and creating irrecoverable configurations. If fewer than two endpoints and no knots are visible, the algorithm is also unable to perform a termination check as that requires performing an incremental Reidemeister, which grasps the cable at the endpoints. In both these cases, SGTM 2.0 performs an endpoint exposure. If two endpoints are visible and no knots are visible, SGTM 2.0 proceeds to the incremental Reidemeister move. If one or two endpoints are visible and there are knots in the scene, it attempts to untangle, beginning by tracing from the visible endpoint(s). Here, if it is not able to confidently trace from either endpoint to a knot, SGTM 2.0 performs a Reidemeister move or endpoint exposure (based on the number of endpoints visible) to increase likelihood of unambiguous traces in future steps. Otherwise, it assesses the cage-pinch network uncertainty on the predicted points. If it is confident, it proceeds with a full cage-pinch dilation. Else, it performs a partial cage-pinch dilation to disambiguate the state.

Incremental Reidemeister Move

This primitive performs the exact same motions as a Reidemeister move, but uses a multi-stage, perception-based approach where the cable is observed at certain waypoints along the action. We use our knot detection network at these intermediate points to determine whether any knots remain. This can be interpreted as ensembling via perturbation of the observation of the same underlying cable topology. Being sensitive to observational uncertainty allows us to eliminate the time-consuming physical tracing action used in [219] for termination.

Exposure Action

When one or more endpoints are missing for an action, we uniformly at random sample a segment of the cable leaving the reachable workspace and pull it towards the center of the workspace to increase visibility. We also do this for unreachable knots we wish to act on.

9.4.4 Sliding and Grasping for Tangle Manipulation 2.0 (SGTM 2.0) Algorithm

Sliding and Grasping for Tangle Manipulation 2.0 (SGTM 2.0) ties together the aforementioned perception components and manipulation primitives to untangle cables. SGTM 2.0 alternates between the perception and manipulation components, using uncertainty from the former to determine whether to untangle or disambiguate the cable state. The algorithm is covered in detail in Figure 9.4.

9.5 Experiments

9.5.1 Experimental Setup

For our experiments, we use the bimanual ABB YuMi robot with an overhead Phoxi camera, operating on a black foam-padded workspace of width $1.0\,m$ and depth $0.75\,m$. Due to hardware constraints, we slightly extend the workspace with cardboard (by 0.1 meters on either side) not previously present in SGTM 1.0, but this does not make a significant difference as the cable mostly remains in the original foam-padded workspace. We use a $2.7\,m$-long white, braided electrical cable with USB adapters on both ends.

We evaluate SGTM 2.0 on 3 tiers of difficulty:

1. **Tier 1:** A cable with 1 overhand or figure-8 knot.
2. **Tier 2:** A cable with 2 overhand and/or figure-8 knots.
3. **Tier 3:** A cable with 3 overhand and/or figure-8 knots.

The knots in all tiers are evaluated equally in both loose and dense configurations and evenly across positions along the cable (closer to an endpoint vs. closer to the middle). Example start configurations are shown in Figure 9.5. The cable is initialized by laying the knot(s) flat on the workspace, raising the endpoints as high as possible without lifting the knot(s), and then dropping the endpoints. We enforce a time limit of 15 minutes for all tiers. Note that the cable initialization procedures in Tiers 1 and 2 of this chapter are *exactly* the same as Tiers 1 and 2 in SGTM 1.0, the prior state-of-the-art [219].

For each tier, we report the average time to fully untangle the cable (for the rollouts that succeed in doing so) as well as the average time to correctly report that the cable is untangled (for the rollouts that succeed in doing so). Additionally, we report the success rates for untangling alone and untangling with termination detection. For Tiers 2 and 3, we present ablation results where SGTM 2.0(-U) represents SGTM 2.0 with the uncertainty-based components removed. For Tiers 1 and 2, we also report the speedup of SGTM 2.0 and SGTM 2.0(-U) from SGTM 1.0. Our baseline for experiments is SGTM 1.0 because to our knowledge, there are no prior algorithms for untangling long cables.

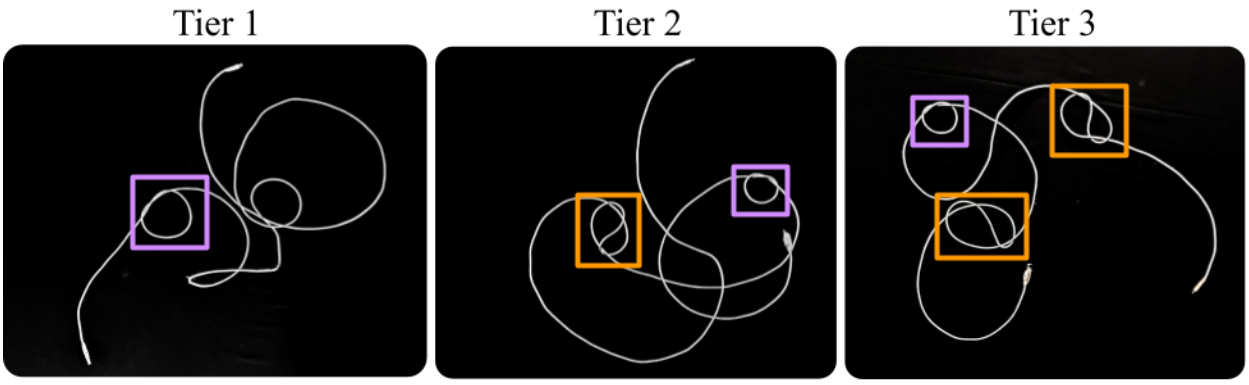


Figure 9.5: **Example starting configurations for all 3 tiers:** Overhand knots are outlined in purple and figure-8 knots are outlined in orange.

Table 9.1: Results from Physical Experiments (84 total trials). Key takeaways include 1) higher success rates in SGTM 2.0, which ablations suggest is a result of interactive perception. Reduced completion times compared to SGTM 1.0 because of novel interactive manipulation primitives.

	Tier 1		**Tier 2**			**Tier 3**	
	SGTM 1.0	SGTM 2.0	SGTM 1.0	SGTM 2.0(-U)	SGTM 2.0	SGTM 2.0(-U)	SGTM 2.0
Knot 1 Success Rate	8/12	**10/12**	10/12	10/12	**12/12**	12/12	12/12
Knot 2 Success Rate	-	-	6/12	7/12	**10/12**	9/12	9/12
Knot 3 Success Rate	-	-	-	-	-	3/12	3/12
Verification Rate	3/12	**7/12**	1/12	5/12	**7/12**	0/12	**2/12**
Avg. # of Actions	**5.0±1.5**	5.6±1.7	12.0	**6.0±1.2**	7.7±1.6	N/A	**12.0±0.0**
Avg. Knot 1 Time (s)	189.8±69.4	**53.1±7.5**	93.3±16.2	128.6±41.9	**75.6±21.0**	88.7±25.5	**69.8±15.1**
Avg. Knot 2 Time (s)	-	-	586.4±165.3	**160.9±26.4**	180.9±26.2	**177.3±28.3**	233.1±57.6
Avg. Knot 3 Time (s)	-	-	-	-	-	**417.7±104.1**	476.7±160.1
Avg. Verif. Time (s)	330.8±127.8	**295.9±61.4**	1079.0	**359.6±74.6**	406.9±22.0	N/A	**704.5±13.5**
Failure Modes (See Section 9.5.3)	-	(A) 2, (B) 1, (C) 1, (D) 1	-	(A) 1, (B) 0, (C) 5, (D) 1	(A) 2, (B) 1, (C) 1, (D) 1	(A) 1, (B) 4, (C) 6, (D) 1	(A) 5, (B) 2, (C) 2, (D) 1

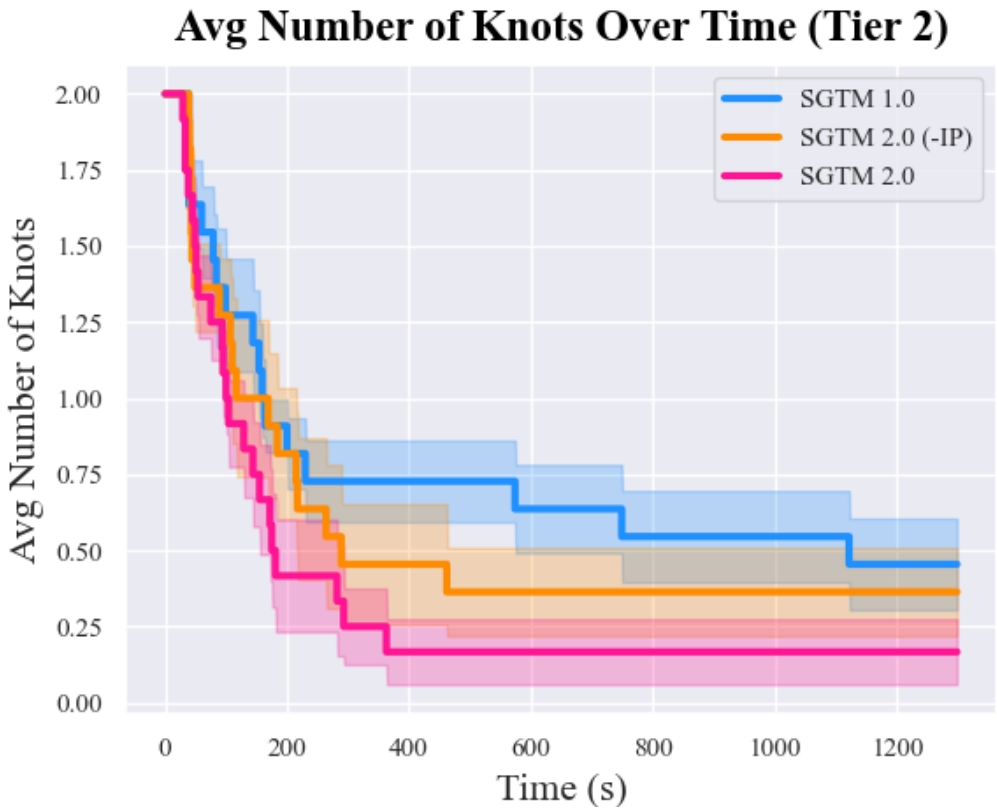


Figure 9.6: **Average Knots over Time (Tier 2)**: SGTM 2.0 most quickly reduces the number of knots on average. SGTM 2.0(-U) is less successful in untangling over the given time period than SGTM 2.0. Further, performance of even SGTM 2.0(-U) exceeds that of the prior state-of-the-art, SGTM 1.0, at 15 minutes, showing the effectiveness of improved untangling primitives like cage-pinch dilations introduced in SGTM 2.0.

9.5.2 Results

Results show that across Tiers 1 and 2, SGTM 2.0 outperforms SGTM 1.0 not only on untangling and verification success rate, but also achieves statistically significant speedups in the untangling time in Tier 1, untangling time for both knots in Tier 2, and verification time in Tier 2. Tier 3 was unachievable in SGTM 1.0 due to algorithmic constraints, but is now possible with SGTM 2.0, which achieves 9/12 successes in untangling 2 out of the 3 knots in Tier 3. In 9.5.3, we discuss difficulties that result in failures, leading to 3/12 untangling success on all 3 knots in Tier 3.

9.5.3 Failure Modes

(A) Timeout, unable to determine termination: This is the most common failure case of SGTM 2.0 across all tiers, especially in Tier 3. Causes include:

1. The system inadvertently manipulates the cable into a state that is difficult to perceive and manipulate, most common in Tier 3 due to higher complexity and a higher chance that a rare failure in disambiguation may accidentally tighten or complicate a knot.

2. A substantial portion of the cable leaves the workspace. The system repeatedly attempts exposure actions, but due to the large mass of cable, the cable continually slips back down into its prior configuration.

3. Though the entire cable may enter a difficult configuration, the algorithm slowly disambiguates it and given more time, may have untangled and terminated.

(B) Cable or knot leaves observable/reachable workspace: This is the next most common failure mode of SGTM 2.0. While performing a cage-pinch dilation or Reidmeister move, one gripper may miss the grasp, causing the entire cable to slide to one side of the workspace and fall off entirely and irrecoverably. This failure mode shows that slack management can be improved in future work.

(C) False termination due to missed knot detection: False termination is the most common failure mode in SGTM 2.0(-U), mostly resolved by SGTM 2.0 with uncertainty-based components. This failure also occurs in SGTM 2.0, largely due to rarer cases where the knotted portion inadvertently lands outside the observable workspace during an incremental Reidemeister move, causing early termination. This can be addressed with improved motion primitives.

(D) Irrecoverable YuMi system error: These relatively rare issues result from the YuMi losing connection to the computer running the algorithm and freezing.

9.5.4 Ablations

We run ablations on Tier 2 and Tier 3 to compare the performance of SGTM 2.0 to the performance of SGTM 2.0(-U), which uses the exact same algorithm as SGTM 2.0, but with the following uncertainty-based components removed:

- Reidemeister move due to tracing uncertainty.

- Ensemble network for keypoint predictions and partial cage-pinch dilation in the case of ensemble uncertainty in the cage-pinch dilation network.

- Intermediate views for incremental Reidemeister move.

We find, as shown in table 9.1, that SGTM 2.0(-U) achieves a lower success rate than SGTM 2.0 on Tier 2. While SGTM 2.0 and SGTM 2.0(-U) achieve the same success rate

on Tier 3, the main failure case for SGTM 2.0 is timeout as higher complexity cases tend to require more time to disambiguate and untangle. In comparison, the main failure case for SGTM 2.0(-U) are false termination. In fact, the most common failure case in SGTM 2.0(-U) across Tier 2 and 3 is false termination, suggesting that sensitivity to observational uncertainty may be important for higher performance. Another implicit failure that results in more false terminations is over-tightening of knots to a diameter $\leqslant 3cm$. If the ensemble cage-pinch network has low confidence, SGTM 2.0 performs a partial cage-pinch dilation rather than a full cage-pinch dilation, preventing over-tightening knots in the case of poorly predicted cage-pinch points. Additionally, if the trace to a knot is uncertain, the Reidemeister move disambiguates the cable state, preventing poor grasps that may tighten or complicate the knots. The ablations suggest that these uncertainty-based primitives may prevent the over-tightening of knots and thus reduce false terminations.

9.6 Discussion and Future Work

The perception approach in this chapter is limited to overhand and figure-8 knots due to reliance on geometric rather than topological features. Additionally, while perception system operates on just RGB data, grasping is reliant on depth. Lastly, this algorithm has been tested on only a single white cable on a black background.

In future work, we will explore cable state estimation and topological knot analysis to generalize to other knot types, and RGB visual servoing methods that could eliminate reliance on depth sensing altogether.

Chapter 10

Learning Efficient Bimanual Folding of Garments

Manipulating two-dimensional deformable objects, such as cloth and garments, is challenging because of their large configuration space, self-occlusions, and complex dynamics. This chapter details how we use a bimanual robot and utilize high-velocity dynamic motions to create a system that folds garments both reliably and efficiently.

10.1 Introduction

Garment handling such as folding and packing are common tasks in textile manufacturing and logistics, industrial and household laundry, healthcare, and hospitality, where speed and efficiency are key factors. These tasks are largely performed by humans due to the complex configuration space as well as the highly non-linear dynamics of deformable objects [231, 251]. Additionally, folding is a long horizon sequential planning problem, as it requires to first flatten or smooth the garment, and then follow a sequence of steps [252, 253] or sub-goals [19] to achieve the desired fold.

Prior work has mainly focused on single-arm manipulation [16, 227, 231, 254] or on complex iterative algorithms [252, 253, 255], requiring a large number of interactions and resulting in long execution times. Recently, Ha et al. [15] proposed a method for smoothing cloth that computes the pick points for a high-velocity dynamic fling action directly from overhead images, and can smooth garments to 80 % coverage in 3 actions on average. However, the proposed 4 DoF action parameterization constrains the two pick poses significantly, in particular by discrete distances and a fixed rotation in between.

We present SpeedFolding, an end-to-end system for fast and efficient garment folding. At first, a novel BiManual Manipulation Network (BiMaMa-Net) learns to predict a pair of gripper poses for bimanual actions from an overhead RGBD input image to smooth an initially crumpled garment. Once the garment has been smoothed to a desired level, determined by a learned smoothing classifier, SpeedFolding executes a folding pipeline (see Fig. 10.1).

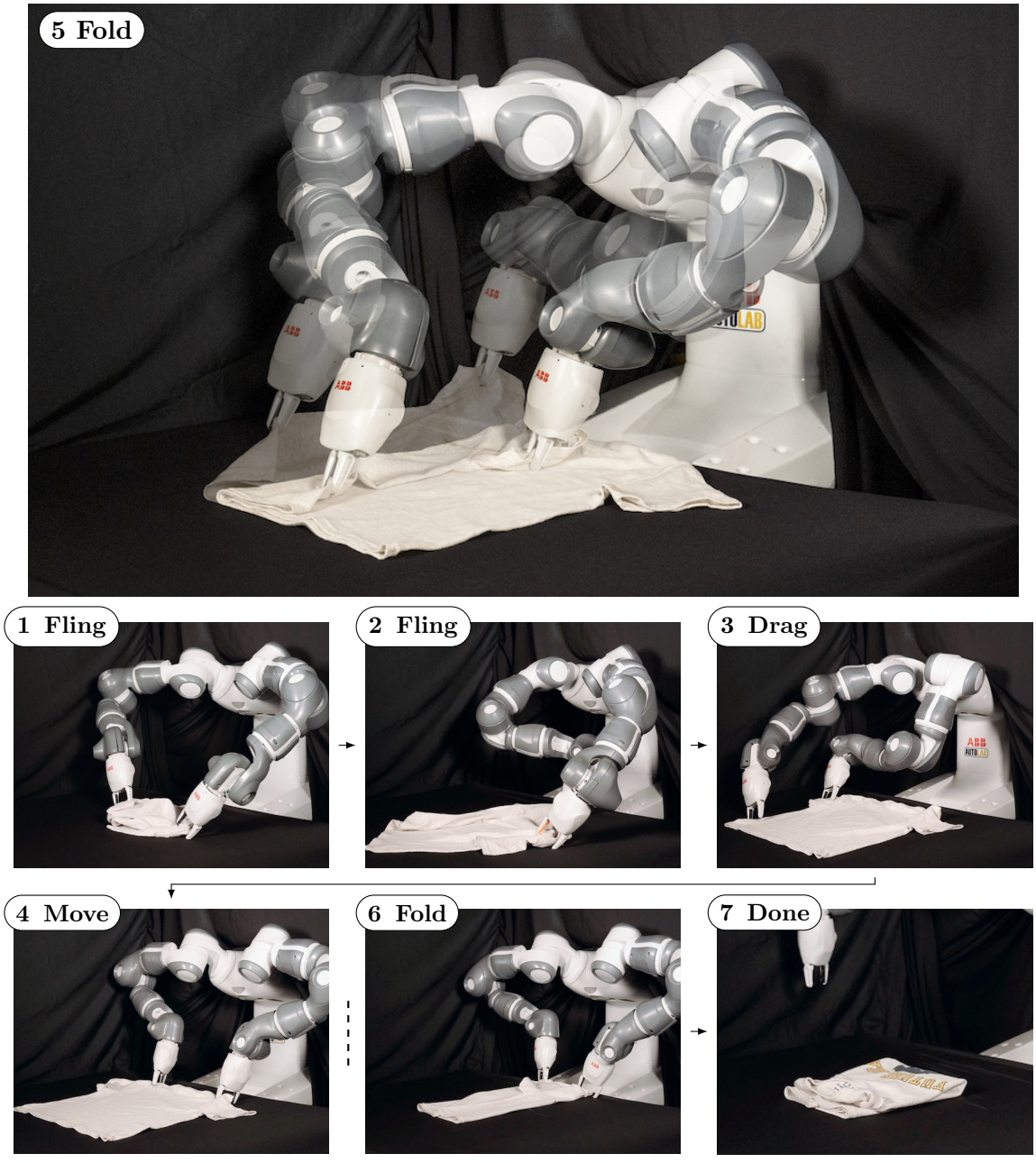


Figure 10.1: SpeedFolding learns to fold garments from arbitrary configurations: Given a crumpled t-shirt, the robot unfolds using fling actions (1, 2), smooths it with a drag action (3) until it is *sufficiently smoothed*. It then moves the t-shirt for better reachability (4), and applies folds (5-6) to achieve the user-defined configuration (7).

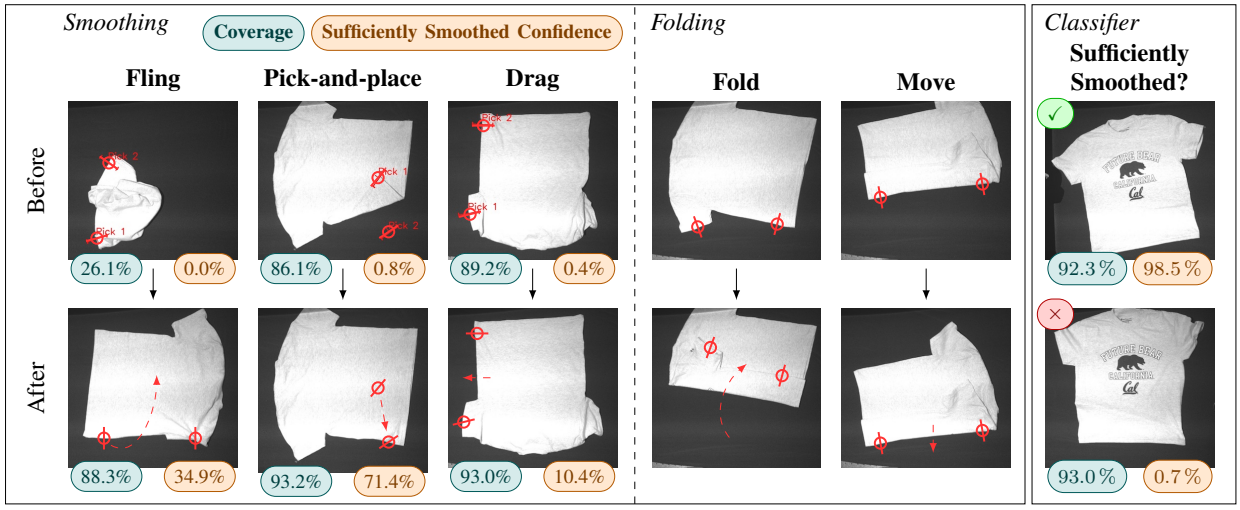


Figure 10.2: **Action Primitives.** Given an overhead RGBD image, BiMaMa-Net selects a smoothing action from a discrete set of primitives (**left** box), and computes a pair of end-effector poses. Coverage calculation (in blue) is insensitive to wrinkles in the fabric (**right** box, **bottom**) and therefore BiMaMa-Net learns to classify configurations as Sufficiently Smoothed (**right** box, **top**) (in yellow). Folding a t-shirt from a smooth configuration is done through a sequence of folding primitives (**center** box).

This chapter provides the following contributions: (1) The BiMaMa-Net architecture for bimanual manipulation that computes two *corresponding* planar gripper poses without any spatial restrictions, with an automated calibration procedure to account for robot reachability constraints; (2) An end-to-end robotic system for efficient smoothing and folding. First, the system learns to smooth a garment to a sufficiently smoothed configuration through self-supervision. Then, the robot folds the garment according to user-defined folding lines; and (3) An experimental dataset from physical experiments that suggests the system can fold garments with a success rate of over 90 %, including garments unseen during training that differ in color, shape and stiffness. Folding a t-shirt takes under 120 s on average, improving baselines by 30 % to 47 % and prior works by 5 × to 10 ×.

10.2 Related Work

Bimanual robotic manipulation has been studied extensively in fields from surgical robotics to industrial manipulation [256]. A dual-arm system extends the workspace, allows for increased payload and for more complex behaviours than a single arm system [19–22], but comes at the cost of higher planning complexity due to the additional DoF and self-collisions [257]. A promising line of research is to employ dual-arm systems for garment manipulation [258]. Garments are especially difficult to control and manipulate due to their large configuration

space, self-occlusions, and complex dynamics [231]. Recent works have mainly focused on garment smoothing from arbitrary configurations [15], or garment folding, assuming the garment has been initially flattened [19]. We present an end-to-end approach to smoothing and then folding garments from initial crumpled configurations.

Garment smoothing aims to transform the garment from an arbitrary crumpled configuration to a smooth configuration [227]. Prior works have focused on extracting and identifying specific features such as corners and wrinkles [252, 253, 259, 260]. Recent methods have used expert demonstrations to learn garment smoothing policies in simulation [16, 227, 231], however these methods learn quasi-static pick-and-place actions that require a large number of interactions on initially crumpled garments. Ha et al. [15] introduced a novel 4 DoF dynamic fling action parameterization learned in simulation that can achieve $\sim 80\,\%$ garment coverage within 3 actions. However, this parameterization is (1) limited to fling actions, (2) fails to fully smooth garments, and (3) induces grasp failures in more than $25\,\%$ of actions. In this chapter we use expert demonstrations and self-supervised learning purely in the physical world to train a novel bimanual manipulation neural network architecture to smooth a garment such that it is ready to be folded.

Garment folding has many applications in hospitals, homes and warehouses. Early approaches rely heavily on heuristics and can achieve high success rates, but have long cycle times on the order of 10 min to 20 min per garment [252, 253, 255, 261, 262]. Recent methods have been focusing on learning goal-conditioned policies in simulation [16, 19, 228, 263] and directly on a physical robot [226]. In this chapter, we compare an instruction-based folding approach that can reliably fold smoothed garments, with a novel folding approach that can fold a t-shirt directly from a non-smooth configuration given prior knowledge about its dimensions.

10.3 Problem Statement

Given a visual observation $o_t \in \mathbb{R}^{W \times H \times C}$ of the garment's configuration s_t at time t, the objective is to compute and execute an action a_t to transfer the garment from an arbitrary configuration to a desired user-defined s^* goal configuration. In particular, s^* is invariant under the garment's position and orientation in the workspace. We assume an overhead observation with a calibrated pixel-to-world transformation, as well as a garment that is easily distinguishable from the workspace.

We consider a dual-arm robot with parallel-jaw grippers executing actions of type $m \in \mathcal{M}$ from a discrete set of pre-defined action primitives. In particular, we parameterize each primitive by two planar gripper poses

$$a_t = \langle m, (x_1, y_1, \theta_1), (x_2, y_2, \theta_2) \rangle$$

for each arm respectively, in which (x_i, y_i) are coordinates in pixel space, and θ_i is the end-effector rotation about the z axis. We further assume a flat obstacle-free workspace and a motion planner that computes collision-free trajectories for a dual-arm robot.

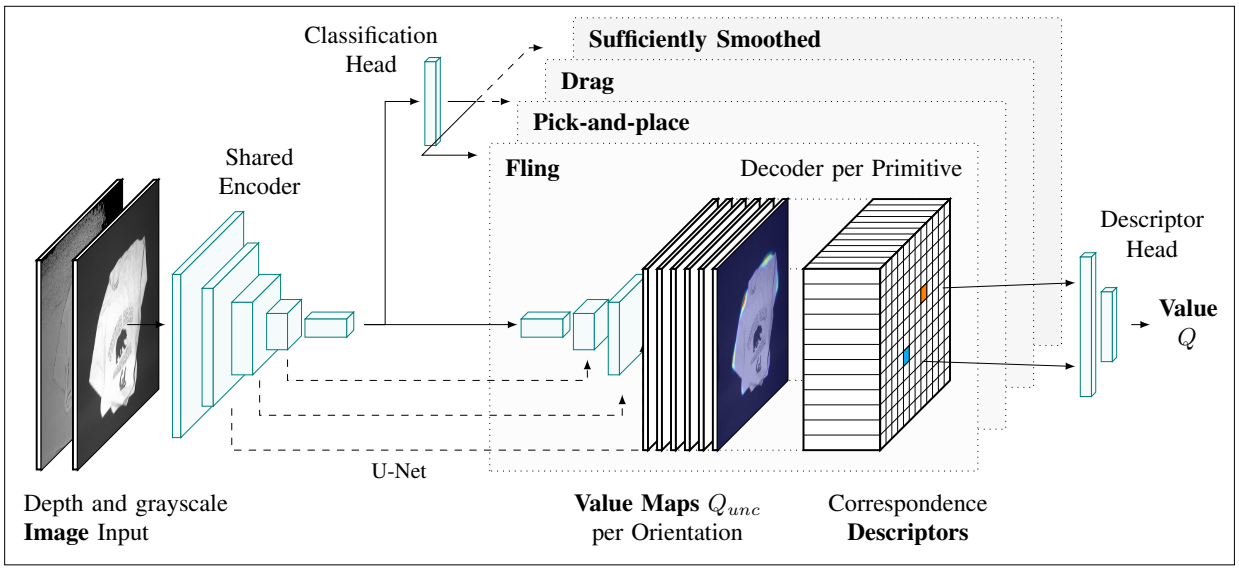


Figure 10.3: **BiManual Manipulation Net (BiMaMa-Net)** architecture first maps an image to a manipulation primitive type via its shared encoder and classification head. Given a primitive, it predicts dense unconditioned value maps for a discrete set of gripper orientations. It then calculates pixel-wise correspondence descriptors. A descriptor pair, representing a bimanual action, is combined in the descriptor head to predict its joint value.

10.4 Method

SpeedFolding uses BiMaMa-Net, a learned garment-smoothing method to bring an initially crumpled garment to a sufficiently smooth configuration, followed by an instruction-based garment folding pipeline.

10.4.1 Action Primitives

We are interested in the set of quasi-static and dynamic action primitives that enable the robot to (1) transfer an arbitrary garment configuration s_t to a folded goal configuration s^* (completeness), (2) reducing the number of action steps (efficiency), and (3) with a reduced number of primitives (minimality). Each action primitive is defined through a pair of poses as well as a motion trajectory. All primitives share a common procedure to reliably grasp the garment with parallel jaw grippers: Each gripper moves 4 cm above the grasp pose a_t, rotates 8° so that one fingertip is below the other, and moves 1 cm towards the direction of the higher fingertip. This motion improves the success for grasping in particular at the edge of the garment. We define following learned primitives (Fig. 10.2 left box):

Fling: Given two pick poses, the arms first pick those points, lift the garment above the workspace and stretch it until a force threshold is reached, measured using the arms'

internal force sensors. Next, the arms apply a dynamic motion, flinging the garment forward and then backward while gradually reducing the height toward the workspace. Similar to [15], we find the fling motion to be robust under change of velocity and trajectory parameters, and therefore we keep these parameters fixed. The fling primitive allows to significantly increase the garment's coverage in a few steps, but often does not yield a smooth configuration.

Pick-and-place: Given a pick and a corresponding place pose, a single arm executes this quasi-static action, while the second arm presses down the garment at a point on a line extending the pick from the place pose. Pick-and-place enables the robot to fix local faults such as corners or sleeves folded on top of the garment.

Drag: Given two pick points, the robot drags the garment for a fixed distance away from the garment's center of mask, leveraging the friction with the workspace to smooth wrinkles or corners, e.g. sleeves folded below the garment.

We define heuristic-based primitives (Fig. 10.2 center box):

Fold: Both arms execute a pick-and-place action simultaneously to fold the garment. The heuristic for calculating the pick and place poses is explained in Sec. 10.4.5.

Move: While similar to drag, this primitive's pick poses and its drag distance are calculated by a heuristic so that the garment's center of mask is moved to a goal target point. Usually, the robot drags the garment towards itself to mitigate reachability issues in subsequent actions. Sec. 10.4.5 provides details about the pose calculation.

We define an additional learned primitive to switch from garment-smoothing to folding (Fig. 10.2 right box):

Sufficiently Smoothed: We find that deciding whether a garment is ready to be folded purely from coverage computation, as done in prior works [15, 16, 227], is not sufficient. In particular, even a high coverage is prone to wrinkles or faults that might reduce the subsequent fold quality significantly (as described in Fig. 10.2). Instead of relying on the coverage, BiMaMa-Net returns a smoothness value given an overhead image. While this primitive is not used to change the configuration of the garment, it is used to switch from garment-smoothing to folding.

10.4.2 BiMaMa-Net for Bimanual Manipulation

Predicting a single pose from an overhead image is commonly done by first estimating a pixel-wise value map per gripper z-axis rotation θ, in which each pixel value represents a future expected reward (e.g., grasp success, increase in garment coverage, etc.), and then selecting the maximum greedily [15, 264–266]. Extending this approach to two *corresponding* planar poses $(x, y, \theta)_{1,2}$ conditioned on each other is however challenging primarily due to the

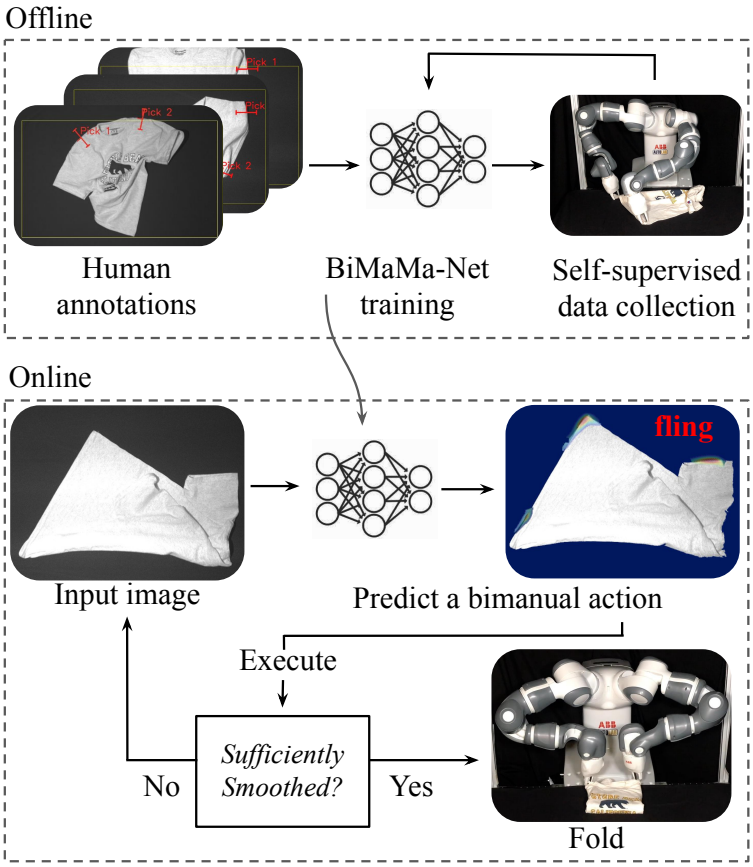


Figure 10.4: **SpeedFolding Pipeline.** We start by manually annotating input images with primitives and gripper poses, train a NN and then iteratively use the NN for self-supervised data collection (**top**). During runtime, we use the NN to predict a primitive and a pair of poses given an input image and execute it on the robot. If the resulting garment configuration is classified as Sufficiently Smoothed the robot will fold the garment, otherwise it will repeat the process.

exponential scaling of possible end-effector poses with the number of dimensions. In particular, this is a multi-modal problem, and the predicted unconditioned value maps $Q_{unc}(x, y, \theta)$ have multiple peaks (as in Fig. 10.6). While unconditioned value maps may provide information relevant for downstream bimanual tasks, such as the grasp success, they provide no information regarding their correspondences. To address this we define *correspondence descriptors*

$$\mathbf{d} = (Q_{unc}, x, y, \sin\theta, \cos\theta, m, \mathbf{e})$$

where $\mathbf{e} \in \mathbb{R}^M$ is a learned embedding for each pixel (disregarding orientations θ) concatenated with the unconditioned value Q_{unc}, positional encodings, and the action primitive type m. Then, the final conditioned value $Q(\mathbf{d}_1, \mathbf{d}_2)$ depends on a descriptor pair.

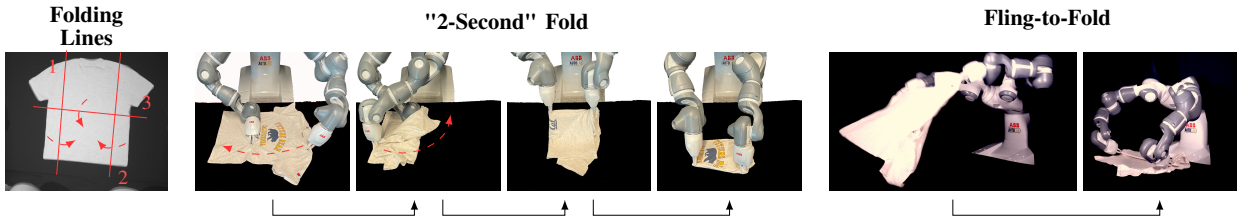


Figure 10.5: **Folding Approaches.** We compare three approaches for folding. Left: A template mask with a sequence of folding lines that is compiled to a number of bimanual pick-and-place actions. Center: A so-called "2-second" folding heuristic that applies only very few steps however is for t-shirts only [271]. Right: A *fling-to-fold* primitive that combines a fling with an immediate folding action. Here, the garment does not need to be fully smoothed, however prior knowledge is required.

Fig. 10.3 shows the complete information flow of BiMaMa-Net: A shared encoder using a ResNext-50 [267] backbone maps an input image (e.g. depth and grayscale) to high-level features. First, a classification head predicts the manipulation primitive m. For a *Sufficiently Smoothed* primitive, no further action is required. For all other learned primitives, a U-Net [268] decoder predicts value maps for a discrete number N of end-effector orientations θ. We choose a U-Net architecture over fully convolutional neural network used in prior robotics manipulation works [15, 265, 269, 270] as U-Nets are better suited for high-resolution inputs that we find necessary for detecting edges and wrinkles for garment smoothing.

Then, *BiMaMa-Net* samples a set of poses from the value map, where pixels with higher values are more likely to get sampled. During training, BiMaMa-Net samples from $p(a|s) \sim \sqrt{Q_{unc}(a,s)}$ to allow for sampling negative examples to better estimate the underlying distribution of action values. For inference, BiMaMa-Net samples from $p(a|s) \sim Q_{unc}(a,s)^2$ which emphasizes action poses with high values. It then calculates the correspondence descriptors for each pose, and a final neural network head combines all descriptor pairs $\mathbf{d}_1$, $\mathbf{d}_2$ to output the final conditioned action value Q.

If two poses are interchangeable (e.g., during a fling or a drag action), a single decoder predicts the value maps per θ. However, if a certain relation between the poses must be maintained (e.g., the conceptual difference between the pick and the place poses in a pick-and-place action) then separate decoders compute two value maps Q^1_{unc} and Q^2_{unc}.

10.4.3 Reachability Calibration

As shown in Fig. 10.6, to ensure reliable garment smoothing and folding, the robot should compute the actions that maximize the expected reward within the reachable space. To find the robot's reachable space, we perform a one-time boundary search along a discretized grid in the action space (x, y, θ) for each gripper, assuming a constant height z above the table. The search, done separately for each θ, starts with a fixed lower value of y and increases x

until the inverse kinematics fails to find a solution. Afterwards, it repeatedly increases y or decreases x so that the search is confined to the continuous boundary at which reachability fails. As a result, we get masks M_l and M_r for the left and right arms

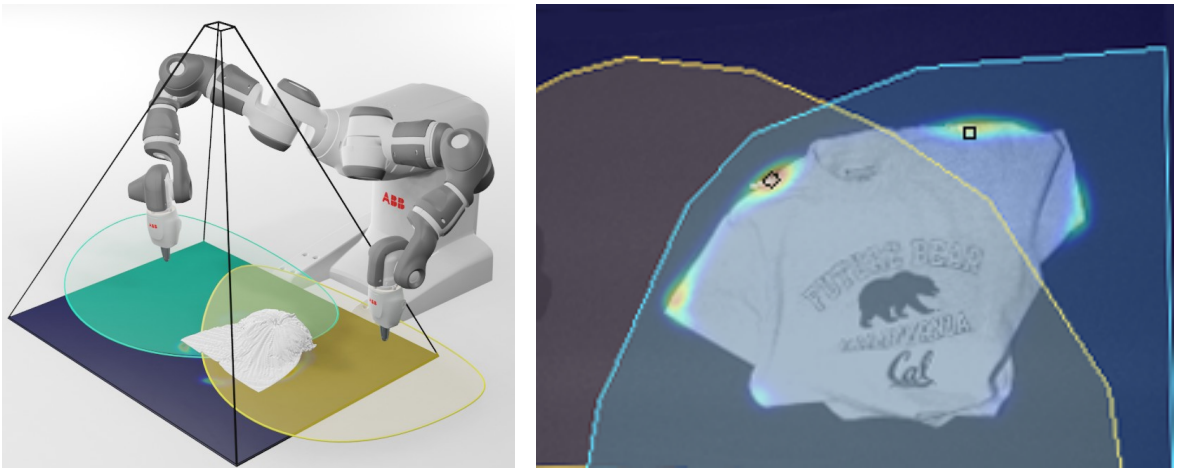
$$M(x, y, \theta) \rightarrow \{0, 1\}$$

that can be incorporated into BiMaMa-Net as spatial binary constraints by restricting the action sampling to the masks. To ensure that each reachability mask contains at least one pose, we create up to four masked value maps from Q^1_{unc} (or Q^2_{unc}) by multiplying them with M_l or M_r: $\{Q^{1l}_{unc}, Q^{1r}_{unc}, Q^{2l}_{unc}, Q^{2r}_{unc}\}$. An action value $Q_{unc} = 0$ is ignored in the sampling process. We then sample and combine the correspondence descriptors from Q^{1l}_{unc} and Q^{2r}_{unc}, and vice versa for Q^{1r}_{unc} and Q^{2l}_{unc}.

We find that using a calibrated reachability mask for each end-effector orientation θ separately significantly reduces the number of false negatives that arise when using approximations, such as a circular mask. After selecting the final, reachable poses $(x, y, \theta)_1$ and $(x, y, \theta)_2$ during runtime, we check for possible collisions due to inter-arm interaction. If a potential collision is detected, the next best action is selected until reachable and collision-free poses are found.

10.4.4 Training for Smoothing

We train BiMaMa-Net via self-supervised real-world learning to predict the manipulation primitive type m and the corresponding action poses $(x, y, \theta)_1$ and $(x, y, \theta)_2$ given an overhead



(a) Workspace (b) Reachability mask

Figure 10.6: **Reachability.** (a) We perform a boundary search to compute separate reachability masks for the left (yellow) and right (blue) robot arms. (b) BiMaMa-Net guarantees at least one pick pose (black) from the value map within each mask.

image of a garment.

In order to scale real-world interaction, the learning process is designed for minimal human intervention. First, we collect examples of smooth configurations to train a classifier outputting the confidence $p(\textit{Sufficiently Smoothed}|s)$. Additionally, let $cov(s)$ be the coverage of the garment at configuration s observed from an overhead perspective, calculated by background subtraction and color filtering. We define the reward r:

$$\begin{aligned} r_t = \max\big(\tanh\big[\alpha\ (cov(s_{t+1}) - cov(s_t)) \\ +\ \beta\ (p(smoothed|s_{t+1}) - p(smoothed|s_t))\big], 0\big) \end{aligned}$$

as the sum of the *change* of coverage and *Sufficiently Smoothed* confidence with tuned weights α and β respectively. It is scaled to $r \in [-1, 1]$ first and then clipped to a non-negative value, so that no change equals zero reward. To ensure continuous training, the robot resets the garment configuration by grasping it at a random position on its mask and dropping it from a fixed height. We iteratively train a self-supervised data collection neural network, interleaving training and execution (Fig. 10.4). The robot explores different actions by uniformly sampling from the set of N_s best actions.

To avoid a purely random and sample-inefficient initial exploration, we kickstart the training with human annotations. We differentiate between self-supervised and human annotated data within the training process in three ways: (1) We set the reward of human annotated data to a fixed r_h. (2) Besides training the value map at the specific annotated pixel position and orientation, we follow [266] and introduce a Gaussian decay centered around each pose as a global target value instead. (3) The classification head is trained only with data that has a reward higher than a tuned threshold $r \geqslant r_c$.

10.4.5 Folding Pipeline

We compare three approaches for folding: *instruction-based folding*, which can be adapted to different garments and different folding techniques, *"2-second" fold*, a known heuristic for surprisingly fast t-shirt folding, and *fling-to-fold* (F2F), a novel technique that can increase the number of folds-per-hour (FPH) by leveraging prior knowledge about the t-shirt's dimensions.

Instruction-based Folding: As shown in Fig. 10.5 (left), given a mask of a smoothed garment, the robot iteratively folds the garment along user-specified *folding lines*. These allow to define the goal configuration of a smooth garment precisely without using high-dimensional visual goal representations [16, 19]. A complete user instruction includes: (1) A binary mask called template and (2) a list of folding lines relative to the template (Fig. 10.5 left). The folding direction is defined with respect to the line according to the right-hand-rule.

To execute the folding lines, a particle-swarm optimizer computes an affine transformation by registering the template with the current image. Afterwards, SpeedFolding calculates

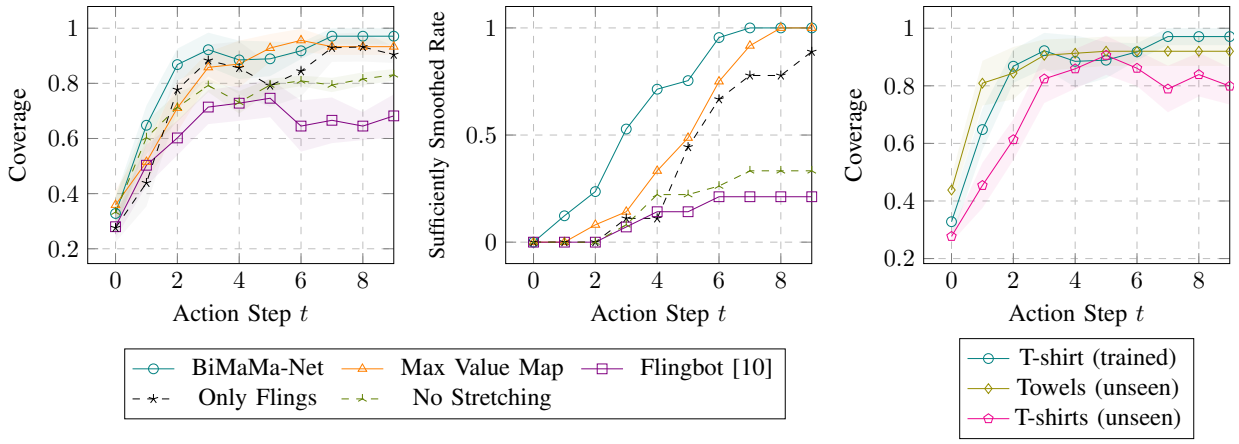


Figure 10.7: **Garment smoothing** until it is Sufficiently Smoothed. We compare the normalized coverage (left) and prediction of the learned *Sufficiently Smoothed* classifier (center) over the number of action steps with different baseline methods. The system is able to generalize to unseen garments of different color, patterns, and material (right).

corresponding poses for a bimanual fold action: Let $\mathbf{p}_{ent}$ be the first and $\mathbf{p}_{exit}$ be the second intersection point of the line and the mask, where the line enters and exits the mask respectively. This splits the mask into a *base* and a *fold-on-top* part. On the contour of the latter, the algorithm finds two pick points $\mathbf{p}_1$ and $\mathbf{p}_2$ so that the area of the four-sided polygon $(\mathbf{p}_{ent}, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_{exit})$ is maximized. We use the normal at the pick point for the gripper orientation θ. The place poses are calculated by mirroring the pick poses at the folding line.

"2-Second" Fold: For specific garments such as t-shirts, there exist heuristics for efficient folding. Given a smooth configuration, the "2-second" fold follows a set of steps that requires using two arms simultaneously (Fig. 10.5), and is therefore well suited for a bimanual robot [271].

Fling-to-fold (F2F): We observe that (1) a fling action while grasping a sleeve and the non-diagonal bottom corner is especially effective and (2) the first fold action grasps the same points. We conclude that these two steps can be merged to reduce imaging and motion time. We implement F2F by adding a learned primitive to BiMaMa-Net that computes these pick points if visible. The primitive's motion is implemented by combining a fling with a consecutive fold action (Fig. 10.5). To ensure that the t-shirt is folded correctly, prior knowledge about the t-shirt's dimension is required to adapt the height of each arm prior to the fold.

Table 10.1: **End-to-end folding** for different neural network architectures, folding approaches, and garments, averaged over 15 trials per experiment. The durations are averaged over successful folds, while the cycle time and FPH are averaged over both successful and unsuccessful folds.

Method	Folding Approach	Garment	Smoothing Actions	Duration [s]	Fold Success	Cycle Time [s]	Folds Per Hour (FPH)
Max Value Map	Instruction	T-shirt	5.1(5)	133.9(74)	80 %	167.4(92)	21.5(12)
BiMaMa-Net	Instruction	T-shirt	3.0(4)	108.7(73)	93 %	116.9(79)	30.8(21)
	"2-Second" Fold		3.0(4)	97.3(48)	53 %	182.4(54)	19.7(6)
	Fling-to-fold		1.8(2)	81.7(43)	93 %	87.9(47)	40.9(22)
Garments Unseen During Training							
BiMaMa-Net	Instruction	Towel	1.7(2)	59.2(38)	87 %	68.1(44)	52.9(34)
		T-shirt	4.8(4)	141.1(87)	80 %	176.3(109)	20.4(13)

10.5 Experiments

We experimentally evaluate the garment smoothing and folding performance of SpeedFolding on a known t-shirt, as well as on two garments unseen during training.

10.5.1 Experimental Setup

We perform experiments on a physical ABB YuMi robot with parallel-jaw grippers. The gripper's fingertips are extended by small 3D printed teeth to improve grasping. A thin sponge mattress is placed on the workspace to allow the grippers to reach below the garment without colliding. A Photoneo PhoXi captures overhead grayscale and depth images of the workspace, generating observations $o_t \in \mathbb{R}^{256\times192\times2}$. As the garment is frequently outside the camera's field of view, a 1080P GESMATEK RGB webcam is mounted above the workspace and used for coverage calculation. Computing is done on a system using an Intel i7-6850K CPU, 32GB RAM, and a NVIDIA GeForce RTX 2080 Ti.

We first perform data collection, and train 10.4.4 on a single t-shirt. Initially, 600 scenes of random garment configuration were recorded and manually annotated in 1 h. After training a first neural network, the robot collected 2200 self-supervised actions in 16 h. To include data of less frequently observed actions, we copied and re-annotated 1500 actions in 3 h, resulting in a dataset of 4300 actions in total. We used a single t-shirt shown in Fig. 10.1 throughout the training. We further perform data augmentation, including random translations, rotations, flips, resizes, brightness and contrast changes. We use $N = 20$ gripper orientations equally distributed over $[0, 2\pi)$ to implement the BiMaMa-Net decoder as described in Sec. 10.4.2. For training, we manually tune $N_s = 50$, $r_h = 0.8$ and $r_c = 0.3$ (see Sec. 10.4.4).

We design a set of garment smoothing and folding experiments to evaluate SpeedFolding. Initial garment configurations are generated by environment resets as described in 10.4.4. Each experiment is averaged over 15 trials. We ignore experiments that terminate early due

to a motion planning error, as this is not the focus of this chapter. A trial is considered unsuccessful if the garment was not successfully folded according to the majority vote of three reviewers or the number of actions exceeded a maximal horizon of $H = 10$. We define a grasp success if the gripper holds the garment *after* an executed action. For known garments, BiMaMa-Net achieves a grasp success rate of over 96 %.

10.5.2 Sufficiently Smoothed

We evaluate garment smoothing using two metrics: The garment coverage, computed from an overhead image, and a binary Sufficiently Smoothed value, predicted using the Sufficiently Smoothed classifier. We compare BiMaMa-Net to two baseline (1) *Max Value Map*, a variant of BiMaMa-Net that computes the pick points directly from the value maps Q_{unc} by computing the maximum over the map to find two pick points without using correspondence descriptors, (2) *Only Flings*, a variant restricted to fling actions only, and (3) *Flingbot*, the pre-trained method from [15] (see Fig. 10.7). Results suggest that BiMaMa-Net is able to smooth a known t-shirt to a Sufficiently Smoothed configuration in ~ 3 fewer steps compared to baselines requiring ~ 5. Although the increase of coverage is similar to Only Flings, the latter reaches a Sufficiently Smoothed configuration later or even fails to do so, confirming the need of additional action primitives to fully smooth a garment.

We note that the FlingBot baseline fails to reach an 80 % coverage as reported in [15], as we observe frequent grasp failures presumably due to differences in the physical setting. We ablate the stretching motion before a fling and observe that stretching leads to higher coverage.

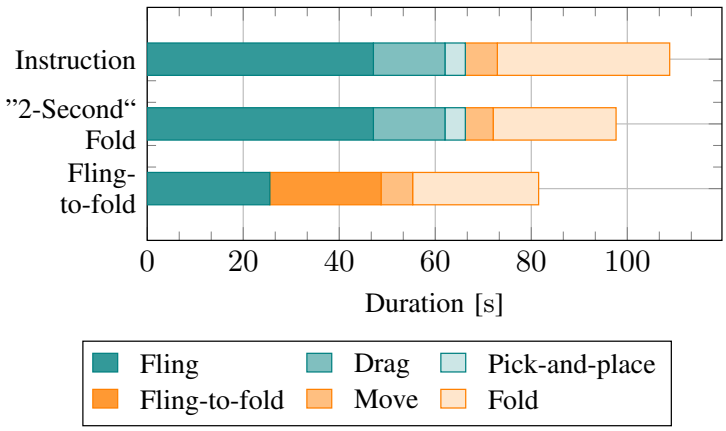


Figure 10.8: **Timings** for calculating and executing the action primitive types depending on the folding approach. Instruction-based and ”2-Second“ fold share the same smoothing actions (blue), but differ in folding (orange). By introducing a combined Fling-to-fold primitive, a smoothed state is not required before folding. However, the ”2-Second“ fold is suited for manipulating a t-shirt only, and Fling-to-fold assumes prior knowledge of the garment.

10.5.3 Folds per Hour

Table 10.1 shows results of end-to-end garments folding experiments. BiMaMa-Net manages to (1) successfully fold garments in over 90 % of the trials on known garments and (2) 30 % faster than the Max Value Map baseline using the Instruction-based folding approach. The "2-second" fold achieves an additional speedup of 10.4 % when executed successfully, however we find that it is sensitive to t-shirt's orientation in a Sufficiently Smoothed configuration and suffers from a low fold success rate. With prior information on the t-shirt's dimensions, F2F uses 40 % less smoothing actions and imaging time. As a result, it achieves a speedup of over 25 % compared to the instruction-based approach, leading to 40.9 folds per hour on average. Calculating an action using BiMaMa-Net takes 126.0(9) ms on our hardware.

10.5.4 Generalization to Unseen Garments

We explore how SpeedFolding, trained on a single t-shirt, can generalize to garments unseen during training. In these experiments we use (1) a t-shirt with a different color and stiffness and (2) a rectangular towel with a different color compared to the original t-shirt. We evaluate SpeedFolding on unseen garments using instruction-based folding, as this is the only approach that easily adapts to general garments. We run the same experiments on the unseen t-shirt with no changes to the BiMaMa-Net model or the folding template. In contrast, when we run the towel experiments we observe that the system fails to classify a Sufficiently Smoothed configuration, as the object's shape is different from that BiMaMa-Net was trained on. To address this, we add 20 Sufficiently Smoothed towel images to the dataset and re-train BiMaMa-Net. Table 10.1 suggests that SpeedFolding can generalize to garments with different color, stiffness and shape.

10.5.5 System Limitations

Grasp failures, especially during a fling motion, can decrease the garment's coverage dramatically. We find that most grasp failures happen due to losing the grip during the stretching motion prior to a fling action. This limitation can be mitigated using improved force feedback or by adding visual feedback. We observe a frequent failure case during top-down grasps while executing the first step of the "2-second" fold. These grasps may require different gripper jaws that are better suited for top-down grasps.

As common with data-driven methods, SpeedFolding can generalize to *similar* unseen garments. For example, textile patterns may be more challenging to detect and classify correctly. This limitation can be addressed through additional data augmentation. Generalization to different garment shapes may also be limited, and can be addressed by adding examples of Sufficiently Smoothed configurations to the dataset, as described in Sec. 10.5.4 for the towel example.

10.6 Discussion and Future Work

We presented SpeedFolding, a bimanual robotic system for efficient folding of garments from arbitrary initial configurations. At its core, a novel BiMaMa-Net architecture predicts two conditioned poses to parameterize a set of manipulation primitives. After learning from 4300 human-annotated or self-supervised actions, the robot is able to fold garments in under 120 s on average with a success rate of 93 %.

While prior works e.g. by Maitin-Shepard et al. [253] or Doumanoglou et al. [252] achieved high success rate for end-to-end cloth folding, cycle times for a single fold were on the order of 3−6 Folds Per Hour (FPH), whereas SpeedFolding achieves 30 FPH to 40 FPH. Similar to Ha et al. [15], the fling primitive can unfold the garment in a few actions. In contrast however, we introduce additional action primitives that enable the robot to reach a sufficiently smoothed configuration. In future work, we will explore methods that can learn to manipulate a novel garment given a few demonstrations.

Part IV

Efficient and Reliable Autonomous Gardening

Chapter 11

AlhphaGarden

In this chapter, we explore various models and optimizations to increase crop yield and maintain plant diversity under limited irrigation in polyculture farming. However, the long growing seasons make this process impractical. We created a simulator that can approximate growth in a real greenhouse 25,000X the speed of natural growth, allowing us to tune irrigation and pruning policies.

11.1 Introduction

Cultivating plants has been an essential human activity for over 10,000 years. Many factors influence the quality and quantity of yield, such as irrigation, pesticide use, weather conditions, and plant disease. Industrial agriculture aims to maximize yield by growing a single plant species in isolation (monoculture). Polyculture farming, on the other hand, involves growing different crops simultaneously in imitation of the diversity of natural ecosystems, and is a sustainable alternative that uses biodiversity to reduce pesticides, disease, and weeds [272–274]. Polyculture is also more practical for confined urban spaces and essential for aesthetic gardens.

With increasing demands for fresh local herbs and produce, in spite of limited resources, polyculture gardens may be increasingly in demand. However, polyculture requires more human labor than monoculture to prune and maintain. A robot with a reliable and sustainable control policy has the potential to increase yield and reduce water consumption. Still, finding an optimal policy is a challenging task. The long time constants for real-world experiments motivates the use of a simulated environment, yet it is difficult to simulate inter-plant dynamics, including competition for light, water and nutrients. In addition, much of the interaction is done in the soil or beneath the leaf canopy and cannot be directly observed, and the action space, in particular pruning, is complicated.

In this chapter we present AlphaGardenSim, an efficient, first order simulator for polyculture farming. AlphaGardenSim models a polyculture garden using first order models to simulate single plant growth, inter-plant dynamics and competition for resources.

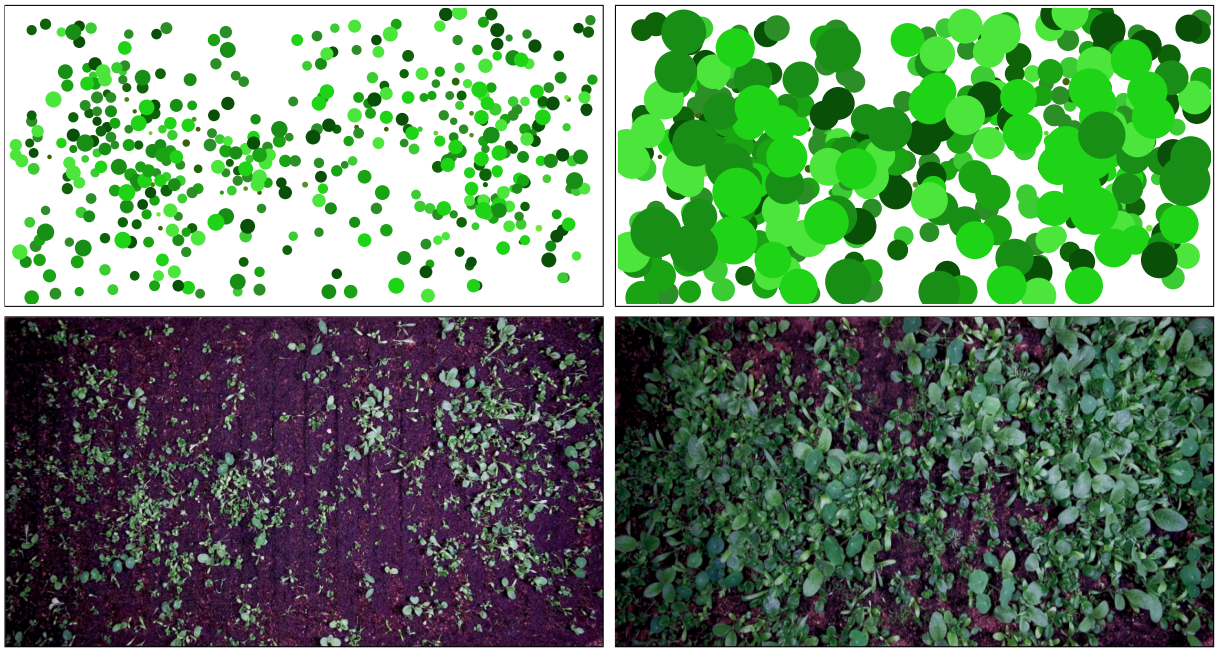


Figure 11.1: **Simulated polyculture garden.** Planted with seeds from 13 edible plant types, and grown for a period of 90 days. A garden simulated with AlphaGardenSim, and a corresponding physical garden initialized with similar seed types and locations, and irrigated every 2 days for a similar period of time. Each shade of green in the simulated garden represents a different plant type. **Left:** The gardens at day 15, a few days after germination. **Right:** The gardens at day 30.

This chapter makes 3 contributions: (1) A fast, first order simulator, AlphaGardenSim, that incorporates parameterized individual plant growth models and inter-plant dynamics to simulate competition over resources between plants in close proximity; (2) An analytic automation policy that yields leaf coverage and garden diversity; and (3) Application of the simulator to tune the policy parameters. Experimental results in simulation suggest that a tuned analytic policy can achieve high coverage and plant diversity compared with other policies, even in the presence of rapidly spreading invasive plants.

11.2 Related Work

Past work in plant growth simulation has predominantly been focused on monoculture agriculture. Widely used simulation models include DSSAT [275] and AquaCrop [276]. However, such models are intended for simulating large scale, monoculture agricultural operations, and are point-based models, which make the assumption that plants are grown homogeneously. Therefore, these models are not well-suited for a polyculture setting, where gardens are heterogeneous.

GeoSim [277] is a tool that adds spatial functionality to point-based agricultural models by leveraging data from a geographic information system (GIS) to run independent simulations at different geospatial points, allowing for heterogeneous simulation. However, to tune a policy for managing a small-scale polyculture garden, it is desirable to simulate at the plant level. Recently, Chebrolu et al. [278] developed a point cloud registration algorithm that enables plant monitoring to analyze growth at the single-plant level. It can be used to tune a single-plant growth model, but does not reveal inter-plant interactions, which are required to provide higher granularity data for polyculture modeling.

There exist individual plant models that model inter-plant competition, but to the best of our knowledge, there does not exist one for a polyculture setting. For example, Damgaard et al. [279] proposed modeling competition between individual plants based on density and size differences between plants, but their work does not explicitly model resource competition, which is important for tuning a policy that affects the distribution of resources in a garden. Price et al. [280] introduced a simulator for individual plant growth and competition with promising results, but their work only modeled plant radii and did not take into account competition for resources other than water. According to Berger et al. [281], a review on individual-based approaches for modeling plant competition, existing models lack consideration for the effect of plants on resource levels in an environment. Thus, we were motivated to develop our own first order simulator for tuning a polyculture gardening policy.

Czárán and Bartha [282] proposed a broad classification of individual-based plant competition models as either grid-based models, or individual-based neighborhood models. Grid-based models discretize a region into a grid of cells that may be occupied by plants, while individual-based neighborhood models represent plants in a continuous space. Furthermore, grid-based models typically use empirical rules to define plant competition, whereas individual-based neighborhood models define explicit mechanisms that regulate competition. One such individual-based neighborhood model is the zone-of-influence model, where a circular zone corresponding to the plant's size defines where a plant acquires resources from. Plants with overlapping zones are in competition with each other, and the growth rate of a plant decreases as more of its zone is overlapped with. While these models allow for greater modeling complexity, grid-based models make simplifying assumptions and reduce computational cost.

Research in agricultural automation has also been conducted specifically in crop modeling and individually controlled plants. Wiggert et al. [283] developed a testbed that enables real-time data collection of plant water stress to automate and optimize plant-level irrigation. Habibie et al. [284] trained a Simultaneous Localization And Mapping (SLAM) algorithm in simulation to automate fruit harvesting in a red apple tree field. While it supports a wide variety of use cases and enabled successful harvesting, plant dynamics were not modeled, and the simulation focused on a single plant type. CoppeliaSim [285] comes closer to simulating a polyculture garden, as plants are able to be controlled separately. This simulator was used to train a crop monitoring green house robot to navigate a greenhouse and identify diseased crops [286]. Even though each plant had unique parameters, they did not model inter-plant interactions.

Local Variables				Global Variables			
$\mathbf{d}(x, y, t)$	$h(x, y, t)$	$w(x, y, t)$	$\mathbf{s}(x, y)$	$\mathbf{p}(k, t)$	$c(t)$	$v(t)$	$s(t)$
Plant Type	Plant Health	Water Amount	Seed Locations	Plant Type Distribution	Total Canopy Coverage	Garden Diversity	Garden Sustainability

Table 11.1: Simulator local and global time-dependent state variables

11.3 AlphaGardenSim

In our simulator, AlphaGardenSim, the garden is modeled as a discrete $M \times N$ grid. The time t is specified in units of days. Each point $p(x, y)$ in the garden contains up to one seed, and up to a fixed capacity of water. We define $D(k)$ as a set of k plant types available in the garden, as well as types *soil* and *unknown*. At each point $p(x, y)$ in the garden at time t, we define $\mathbf{d}(x, y, t)$, a vector of length k representing the distribution of the plant (or soil) types that is visible overhead at point $p(x, y)$, i.e., the canopy cover in the garden. For example, when the garden is planted with plant type $D[3]$ at point $p(x, y)$ at time 0, then $\mathbf{d}(x, y, 0) = \big[0, 0, 1, 0, \ldots, 0\big]^T$. This can change over time if a plant from a neighboring point expands and occludes the plant seeded at $p(x, y)$. We define the seed locations in the garden as $\mathbf{s}(x, y)$, the health of the plant at point $p(x, y)$ at time t as $h(x, y, t)$, and the amount of soil moisture available at point $p(x, y)$ at time t as $w(x, y, t)$. We define the following global quantities over the entire garden; $\mathbf{p}(k, t)$ holds the global population in the garden as a distribution over point types D, $c(t)$ is the total canopy coverage w.r.t. the garden size at time t, $v(t)$ is the diversity in the garden at time t, and $s(t)$ represents the sustainability in the garden, i.e. the savings in irrigation compared to a uniform baseline irrigation policy.

11.4 Modeling

11.4.1 Plant Growth Model

To represent each plant, we adopt the model proposed by [280] which tracks the radii of plants for the purposes of water competition. However, we add height as a simulated attribute, as this is necessary to model competition for sunlight. This yields efficient calculation of plant growth, while still modeling a variety of plant types and interactions. The overall growth curve of a plant is modeled as exponential in its radius, but with adjusted rates at each step when competition and limited resources are taken into account. This is consistent with experimental evidence suggesting that plants grow exponentially in ideal

conditions [287].

AlphaGardenSim executes a sequence of updates at each timestep: irrigation, lighting, water use and plant growth.

In the irrigation step, each of the $M \times N$ coordinate points accepts a water amount of 0 or 1. Values should be chosen in a garden-specific manner, such that 0 represents no water and 1 represents the maximum amount of water a unit area of soil can hold.

We assume a fixed amount of light is available at every point in the garden, so that plants receive light based on the size of their unoccluded leaf area (the portion of their bounding circle that is not covered by taller plants), as demonstrated in Fig. 11.2. To estimate the area of partially overlapping circles, we use an approximation. For a plant i with radius r_i, let L_i be the set of garden coordinates which are less than distance r_i away from its center coordinate. The approximate unoccluded leaf area is then simply the number of coordinates L_i which do not belong to L_j, for any other plant j with a taller height. Light is distributed in an exponentially decaying fashion, where the i^{th} tallest plant at $p(x, y)$ receives LD^i amount of light from $p(x, y)$, where LD is a light decay coefficient in the range $(0, 1)$.

Next, once each plant has been allocated light, we model the water use of each plant. Here, our assumption is that plant growth potential is proportional to the amount of sunlight they can intercept, but actual growth is dependent on the plants' access to sufficient water resources, allowing for transpiration and gas exchange, and therefore photosynthesis [288]. The amount of water required by a plant therefore depends on the sunlight obtained from the previous step, and by extension the size of the plant itself.

From this assumption, the maximum water amount desired by a plant is modeled as:

$$w_{max} = \frac{c_2}{c_1}\sqrt{L_u}$$

where L_u is the unoccluded leaf area of the plant, equivalent to the maximum amount of sunlight it can receive, and c_1 and c_2 are plant-specific parameters that adjust its growth pattern. Larger values for c_1 correspond to higher water use efficiency, i.e. higher biomass accumulation per unit of water. Larger values for c_2 correspond to higher overall productivity, i.e. a higher attainable growth at each timestep [289].

While the simulator uses a grid-based model, it draws from the zone-of-influence approach described by [281], allowing a plant to only uptake water from a circular zone defined by its radius, i.e. from coordinate points that are within radius distance of the plant's central coordinate. To model the sharing of water resources by plants in the same general area, Firbank and Watkinson [290] used relative sizes to allocate proportions of water to each plant. We take a similar approach, but add randomness in order to simulate more aggressive competition. At each coordinate, we iterate in a random order through the plants whose leaf areas cover that coordinate. For each such plant, we allow it to use the maximum amount of water it desires from this coordinate, defined as w_{max} divided by its total approximate leaf area, provided there is still water remaining. Thus, plants will still take a proportion of the water in their ecological neighborhood in expectation, but the randomness allows for some plants to take more water than usual and dominate their surrounding competition.

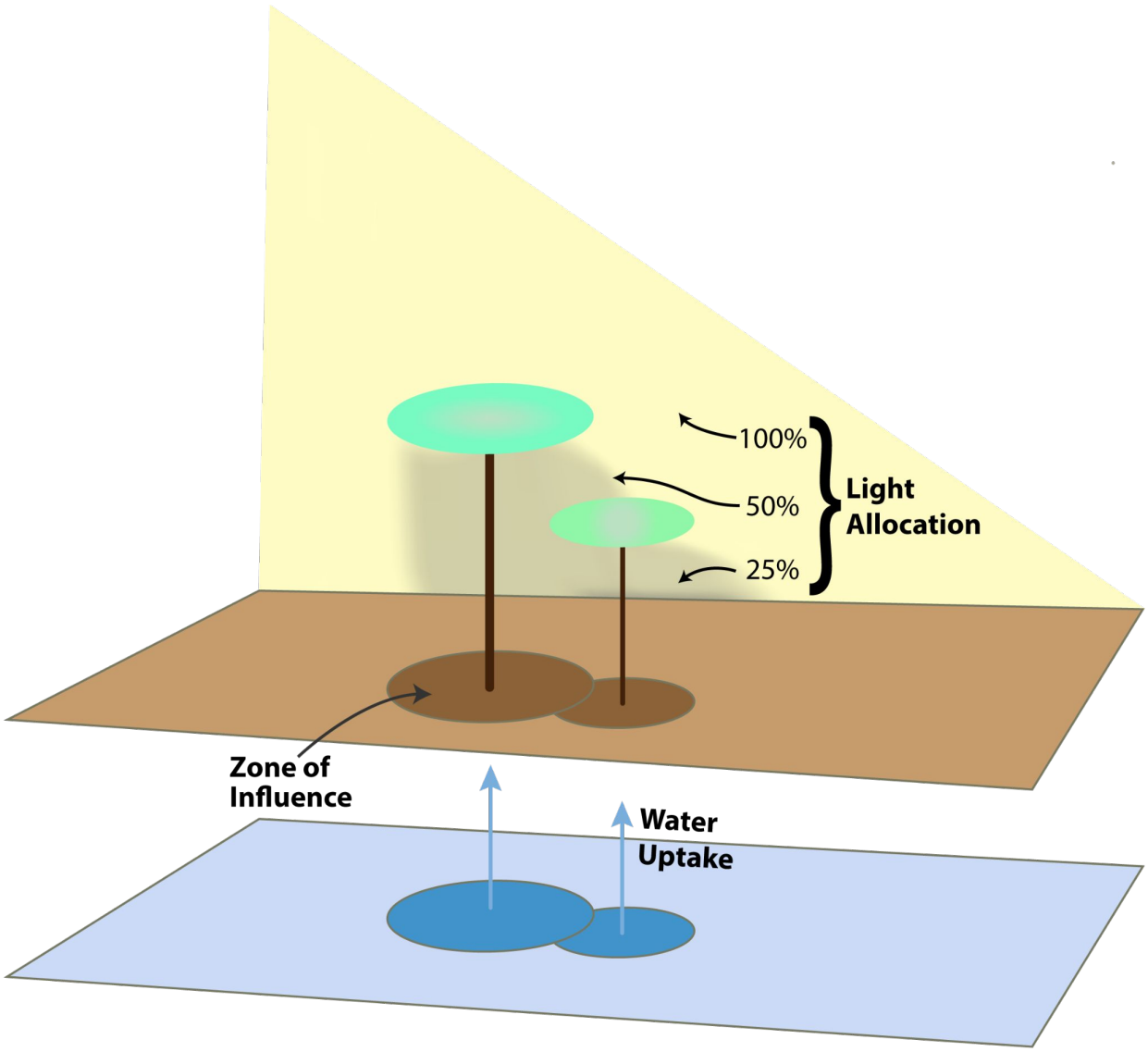


Figure 11.2: **Light and Irrigation Models.** Each plant receives light based on the size of its unoccluded leaf area in the grid, i.e., the number of grid points visible overhead, while occluded points allocate light in an exponentially decaying fashion. The plant's water uptake is then drawn from its neighboring grid points, to fulfill its growth potential. The plant is limited by the amount of light it intercepts and the amount of water available in its zone-of-influence.

Furthermore, this approach causes a plant to grow slower in expectation as more of its zone overlaps with the zones of other plants, in accordance with the zone-of-influence model.

Once the water at each coordinate has been allocated, we compute a growth value for each plant, defined as:

$$G = c_1 \cdot \min(w, w_{max})$$

where w is the actual amount of water this plant was able to uptake. Note that the water allocation process enforces that a plant obtains no more than $w_{max} = \frac{c_2}{c_1}\sqrt{L_u}$ units of water in the best case, so this is essentially $G = c_1 w$, a linear function of the water used.

Each plant must divide its growth value G between vertical growth (increasing its height) and radial growth (increasing its radius). We make this assumption because plants are inherently phototropic and adjust their growth to seek light and escape shade [291], thus they should divide G strategically to ensure maximum unoccluded leaf area. There is a tradeoff: plants will typically seek to increase their radius as much as possible, but if their leaves are occluded by taller plants, they will increase their height in order to outgrow competitors and gain an advantage. We define $l_{o,i}$ and $l_{u,i}$, the number of points where plant i is occluded and unoccluded respectively, and model this dynamic as follows:

$$\begin{aligned} l_p &= \frac{l_{u,i}}{l_{u,i} + l_{o,i}} \\ r_i &= \max(k_1, \min(l_p, k_2)) \\ G_{radial} &= r_i G \\ G_{vertical} &= (1 - r_i) G \end{aligned}$$

where k_1 and k_2 are plant-specific constants indicating lower and upper bounds on the proportion of G the plant is willing to allocate to purely radial growth. This is reflective of the genetically ingrained habit and morphology of the individual species. For instance, higher values of k_1 reflect a rosette habit, in which very little vertical growth occurs between individual leaves until the transition to reproduction [292]. We estimate k_1 and k_2 by taking a ratio of height and radius measurements collected from each plant species in an experimental real garden, as can be seen in the project's open-access repository. In the future, we intend to tune these parameters with more data.

Finally, we increment the radius and height of each plant according to the computed radial and vertical growth values, G_{radial} and $G_{vertical}$.

11.4.2 Life Cycle

Each plant is modeled with a biologically standard life cycle trajectory, consisting of five stages: germination, vegetative, reproductive, senescence, and death [293, 294]. Plants progress from one stage to the next after a number of timesteps sampled from a discretized Gaussian distribution, specific to the plant type, as the model assumes that plants of the same type share transition times between stages [295].

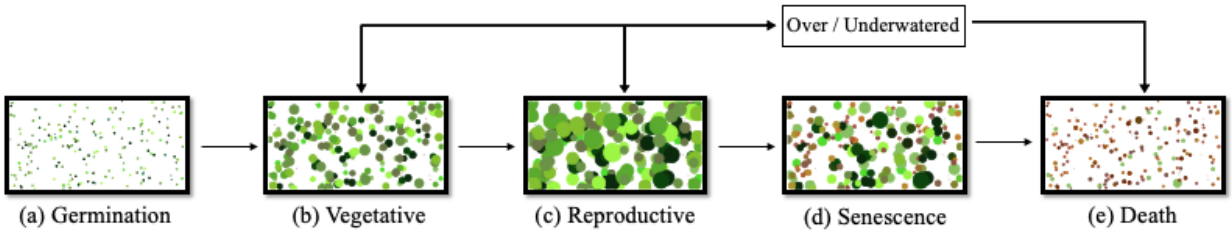


Figure 11.3: Each plant is modeled with a life cycle trajectory, consisting of five stages: germination, vegetative, reproductive, senescence, and death. When plants get underwatered or overwatered, their radius decays exponentially and their color turns brown, and after a short period they move to the death stage. However, if they receive their desired water amount prior to that, they return to their original stage, hence it is reversible. Also, since transition times between life stages are unique to the plant species, life stages of different species overlap in the snapshot above, for example in the top image some plants have already germinated while others are about to germinate.

Germination

When a seed is planted, it begins the germination phase. During this phase, it only occupies the single coordinate point where it was planted, and has both 0 radius and height. It uses light and water according to the model specified in IV.A, but does not change in radius or height. When the plant transitions to the next stage, it gains a nonzero radius and height, each sampled from a plant type specific Gaussian distribution.

Vegetative

During the vegetative phase, the plant obtains resources and grows according to the model specified in IV.A, unless it experiences stress from over or underwatering, as further detailed in IV.C.

Reproductive

During the reproductive phase, the plant behaves similarly to the vegetative phase, except it does not change in radius or height, unless it experiences stress from over or underwatering, as further detailed in IV.C.

Senescence

During the senescence phase, the plant obtains resources according to the model specified in IV.A, but its desired water amount is reduced to a proportion of its usual desired water amount, which linearly decreases to 0 over time. More specifically, if w_d is its desired water amount, should it have been in the vegetative or reproductive stage, then its adjusted desired

water amount, $\widetilde{w}_d$, is calculated as:

$$\widetilde{w}_d = \frac{1-t}{t_s} w_d$$

where t is the amount of time the plant has spent in the senescence stage, and t_s is the total duration of the senescence stage. However, the plant's accumulated resources no longer go towards its growth. Instead, the radius of the plant decays exponentially to a proportion of its radius at the beginning of the stage. This proportion is a parameter associated with the plant type. Plant height does not change during this phase.

Death

When the plant dies, it no longer uses resources nor changes in radius or height. However, it continues to occupy space in the garden, potentially occluding plants.

Fig. 11.3 shows snapshots from a simulated garden at every stage, where the top image corresponds with the germination phase and the bottom corresponds with the death phase.

11.4.3 Water Stress

To model the detrimental effects of suboptimal irrigation, the plant model includes a specific response to over and underwatering when in the vegetative and reproductive stages. While in these stages, a plant is considered overwatered if the total amount of soil moisture within its radius, $w(t)$, exceeds its desired water amount, w_d. More concretely, a plant is overwatered if:

$$w(t) \geqslant T_o w_d$$

where T_o is a threshold parameter associated with plant type. Similarly, a plant is considered underwatered if the amount of water it uptakes, $\tilde{w}(t)$, falls below some proportion of its desired water amount:

$$\tilde{w}(t) \leqslant T_u w_d$$

where T_u is also a plant type specific threshold parameter.

While the plant remains in either stressed state, its radius decays exponentially to some fraction of its prior radius, similar to its behavior in the senescence phase. Furthermore, we visualize the effects of stress on plants by modifying their color to become progressively more brown as they continue to be stressed. If a plant remains in either stress state after a number of timesteps, then the plant immediately moves to the death stage. If a previously stressed plant is no longer stressed, then for the current timestep the plant's radius will decay, but it will return to its normal behavior the next timestep. Afterwards, the plant will continue its normal behavior with no lingering effects. The duration of the plant's stage is not affected by its stress during the stage. The plant's stress state persists when it transitions from the vegetative stage to the reproductive stage.

11.4.4 Irrigation

For each discrete point in AlphaGardenSim, soil moisture dynamics are independently simulated, without interactions between different spatial locations. One-dimensional soil moisture dynamics can be modeled according to the Richards equation [296], which takes the form of the following nonlinear partial differential equation:

$$\frac{\partial s}{\partial t} = -\frac{\partial q}{\partial z} - S(h)$$

where s is the volumetric water content in the soil, t is time, $\frac{\partial q}{\partial z}$ is a term that describes the flow of soil water as a result of differences in soil water potential, and $S(h)$ is a term that describes the soil water extraction by plant roots.

Soil moisture dynamics for each spatial point $p(x, y)$ is based on the following discrete-time linear approximation of this differential equation, proposed by Tseng et al. [297]:

$$w(x, y, t) = w(x, y, t-1) - d + a(x, y, t) - U(x, y, t)$$

where $w(x, y, t)$ is the soil moisture content at point $p(x, y)$ during time t, $a(x, y, t)$ is the amount of irrigation applied, and $U(x, y, t)$ is the plant water uptake rate. d specifies local water loss at each point, and accounts for all contributing factors, such as soil drainage and evaporation. In AlphaGardenSim, d is approximated by a constant quantity uniform for each point in the garden, $U(x, y, t)$ is accounted for by the stochastic plant water uptake model, and $a(x, y, t)$ is specified by an irrigation policy.

Soil moisture levels are initialized according to an i.i.d. Gaussian distribution $N(\mu, \sigma^2)$ for each point in the garden at the start of each episode, with an irrigation amount also specified for each point. Additionally, we enforce that soil moisture content remains non-negative throughout the episode. Thus, soil moisture dynamics can be described by the following equations:

$$w(x, y, 0) \sim \max(N(\mu, \sigma^2), 0) + a(x, y, 0)$$

$$w(x, y, t) = \max(w(x, y, t-1) - d + a(x, y, t) - U(x, y, t), 0)$$

11.4.5 Diversity

Plants in a polyculture garden must be pruned to prevent invasive species from dominating. If an invasive plant's growth is too aggressive in relation to other plants, it uses more resources and hinders their growth. We seek to diversify the garden through pruning. We model diversity as the entropy of the global population in the garden:

$$v(t) = H\left(\mathbf{p}\left(k, t\right)\right)$$

We normalize the entropy, so that when $\mathbf{p}(k, t)$ is uniform, $\bar{H}\left(\mathbf{p}\left(k, t\right)\right)$ is 100%, whereas if $\mathbf{p}(k, t)$ is completely unbalanced, for instance $\mathbf{p}(k, t) = [1, 0, 0, ..., 0]^T$, then its normalized

entropy is 0%. Since $\mathbf{p}(k,t)$ represents the leaf coverage ratio per plant type, if $\mathbf{p}(k,t)$ is uniform, the garden is maximally diversified, and vice versa. To increase the diversity in the garden, an agent can prune plants that spread more than others. In AlphaGardenSim, pruning reduces the radius of the pruned plant by a fixed percentage.

11.5 Experiments

We evaluate the performance of different polyculture farming automation policies by assessing their ability to reduce water use and maximize plant diversity in AlphaGardenSim, and we explore the robustness of the policies to varying garden settings, some of which include invasive species. In our experiments, we use a set of 13 edible plant types with different germination times, maturation times and growth rates, sampled from plant-specific Gaussian distributions, as specified in the project's open-access repository.

At the beginning of the simulation, n points are randomly picked and planted with seeds of k types. The seed locations $\mathbf{s}(x,y) = \mathbf{d}(x,y,0)$ are saved and remain fixed during the entire simulation period. As plants grow and accumulate resources from the garden, each plant's health level $h(x,y,t)$ is tracked, determined by its water-stress level in the previous days and the current amount of water accessible at the plant's location (see IV.C).

The state of the garden at each point $p(x,y)$ is defined as the tuple of all local and global quantities (see Table 11.1):

$$\mathbf{S}(x,y,t)= \\ (\mathbf{d}(x,y,t), w(x,y,t), h(x,y,t), \mathbf{p}(k,t), v(t), s(t))$$

To simulate sensor precision limitations, we define $\tilde{\mathbf{S}}(x,y,t)$, a sector of size $\frac{M}{10} \times \frac{N}{10}$ centered at $p(x,y)$ representing the area observable at time t. In this setting, a policy's access to local quantities $\mathbf{d}(x,y,t)$, $w(x,y,t)$ and $h(x,y,t)$ is limited to points within the sector's boundaries. In addition, we define A, the set of actions an agent can apply; watering, pruning, both, or none. For each observation, an agent chooses $a(x,y,t) \in A$, the action applied at point $p(x,y)$ at time t.

11.5.1 Experimental Setup

For experiments, we initialize a 150×300 garden with m plants, each sampled with replacement from k plant types and seeded at points $s(x,y)$, randomly sampled from all garden points. At each time step in the garden, we sample m sectors centered at each $s(x,y)$, as well as an additional $\frac{m}{10}$ sectors centered at non-seed points $p(x,y)$. One hundred garden days are simulated. For light distribution, we choose a light decay LD of 0.5.

We allow each action $a(x,y,t)$ to affect a small window in the center of a sector. During irrigation, we update each $w(x,y,t)$ of an 11×11 square centered around $s(x,y)$ as detailed in IV.D with $a(x,y,t) = 1$. For pruning, we create a 5×5 window centered around $s(x,y)$ and collect all plants that are visible inside the window from overhead. Then we prune each

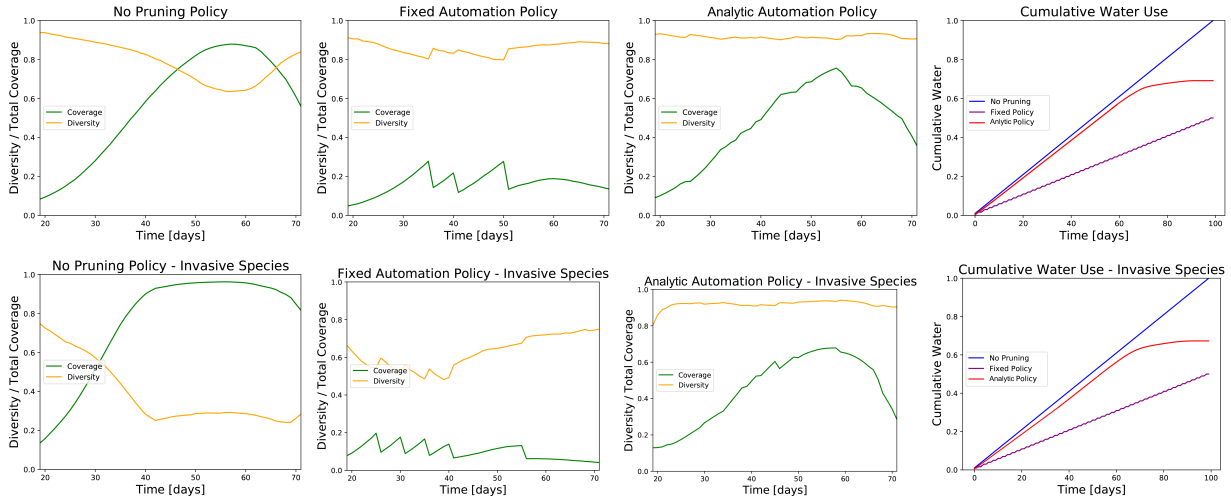


Figure 11.4: **Simulation Experiments.** The total plant coverage and diversity in a garden over a period of 50 days out of a 100 day growth period, as a result of using two different automation policies and a policy that does not prune to provide a baseline for growth and diversity. We focus on the period between days 20 and 70 because prior to day 20 the policies do not prune, and after day 70 plants start to die. We can see that there is a trade-off between the coverage and diversity. **Top row:** each policy acts on a general garden. With no pruning, the coverage in the garden is high while the diversity is low. The fixed policy manages to increase the diversity, at the price of reducing the coverage dramatically. The analytic policy finds the balance between the two metrics and achieves high diversity and coverage. **Bottom row:** the policies act on gardens that contain invasive species. If not pruned, the invasive species spreads rapidly until it covers the majority of the garden. The fixed policy tries to slow the spread but fails since it prunes both the invasive species and the other plants, and the invasive species spreads between pruning intervals. The analytic automation policy achieves worse performance than before, but manages to keep a balanced garden by pruning areas covered by the invasive species more frequently.

plant, reducing each of their radii by 15%. Pruning actions in the simulator are only applied after a specified number of days, to allow plants to germinate and grow. We set this prune delay to 20 days, after most plants reach the vegetative stage.

Plant overwatering and underwatering thresholds are closely related to the plant's neighborhood. An overwatered threshold of $T_o = 100$ was chosen as it is the maximum amount of water in the 10×10 square around the plant. We chose such a square to promote initial plant growth in our experiments. We set the underwatered threshold to $T_u = 0.1$.

At each time step, the sectors are sampled in a random order, with the policy acting on each observation independently. Once all actions are accumulated, we update the garden's water grid with irrigation amounts and update pruned plant radii. Afterwards, we distribute light, allow plants to use water and grow, and simulate water drainage and evaporation.

11.5.2 Policies

We implement two policies: a fixed policy that irrigates according to a fixed schedule and prunes uniformly, and an analytic policy that adapts according to the plants' needs and the garden's current diversity.

Fixed Policy

The fixed policy acts according to a fixed irrigation schedule, irrigating each sector it observes every two days. This method resembles the one applied in many farms and greenhouses, where an array of drippers or sprinklers irrigates in fixed time intervals at fixed locations. In addition, every 5 days, the policy prunes all the plants that grew beyond a fixed radius, limiting plants from overspreading. To prevent overpruning, we adjusted pruning to reduce a plant's radius by 5%.

Analytic Policy

The analytic policy utilizes soil moisture and plant health to dynamically prune and irrigate each sector it observes. The policy chooses among four actions: irrigate, prune, irrigate and prune, or do nothing. The policy independently decides whether to irrigate and whether to prune, and these decisions are combined to produce an action.

If a sector contains no plants or only dying plants, the policy does not irrigate. If there are any underwatered plants inside the irrigation window, irrigation is selected. Otherwise, we sum $w(x, y, t)$ for all $p(x, y)$ in $\tilde{\mathbf{S}}$. To account for overwatering, wherever $h(x, y, t)$ indicates an overwatered plant, we double $w(x, y, t)$ in the summation. If the final sum is less than a threshold, the policy irrigates.

To decide when to prune, the policy normalizes $\mathbf{p}(k, t)$. If any plant type inside the pruning window has normalized coverage greater than a threshold, the policy will choose to prune.

11.5.3 Evaluation

We evaluate each policy on 20 randomly seeded experiments per setting, using the following metrics:

1. Total Plant Coverage - We sum the total coverage in a single experiment over days 20 to 70 of the growing period, taking into account only the coverage of the plants, ignoring the uncovered space labeled as *soil*:

$$TC = \sum_{k \in (K \setminus soil), t} \mathbf{p}(k, t)$$

2. Average Diversity - We average the diversity in a single experiment, ignoring the first 20 days and last 30 days of the growing period (i.e. averaging between days 20 and 70

of the growing period, $T = 50$):

$$AD = \frac{\sum_t (v(t))}{T}$$

During the beginning and end of the growing period, the diversity is always high, since all plants are very small (germinating or dying). Therefore, these diversity measurements are not reflective of the policy's performance.

3. Water Usage - We sum the water used in a single experiment over the entire growing period (100 days):

$$WU = \sum_{x,y,t} w(x, y, t)$$

We conduct a first set of experiments on a general setting consisting of 200 plants from 10 types, for a growing period of 100 days. This setting represents a general and unbiased setup where there is a large variety of plants growing in close proximity. This leads to increased competition for resources, and similar initial conditions for each plant in expectation. We evaluate and compare the performance of the two automation policies, and an additional policy that provides maximal irrigation and does not prune as a baseline. Averaged over 20 test gardens, the results are summarized in Table 11.2. The analytic policy balances pruning and irrigation and achieves higher diversity with respect to the other policies. The top row in Fig. 11.4 shows the total coverage and diversity in a garden during a single experiment, and provides intuition for the performance of each automation policy; when irrigation is not limited and the garden is not pruned, the total coverage reaches its maximal peak, but the diversity is low due to the competition between plants. A fixed automation policy manages to maintain diversity to some extent, at the cost of overpruning. The analytic automation policy maintains high diversity, but seems to be limited by slow growing plants, which cause it to heavily prune faster growing plants. Nevertheless, it achieves high coverage and diversity. It also provides sufficient irrigation, while the fixed policy underwaters and the no-pruning baseline overwaters.

To examine the effects of an invasive species in the garden, we conduct a second set of experiments in which we replace one of the plant types with an invasive species. Invasive species grow rapidly and consume the resources in the garden. To simulate an invasive species, we create a plant with a short germination period, long vegetative and reproductive periods, and large maximal radius. Fig. 11.5 shows a comparison between the growth of a garden with and without an invasive species, marked with a red color, during 3 phases: germination, the reproductive phase, and after most plants have died. From the second row of Fig. 11.4 we can see that if not pruned, the invasive species has a negative impact on diversity. The invasive species spreads rapidly until it covers the majority of the garden, and stays dominant during a long reproductive phase, preventing other plants from growing. The fixed automation policy struggles to hold back the spread, as after each pruning, the invasive species recovers and continues to spread. The analytic automation policy performs

Policy	Coverage	Diversity	Water Use
No Pruning	**30.13**	0.78	100.00
Fixed	9.27	0.87	**50.00**
Analytic	19.91	**0.91**	67.80
No Pruning w/ Invasive	**37.30**	0.49	100.00
Fixed w/ Invasive	6.64	0.72	**50.00**
Analytic w/ Invasive	17.00	**0.89**	67.21

Table 11.2: Policy evaluations averaged over 20 test gardens with and without invasive species. Coverage is the sum over 50 days of percentage of garden covered at each day, diversity is the average of the normalized entropy of leaf distribution over 50 days, and water use is the sum of the water applied over 100 days. Results show a tradeoff between maximizing yield and maximizing diversity, and suggest that a tuned analytic policy can balance those to achieve gardens with high coverage and diversity.

better, increasing the pruning rate where there are invasive plants, and reducing it where there are none, giving other plant types the chance to grow.

11.6 Discussion and Future Work

We present AlphaGardenSim, a fast, first order simulator that uses plant growth models to simulate inter-plant dynamics and competition for light and water. We use it to tune an analytic automation policy that achieves high coverage and diversity, even in the presence of invasive species. A key factor AlphaGardenSim does not model is the effects of companion plants, that can assist neighboring plants by repelling pests or providing nutrients and shade. In future work, we will further tune the simulator with real world measurements. We will then apply the simulator to learning policies that optimize leaf coverage and diversity, and evaluate the performance of the resulting policies on real gardens.

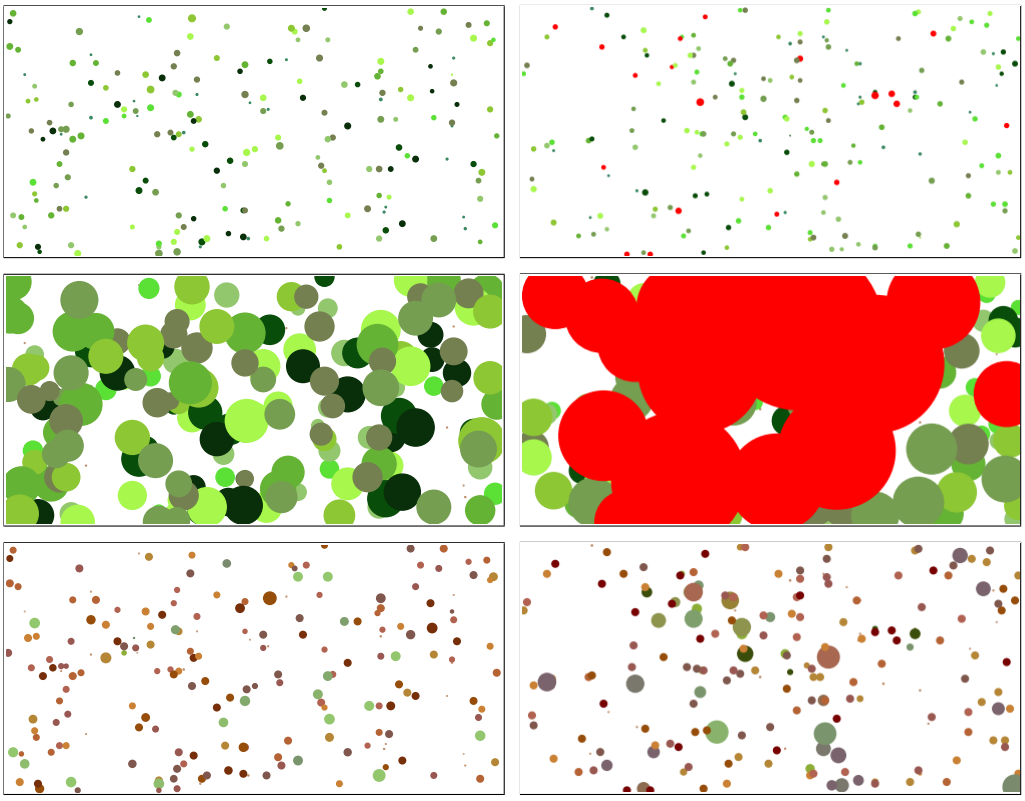


Figure 11.5: Invasive species grow rapidly and spread, consuming the resources in the garden and preventing the growth of other plants. We see the comparison between a regular garden (left column) and a garden with an invasive species (right column). **Top row:** Germination phase, both gardens look similar but the invasive species starts to germinate earlier than other plant types. **Middle row:** At the peak of the reproductive phase, the right garden is dominated by the invasive species. **Bottom row:** At the end, all plants die and the gardens reach a similar state.

Chapter 12

Learning Efficient Policies for Polyculture Farming with Optimized Seed Placements

In the previous chapter, we developed a simulator to tune automation policies for polyculture farming, though these policies varied significantly in performance. To enhance reliability, we optimized seed placement, developed a policy that can plan over a short horizon, and improved the underlying irrigation and growth models. The new policy was more reliable but less computational efficient. We then train a learned model to reduce the computation time and increase its efficiency.

12.1 Introduction

Polyculture farming, where multiple plant species are intercropped simultaneously and in close proximity, is a form of agricultural cultivation used for centuries that has been shown to enhance pest control, reduce weeds, limit soil erosion, and provide better use of light, water and soil nutrients [272–274, 298]. It is known that specific mixtures of cultivted plant species can result in higher overall yield [299]. Examples of mutually beneficial polycultures developed prior to industrial agriculture include maize-bean mutualisms, where maize provides a structural scaffold for the nitrogen-fixing leguminous vines [300], and intercropping of deep rooted native shrubs into grain cultivation, which improves water availability in arid regions [301]. More contemporary examples include shade-grown coffee, where species diverse agroforestry practices that can included cacao and banana intercropping provide not only canopy shade for coffee, increasing yields, but also provides needed habitat for birds, butterflies and other species [302]. Monoculture farming, as typically practiced in large-scale, industrial applications, is often characterized by heavy agrichemical inputs, such as chemical fertilizers and pesticides [303, 304], and increased vulnerability to disease and pestilence. The lack of long-term sustainability of industrial agriculture [305], and its implications for

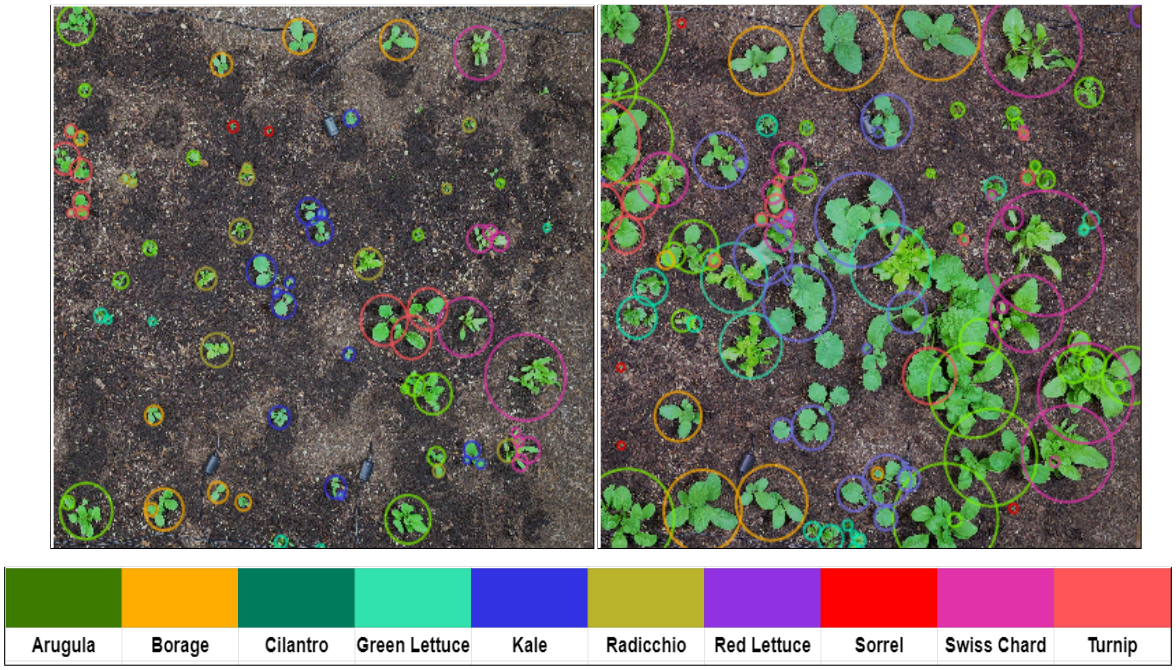


Figure 12.1: **Tuning Plant Simulation Parameters Using the Physical Testbed**. Using seeds from 10 edible plant species the seed locations were computed with a seed placement optimization process that leverages companion plants relationships to increase plant coverage and diversity. **Left:** Garden at day 17. **Right:** Garden at day 25. Plant circles as shown are predicted using the algorithm described in Section III(c).

human food security, has sparked renewed interest in polyculture [306–308].

One drawback is that polyculture farming requires more human labor than monoculture farming due to variations in germination times and growth rates.

We are exploring the use of a robot with a learned control policy to automate polyculture farming and assist - not replace - farm workers. Due to the large time constants in nature, learning such a policy through real world experiments could require many years. In prior work we introduced AlphaGardenSim [306], an efficient, open access, first order simulator for polyculture farming. The simulator models inter-plant dynamics through competition for resources, but did not take into account relations between specific plant species. In this chapter, we explore inter-plant influence, seed placements, irrigation and pruning. This chapter makes 4 contributions: (1) AlphaGardenSim 2.0, an open-source polyculture plant simulator tuned with real world measurements from a physical testbed; (2) Significant extensions to the AlphaGardenSim 1.0 simulator to model companion plant effects; (3) A seed placement algorithm that uses companion plant relations to generate seed placement plans that yield high coverage and plant diversity; and (4) A supervised-learning policy that optimizes plant coverage and diversity over a short horizon.

12.2 Related Work

In 1995, Goldberg et al. [309, 310] presented the Telegarden, an art installation that allowed anyone on the Internet to interact with a garden by planting and watering plants. Wiggert et al. [283] created a testbed that enables real-time data collection for precision irrigation based on observed plant growth and water stress. Fernando et al. [311] use a greenhouse to evaluate mobile robot monitoring of plant health and soil moisture. We build on these prior works by creating a robot-assisted garden which also facilitates real-time data collection for automated control.

Few simulators include the option to model growth of multiple species in a garden [312]. Simulators such as DSSAT [275] and AquaCrop [276] simulate large-scale monoculture farms. Whitman et al. [313] use Gaussian processes to predict weed growth across a farm. Our prior work, AlphaGardenSim [306] simulates a polyculture garden using first order models of single plant growth, simulating inter-plant dynamics and competition for water and light, but was prone to error when the plants have significantly distinct germination times, growth rates, and poor initial placements.

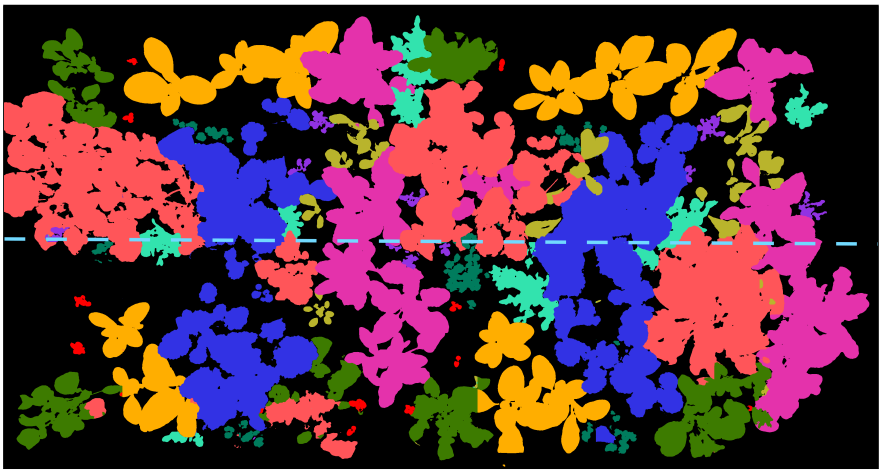
Gou et al. [314] propose a model to simulate the growth of two species in a strip-relay intercropping system. They propose a method to calibrate the plant specific parameters given observed field data. However, this model only takes into account light competition, assuming that irrigation is sufficient. The model allows for analysis of different seed placements on plant growth but restricts to the strip intercropping environment. Tan et al. [315] build on Gou et al. and include the effects of water acquisition suggesting that plants use land and water more efficiently in intercropping. However, their model does not allow for exploring spatial patterns beyond the strip-relay setting and limits to two species.

Both Gou et al. [314] and Tan et al.[315] do not make explicit use of plant characteristics to define plant inter-relations. Yu [316] use a simulated functional-structural plant model to investigate which plant traits contribute to complementary relationships and the effects of different plant placements, assuming irrigation provides sufficient water for all plants.

These simulators consider the polycultural setting, but they either do not model the light and water competition simultaneously and/or are limited by the placement geometry that they consider. We present extensions to AlphaGardenSim to incorporate plant relationships and consider inter-plant cooperation.

12.3 Plant Phenotyping

We fabricated a 1.5×3m garden bed in the UC Berkeley greenhouse and mounted a commercial FarmBot gantry robot system [317] to tune and test AlphaGardenSim on real plants. We use a high-resolution overhead camera that has a full view of the garden, soil moisture sensors and an automated pruning tool. A uniform and nutrient rich soil was used to reduce stochastic effects in nutrient availability.



Name	unknown / soil	kale	swiss chard	turnip	arugula	borage	green lettuce	cilantro	sorrel	radicchio	red lettuce
% Coverage	48.4	11.8	10.1	9.7	7.4	7.0	1.7	1.0	0.3	0.2	0.1
Baseline IoU	0.88	0.78	0.81	0.55	0.40	0.82	0.72	0.46	0.70	0.13	0.39
IoU with augmentation	0.92	0.92	0.93	0.81	0.72	0.95	0.87	0.71	0.76	0.57	0.65

Figure 12.2: **Learned Plant Segmentation Model**. The figures above (from top to bottom) show an overhead image from October 6, 2020, and the classifier output from the network with augmented data. The overhead image is split in half as shown by the blue line. The top half is for training while the bottom half is for testing. Below, the table shows how much of the garden is covered by each plant and its respective IoU score based on the bottom half only. By adding augmented data, the model was able to more accurately classify unseen leaves when compared to the baseline with no augmented data. Low IoU for radicchio and red lettuce is consistent with a low percent of coverage.

Camera The digital camera, mounted 2m above the garden bed, is a Sony SNC-VB770 and takes images every 24 hours. Its 35 mm sensor has a maximum 4240×2832 resolution (1.4x higher than 4K) image mode, and a 3840×2160 (4k) video mode. We also selected a 20 mm focal length Sony lens designed for SLR-type mirror-less cameras of comparable quality. Its optical design minimizes aberrations and distortion, providing a clear detailed image with no fishseye effect.

Leaf Segmentation See Fig. 2. We implemented semantic segmentation to study plant canopy coverage. To estimate the canopy distribution from overhead images, we predict a plant type, or "unknown," for each pixel in the overhead image, using the UNet [268] architecture with a ResNet34 [318] backbone pre-trained with weights from ImageNet [319] data. We then use two 2,000×3,780 and one 1,630×3,478 overhead images taken of the garden on September 26, September 30, and October 6, 2020, with hand-labeled ground truth masks of plant phenotypes, and divide each image into a top half and bottom half as shown in Fig. 12.2. We extract 48 512×512 patches from the top half for training. We then modifwey these patches through actions such as shearing, shifting, and scaling.

A key challenge is generalizing across all garden days and plant life stages. To address this, we extract individual leaves from October 1 to 22, 2020 to get samples of various sizes, plant health, lighting, and texture. We augment the dataset by overlaying individual leaves on top of each patch. By varying the position and pose of each augmented leaf, we create 100 patches of training images from a single overhead image. This additional augmented data improves network robustness as shown in Fig. 12.2.

We train using 3,180 patches, 1,500 of them from augmented data, and 1680 from the original data. Training uses categorical cross-entropy loss over 100 epochs and utilizes a 75-25 train-validation split. The output is a 512×512×11 array, with 11 softmax likelihoods, representing ten plant phenotypes (or "unknown"). We classify a pixel by choosing the largest likelihood, and create a predicted mask for an overhead image by combining the classified 512×512 patches. Fig. 12.2 shows the network's prediction on the bottom half of the image from October 6, 2020, which is unseen to the network. When evaluated, the model has a mean IoU of 0.80. The model performs well in identifying plant types with high coverage, but has lower accuracy in plants that are not common in the overhead image.

Converting Segmentation Masks to Circles See Fig. 12.1. To convert the pixel-wise segmentation masks into the circular model used in AlphaGardenSim [306] we track plant centers and radii. We define the plant center as the average over all pixel locations in the plant's segmentation mask. We define the radius as the distance from the new center to the farthest point on its contour and note that although the seed locations are known, plant centers change over time due to phototrophy [320] and irrigation [321]. Given the centers and radii of all plants on day $t-1$ as a prior, we use three heuristics to guide the circles update on day t: the previous center, a minimum-, and a maximum-radius estimate. The radius estimates are computed by finding the maximal and minimal observed radius per day for

each plant type using real world measurements as described in Section V. We use a breadth-first-search (BFS) algorithm, traversing radially outward to update each of the circles. The BFS terminates when either the max radius estimate is achieved or the percentage of new pixels discovered during the previous two iterations is lower than a threshold (1%). After termination, the new center is the center of mass of the pixels within the circle and the new radius is the distance from the termination point to the center, as shown in Fig. 12.1.

12.4 Irrigation Model

We utilized six TEROS-10 [322] volumetric water content (VWC) soil sensors connected to a ZL6 Data Logger [322] to measure soil moisture. The first set of experiments refined the irrigation application parameter, $a(x, y, t)$, the amount of irrigation applied at point (x, y). We identified the flow rate from the FarmBot nozzle to be 0.083 L/s. The area of influence from the nozzle is a circle of 0.04m radius.

We then used the soil moisture sensors to determine radial flow. By watering at varying distances spanning from 0.04m to 0.10m from the center of a soil moisture sensor, we determined a model as follows: beginning outside of the 0.04m radius, the water gain is roughly half that when compared to water gain within the radius. This trend continues each additional 0.01m away from the center watering point up until 0.09m. Let $w(x, y)$ be the VWC centered at at point (x, y) in the garden. Thus, $\Delta w(x_d, y_d) = (1/2)^d * gain$ where d is distance measured in 0.01m outside of the 0.04m radius, (x_d, y_d) is a point $d + 0.04$m away from (x, y), and $gain$ is the moisture gain for soil directly under the nozzle.

We studied changes in soil moisture content from irrigation procedures to tune the local water loss parameter, d. Using soil moisture over time curves produced from irrigation experiments in which we watered at different frequencies, we identified a water gain and water loss period post watering event. In the water loss period, the change in soil moisture over time t in hours fits nicely to a weighted, negative logarithmic decay: $\Delta w = -0.01675 \cdot \ln t$.

Furthermore, we used the soil moisture sensors to tune the prior soil moisture content parameter, $w(x, y, t - 1)$. Here, the quantity that is important for identifying real world irrigation policies is the soil's specific maximal VWC which describes how much moisture the soil can store [323]. By saturating several different samples of soil that we used in the physical testbed, we discovered the max VWC of our soil to be around 0.3, and capped the $w(x, y, t - 1)$ accordingly.

Through the execution of irrigation and soil moisture experiments in the physical testbed, we made adaptations to parameters based on Richards equation for soil moisture dynamics used in [306]:

$$w(x, y, t) = \max(w(x, y, t - 1) - d + a(x, y, t) - U(x, y, t), 0)$$

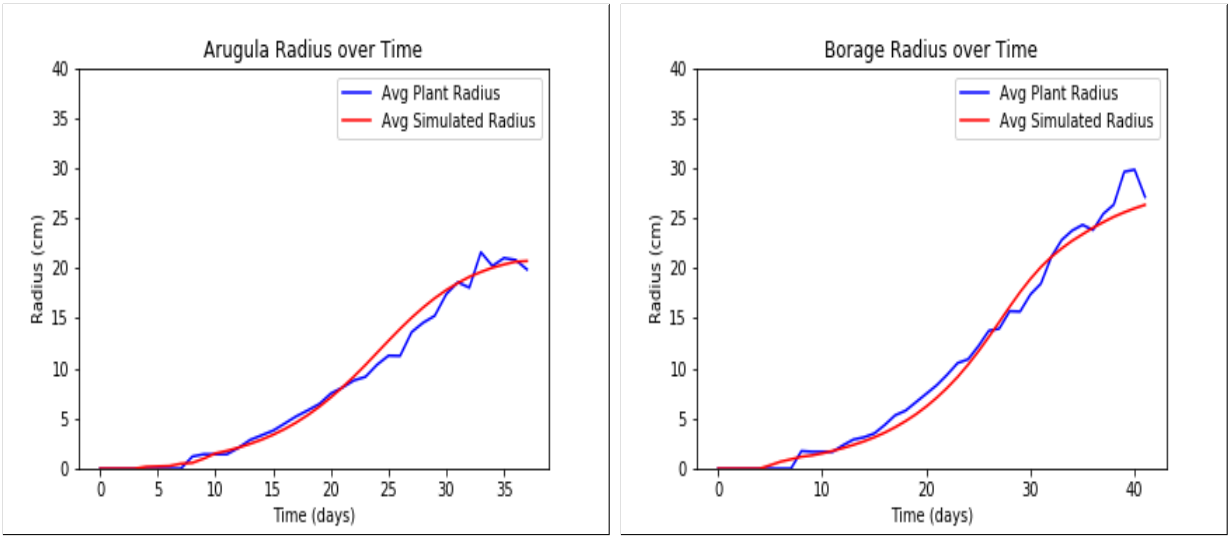


Figure 12.3: Plots of growth curves, radius (cm) over time (days), for both simulated and real world plants. Borage was occluded by other plants after day 40. Arugula was occluded starting on day 35. The blue curve is real world radius. The red logistic curve is simulated radius from AlphaGardenSim.

12.5 Growth Analysis

The standard agriculture parameters that dictate growth are germination time, maturation time, and maximum radius. Individual plants also depend on the light, water, the plant starting radius and height, and the number of days the plant remains in its growth phase versus its wilting phase. The starting radius and height in the simulation and the number of days in which a plant grows before wilting are sampled from a normal distribution centered around the values defined above.

Through the use of overhead photos of the physical testbed, we can measure each plant's radial growth. Similar to how growth is modeled in our simulator, we annotated every plant with a point at its center and a point at its outermost radius in an image from every day since seeding. We then made a rough conversion of pixel coordinates to real world coordinates in *cm*. These coordinates were then used to find plant radius in *cm* for that day.

By analyzing the growth of the one-hundred-twenty plants in the garden over 46 days, and averaging the growth of a plant with others of its same species we created a new growth function in AlphaGardenSim:

$$r(t) = \frac{r_1}{1 + r_1 \cdot e^{-c_1 t}}$$

where r_1 is the plant's radial growth potential, which controls how large the plant will grow, and c_1 is the plant's radial growth rate, which controls how fast the plant will grow. Both values were fitted using measurements from the garden, as shown in Fig. 12.3. The growth

Plant Type	g_0	g_1	m_0	m_1	r_1	c_1	$c(35)$	$e(35)$
Borage	7	7	49	55	60	0.09	3107	6.61
Kale	3	7	62	55	65	0.10	7450	5.41
Swiss Chard	7	7	53	50	47	0.11	5536	9.93
Turnip	3	7	42	47	53	0.11	3961	10.04
Green Lettuce	7	9	43	52	27	0.08	232	7.46
Arugula	5	8	45	52	40	0.10	1133	5.50
Sorrel	7	15	53	70	8	0.08	59	9.58
Cilantro	7	10	53	65	20	0.09	23	10.76
Red Lettuce	5	12	45	50	28	0.09	10	11.61
Radicchio	5	9	83	55	53	0.09	53	9.28

Table 12.1: Growth Analysis: Where g_0 (days) is original germination time, g_1 (days) is tuned germination time, m_0 (days) is original maturation time, m_1 (days) is tuned maturation time, r_1 is radial growth potential, c_1 is radial growth rate, $c(35)$ (cm^2) is the simulated canopy coverage on day 35, and $e(35)$ (cm) is the mean absolute error on day 35 between simulated and average real world radius. Original values were taken from published plant tables [324]. Growth time is found by subtracting g_1 from m_1. Sorrel not only germinated later than other plants, but also had a growth potential and growth rate that was minuscule compared to other plants in the physical testbed.

parameters of all ten plant species were tuned in the simulator to match real measured values, as shown in Fig 12.1. The mean absolute error (MAE) between the simulated plants and physical testbed plants is displayed in Table 12.1, along with growth parameters. It should be noted that we observe substantial plant overlap by day 35, and after this day it was difficult to identify a plant's outermost radius. Thus, the MAE is taken on day 35 rather than day 46.

12.6 Companion Planting

Companion planting is an ancient technique of polyculture where mutually beneficial plant types are placed in proximity to each other. A positive or negative relationship between companion plants can exist due to above and below ground interactions [325], [326], [327]. Above ground includes physical environment changes such as providing shade, protecting

against weather damage, and supplying structural support. Below ground interactions include providing nitrogen which fertilizes the soil, root-root activity and allelopathy, which occurs when a plant releases toxic chemicals that inhibit growth of other plants. While some crops such as grain and fruit trees require uniform spacing for optimal growth or harvesting, some such as leafy greens do not and can take advantage of large seed beds such as the one in the physical testbed. This motivates a method to find a garden seed placement that exploits plant relationships, which can lead to different yields and more or less efficient use of resources [328], [329].

Modeling Companionship Consider a garden with N seeds: $\{s(1), \cdots, s(N)\}$. Denote $\mathcal{K}$ as the set of plant types in the garden, $p(i) \in \mathcal{K}$ as the plant type of seed i, and $l(i) = (x_i, y_i)$ as the location of seed i. Let $r_k^{\max}$ denote the expected maximal radius of plants of type k. To model plant interrelationships we use the plant relationship matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{K}| \times |\mathcal{K}|}$. $\mathbf{C}_{i,j}$ stores a number that describes the level of companionship between plants of type i and j, which are not necessarily symmetric. In simulation, $\mathbf{C}$ is used to calculate a local plant specific companionship factor c. For a given plant i,

$$c_i = \sum_{j \in [1, \cdots, N], j \neq i} \frac{\mathbf{C}_{p(i),p(j)}}{\|l(i) - l(j)\|_2^2}$$

The strength of the companionship decays as the distance between them grows.

In AlphaGardenSim 2.0, each plant's daily radius grows according to a factor $\tilde{G}$, determined by the local water and light resources and competition. The effects of companionship are modelled by a change to the growth value in the simulator. The growth value is updated to be $G = \tilde{G} \cdot c$.

The $\mathbf{C}$ matrix was determined using the one-hundred and twenty annotations provided by analyzing growth rates in the physical testbed. Plants in the same relative location on each side of the garden were compared to one another as well as the average growth; if the same plant on both sides exhibited either exaggerated growth or stunted growth, the neighbors were noted and assigned positive or negative scalar values, respectively, to indicate companionship between plants. These scalar values were then tuned to minimize the MAE between simulated and real world individual plants.

Seed Placement Given two plants, the larger their relationship score the closer they would prefer to be. Then for a garden of width W and height H with N seeds, the following problem is solved to compute seed coordinates (x_i, y_i) for every plant i:

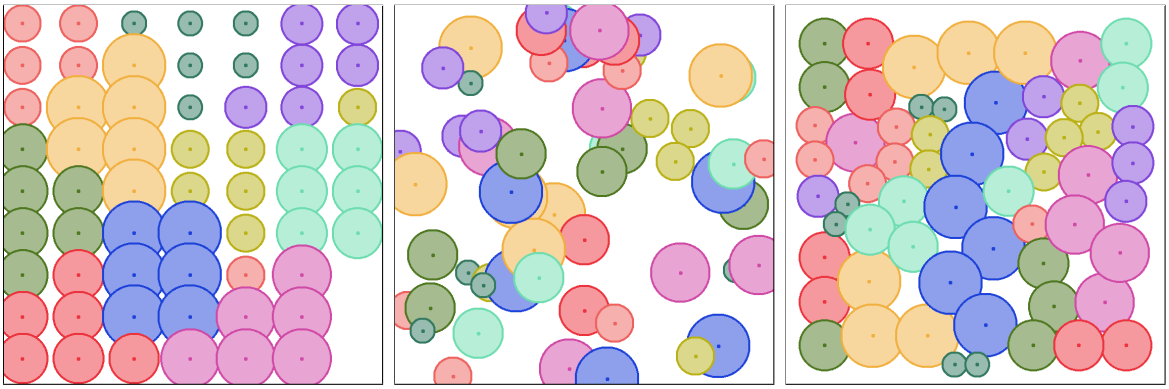
$$
\begin{aligned}
\max \quad & \sum_{i,j\in[N],i\neq j} \frac{\mathbf{C}_{p(i),p(j)}}{\|l(i)-l(j)\|_2^2} & \\
\text{s.t.} \quad & r_{p(i)}^{\max} \leqslant x_i < W - r_{p(i)}^{\max} & \forall i \in [N] \\
& r_{p(i)}^{\max} \leqslant y_i < H - r_{p(i)}^{\max} & \forall i \in [N] \\
& \alpha(r_{p(i)}^{\max} + r_{p(j)}^{\max}) \leqslant \|l(i)-l(j)\|_2 & \forall i \neq j \in [N]
\end{aligned}
$$

The objective is to seed plants with a positive symbiotic relationship close to each other and vice versa when the relationship is negative. The first and second constraints ensure that seed locations are within the garden boundaries. Finally the last constraint ensures that plant radii do not overlap more than $100(1-\alpha)$ percent. α is a parameter in $[0,1]$ that specifies the maximal level of overlap between plants. A larger α results in more plants overlapping and thus interacting with each other. However if α is too large, plants tend to create very tight clusters of companion species, resulting in a difficulty for these plants to grow and a limited coverage. In this case, the negative effects of competition between neighbouring plants outweigh the benefit brought by the proximity of companion plants. To set α for a 150cm $\times$ 150cm garden with 10 species and 6 seeds of each type, gardens were generated with $\alpha \in [0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95, 1.0]$. Each garden was simulated 5 times and $\alpha = 0.8$ producing the highest average coverage and diversity was selected. The resulting seed placement is shown in Fig. 12.4.

12.7 Pruning and Irrigation Policies

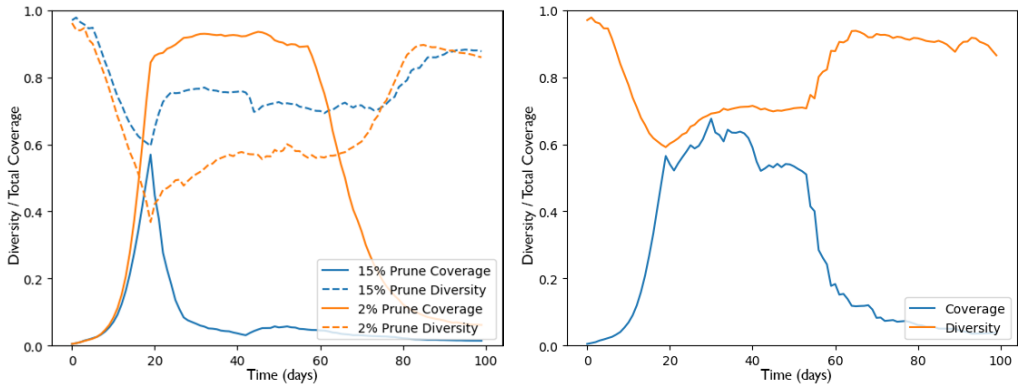
Analytic Automation Policy In prior work [306], we presented an analytic automation policy, Fixed Pruning, with hand tuned parameters. For $D(k)$, a set of k plant types available in the garden, the policy observes the local canopy coverage, plant health, and soil moisture levels defined by a $\frac{H}{10} \times \frac{W}{5}$ sector of the garden. In addition, the policy observes $\mathbf{p}(k,t)$, the global population in the garden as a distribution over point types D. It applies one of four actions in each sector: irrigate, prune, irrigate and prune, or null. When pruning, the policy creates 5×5cm pruning window centered around a target plant, simulating the inaccuracy of a pruner, and reduces the radii of all plants visible in the window by a fixed ratio U. To decide to prune, the policy checks if the proportion of any plant type in the pruning window is higher than a uniform threshold. To evaluate the policy's performance, we compute the average of canopy coverage, plant diversity and water usage across days 20 to 70. We focus on these 50 days since during these days both the fast and the slow growing plants are in a growing stage.

However, as described in Section VIII(b), experiments suggest that Fixed Pruning struggles to manage plants with significant differences in germination times, maturation times and max radii.



(a) Greedy (b) Random (c) Optimized

Figure 12.4: **Left:** Seed placement achieved by greedy neighbourhood swap algorithm which ignores plant maximal radii, only taking into account companionship scores. This produces an artificial looking garden with limited interactions between plants of different species. **Middle:** Random seed placement, producing a sparse garden with seeds too close together, limiting the achievable coverage and diversity due to competition. **Right:** Seed placement obtained by using the optimization described in Section V. Small clusters of plants in irregular shapes are spread across the garden, allowing for a variety of interactions between different plant species.



(a) Fixed Pruning (b) Variable Pruning

Figure 12.5: **Left:** Simulation results for Fixed Pruning with fixed prune levels of 15% and 2%. 15% prune level achieves good diversity but low coverage. 2% prune level is the opposite. **Right:** Variable Pruning on gardens over 100 days with the fast and slow plant types from Table 12.2. Favoring coverage initially, the policy uses variable pruning levels to achieve an average coverage of 0.51. As Variable Pruning begins to value diversity more, it uses higher prune rates beyond day 50 to achieve good coverage and diversity.

Plant Type	Germination (days)	Maturation (days)	Max Radius (cm)
Fast growing	9.8	99.4	140.4
Slow growing	25.6	156.0	42.4

Table 12.2: Average germination time, maturation time, and max radii of 5 fast and 5 slow growing plant types. We vary germination times, maturation times and max radii of plants to identify combinations, such as the one in the table, on which the analytic policy achieves significantly lower coverage, as shown in Fig. 12.5(a).

1-Step Lookahead Policy To address this limitation, we introduce Variable Pruning, a policy which dynamically selects a pruning level $U(t) \in \mathcal{U}$ for each day t from a discrete set of six pruning levels $\mathcal{U}$, by taking a 1-step lookahead, simulating the potential coverage c_{u_i} and diversity values d_{u_i} that would result from choosing pruning level $u_i \in \mathcal{U}$ on the current state of the garden. With the simulated results, the policy uses tunable weights w_c and w_d to favor either coverage or diversity at different times of the growing period. To favor coverage during early growing periods and diversity later on, we set $w_c = 1 - \frac{\tilde{t}}{50}$ and $w_d = \frac{\tilde{t}}{50}$ where $\tilde{t} = t - 20$, starting to affect after the prune delay ends and lasting for 50 days until day 70. To favor diversity early and coverage later, the weights are swapped. The policy uses a weighted sum to determine which pruning level is preferable for day t:

$$U(t) = \max_{u_i \in \mathcal{U}} (w_c \cdot c_{u_i} + w_d \cdot d_{u_i})$$

Once Variable Pruning chooses $U(t)$, it uses Fixed Pruning to determine actions for the sectors observed each day to maximize both diversity and coverage.

Learned Policy The computation time of Variable Pruning however, increases with $|\mathcal{U}|$. Each day, Variable Pruning must run the simulator $|\mathcal{U}|$ more times than Fixed Pruning.

To reduce the computational cost of Variable Pruning, we train a deep supervised learned policy, Learned Pruning, mapping prune level $U(t)$ demonstrations to full garden observations as depicted in Fig. 12.6.

12.8 Simulation Experiments

Seed Placement We compare the coverage and diversity achieved on two types of gardens shown in Fig. 12.4: (i) random seed placement and (ii) optimized seed placement using six seeds for each of the ten plant types in Table 12.1. Averaging over 10 simulations each, the optimized garden achieves over 60% more coverage and 10% more diversity than the randomly seeded gardens.

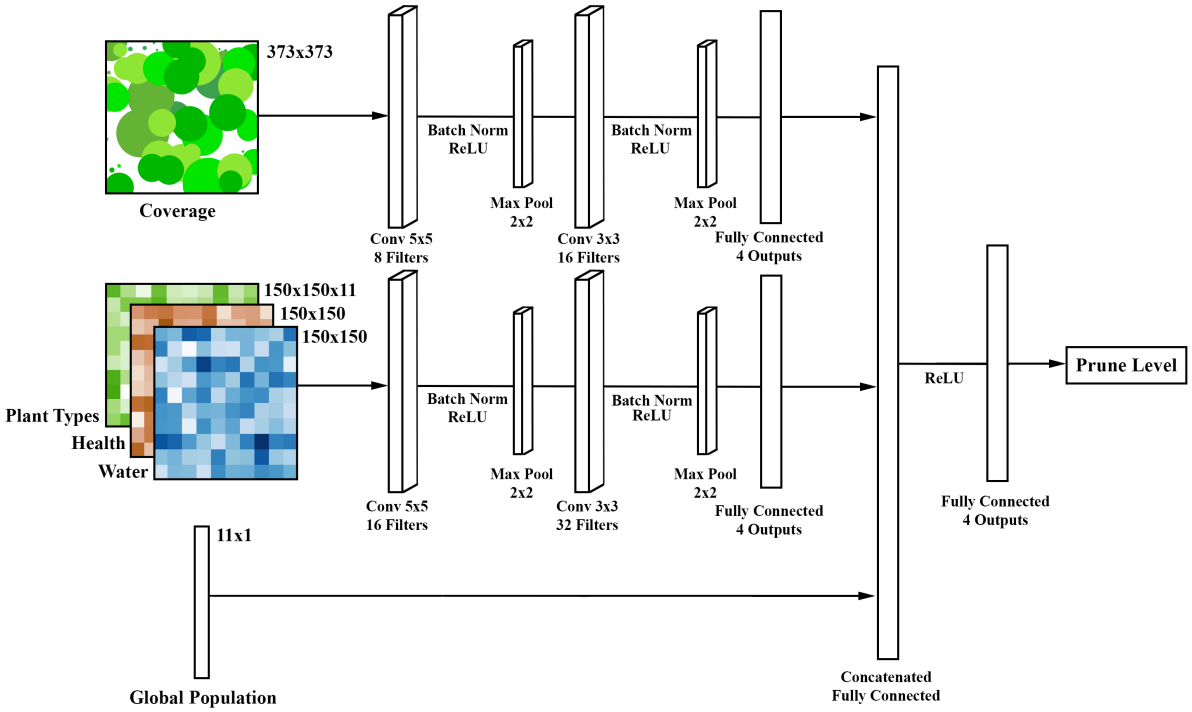


Figure 12.6: Learned Pruning Policy. A deep CNN with 18,244 parameters. The network takes three inputs: an RGB image of the full garden, the distribution of plant types, plant health and water levels, and the global population distribution with soil. A prune level is predicted for the input observation.

Fixed Pruning Performance To illustrate the shortcomings of Fixed Pruning on plants with different germination times, maturation times and max radii, we simulated 2 fixed pruning policies, with 15% and 2% pruning levels respectively, on a garden with 100 plants, 10 plants from each of the 10 plant types in Table 12.2 where faster growing plants grow 2X-20X faster than slower ones. Illustrated in Fig. 12.5(a), since 5 species grow significantly faster than the other 5, garden diversity rapidly drops during days 10 to 20. To achieve uniform plant diversity, 15% heavily prunes the faster plants to match the size of the slower growing plants resulting low coverage. In contrast, 2%'s pruning fails to keep up with the fast growing plants, resulting in lower plant diversity compared to 15%.

Variable Pruning Performance To compare Fixed Pruning and Variable Pruning, we evaluate their performances on two sets of 10 plant types. We initialize 150 × 150cm sized gardens with 100 plants each sampled with replacement from the plant types. For Variable Pruning, we set $w_c = 1 - \frac{\tilde{t}}{50}$ and $w_d = \frac{\tilde{t}}{50}$ to favor coverage early and diversity later. After experimenting with different pruning levels, we provided Variable Pruning these pruning levels: $\mathcal{U} \in (5\%, 10\%, 16\%, 20\%, 30\%, 40\%)$. The first set of plants are from Table 12.1 and

Metric	Fixed	Variable	Learned
Avg coverage	0.38	**0.44**	-
Avg diversity	**0.92**	0.91	-
Avg water use	0.06	0.06	-
Avg coverage	0.24	**0.51**	0.50
Avg diversity	**0.77**	0.73	0.73
Avg water use	0.08	0.08	0.08
Computation time (seconds)	-	987.56	**0.92**

Table 12.3: Policy evaluations of Fixed Pruning, Variable Pruning and Learned Pruning averaged across 20 test gardens. **Top 3 rows:** use the 10 plant types from Table 12.1. **Bottom 4 rows:** use the plant types from Table 12.2.

have similar growth parameters. From experiments, we found that a fixed prune level of 15% for Fixed Pruning leads to the highest coverage and diversity values on the plant set. Results averaged across 20 test gardens with random seed placements are summarized in Table 12.3. While Variable Pruning achieves higher coverage due to its ability to favor coverage over diversity during early growing periods, both policies achieve similar diversity and water use. This is expected as a fixed prune level is able to handle plant types with similar growth patterns.

Using the same parameters for both policies, we evaluate their performances on the fast and slow growing species from Table 12.2. Results are presented in Table 12.3 and Fig. 12.5. Fixed Pruning achieves low coverage on these gardens, killing the plants at the beginning of the growing period. The high diversity achieved afterwards represents a uniform but empty garden. Variable Pruning, by favoring coverage early on, initially uses a small prune level of 5%. As w_d increases and w_c decreases over time, the policy shifts to favoring diversity and uses higher prune levels between 10% and 40%. As a result, Variable Pruning achieves high coverage and diversity.

Learned Pruning Performance To achieve a computationally efficient policy, we train Learned Pruning to map full garden observations from gardens with the plant types in Table 12.2 to prune levels. We simulate Variable Pruning on 10,000 gardens with randomized seed locations to collect prune level demonstrations. The network is trained with 800K demonstrations for 30 epochs with the Adadelta [47] optimizer and mean squared error loss. Table 12.3 summarizes results averaged across 20 test gardens withheld from the training dataset. Learned Pruning achieves comparable performance to Variable Pruning but is over 1000X faster in predicting $U(t)$ for days 20 to 70.

12.9 Discussion and Future Work

This chapter presents a physical polyculture farming testbed for estimating plant growth parameters and inter-plant companion effects. We use the estimated parameters to tune AlphaGardenSim 2.0 parameters, and developed an optimization algorithm that uses companion plant relations to generate a seed placement which yields high coverage and plant diversity. We trained a supervised-learned policy that is able to achieve high leaf coverage and plant diversity 1000X faster than a lookahead policy. In future work we will estimate stochastic models of growth parameters observed in the physical testbed and use these to optimize seed placements for subsequent growth cycles. This seed placement algorithm, learned plant phenotyping model, and learned irrigation and pruning models will be combined into a fully automated controller that will operate irrigation and pruning tools over multiple plant growth cycles.

Part V

Efficient and Reliable Autonomous Suturing

Chapter 13

Automating 2D Suture Placement

In this chapter, we develop an optimization to improve the reliability of autonomous suture placement for non-linear wounds. The goal is to find the number and locations of entry and exit suture points, and the quality depends on various factors, including the forces that are applied to the wound in the process.

13.1 Introduction

Suturing is the process of sewing a wound or laceration closed to allow it to heal naturally. It is extremely common in surgery and trauma care and involves long sequences of precise, repetitive movements – something that is often burdensome to humans.

This work focuses on one aspect of automating suturing: the sub-task of *suture planning*, which is to find an appropriate sequence of needle insertion and extraction points. Having a high quality placement of suture is crucial for healing, as having sutures that are too close or tight may lead to ischemia (under-supply of oxygen to the tissue), while having sutures too far apart may lead to insufficient closing force on the wound to ensure the edges stay together [330]. Furthermore, sutures may exert shear forces along the wound, which cause more pronounced scarring and lead to cosmetically unappealing results [331] [332].

Thus, it is ideal to plan sutures by directly optimizing the forces they are expected to generate. Humans cannot directly estimate closure and shear forces as they suture, and typically rely on rules of thumb, intuition, and experience to guide suture placement.

This chapter makes five contributions: (1) A novel objective, optimizing Elliptical Force on the wound, based on an extension of the well-known "diamond force model" [333, 334] to nonlinear 2D wound shapes; (2) A novel formulation of planar, non-linear suture planning as a constrained optimization problem; (3) An analysis of the SP2DEEF algorithm's output on wounds of varying degrees and curvature; (4) Experiments comparing the suture placements from SP2DEEF to those chosen by a human surgeon; and (5) An interactive interface that allows surgeons to upload an image, trace the wound curve, view proposed sutures and edit suture placement.

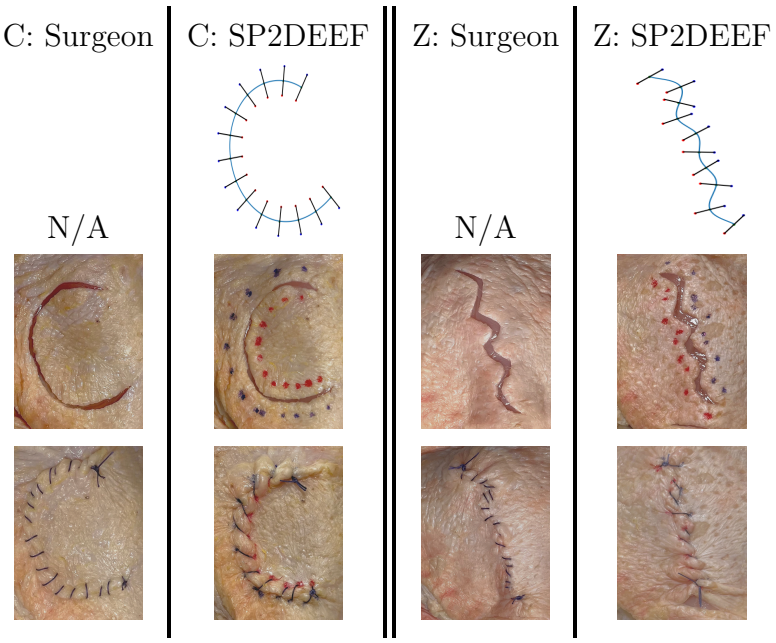


Figure 13.1: Physical Experiments with chicken skin. Top row: SP2DEEF's outputted placements; Middle row: Surgeon's initial state and physical ink markings of SP2DEEF's placements; Bottom row: after suturing. Left half: C-shaped wound; Right half: Z-shaped wound. First and third columns: Surgeon's placement; second and fourth columns: SP2DEEF's placement. SP2DEEF's placements were evaluated as equal or better than the surgeon's placement by the surgeon.

13.2 Related Work

Automated and robot-assisted suturing has seen considerable study over the last decade, with particular focus on two sub-tasks: *2D suture planning*, which considers where to place sutures (the focus of this work), and *3D needle path planning*, which considers how to guide the needle to best accomplish a desired suture. Automating either of these sub-tasks can provide valuable assistance to a human surgeon, while having both may potentially lead to a fully automated suturing system.

13.2.1 Needle Path Planning

Much of the work on autonomous suturing focuses on needle path planning in the vertical plane (orthogonal to the wound line), as pushing a needle through deformable tissue represents a major challenge for sensing and control. In general, needle path planning focuses on a single suture at a time.

Nageotte et al. [335] proposed a kinematic analysis and geometric modeling of the problem of the stitching task in laparoscopic surgery. The work particularly uncovers the degree

of uncertainty which the surgeon has with regard to where the tip of the needle is; that is, the exit point may not be exactly where desired.

Schulman et al. [336] applied the *transfer trajectory algorithm* to take trajectories from human demonstrations and adapt them to new environment geometry for suture needle path planning. Another approach developed by Sen et al. [337], was one of sequential convex optimization. As with the other papers, the optimization was over the execution of a single suture. Their approach considers multiple sutures, but the suture locations are chosen by linear interpolation of the start and end points, and each suture is treated independently thereafter. This method does not capture the constraints that the curvature of a wound might place on suture placement. Later, Shademan et al. [338] studied suturing on intestinal tissue. Various constraints for good suturing are discussed in this work: for instance, that sutures should be perpendicular to the wound, and the gaps between the sutures must be small enough to avoid leakage, but not too small as to prevent bloodflow. However, they are primarily used as assumptions that hold for a set of planned sutures that have been decided beforehand.

Other papers consider additional aspects of needle path planning: for example, Pedram et al. [339] presented an algorithm that takes in the desired suture entry and exit points on a wound as input and computes the needle shape, diameter and path so that the execution of the suture satisfies recommended suturing guidelines. These guidelines included: a minimization of tissue trauma, orthogonal sutures, positioning the needle to allow for successful grasps, suture symmetry and being able to enter and exit at the points defined by the surgeon. The optimization weights were selected based on the recommendation of surgeons and fine-tuned during simulations. Jackson et al. [340] proposed a Kalman filter to model the internal deformation force generated by a needle as it is driven through tissue. Similarly, Pedram et al. [341] described a needle stitch path planning algorithm which deals with optimal motion of the needle inside the tissue, with the goal of entering the tissue perpendicularly; reaching specific suture depth; and minimizing tissue trauma.

13.2.2 The Suture Planning Problem

There is prior work in planning the location of suture entry and exit points in the horizontal plane, for specific wounds, and in certain controlled settings. In particular, robotic minimally-invasive surgery (RMIS) must conform to the robot's kinematic constraints and the potentially very tight space in which the operation takes place.

Saeidi et al. [23] describe a planning algorithm for autonomous suturing using a segmented point cloud and demonstrated its application with the Smart Tissue Autonomous Robot (STAR) system. However, their technique assumed a straight-line wound (as can be expected in surgery, where the wound is the result of a surgical incision). Similarly, Thananjeyan et al. [342] studied suture planning for a circular wound.

The primary constraints in both works were limitations to the robot's movement ability, either from kinematics or from a sharply bounded workspace, with the objective being to

ensure evenly-spaced sutures with the gaps between them being as close as possible to an ideal distance.

However, when dealing with more complex wound shapes, sutures may interact with each other in more complex ways, and thus planning and evaluating an effective set of sutures may require a more detailed model of how the sutures hold the wound together.

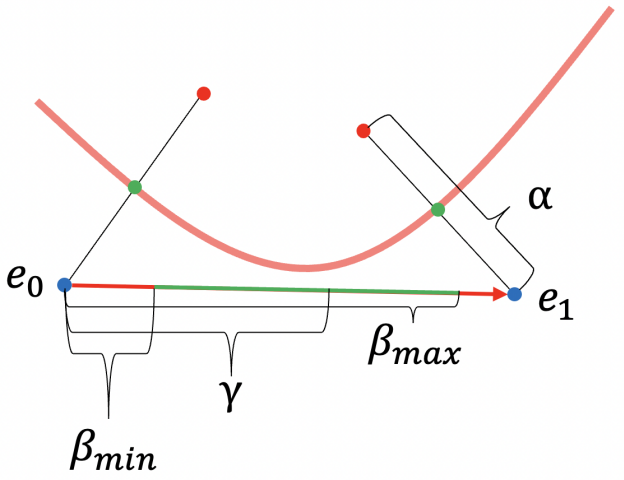


Figure 13.2: Wound centerline shown in red, with two entry points (red) and two exit points (blue). Consider the distance between extraction points e_0, e_1. If the distance is between β_{min} and β_{max}, i.e. the green zone, it meets the constraints. Hence e_0 and e_1 violate constraints.

13.3 Problem Statement

Given an image with points along the wound selected by the surgeon, the suture planning problem consists of choosing the number and placement of sutures to best close the wound. We denote the number of sutures as n, and, for most of what follows, we treat it as fixed and consider the optimization problem which represents the task of finding the best placement of n sutures; we discuss how n is selected in Section 13.4.2. The parameters of this problem come in two types: the first type denote physical constraints or objectives. For example, the minimum allowable distance between any two skin puncture points or the ideal distance between the insertion and extraction points of a single suture (which we refer to as the *suture width*, α) We represent such constraints with Greek letters. The second kind of parameter are scalar weights which set the relative importance of different components of the objective function, and are represented by $c^\star$ where the superscript $\star$ denotes the corresponding component.

Let α be the *suture widths*, that is, the distance between the insertion and extraction points of a single suture. These two points are placed at equal distance apart from the wound such that their midpoint lies on the wound curve. Additionally, no two insertion

and/or extraction points should be closer than a certain minimum distance, denoted $\beta_{\min}$, and no two consecutive insertion and/or extraction points should be further than a certain maximum distance, denoted $\beta_{\max}$. We also define the *suture distance* which is the straight line distance between the midpoints of two stitches. Guidance for surgeons suggests that an ideal suturing distance is 5mm [343]. Let γ be this target distance between sutures, and let ℓ be the length of the wound.

Let L^d be the Mean Squared Error (MSE) defined as average difference between computed suture distance and the target suture distance γ. Let $L^{\text{var\_center}}$ be the variance of computed suture distances. Let $L^{\text{var\_ins\_ext}}$ be the variance of the distances between consecutive suture insertion and extraction points. In this term, we sum up the variance of the distance between insertion points and the variance of the distance between exraction points. Let L^f be the MSE between the closure force at each point along the wound curve and an ideal value. Similarly, Let L^{shr} be the MSE between the shear force at each point of the wound and an ideal value. Note that these target values are defined directly from the surgeon's input to the system.

Assume the suture planning is constrained to a given suture width α. Further assume that sutures are constrained to be orthogonal to the wound curve; therefore it is sufficient to specify the number n of sutures and at which points $0 \leqslant s_1 \leqslant \ldots \leqslant s_n \leqslant t_{\max}$ along the wound they are placed, to describe a complete suture plan of needle insertion and extraction points. Here, s_i serve as our decision variables (the center of suture i being $w(s_i)$). Additional we impose hard constraints that concern minimum and maximum distances allowed between consecutive insertion and extraction points. We denote these as constraints $A^{\min}, A^{\max}$. Finally, we enforce that sutures should not 'cross.' That is, if we draw a line from corresponding insertion point to extraction point for each suture, it should be the case that none of these lines are crossing. This constraint is denoted as A^c.

Each of the optimization terms $L^{\cdot}$ is weighted with a factor $c^{\cdot}$. The objective is then to find the sequence of s_i for all $i \in \{1, ..., n\}$ satisfying:

$$\begin{aligned} \min \quad & c^d L^d + \\ & c^{\text{var\_center}} L^{\text{var\_center}} + \\ & c^{\text{var\_ins\_ext}} L^{\text{var\_ins\_ext}} + \\ & c^f L^f + c^{\text{shr}} L^{\text{shr}} \\ \text{s.t.} \quad & \\ & A^{\min}, A^{\max}, A^c \\ & 0 \leqslant s_1 \leqslant \ldots \leqslant s_n \leqslant t_{\max} \end{aligned} \tag{13.3.1}$$

Note that the number of sutures n is fixed in this problem; to minimize the loss over all possible suture plans, the problem needs to be solved with various different values of n, which we choose heuristically; see Section 13.4.2 for details.

The objective function consists of three geometric cost terms (L^d, $L^{\text{var\_center}}$ and $L^{\text{var\_ins}_{\text{e}}\text{xt}}$) and two force model-based cost terms, both measuring the mean squared error. Its solution represents a sequence of suture midpoints which should be close to evenly spaced while also

explicitly ensuring good closure forces over the wound. We will set the weights to emphasize L^f (closure forces), with the other components there to provide numerical stability and refinement.

See sections 13.4.3, 13.4.4 and 13.4.5 for a complete mathematical description of the objective function and constraints.

13.4 Method

The SP2DEEF algorithm is divided into three distinct phases:

1. *Input*, in which the system queries the surgeon for points along the wound curve, desired suture width, and scaling information;
2. *Optimization with elliptical force model*, in which the system plans a set of sutures to minimize an objective function under constraints, utilizing an explicit model to estimate forces applied by the sutures to the wound;
3. *Adjustment*, in which the surgeon can optionally adjust the suture plan computed by the system.

13.4.1 Input

The interface first collects surgeon input, via a clicking interface, as depicted in Fig. 13.6. The surgeon selects two points on the image and inputs the measured distance between those points as well as the desired suture width.

The program scales the image based on the calibration points provided. The surgeon then clicks a sequence of points on the image tracing the shape of the wound. We then use the scipy interpolation function `scipy.interpolate.splprep` to generate a smooth B-spline that approximates a curve that goes through all points specified. We then generate an initial placement of sutures to 'warm start' the optimization. To that end, the algorithm computes the initial number of sutures by dividing the length of the wound curve by the ideal suture distance γ. It then places the corresponding number of equally spaced suture midpoints along the wound curve. The insertion and extraction points are determined by the perpendicularity constraint between the sutures and the curve as well as by the surgeon-specified suture width.

13.4.2 Optimization

SP2DEEF solves the placement of sutures on a wound as a constrained optimization problem. In order for the healing process to occur, the two sides of the wound must be brought together. Ideally, we would use a detailed model to predict skin deformation after suture placement. However, the mechanical behavior of skin is complex and influenced by the location on the

body of the wound and the patient's height, weight, and age, amongst other factors, [330] making full modeling impractical. Instead of this, surgeons usually employ heuristics to plan where to place sutures. These heuristics state that sutures should be orthogonal to the wound, or that they should be spaced evenly and as close as possible to some ideal distance apart. However, relying exclusively on such rules can lead to solutions which leave parts of the wound insufficiently closed, because of insufficient closure force from sutures in the region. Thus, it is desirable to combine surgical heuristics with a model of the forces imparted by the sutures to ensure that the entire wound is held sufficiently closed by the sutures. This 'hybrid' optimization problem leverages surgeons' experience while also ensuring that each point along the wound receives acceptable closure forces.

We assume that the wound lies on a 2D plane and place the sutures using only a 2D image of the wound as seen by an overhead camera; we assume that the wound has been positioned and rotated to minimize the distortion incurred by perspective effects. The wound is thus represented by a parametric spline computed by interpolating over points clicked by the surgeon on the image, which we denote as a function $w(t) = (x(t), y(t))$ where $w : [0, t_{\max}] \to \mathbb{R}^2$.

Since this optimization problem is in general nonconvex, we solve it with the SLSQP algorithm [344], which performs constrained optimization using linear approximation.

To find the best number n of sutures, we execute a bi-linear search from an initial naive estimation $\hat{n}$, as for a typical wound there is only a small range of 'reasonable' values of n. We use $\hat{n} = \lfloor \ell/\gamma \rfloor$, as this is the number of sutures resulting from naively spacing the sutures at distance γ along the length-ℓ wound, and is therefore likely to be close to the best number of sutures. We estimate ℓ via linear approximation and compute $\hat{n}$; then we solve the optimization problem (13.3.1) for all integers n such that $0.5 * \hat{n} \leqslant n \leqslant 1.4 * \hat{n}$ and return the suture plan with the lowest loss.

For evaluating the hyperparameter settings we use the same loss as our training loss.

13.4.3 Suture regularity constraints and objectives

The suture width α is given by the surgeon in the input phase of the process and the number of sutures n is selected by the algorithm. The other parameters $\beta_{\min}, \beta_{\max}, \gamma$, are assumed to be constant. For our experiments, we set these values based on consulting an expert in the field of surgery[1] These parameters are depicted in Fig. 13.2. We also include several key conditions for suture placement:

1. All sutures should cross the wound at their midpoint and be orthogonal to the wound at that point.

2. Sutures should have width (distance between insertion and extraction points) of α.

[1] we consulted our co-author Danyal Fer, MD, Department of Surgery, University of California San Francisco East Bay.

The hyperparameters $\alpha, \beta_{\min}, \beta_{\max}, \gamma, \epsilon$ are set via... Since in general these conditions cannot be perfectly satisfied, some conditions are hard-coded as constraints and others are encoded via penalty terms in the objective function. Conditions (i) (orthogonality of sutures to the wound) and (ii) (suture width) mean that if the suture crosses the wound at some $w(s)$, the insertion and extraction points are uniquely determined (up to swapping); therefore, as discussed above, our decision variables are the points $0 \leqslant s_1 \leqslant \ldots \leqslant s_n \leqslant t_{\max}$ along the wound at which we want the sutures to cross, with suture i crossing at $w(s_i)$. Since the wound w is represented as a B-spline, it has a well-defined derivative $w'(t) = (x'(t), y'(t))$ at each t. Then, interpreting $w(t), w'(t)$ as vectors in $\mathbb{R}^2$, the insertion and extraction points are

$$a^0(s_i), a^1(s_i) = w(s_i) \pm \frac{\alpha}{2} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{w'(s_i)}{\|w'(s_i)\|} \tag{13.4.1}$$

Here a^0 corresponds to the insertion point and a^1 corresponds to the extraction point and both are vectors in $\mathbb{R}^2$. The side that corresponds to insertion and the side which corresponds to extraction was chosen arbitrarily and can be swapped if desired.

Conditions (iii) and (iv) are coded as the constraints labeled A^{crs} and $A^{\min}, A^{\max}$ respectively. A^{crs} states that for any $i \neq j$, the line segments $(a^0(s_i), a^1(s_i))$ and $(a^0(s_j), a^1(s_j))$ cannot cross; $A^{\min}$ states that no two different insertion and/or extraction points can be within $\beta_{\min}$ of each other; and $A^{\max}$ states that for any $i \in [0, n]$ and $z(\cdot) \in \{a^0(\cdot), a^1(\cdot), w(\cdot)\}$

$$\|z(s_{i+1}) - z(s_i)\| \leqslant \beta_{\max} \tag{13.4.2}$$

(no consecutive suture insertion points, extraction points, or midpoints can be more than $\beta_{\max}$ apart).

Condition (v) states that the distance between the sutures should be as close as possible to the ideal suture distance γ while condition (vi) ensures that the sum of variances of the insertion, center and extraction points is low. To measure how well a set of sutures satisfies conditions (v) and (vi), we use mean squared error and variance. To encode the constraint concerning the endpoints of the wound, we add 'phantom' sutures at $s_0 = 0$ and $s_{n+1} = t_{\max}$, giving:

$$L^{\text{idl}}(s_1, \ldots, s_n) = \sum_{z(\cdot)} \frac{1}{n+1} \sum_{i=0}^{n} (\|z(s_{i+1}) - z(s_i)\| - \gamma)^2 \tag{13.4.3}$$

$$L^{\text{var}}(s_1, \ldots, s_n) = \sum_{z(\cdot)} \operatorname{Var}\big(\{\|z(s_{i+1}) - z(s_i)\|\}_{i=0}^{n}\big) \tag{13.4.4}$$

where $\sum_{z(\cdot)}$ indicates summing the function inside three times, with the points $z(s_i)$ being $a^0(s_i)$, $a^1(s_i)$ and $w(s_i)$, i.e. we take the average difference squared from ideal and the variance using the insertion points, the extraction points, and the midpoints and add them up. Conditions (i), (ii), (iii) and (iv) are treated as constraints while (v) and (vi) are encoded as penalty terms and in the objective function and optimized over.

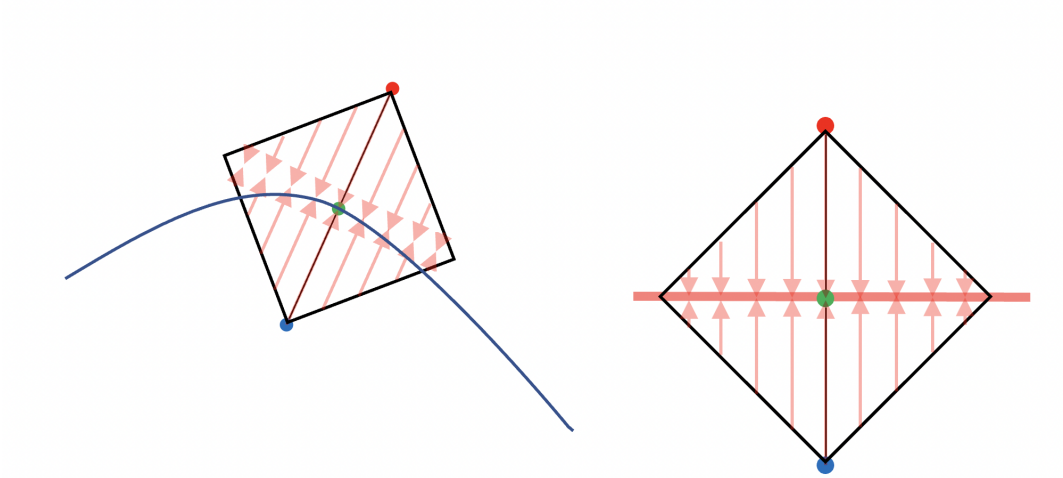


Figure 13.3: **Diamond Force Model.**. The diamond model does not generalize well to non-linear wounds. As shown to the left, the curve pulls away from the diamond line, and thus it is non-obvious how to calculate distances.

13.4.4 Generalizing the Diamond Force Model

We supplement the suture regularity constraints and objectives with a model that aims to quickly estimate the forces applied by the sutures to the wound, inspired by the diamond force model [333, 334] on a linear wound, in which the force imparted on the wound by a suture has a particular intensity at the crossing point and drops off linearly (to a minimum of 0) according to distance from the crossing point. An interesting feature of this model is that placing each suture at the point where the force from the previous suture drops to 0 yields a constant force across the wound. However, this model only considers linear wounds and does not generalize well when applied to curved wounds, as depicted in Fig. 13.3. We extend this to curved wounds by modifying it so that the forces imparted from an insertion or extraction point on the skin are parallel to the suture and decrease linearly (to a minimum of 0) based on an elliptical norm aligned with the suture. We choose an elliptical norm to represent the fact that the force of pushing on an elastic medium is transferred more strongly to points in line with the force than those to the side. We normalize the magnitude of the forces in our model by letting 1 unit of force be the ideal amount applied to close any given point on the wound, which is the amount applied by a suture to its midpoint. Since by symmetry the insertion and extraction points of a wound exert the same amount of force on the center, in opposite directions (since the suture center is the midpoint of the insertion and extraction points), this means that each insertion or extraction point exerts 0.5 units of force on its respective center. Thus, given a parameter $0 < \varepsilon \leqslant 1$ denoting the ratio of the shorter axis to the longer axis of the ellipse which defines the norm (to be discussed later), in our model

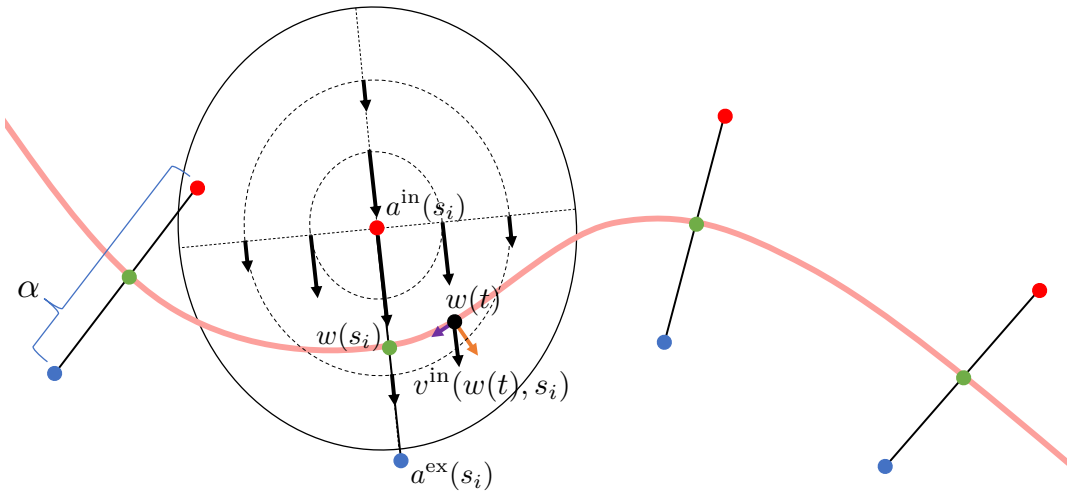


Figure 13.4: **Elliptical Force Model.** Sutures (width α) with force model around insertion point $a^0(s_i)$ depicted as an ellipse showing the region of nonzero force imparted from $a^0(s_i)$, with forces decreasing linearly from the center, with isocontours of force being ellipses. Purple and orange arrows show shear and closure forces generated at the wound point $w(t)$ by $a^0(s_i)$.

the insertion point $a^0(s_i)$ has a suture-aligned distance from a point $z \in \mathbb{R}^2$ of

$$d^0(z, s_i) = \sqrt{d^0_{\parallel}(z, s_i)^2 + (d^0_{\perp}(z, s_i)/\varepsilon)^2} \tag{13.4.5}$$

where $d^0_{\parallel}(z, s_i)$ is the distance between $a^0(s_i)$ and z on the axis parallel to $a^1(s_i) - a^0(s_i)$ and $d^0_{\perp}(z, s_i)$ is the distance between $a^0(s_i)$ and z on the axis perpendicular to $a^1(s_i) - a^0(s_i)$; we define the distance $d^1(z, s_i)$ from the extraction point $a^1(s_i)$ analogously. Then, the force exerted on z from $a^0(s_i)$ is the vector

$$v^0(z, s_i) = \frac{1}{2} \frac{\max(\eta - d^0(z, s_i), 0)}{\eta - \alpha/2} \frac{a^1(s_i) - a^0(s_i)}{\alpha} \tag{13.4.6}$$

where $\eta = \sqrt{(\alpha/2)^2 + (\gamma/\varepsilon)^2}$. The force $v^1(z, s_i)$ imparted by an extraction point $a^1(s_i)$ on z is defined analogously. The suture force model is shown in Fig. 13.4. Note that the force vector is parallel to $a^1(s_i) - a^0(s_i)$ and (since $\alpha = \|a^1(s_i) - a^0(s_i)\|$ by definition) has the following properties: (a) $\|v^0(w(s_i), s_i)\| = 1/2$ for any i (insertion points, and by symmetry extraction points, exert a force of magnitude $1/2$ on the suture center); and (b) on a linear wound with sutures placed an ideal γ distance apart, each suture center lies on the boundary of the regions in which the previous and next suture exert forces of positive magnitude. We then set $\varepsilon = 0.77$ so that ideally-spaced sutures on a linear wound exert forces which are as constant as possible along the length of the wound. These properties extend the diamond force model to non-linear wounds.

13.4.5 Closure and shear force objectives

Given the model described in Section 13.4.4 and Fig. 13.4 of how sutures exert force on skin, what does this imply about the quality of the sutures? While the insertion and extraction points are placed so that the suture (and thus the forces it exerts) are orthogonal to the wound at the crossing point, if the wound is not linear then at other points on the wound the exerted force may have both a *closure force* component orthogonal to the wound and a *shear force* component parallel to the wound. The total force exerted by insertion points (which push on one side of the wound) and the extraction points (which push on the other side of the wound) on $w(t)$, are denoted respectively as

$$F^0(t) = \sum_{i=1}^{n} v^0(w(t), s_i) \text{ and } F^1(t) = \sum_{i=1}^{n} v^1(w(t), s_i) \tag{13.4.7}$$

The total closure force at point $w(t)$ on the wound is:

$$f^{\mathrm{c}}(t) = (F^0(t) - F^1(t))^\top \frac{a^1(s_i) - a^0(s_i)}{\alpha} \tag{13.4.8}$$

(note the $1/\alpha$ term to normalize the suture width). This value is normalized so that the ideal value is 1 at every $w(t)$.

To get the total shear force at a given point t on the wound, we take the components of the forces parallel to the wound:

$$f^{\mathrm{s}}(t) = (F^0(t) - F^1(t))^\top \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{a^1(s_i) - a^0(s_i)}{\alpha} \tag{13.4.9}$$

Since shear forces do not hold the wound together but may cause misalignment, the 'ideal' magnitude of the shear force is 0 at every t.

13.4.6 Force closure objective

We take m points $w(t_1), \ldots, w(t_m)$ and use the model above to estimate the closure and shear forces at these points. The average closure force penalty is then

$$L_{\mathrm{c}} = \frac{1}{m} \sum_{j=1}^{m} (1 - f^{\mathrm{c}}(t_j))^2 \tag{13.4.10}$$

(penalizing deviation from the ideal value of 1) and the average shear force penalty is

$$L_{\mathrm{s}} = \frac{1}{m} \sum_{j=1}^{m} f^{\mathrm{s}}(t_j)^2 \tag{13.4.11}$$

(penalizing deviation from the ideal value of 0). As with the regularity objective function, these components of the objective function are given weights c_{c} and c_{s}, respectively.

Table 13.1: Suture Placement Results

Loss Type	Algorithm	Spline 1	Spline 2	Spline 3	Spline 4	Spline 5
Variance -	Baseline	0.00	**0.02**	2.06	2.79	**0.77**
center points	Optimized Placement	0.00	0.62	**1.51**	**1.24**	24.92
Variance - insertion /	Baseline	0.00	**0.05**	**28.12**	18.50	**47.66**
extraction points	Optimized Placement	0.00	1.25	28.40	**13.65**	87.12
Ideal distance	Baseline	8.48	**5.34**	106.50	86.68	**185.58**
loss	Optimized Placement	**4.50**	9.01	**64.88**	**25.95**	527.79
Closure Force	Baseline	3431.70	9410.64	9828.90	7776.41	8283.97
loss	Optimized Placement	**0.61**	**5.68**	**0.01**	**0.00**	**9.54**
Shear Force	Baseline	0.00	54.86	**29.00**	104.66	**17.52**
loss	Optimized Placement	0.00	**0.55**	39.13	**32.00**	113.43
Total	Baseline	51483.98	141439.78	147878.44	117400.61	124827.93
loss (weighted)	Optimized Placement	**13.65**	**111.9**	**449.2**	**282.73**	**2059.8**
Number of	Baseline	6	13	40	15	51
Sutures	Optimized Placement	7	14	39	15	53

13.4.7 Parameter settings

We take α (suture width) as input from the user while we chose the values of $\beta_{\min}$, $\beta_{\max}$ (min and max distances between sutures) and γ (ideal distance between sutures) by consulting an expert in the field of surgery.

We developed the constraints using surgical suturing guidelines, setting $\beta_{\min} = 2.5$mm between two sutures as consulted by an expert surgeon and $\beta_{\max} = 10$mm between consecutive sutures so as to not leave parts of the wound unclosed.

13.4.8 Adjustment

Finally, we provide a GUI for the surgeon to interact with. They are presented with the suture points, superimposed over the wound. The surgeon is able to drag the points to the desired place.

13.5 Experiments

13.5.1 Synthetic Splines

Our loss function contains weights on all 5 loss terms, which we manually tuned to the values: $c^d = 1$; $c^{var_c enter} = 12$; $c^{var_i ns_e xt} = 6$; $c^f = 15$; $c^s = 5$.

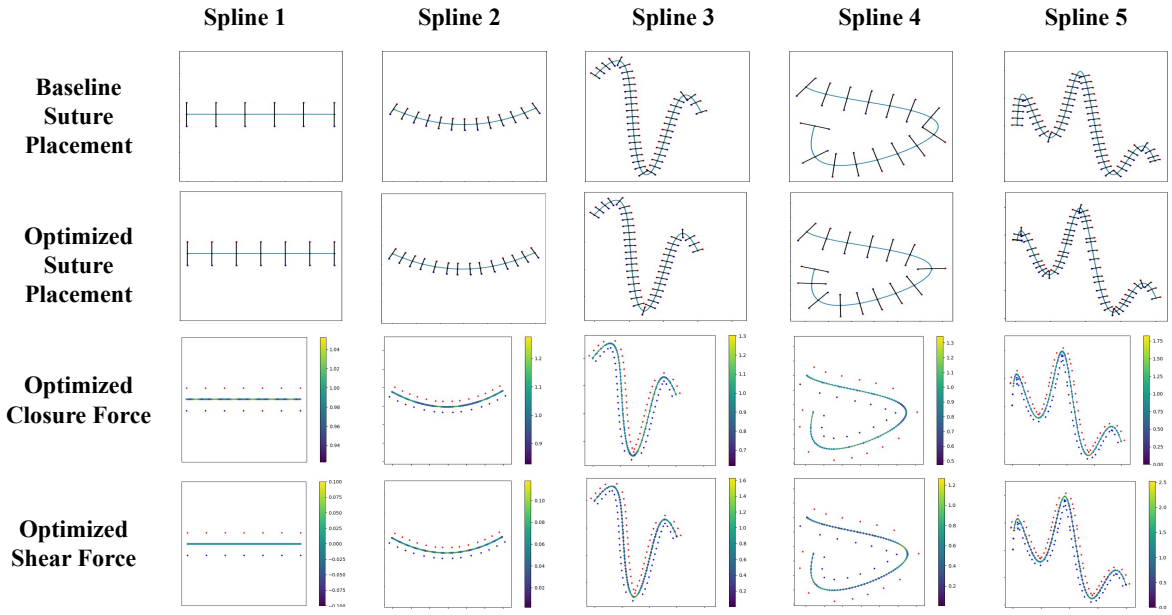


Figure 13.5: Results analyzing the suture placements for 5 synthetic splines. The top two rows present the difference in the sutures placed using the baseline, as compared to the optimization algorithm. The bottom rows present the shear forces and closure forces created by the optimization suture placement.

Baseline Algorithm: We evaluate a baseline algorithm, which evenly places sutures along the curve spaced at the ideal distance, on the splines. This can be implemented by starting at one end of the wound, travelling the ideal distance along the wound, placing sutures at increments of the ideal distance until the end of the wound.

We tested both SP2DEEF and the baseline on 5 splines generated manually. To evaluate the robustness of the algorithms we created simple splines and splines with a higher degree and curvature. The suture placement points generated by the algorithms are depicted in Fig. 13.5. Our results show that the suture placements generated by SP2DEEF are relatively consistent, robust and well spaced, especially for the simpler splines. SP2DEEF also deals well with sharp curves (depicted in Spline 3) as it places a higher number of sutures close to the point of curvature.

SP2DEEF tended to choose a different number of sutures, when compared with the baseline. An optimal placement over an optimal number of sutures results in a lower closure force, while other terms tend to remain the same: the closure force that each suture exerts on the wound decreases sharply as the curve turns away from being perpendicular to the wound. To maintain sufficient closure force, it is necessary to increase the number of sutures. The baseline algorithm was not able to provide consistent closure force, as it is able to vary

neither the number of sutures nor the placement to account for curvature.

While it reported high closure force losses, the baseline did well according to the variance and ideal distance metric, since it was initialized to be evenly spaced. The variances for the baseline aren't always 0, because we used a greedy algorithm which approximately placed sutures evenly. Furthermore, the insert and extract points may have additional variance on curved segments of the wound even if the centers are approximately evenly spaced.

One major improvement SP2DEEF saw was that it always avoided sutures crossing over and heavily penalized cases in which the sutures were placed too close to each other, in contrast to the baseline algorithm, which did both of these.

Overall, SP2DEEF outperforms the baseline; we believe that closure force is the most important metric as the main purpose of sutures is to provide closure forces to close a wound. Our algorithm finds a placement almost as well spaced as the baseline, with similar shear forces, while making sure that all the normalized closure forces are very close to the ideal. Although it does not outperform the baseline in the variance (and the shear sometimes), it makes up for this by returning a placement that optimizes the closure force.

13.5.2 Physical Experiments on Chicken Skin

We tested SP2DEEF in a physical experiment using chicken (thigh) skin. To find the most realistic wounds possible, we had a surgeon cut two wounds into the tissue with a scalpel. These two wounds, as can be seen in Fig. 13.1, were chosen by the surgeon for their relevance to clinical wounds seen in practice, as well as their difficulty. As a baseline, we had a surgeon suture the wound as they saw fit. We then fed an image of the wound without sutures into our pipeline, clicked points to trace the wound shape, and recorded the optimized placement of the sutures, the output of our optimizer. We manually marked this placement on the phantom with ink, and had the surgeon implement the autonomous placement by suturing on top of the ink markings with USP Size 2-0 thread and a suture needle type GS-22 as depicted in Fig. 13.1.

Our placement was able to achieve wound closure on the phantom: we consulted an expert surgeon who evaluated the sutures as equal or better than a human surgeon. In the test case, having SP2DEEF's placement marked on the skin helped the surgeon place sutures with consistent spacing throughout the whole wound. In contrast, in the baseline case, with no visual guides, the Surgeon stitches developed uneven spacing between suture locations as the wound was deformed during the suturing process.

13.6 Discussion and Future Work

13.6.1 Limitations

This chapter has the following limitations:

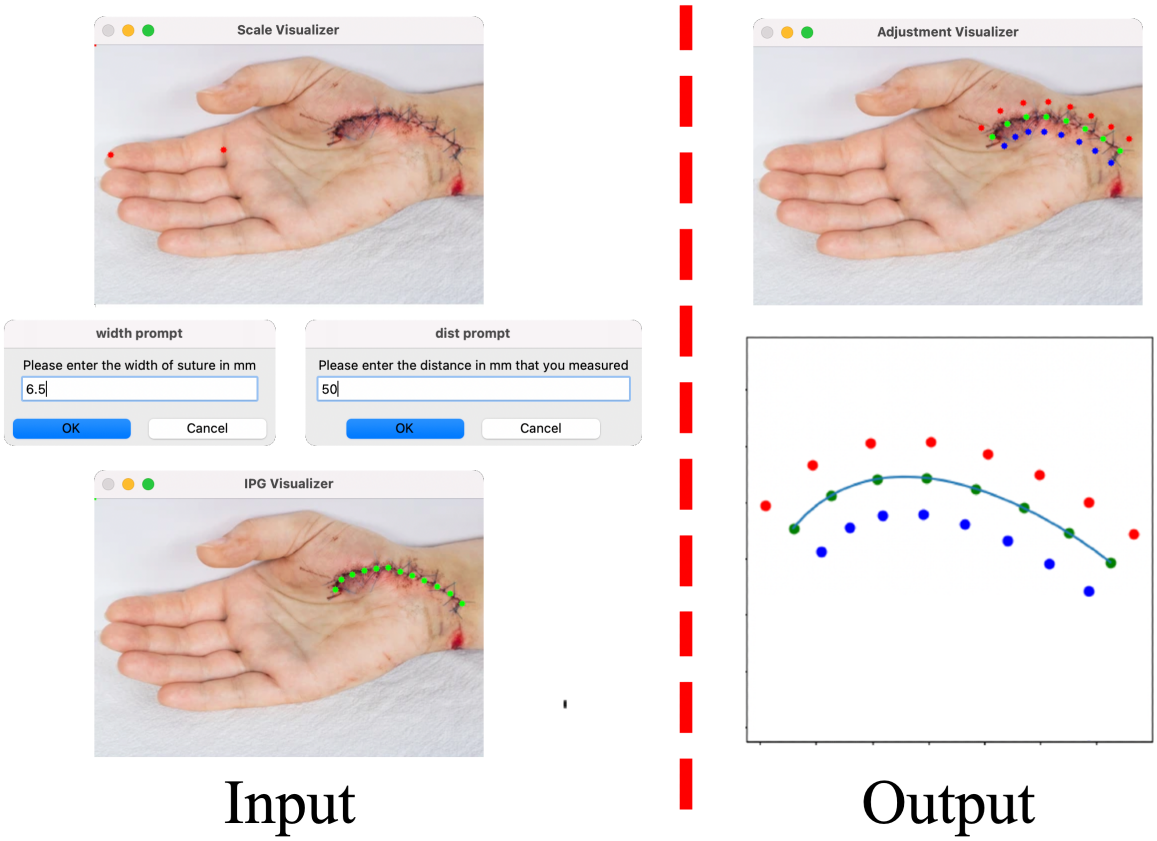


Figure 13.6: **The full autonomous pipeline for optimizing surgical suture placement.**. Input: Surgeon selects two points on the image (finger length) and inputs the measured distance between those points as well as the desired suture width. The surgeon is further asked to click points along the wound. Output: The wound fitted as a Bézier curve (green) with the result of the suture optimization (red and blue points). The suture plan is overlaid on the original wound image.

- The wound is represented as a spline, and therefore the representation cannot have sharp corners (or branches).
- No information about the wound depth or width is incorporated.
- SP2DEEF assumes a fixed suture width as opposed to being able to modify it for particular sutures. Some better solutions may therefore be missed, particularly when tight wound curves are involved.
- The explicit model of suture forces is only a rough approximation of how skin behaves under tension.
- No objective, external measure of suture placement quality was used to evaluate the algorithm.
- The sutures may be re-planned during the operation due to the wound changing shape when partially sutured.

13.6.2 Future work

Prior work on the suture planning problem focused primarily on achieving evenly placed sutures on known wound shapes under kinematic or workspace constraints. This suggests the possibility of combining such constraints with the wound-focused constraints and objective developed in this work, for instance by modifying the objective function to include a consistency measure and also account for the difficulty of executing the planned sutures.

Chapter 14

Autonomous Suture Tail-Shortening

In the previous chapter, we proposed a method for reliable suture planning. In this chapter, we tackle the problem of autonomous robot suture tail-shortening. Similar to SGTM 2.0 for long cable untangling in chapter 9, where perception is limited and the dynamics are hard to model, we use interactive perception to increase the reliability of the system.

14.1 Introduction

Many steps in suturing, such as tail-shortening (where a thread is pulled through a suture to a desired length) and knot tying, require accurate thread tracking, which is particularly challenging due to the thin and flexible nature of suturing thread, as well as its propensity for self-intersections and partial occlusions.

In this chapter, we propose a novel interactive perception system for tracking suture thread in 3D, which we apply to track thread in the autonomous tail-shortening task described in Figure 14.1, requiring precise tracking to avoid pulling the thread too far out of the suture.

The learned 2D suturing thread detection model is trained using the Labels from UltraViolet (LUV) method [345] for self-supervised data collection, which has previously been shown to be effective for detecting cables and surgical needles. We extend it to surgical threads and combine the detection network with a 3D tracking method for temporal stability. Modeling the thread in 3D is a non-trivial task due to its complex shape and unclear endpoints. To address this issue, we model the thread as a 3D Non-Uniform Rational B-Spline (NURBS) [346] based on stereo images of the scene. We adapt the thread model across frames by optimizing spline control points to minimize the error between the current detections and the reprojection of the 3D spline into the images.

Using NURBS optimization alone can break down in complex thread configurations because of false-positive detections or self-intersections. To address this, we develop an analytic 2D tracing approach based on prior work for cable untangling [347], which is used as a prior

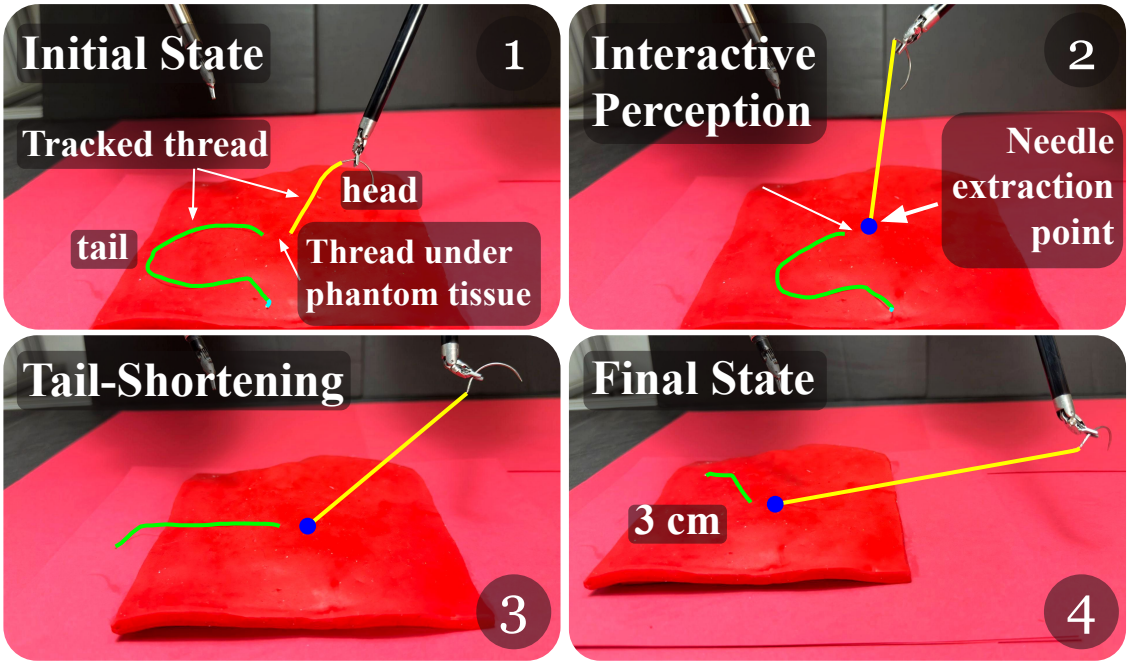


Figure 14.1: **Surgical Suture Tail-shortening with 3D Thread Tracking.** We present a thread tracking method which we use for automating surgical suture "tail-shortening", i.e. pulling thread through tissue until a desired length of thread remains exposed. Initially, the robot grasps the needle close to the wound. It uses interactive perception to determine which portions of the reconstructed thread are on the needle side (head) vs slack side (tail). The system accomplishes tail-shortening by visually servoing the needle driver until a desired tail length remains.

to prevent the NURBS optimization from collapsing in the presence of distracting false detections or challenging thread configurations.

Experiments conducted on a physical Intuitive Surgical da Vinci Research Kit (dVRK) RSA demonstrate that the system utilizing our thread tracker achieves 18/20 successful trials on the tail-shortening task (shown in Figure 14.1).

This chapter makes three contributions: (1) A 3D surgical thread tracking algorithm, described in Figure 14.2, that combines a learned thread detection module trained on data collected in a self-supervised fashion with a NURBS spline optimization; (2) An interactive perception approach to suture thread tail-shortening which utilizes the thread tracker with visibility-maximizing manipulation to estimate the length of remaining thread tail; and (3) Data from experiments evaluating the perception components individually and applied to suture tail-shortening, achieving a 90% success rate.

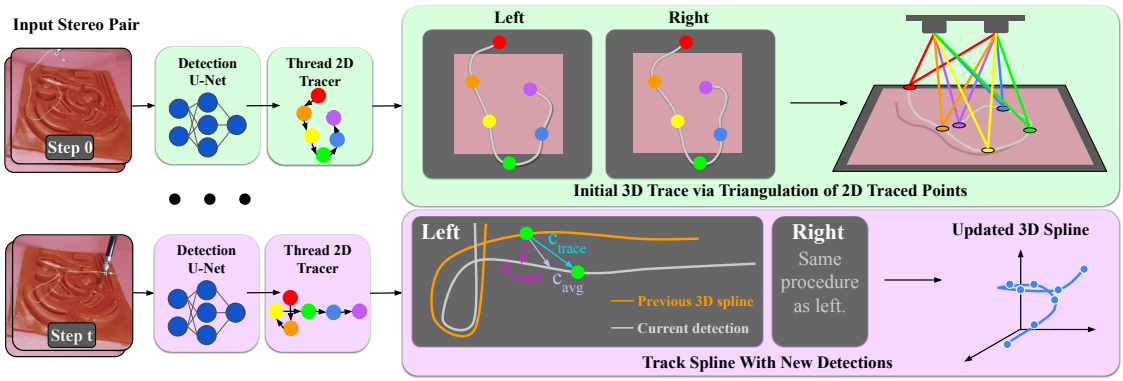


Figure 14.2: **Overview of the first 4 modules: 2D Surgical Thread Detection, 2D Tracing, 3D Tracing, and 3D Tracking**. *Left:* For every stereo pair of images, we predict thread segmentation masks, then run a 2D tracer to compute the sequence of pixels along the thread. *Top right:* To initialize the 3D spline of the thread, we match points meeting both stereo image and tracer topology constraints and triangulate their positions in 3D. We initialize the 3D trace by fitting a 3D spline to these points. *Bottom right:* To update the 3D spline with new frames, we compute correction vectors in 2D as an average of vectors which push the projected 3D spline onto the new detection and push each projected 3D point to its corresponding point on the 2D trace. We then triangulate the correction vectors across both images and apply them to the 3D spline to perform an update.

14.2 Related Work

Detecting surgical thread from an RGB image has been previously explored in a number of different settings. Early approaches relying on analytic curvilinear detectors [346, 348] work well when the thread is isolated and clearly visible, but fail in realistic scenes with shadows and occlusions. Similarly, Joglekar et al. [349] assume that thread detections can be obtained from color segmentation; however, this may fail due to light glare, sensor noise, materials covering the thread (e.g., blood), and varying lighting conditions. Learning-based approaches generalize better to different backgrounds and lighting conditions, but require manual collection of large datasets. Lu et al. [350] train a U-Net [268] using semi-supervised learning leveraging hand-labeled images for supervision which are time consuming to obtain. We use a self-supervised data collection method that extracts labels autonomously using UV light [345], allowing the system to collect 10 labeled images per minute.

Lu et al. [350] propose using a 3D graph to represent the triangulated 3D candidate thread points. The method then computes a minimum energy path through the graph and uses it as the 3D model of the thread. Joglekar et al. [349] propose using a minimum variation spline to represent the suture. This results in a smooth reconstruction with less tight curvature and yields a confidence value along the spline model which is useful to chose a grasp point along the thread. Both methods mentioned above fully reconstruct the model on each frame, ignoring prior frames, making them more susceptible to one-off missing or

false detections. Padoy et al. [348] assume the 3D spline has been initialized in advance, and focus on tracking the spline across frames. However, this work assumes that the length of the thread is constant, which limits its applicability to certain applications like tail shortening or knot tying. Jackson et al. [346] propose an approach to jointly trace and reconstruct a 3D spline from stereo images as well as a tracking method using pixel-space error minimization. However, their approach assumes a known initial tracing point, manually defined using a space mouse. In contrast, we leverage tracing of 2D splines to address missing or occluded parts of the thread and use an approach which does not rely on a user-defined seed point. Furthermore, our method tracks the spline across frames, increasing its robustness to noisy detections.

Goldberg et al. [243] investigate how a robot can use active perception to recognize the shape of an object by moving a touch sensor to trace its contours. Bajcsy [118] defines *active perception* as the search for models and control strategies for perception which can vary depending on the sensor and the task goal, such as adjusting camera parameters [119] or moving a tactile sensor in response to haptic input [243].

Similarly, *interactive perception*, as explored by Bohg et al. [220], utilizes robot interactions to enhance perception. Interactive perception has been used in robotic manipulation to extract kinematic and dynamic models from physical interactions with the environment [351] and to improve the understanding of a scene in the presence of occlusions and perception uncertainty [220, 244, 246]. Murali et al. [352] leverage feedback from visual and tactile sensors to estimate the pose of partially occluded objects in cluttered environments. Danielczuk et al. [246] propose the mechanical search problem, where a robot retrieves an occluded target object from a cluttered bin through a series of targeted parallel jaw grasps, suction grasps, and pushes. Novkovic et al. [244] use a robot to move a camera and interact with the environment in order to find a hidden target cube in a pile of cubes, while Shivakumar et al. [347] use interactive perception to reduce perception uncertainty when untangling long cables.

In this chapter we propose an interactive perception-based approach to surgical suture tail-shortening. The robot tensions the thread to create a sharp angle between the taut thread on the extraction side of the suture and the slack thread on the insertion side. This forces the thread into a linear configuration to facilitate perception. The robot then pulls the thread through the suture until the desired length of slack thread is detected at the tail.

14.3 Problem Statement

Using stereo RGB images, we want to accurately track the state of a surgical thread and use these state estimates to automate the task of surgical tail-shortening.

We define the workspace using a cartesian (x, y, z) coordinate system. The workspace consists of a bimanual dVRK robot [353]; a Simulab TSP-10 human organ phantom; and a fixed ZEDm RGB stereo camera, which outputs images at 1280x720 pixel resolution. The camera is angled at the robot and phantom such that the whole reachable workspace of the

robot is captured in the field of view. We work with undyed (beige) Polysorb^{TM} surgical suture thread from Covidien. The sutures are of variable length between 10 and 40 cm, with 2-0 USP Size (0.35-0.399 mm in diameter) and are attached to a GS-21 needle or similar. The length, diameter and needle size of the suture are unknown to the algorithm.

We make the following assumptions:

1. The robot-to-camera transform is known.
2. During test time, the thread and phantom configurations lie within the training data distribution. However, their pose does not necessarily correspond exactly to any pose seen in training.

14.4 Method

We decompose the problem of thread modeling and autonomous robot suture tail-shortening into five modules:

1. *Learned 2D Surgical Thread Detection*: uses a convolutional neural network to segment the surgical thread in a physical mockup of a surgical environment. This module takes as input an RGB image of dimension 1280 x 720 and returns a pixel-wise probability mask of the thread's location.
2. *2D Surgical Thread Tracing*: given the detection probability masks with potential gaps in the detections and false positives, identifies the sequence of image points along the thread.
3. *3D Surgical Thread Tracing*: computes a 3D representation of the suture thread based on the traced thread. This algorithm takes the traces from 2 rectified stereo images as input and returns a 3D NURBS spline.
4. *3D Surgical Thread Tracking*: adapts the 3D spline model to the current view of the scene. This module takes the traces from the current pair of rectified stereo images as well as the previous 3D spline as input and outputs an updated 3D spline.
5. *Surgical Suture Tail-Shortening*: This module performs the surgical tail-shortening task using an interactive perception approach which leverages the 3D spline model computed by the previous modules.

An overview of how these modules are combined is shown in Figure 14.2.

14.4.1 Module 1: Learned 2D Surgical Thread Detection

We train a neural network to segment surgical thread from scenes using the self-supervised training approach proposed in Thananjeyan et al. [345]. To collect labels automatically, we

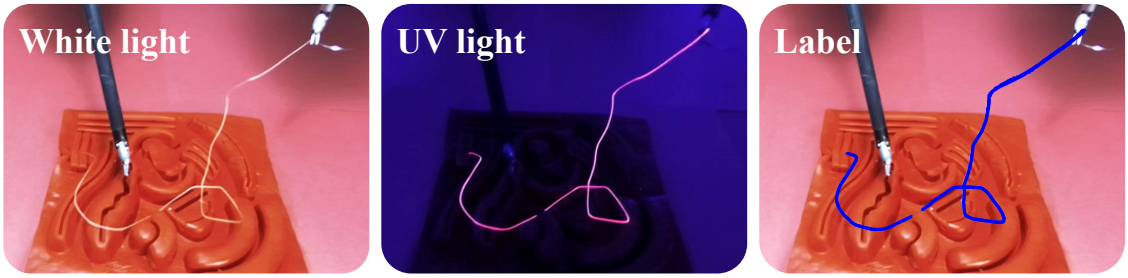


Figure 14.3: **2D Surgical Thread Detection Data Collection.** The left image shows the dVRK gripper holding a suture thread under white light. The middle image depicts the same scene under UV light. The thread painted with UV fluorescent color lights up and can be segmented via color thresholding. The right image displays the extracted label used for training.

paint the surgical thread with a UV-florescent paint. This paint is invisible under visible light but shines when illuminated with UV light. For each scene, the robot arms are moved to a new random position in the workspace, changing the thread configuration and RGB stereo images are recorded under both visible and UV light.

The label masks are extracted from the UV images using color segmentation. Train, validation, and test sets are split from disjoint subsets of scenes to ensure no cross-contamination of the sets from the same state. Using this self-supervised data collection technique, we are able to acquire 10 labeled images per minute (visible light stereo images with corresponding labels extracted under UV light). The image size of 1280 x 720 pixels is chosen to maximize the tradeoff between resolution (which aids in segmenting the thin thread) and inference speed (2.5 FPS on our test computer using an NVIDIA 2080 GPU).

We train a U-Net [268] to detect surgical thread from a single RGB image. We train our model on 1320 images of size 1280 x 720 for 400 epochs. We specifically choose not to upweight false negatives, as would be expected from the ratio of background pixels to thread pixels, as this yields predictions that are biased towards precision over recall. This is desirable because the 2D tracer is able to bridge missing thread detections but can get confused by false positives.

14.4.2 Module 2: 2D Surgical Thread Tracing

We adapt the analytic cable tracing method from Shivakumar et al. [347] to trace the path segments from the 2D thread detection masks. However, instead of generating all possible global paths, this work leverages heuristic scoring rules similar to those proposed by Viswanath et al. [354] and Keipour et al. [355] to generate a single global trace. In contrast to the learning-based method proposed in [354], which detects and traces cables simultaneously, we propose an analytical method. The method proposed in [355] is similar in the sense that it uses scoring functions that prioritize traces which cover more of the cable and have lesser changes in angle. However, they model the thread as a chain of cylinders whereas we

fit a 2D spline onto the traced detections to bridge gaps. The analytic thread tracer locally traces contiguous segments and greedily stitches them together, as described in Algorithm 5.

Algorithm 5 2D Surgical Thread Tracing Algorithm

Require: $D \leftarrow$ pixelwise thread detection
  mask $\leftarrow D >$ thresh$_{\text{d}}$
  mask $\leftarrow$ mask $-$ (conn components with area $<$ thresh$_{\text{a}}$)
  path_segs $\leftarrow$ []
  while sum(mask) $>$ thresh$_{\text{s}}$ **do**
    start_point $\leftarrow$ argmax(D)
    paths $\leftarrow$ sgtm2tracer(mask, start_point).
    best_path $\leftarrow \text{argmax}_{\text{p}\in\text{paths}}$ score(path)
    On mask, set points along best_path to 0.
    Append best_path to path_segs.
  while length of path_segs > 1 **do**
    find i, j within path_segs with lowest matching cost
    new_seg $\leftarrow$ merge of path_segs[i] and path_segs[j]
    add new_seg to path_segs
  return path_segs[0]

14.4.3 Module 3: 3D Surgical Thread Tracing

As in prior work [346], we model the suturing thread as a 3D NURBS parametric curve. Instead of jointly tracing and reconstructing the thread, we use a dedicated 2D tracer to compute the sequence of thread pixels in both images before reconstructing the 3D thread model. The spline parameter $t \in [0, 1]$ describes the normalized distance along the spline.

To start the 3D tracing method, a 2D NURBS spline defined by 32 control points is fitted to the traces in both images using a least squares approximation. The number of control points is chosen to allow a sufficient amount of flexibility to the spline so that it can approximate tight curves common in suturing thread.

Next, we triangulate these 2D splines into 3D to estimate the thread state. We therefore propose the following stereo-matching approach: The left trace spline point p_i^L is located at spline parameter t_i^L along the spline and has pixel coordinates $[u_i^L, v_i^L]$ for width and height respectively, starting from the top left corner. For each point along the left spline p_i^L, a corresponding point on the right spline $p_{j(i)}^R$ is found which minimizes the difference between spline parameters t_i^L and $t_{j(i)}^R$ and satisfies rectified stereo image properties. Specifically, the right image point should have the same vertical coordinate than the left image point except for a tolerance of up to $\alpha = 5$ pixels (condition a). $p_{j(i)}^R$ must be further left within the image than p_i^L (condition b). The right spline candidates must be further along the spline than the

last matched right spline point (condition c). t_i^L and $t_{j(i)}^R$ must be within a distance $\beta = 0.05$ (condition d). For a given value of i, we seek to solve

$$j(i) = \underset{j}{\operatorname{argmin}} |t_j^R - t_i^L|$$

such that a) $|v_i^L - v_j^R| \leqslant \alpha$, b) $u_j^R \leqslant u_i^L$, c) $t_j^R > t_{j(i-1)}^R \forall i$, d) $|t_j^R - t_i^L| \leqslant \beta$.

The matched points are then triangulated using the camera intrinsics to obtain their 3D position. A 3D NURBS spline model is then fitted to the triangulated points using least-squares optimization. The values for α, β and a rejection threshold for bad reconstructions were set empirically as a trade-off between reconstruction quality and number of discarded frames.

14.4.4 Module 4: 3D Surgical Thread Tracking

Inspired by Jackson et al. [346], we compute 200 correction vectors to update the coordinates of the 3D spline control points between frames. The number of correction vectors was set as a trade-off between tracking accuracy and computation speed. The thread tracking computation time is under 2.5 FPS which is the frame rate of the 2D learned surgical thread detection module as described in Section 14.4.1. Instead of an energy minimization approach to compute correction vectors, we compute correction vectors using the 2D splines fitted on the current stereo traces. The 2D correction vectors are obtained as a sum of two vectors, c_{mask} and c_{trace}. c_{mask} is a vector in image space pointing towards the closest point on the prediction mask. c_{trace} matches the point of the 2D spline fitted on the 2D trace at parameter t with the point at parameter t of the projected 3D spline. The 2D correction vectors from both stereo images are triangulated to find 3D correction vectors. Both 3D correction vector terms are averaged to obtain the final set of correction vectors.

Using only the distance correction c_{mask}, the 3D spline tends to collapse as the segmentation mask of the thread does not constrain the 3D spline along the length of the thread. This is mitigated by the second correction vector, c_{trace}, which assigns a fully constrained pixel location to each point along the projected 3D spline.

Given the correction vectors, an updated set of control points is computed using the least square control point update described by Jackson et al. [346].

14.4.5 Module 5: Surgical Suture Tail-Shortening

Initially, the thread passes through the phantom at one suture, with the needle held by one dVRK gripper. Excess thread of between 12-16 cm exists on the needle insertion side of the suture and needs to be pulled through while an unknown amount of thread has already been pulled through the suture. First, the robot uses interactive perception to estimate the needle extraction point by pulling the needle side of the thread taut. This is achieved by moving the needle upwards in positive z direction. The algorithm detects the needle end of the spline to be the one that has the highest z-coordinate. The system detects the taut segment of string by computing the tangent along the 3D spline and identifying a constant tangent

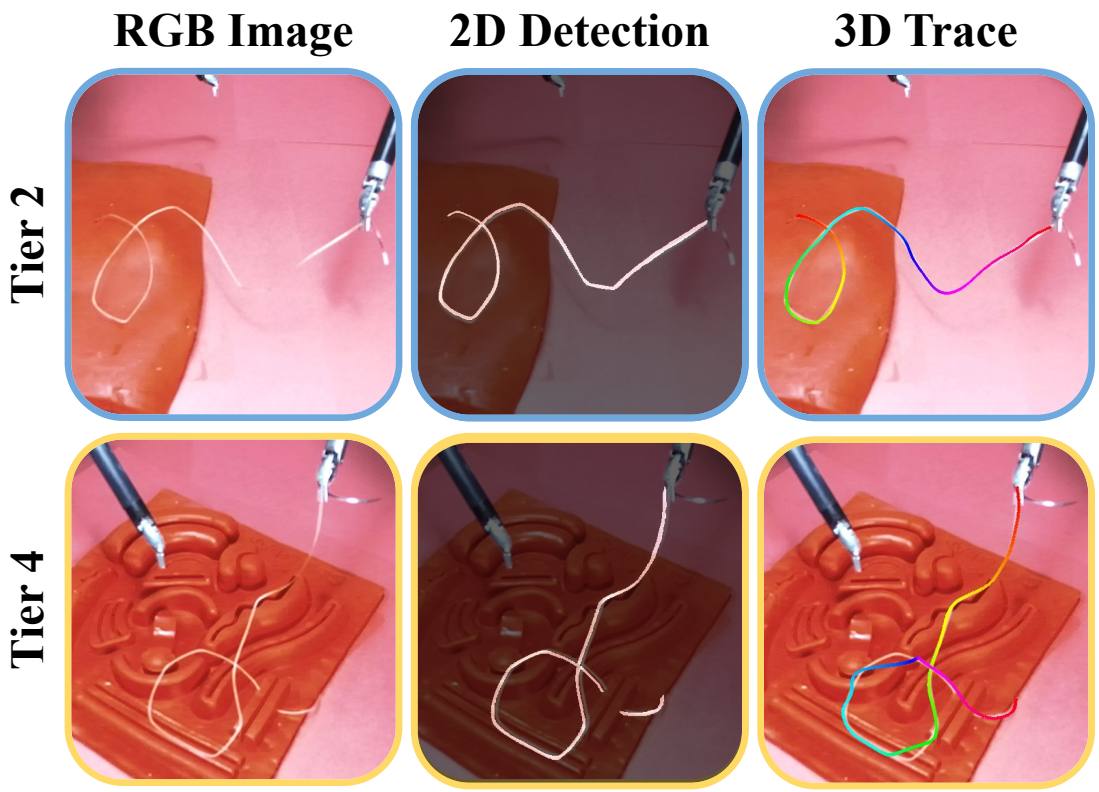


Figure 14.4: **Example 2D Thread Detections and 3D Traces**. 2 example executions of the 3D thread tracing method. *Left* shows the left camera's input RGB image, *Middle* shows the 2D thread detection prediction from the neural network, and *Right* shows the resulting reconstructed 3D spline reprojected into the camera image. The color indicates the path from red to orange.

direction segment. The angle between the taut segment tangent and the following thread tangent is computed. Pulling the thread upwards leads to a sharp angle in the thread which is identified as the needle extraction point. The 3D thread spline is split into a slack-side and a needle-side at the extraction point and the length of the slack-side thread is computed.

The robot conducts the actual tail-shortening by performing a horizontal motion away from the computed wound location (i.e. the extraction point). We continuously run the 3D thread tracking module during this motion, terminating when the thread tail is less than 3cm.

Table 14.1: 2D Surgical Thread Detection Results

	Tier 1	Tier 2	Tier 3	Tier 4	Overall
Recall (%)					
Color thresholding	32	32	14	40	30
Learned segment. (ours)	**85**	**79**	**80**	**86**	**83**
Precision (%)					
Color thresholding	24	27	5	10	17
Learned segment. (ours)	**93**	**93**	**87**	**90**	**91**
IoU (%)					
Color thresholding	16	17	4	9	12
Learned segment. (ours)	**80**	**75**	**72**	**79**	**77**

Table 14.2: 3D Surgical Thread Tracing Results

	Tier 1	Tier 2	Tier 3	Tier 4	Overall
Mean Reproj. Err. (pix)	0.58	1.14	2.50	1.10	1.33
Max Reproj. Err. (pix)	13.34	20.61	62.16	43.17	62.16

14.5 Experiments

14.5.1 Modules 1-3: 2D Thread Detection and 2D & 3D Tracing

Setup

We test the first 3 modules by using the workspace described in Sec 14.3 The experiments begin with the needle in the right end-effector and the tip of the thread going through the phantom. We collect test examples from 4 difficulty tiers:
Tier 1: No self-intersection in thread, reversed phantom. **Tier 2**: $\geqslant 1$ self-intersection in thread, reversed phantom. **Tier 3**: No self-intersection in thread, phantom facing up. **Tier 4**: $\geqslant 1$ self-intersection in thread, phantom facing up. We collect and label stereo images for 5 scenes per tier for a total of 10 images per tier.

Metrics

In this experiment, we evaluate the IoU, precision, and recall metrics of the segmentation mask with respect to human-labeled ground truth segmentation masks. To evaluate the 3D model of the thread, we report the reprojection error between the human-labeled ground truth thread segmentations and the projection of the 3D model of the thread into both stereo images.

Results and Failure Modes

Results for 2D thread detection and 3D thread tracing are presented in Table 14.1 and Table 14.2 respectively. Example detection masks and reconstructions are shown in Figure 14.4. Comparison with the color thresholding baseline clearly shows that the learned method is able to detect thread in low contrast scenes and in the presence of light reflections on the phantom. Note that while detections can miss segments of highly difficult thread (recall of 83%), the precision of predictions is 90%. Our detection model shows slightly better recall and IoU performance in the more difficult Tier 4 with respect to Tier 2. This difference is due to a scene in Tier 2 in which the thread has a particularly low contrast with the background and is thus not detected. The discrepancy lies in the error bars of this experiment. Lu et al. [350] report an average IoU of 85% with a recall of 93%. While these are impressive results, all their scenes have suturing threads lying on the ground plane without any tools that can cast shadows or reflective configurations that make segmentation more challenging.

The 3D surgical thread traces have an overall mean error of 1.33 pixels, showing that the reconstruction approximates the spline well in general. The 3D tracing fails on parts of the thread in 3 scenes, resulting in the high maximum reprojection error. These errors are mainly due to reflective bright edges on the phantom which lead to erroneous detections and 3D traces. Joglekar et al. [349], report reprojection errors between mean 0.4 and 1.1 pixels on 10 real scenes with printed surgical backgrounds. These results seem comparable to ours even though performance remains highly dependant on the particular scenes, making results hard to compare objectively.

14.5.2 Module 4: 3D Surgical Thread Tracking

Setup

Using the workspace setup described in Section 14.3, we thread the needle through the phantom and place it in the right gripper of the dVRK. We evaluate the thread tracking system on two trajectories: the “no loop” trajectory, a line in the image plane, and the “one loop”, an elliptical track above the phantom. The first trajectory avoids any occlusion or self-intersection of the thread, while the second incurs a challenging, self-crossing configuration.

Table 14.3: 3D Surgical Thread Tracking Results

	Mean Reproj. Err.	Max Reproj. Err.
No loop		
No tracking	**0.30 pix**	9.22 pix
With tracking	0.39 pix	**5.39 pix**
One loop		
No tracking	5.37 pix	94.92 pix
With tracking	**1.28 pix**	**16.26 pix**

No tracking refers to computing a new 3D trace for every frame.
With tracking refers to using Module 4 and leads to a significantly better mean reprojection error in the more difficult "One loop" case.

Metrics

We report the same metrics as in the single-frame 3D tracing (Section 14.5.1) experiments. We manually label ground-truth segmentation masks in stereo images taken every 1 cm along the trajectory and evaluate the 3D thread model against them. This leads to 10 evaluation frames for the "No loop" trajectory and 9 for the "One loop."

Results and Failure Modes

The results in Table 14.3 suggest that the method is able to track the thread reliably in both configurations. The "One loop" trajectory sees higher reprojection errors, as it presents a more challenging thread configuration for the tracer. The full tracking pipeline achieves a mean reprojection error of 0.39 pixels on the intersection-free trajectory and a 1.28 pixels on the loop-forming trajectory. Padoy et al. [348], report a mean reprojection error of 1.21 pixels. However, they use only short threads in configurations which present no self-intersections.

14.5.3 Module 5: Surgical Suture Tail-Shortening

Setup

We additionally test automated suture tail shortening using the workspace setup described in Section 14.3, with the needle threaded fully through the reversed phantom and held by the right end-effector. The tail of the thread is then placed arbitrarily in the workspace such that the entire suture thread is within the view of both the right and left stereo cameras.

Metrics

We define a successful tail-shortening maneuver as a termination in which the final tail length lies within 1 cm of the desired value of 3 cm. We report the success rate of our pipeline on the tail-shortening task, as well as the mean absolute error between the achieved and desired length and the average time to completion for this task.

Results and Failure Modes

The proposed method achieves 18 successes out of 20 trials with a mean absolute tail error of 0.53cm. The mean time to completion is 106.8 seconds. The method achieves a success rate of 90%, indicating that our pipeline provides high-confidence 3D traces using interactive perception. The main failure case occurs during the 3D tracing of the spline due to false detections on the human organ phantom.

14.6 Discussion and Future Work

The primary limitation is the uncertainty about the ability of the learned 2D surgical thread detection to generalize to new thread or phantoms, which will be addressed in future work. Also, the learned thread segmentation method remains vulnerable to false positive detections of light reflections from the edges of the human organ phantom. This could be mitigated in future work by adapting the lighting setup of the workspace.

Part VI

Conclusion

Chapter 15

Concluding Remarks

Efficiency and reliability are key objectives in practical applications of robot manipulation. Yet, increasing one can result in decreasing the other, as seen in Chapter 2 where increasing the efficiency and speed of an optimizing motion planner compromised system reliability and in Chapter 8 where increasing perception reliability for grasping slowed down the system, reducing efficiency. It is a common practice to develop a proof of concept showcasing that robots are able to perform a task, leaving it to users to optimize for efficiency and reliability. However, this is rarely straightforward, and as a result, many developments never get adopted. To facilitate the adoption of robot manipulation research in real-world applications, this thesis explores various tasks from different industries, with the goal of developing methods that are both efficient and reliable.

Across domains, applications and even robots, we observe that constrained optimization is effective for enhancing the efficiency and reliability of the system when we have a reliable model, as described in parts I, II, IV and V. However, in scenarios with high uncertainty in the dynamics of the system, or when we do not have a reliable model, methods involving active and interactive perception, along with self-supervised learning, prove to be more effective, as described in parts II, III, and V.

The primary contributions of this thesis are:

- We frame the objectives of increasing the efficiency and reliability of manipulation systems as constrained optimization problems, as shown in Chapters 2 to 5, 7, 12 and 13.

- We propose a procedure for warm starting an optimizing motion planner with an approximation from a deep neural network to reduce the planning time and improve the efficiency as described in Chapter 2.

- We formalize, tune, and learn acceleration constraints and 3D reconstruction objectives to improve the success rate and the reliability of motion planning and grasping as described in Chapters 3, 4, and 8.

- We develop active and interactive perception primitives to increase the system's reliability when it is difficult to estimate its state, as presented in Chapters 6, 9 and 14.
- We use self-supervised learning to develop an efficient and reliable system when it is difficult to model its dynamics, as described in Chapters 4, 10, and 14.
- We introduce an efficient polyculture farming simulator that integrates parameterized models of plant growth, as well as plant interactions, and train a policy to optimize plant coverage and diversity over a short horizon, as presented in Chapters 11 and 12.

Looking forward, the aim is to bridge the gap further between robotic manipulation research and its practical, commercial applications. The concluding section of this thesis will discuss lessons learned and propose future research directions.

15.1 Lessons Learned

Throughout my PhD, I've had the privilege of working on many research projects with a broad range of topics and collaborating with a large number of collaborators. In the following subsection, I'll discuss some of the important principles that have shaped my approach, hoping they will be of value to future readers.

15.1.1 Physical Experiments

Having a simulation of the robot and its environment is invaluable for quickly testing various hypotheses. However, in robotics, it is essential to also test the methods on a physical robot. This real-world testing reveals unique behaviors that highlight the limitations of the methods being developed. For instance, it was only through physical testing that we discovered the end-effector acceleration constraint, discussed in Chapter 3, was ineffective for suction grasping. This finding motivated the further research detailed in Chapter 4.

From conducting thousands of physical experiments, I've learned a key lesson: begin testing as early as possible. Physical experiments not only provide the most significant insights but also expose the majority of challenges, which can stem from unsuitable hardware, discrepancies between simulations and the physical environment, or incorrect assumptions. Starting early is important for the success of any robotics project.

15.1.2 Rejected Papers

Receiving a rejection from a journal or conference can be quite demotivating, especially after the extensive effort of working on a paper, incorporating feedback from collaborators, and making last-minute updates—only to receive a rejection notice months later. This experience can certainly be disappointing.

However, such rejections are part of the research process and can be beneficial. Detailed and high-quality feedback from reviewers can be invaluable, allowing us to improve our work significantly. This process of refinement and repeated exploration is the essence of research—continually searching and reevaluating until a breakthrough is achieved.

For instance, one of my papers was initially rejected from a conference, but the comprehensive feedback we received was instrumental in improving it. We applied the suggestions, and several months later, when the revised paper was accepted by a top-tier journal, we were not disappointed anymore.

15.2 Opportunities for Future Work

There are a number of exciting areas to explore in future work related to several chapters of this dissertation.

15.2.1 Reactive Motion Planning Around Moving Obstacles

GOMP-based methods, which we discuss in Chapters 2 to 5, are designed to plan open-loop, collision-free trajectories in environments where obstacles are fixed. These methods can plan trajectories in milliseconds, and we could potentially extend them to manage dynamic scenarios involving moving obstacles. One possible approach is to extend methods like DJ-GOMP, which we introduced in Chapter 2, using deep learning to accelerate re-planning trajectories as obstacles move.

However, several challenges arise when adapting these methods for moving obstacles. These include predicting future positions of the obstacles rather than merely reacting to their current locations, ensuring the system can safely stop if it cannot find a collision-free path, and effectively executing these trajectories on physical robots. Each of these aspects presents significant hurdles that need to be addressed.

15.2.2 Grasping Transparent Objects in Real-Time

The methods described in Chapter 8 have significantly reduced the cycle time for grasping transparent objects using NeRF, bringing it down from hours to several seconds. However, to make these methods suitable for industrial applications, we need to further accelerate the process by an order of magnitude. One strategy could involve incorporating grasp prediction directly into the NeRF model as an additional learned feature. Alternatively, exploring different models, such as Gaussian Splatting [356], might also achieve the desired speed-up.

15.2.3 Suture Planning in Real-Time

One challenge with open-loop planning is that it can be inaccurate due to dynamic changes in the environment. A potential solution is real-time replanning to adjust as changes occur,

such as a wound changing shape due to the forces applied during suturing. However, existing methods for this are often slow to converge. Similar to the optimization approach discussed in Chapters 2–5, which deals with a nonconvex optimization objective, we address this issue using Sequential Quadratic Programming (SQP). In this approach, the SQP is initiated with points that are evenly spaced along the wound.

A promising strategy for improving real-time performance could be training a neural network to provide a "warm start" for the optimization process. This technique is similar to the one used in DJ-GOMP, described in Chapter 2, where the neural network accelerates the optimization by predicting a near-optimal solution, thereby enhancing efficiency and computation time.

Bibliography

[1] Matthew T Mason. "Toward robotic manipulation". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 1–28.

[2] L E Kavraki, P Svestka, J.-C. Latombe, and M Overmars. "Probabilistic roadmaps for path planning in high dimensional configuration spaces". In: *IEEE Trans. Robotics and Automation* 12.4 (1996), pp. 566–580.

[3] Sertac Karaman and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning". In: *Int. J. Robotics Research* 30.7 (June 2011), pp. 846–894.

[4] Steven M LaValle and James J Kuffner. "Rapidly-exploring random trees: Progress and prospects: Steven m. lavalle, iowa state university, a james j. kuffner, jr., university of tokyo, tokyo, japan". In: *Algorithmic and computational robotics* (2001), pp. 303–307.

[5] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." In: *Robotics: Science and Systems*. 2013, pp. 1–10.

[6] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. "STOMP: Stochastic trajectory optimization for motion planning". In: *2011 IEEE international conference on robotics and automation*. IEEE. 2011, pp. 4569–4574.

[7] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. "CHOMP: Gradient optimization techniques for efficient motion planning". In: *2009 IEEE International Conference on Robotics and Automation*. IEEE. 2009, pp. 489–494.

[8] Marc Toussaint. "Newton methods for k-order Markov constrained motion problems". In: *arXiv preprint arXiv:1407.0414* (2014).

[9] Jeffrey Ichnowski, Michael Danielczuk, Jingyi Xu, Vishal Satish, and Ken Goldberg. "GOMP: Grasp-Optimized Motion Planning for Bin Picking". In: *2020 International Conference on Robotics and Automation (ICRA)*. IEEE. May 2020.

[10] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio, and Ken Goldberg. "Dex-Net 2.0: Deep Learning to Plan Robust Grasps with Synthetic Point Clouds and Analytic Grasp Metrics". In: *Robotics: Science and Systems (RSS)*. 2017.

[11] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. "NeRF: Representing scenes as neural radiance fields for view synthesis". In: *European Conference on Computer Vision*. Springer. 2020, pp. 405–421.

[12] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. "Planar robot casting with real2sim2real self-supervised learning". In: *arXiv preprint arXiv:2111.04814* (2021).

[13] Hirofumi Nakagaki, Kkosei Kitagi, Tsukasa Ogasawara, and Hideo Tsukune. "Study of insertion task of a flexible wire into a hole by using visual tracking observed by stereo vision". In: *Proceedings of IEEE international conference on robotics and automation*. Vol. 4. IEEE. 1996, pp. 3209–3214.

[14] Yu She, Shaoxiong Wang, Siyuan Dong, Neha Sunil, Alberto Rodriguez, and Edward Adelson. "Cable manipulation with a tactile-reactive gripper". In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1385–1401.

[15] Huy Ha and Shuran Song. "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding". In: *Conference on Robot Learning*. PMLR. 2022, pp. 24–33.

[16] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. "Visuospatial foresight for multi-step, multi-task fabric manipulation". In: *Robotics: Science and Systems (RSS)* (2020).

[17] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. "Learning visible connectivity dynamics for cloth smoothing". In: *Conference on Robot Learning*. PMLR. 2022, pp. 256–266.

[18] Lawrence Yunliang Chen, Huang Huang, Ellen Novoseller, Daniel Seita, Jeffrey Ichnowski, Michael Laskey, Richard Cheng, Thomas Kollar, and Ken Goldberg. "Efficiently learning single-arm fling motions to smooth garments". In: *The International Symposium of Robotics Research*. Springer. 2022, pp. 36–51.

[19] Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. "FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy". In: *Conference on Robot Learning*. PMLR. 2022, pp. 192–202.

[20] Puttichai Lertkultanon and Quang-Cuong Pham. "A certified-complete bimanual manipulation planner". In: *IEEE Transactions on Automation Science and Engineering* 15.3 (2018), pp. 1355–1368.

[21] Shogo Hayakawa, Weiwei Wan, Keisuke Koyama, and Kensuke Harada. "A Dual-Arm Robot That Autonomously Lifts Up and Tumbles Heavy Plates Using Crane Pulley Blocks". In: *IEEE Transactions on Automation Science and Engineering* (2021).

[22] Yang Hu, Lin Zhang, Wei Li, and Guang-Zhong Yang. "Robotic sewing and knot tying for personalized stent graft manufacturing". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 754–760.

[23] Hamed Saeidi, Hanh ND Le, Justin D Opfermann, Simon Léonard, A Kim, Michael H Hsieh, Jin U Kang, and Axel Krieger. "Autonomous laparoscopic robotic suturing with a novel actuated suturing tool and 3D endoscope". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 1541–1547.

[24] Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. "Learning ambidextrous robot grasping policies". In: *Science Robotics* 4.26 (2019), eaau4984.

[25] Alan Kuntz, Chris Bowen, and Ron Alterovitz. "Fast anytime motion planning in point clouds by interleaving sampling and interior point optimization". In: *Robotics Research* (2020), pp. 929–945.

[26] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[27] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2006, pp. 535–541.

[28] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. "Policy distillation". In: *arXiv preprint arXiv:1511.06295* (2015).

[29] René Traoré Kalifou, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Natalia Diaz-Rodriguez, and David Filliat. "Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer". In: *ICML Workshop on Multi-Task and Lifelong Learning*. 2019.

[30] René Traoré, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz-Rodríguez, and David Filliat. "DISCORL: Continual reinforcement learning via policy distillation". In: *arXiv preprint arXiv:1907.05855* (2019).

[31] Igor Mordatch and Emo Todorov. "Combining the benefits of function approximation and trajectory optimization." In: *Robotics: Science and Systems*. Vol. 4. 2014.

[32] Matthew E Taylor and Peter Stone. "Transfer learning for reinforcement learning domains: A survey". In: *Journal of Machine Learning Research* 10.Jul (2009), pp. 1633–1685.

[33] Teguh Santoso Lembono, Carlos Mastalli, Pierre Fernbach, Nicolas Mansard, and Sylvain Calinon. "Learning How to Walk: Warm-starting Optimal Control Solver with Memory of Motion". In: *arXiv preprint arXiv:2001.11751* (2020).

[34] Nicolas Mansard, Andrea DelPrete, Mathieu Geisert, Steve Tonneau, and Olivier Stasse. "Using a memory of motion to efficiently warm-start a nonlinear predictive controller". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2986–2993.

[35] Gu Ye and Ron Alterovitz. "Guided motion planning". In: *Robotics research*. Springer, 2017, pp. 291–307.

[36] Pieter Abbeel, Dmitri Dolgov, Andrew Y Ng, and Sebastian Thrun. "Apprenticeship learning for motion planning with application to parking lot navigation". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 1083–1090.

[37] M Rana, Mustafa Mukadam, Seyed Reza Ahmadzadeh, Sonia Chernova, and Byron Boots. "Towards robust skill generalization: Unifying learning from demonstration and motion planning". In: *Intelligent robots and systems*. 2018.

[38] Brian Ichter, James Harrison, and Marco Pavone. "Learning sampling distributions for robot motion planning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 7087–7094.

[39] Matt Zucker, James Kuffner, and J Andrew Bagnell. "Adaptive workspace biasing for sampling based planners". In: (2008).

[40] Ahmed H Qureshi, Anthony Simeonov, Mayur J Bency, and Michael C Yip. "Motion planning networks". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 2118–2124.

[41] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 2011, pp. 627–635.

[42] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[43] Steven J Rennie, Vaibhava Goel, and Samuel Thomas. "Annealed dropout training of deep networks". In: *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2014, pp. 159–164.

[44] Universal Robotics. *UR5 Collaborative Robot Arm*. `https://web.archive.org/web/20190815054522/https://www.universal-robots.com/products/ur5-robot/`. [Accessed 2019-08-15].

[45] Robotiq. *2F-85 and 2F-140 Grippers*. `https://robotiq.com/products/2f85-140-adaptive-robot-gripper`. [Accessed 2019-05-19].

[46] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. "OSQP: an operator splitting solver for quadratic programs". In: *Mathematical Programming Computation* (2020).

[47] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012). arXiv: `1212.5701`.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1026–1034.

[49] Tobias Kunz and Mike Stilman. "Time-optimal trajectory generation for path following with bounded acceleration and velocity". In: *Robotics: Science and Systems VIII* (2012).

[50] Jeffrey Ichnowski, Yahav Avigal, Vishal Satish, and Ken Goldberg. "Deep learning can accelerate grasp-optimized motion planning". In: *Science Robotics* 5.48 (2020).

[51] Zhenwang Yao and Kamal Gupta. "Path planning with general end-effector constraints". In: *Robotics and Autonomous Systems* 55.4 (2007), pp. 316–327.

[52] Yinkang Li, Xiaolong Hao, Yuchen She, Shuang Li, and Meng Yu. "Constrained motion planning of free-float dual-arm space manipulator via deep reinforcement learning". In: *Aerospace Science and Technology* 109 (2021), p. 106446.

[53] Kevin M Lynch and Matthew T Mason. "Dynamic underactuated nonprehensile manipulation". In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS'96*. Vol. 2. IEEE. 1996, pp. 889–896.

[54] Kevin M Lynch and Matthew T Mason. "Dynamic nonprehensile manipulation: Controllability, planning, and experiments". In: *The International Journal of Robotics Research* 18.1 (1999), pp. 64–92.

[55] Siddhartha S Srinivasa, Michael A Erdmann, and Matthew T Mason. "Using projected dynamics to plan dynamic contact manipulation". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2005, pp. 3618–3623.

[56] Seungsu Kim, Ashwini Shukla, and Aude Billard. "Catching Objects in Flight". In: *IEEE Transactions on Robotics* 30.5 (2014), pp. 1049–1065.

[57] Caio Mucchiani and Mark Yim. "Dynamic Grasping for Object Picking Using Passive Zero-DOF End-Effectors". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3089–3096.

[58] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. "Tossingbot: Learning to throw arbitrary objects with residual physics". In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1307–1319.

[59] Chen Wang, Shaoxiong Wang, Branden Romero, Filipe Veiga, and Edward Adelson. "SwingBot: Learning Physical Features from In-hand Tactile Exploration for Dynamic Swing-up Manipulation". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[60] Quang-Cuong Pham, Stéphane Caron, and Yoshihiko Nakamura. "Kinodynamic Planning in the Configuration Space via Admissible Velocity Propagation." In: *Robotics: Science and Systems*. Vol. 32. 2013.

[61] Quang-Cuong Pham, Stéphane Caron, Puttichai Lertkultanon, and Yoshihiko Nakamura. "Planning truly dynamic motions: Path-velocity decomposition revisited". In: *arXiv preprint arXiv:1411.4045* (2014).

[62] Puttichai Lertkultanon and Quang-Cuong Pham. "Dynamic non-prehensile object transportation". In: *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE. 2014, pp. 1392–1397.

[63] Michael Posa and Russ Tedrake. "Direct trajectory optimization of rigid body dynamical systems through contact". In: *Algorithmic foundations of robotics X*. Springer, 2013, pp. 527–542.

[64] 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* 33.1 (2014), pp. 69–81.

[65] Kris Hauser. "Fast interpolation and time-optimization with contact". In: *The International Journal of Robotics Research* 33.9 (2014), pp. 1231–1250.

[66] Jingru Luo and Kris Hauser. "Robust trajectory optimization under frictional contact with iterative learning". In: *Autonomous Robots* 41.6 (2017), pp. 1447–1461.

[67] Jay D Bernheisel and Kevin M Lynch. "Stable transport of assemblies: Pushing stacked parts". In: *IEEE Transactions on Automation science and Engineering* 1.2 (2004), pp. 163–168.

[68] Ash Yaw Sang Wan, Yi De Soong, Edwin Foo, Wai Leong Eugene Wong, and Wai Shing Michael Lau. "Waiter Robots Conveying Drinks". In: *Technologies* 8.3 (2020), p. 44.

[69] Praneel Acharya, Kim-Doang Nguyen, Hung M La, Dikai Liu, and I-Ming Chen. "Nonprehensile Manipulation: a Trajectory-Planning Perspective". In: *IEEE/ASME Transactions on Mechatronics* 26.1 (2020), pp. 527–538.

[70] Hyeonbeom Lee and H Jin Kim. "Constraint-based cooperative control of multiple aerial manipulators for handling an unknown payload". In: *IEEE Transactions on Industrial Informatics* 13.6 (2017), pp. 2780–2790.

[71] Hung Pham and Quang-Cuong Pham. "Critically fast pick-and-place with suction cups". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 3045–3051.

[72] Suei Jen Chen, Bjorn Hein, and Heinz Worn. "Using acceleration compensation to reduce liquid surface oscillation during a high speed transfer". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 2951–2956.

[73] Wisnu Aribowo, Takahito Yamashita, and Kazuhiko Terashima. "Integrated trajectory planning and sloshing suppression for three-dimensional motion of liquid container transfer robot arm". In: *Journal of Robotics* 2015 (2015).

[74] Ken'ichi Yano and Kazuhiko Terashima. "Robust liquid container transfer control for complete sloshing suppression". In: *IEEE Transactions on Control Systems Technology* 9.3 (2001), pp. 483–493.

[75] Mahmut Reyhanoglu and Jaime Rubio Hervas. "Nonlinear modeling and control of slosh in liquid container transfer via a PPR robot". In: *Communications in Nonlinear Science and Numerical Simulation* 18.6 (2013), pp. 1481–1490.

[76] Luca Consolini, Alessandro Costalunga, Aurelio Piazzi, and Marco Vezzosi. "Minimum-time feedforward control of an open liquid container". In: *IECON 2013-39th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2013, pp. 3592–3597.

[77] Lorenzo Moriello, Luigi Biagiotti, Claudio Melchiorri, and Andrea Paoli. "Control of liquid handling robotic systems: A feed-forward approach to suppress sloshing". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 4286–4291.

[78] John YS Luh, Michael W Walker, and Richard PC Paul. "On-line computational scheme for mechanical manipulators". In: (1980).

[79] Jeffrey Mahler, Matthew Matl, Xinyu Liu, Albert Li, David Gealy, and Ken Goldberg. "Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 1–8.

[80] Tae Myung Huh, Kate Sanders, Michael Danielczuk, Monica Li, Ken Goldberg, and Hannah S Stuart. "A Multi-Chamber Smart Suction Cup for Adaptive Gripping and Haptic Exploration". In: *arXiv preprint arXiv:2105.02345* (2021).

[81] Jeffrey Ichnowski, Yahav Avigal, Yi Liu, and Ken Goldberg. "GOMP-FIT: Grasp-Optimized Motion Planning for Fast Inertial Transport". In: *2022 International Conference on Robotics and Automation (ICRA). (to appear)*. IEEE. 2022.

[82] Chonhyon Park, Jia Pan, and Dinesh Manocha. "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments". In: *Twenty-Second International Conference on Automated Planning and Scheduling*. 2012.

[83] Donato Maragno, Holly Wiberg, Dimitris Bertsimas, S Ilker Birbil, Dick den Hertog, and Adejuyigbe Fajemisin. "Mixed-Integer Optimization with Constraint Learning". In: *arXiv preprint arXiv:2111.04469* (2021).

[84] Patryk Kudła and Tomasz P Pawlak. "One-class synthesis of constraints for Mixed-Integer Linear Programming with C4. 5 decision trees". In: *Applied Soft Computing* 68 (2018), pp. 1–12.

[85] Andrea Bartolini, Michele Lombardi, Michela Milano, and Luca Benini. "Neuron constraints to model complex real-world problems". In: *International Conference on Principles and Practice of Constraint Programming*. Springer. 2011, pp. 115–129.

[86] Michele Lombardi, Michela Milano, and Andrea Bartolini. "Empirical decision model learning". In: *Artificial Intelligence* 244 (2017), pp. 343–367.

[87] Luc De Raedt, Andrea Passerini, and Stefano Teso. "Learning constraints from examples". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[88] Adejuyigbe Fajemisin, Donato Maragno, and Dick den Hertog. "Optimization with constraint learning: A framework and survey". In: *arXiv preprint arXiv:2110.02121* (2021).

[89] Fabio Ruggiero, Vincenzo Lippiello, and Bruno Siciliano. "Nonprehensile dynamic manipulation: A survey". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1711–1718.

[90] Huy Ha and Shuran Song. "FlingBot: The Unreasonable Effectiveness of Dynamic Manipulation for Cloth Unfolding". In: *Conference on Robotic Learning (CoRL)*. 2021.

[91] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. "Planar Robot Casting with Real2Sim2Real Self-Supervised Learning". In: *arXiv preprint arXiv:2111.04814* (2021).

[92] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. "JAX: Autograd and XLA". In: *Astrophysics Source Code Library* (2021), ascl–2111.

[93] Lars Berscheid and Torsten Kröger. "Jerk-limited Real-time Trajectory Generation with Arbitrary Target States". In: *arXiv preprint arXiv:2105.04830* (2021).

[94] R. Kolluru, K.P. Valavanis, and T.M. Hebert. "Modeling, analysis, and performance evaluation of a robotic gripper system for limp material handling". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 28.3 (1998), pp. 480–486.

[95] Hannah S. Stuart, Matteo Bagheri, Shiquan Wang, Heather Barnard, Audrey L. Sheng, Merritt Jenkins, and Mark R. Cutkosky. "Suction helps in a pinch: Improving underwater manipulation with gentle suction flow". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), pp. 2279–2284.

[96] Angel J. Valencia, Roger M. Idrovo, Angel D. Sappa, Douglas Plaza Guingla, and Daniel Ochoa. "A 3D vision based approach for optimal grasp of vacuum grippers". In: *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*. 2017, pp. 1–6.

[97] Jeffrey Ichnowski and Ron Alterovitz. "Motion Planning Templates: A Motion Planning Framework for Robots with Low-power CPUs". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 612–618.

[98] Tobia Marcucci, Mark Petersen, David von Wrangel, and Russ Tedrake. "Motion planning around obstacles with convex optimization". In: *arXiv preprint arXiv:2205.04422* (2022).

[99] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. "Segment Anything". In: *arXiv:2304.02643* (2023).

[100] Xu Zhao, Wenchao Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. *Fast Segment Anything*. 2023. arXiv: `2306.12156 [cs.CV]`.

[101] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981).

[102] Jia Pan, Sachin Chitta, and Dinesh Manocha. "FCL: A general purpose library for collision and proximity queries". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012, pp. 3859–3866.

[103] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. *Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning*. 2021.

[104] Jeffrey Ichnowski and Ron Alterovitz. "Scalable multicore motion planning using lock-free concurrency". In: *IEEE Transactions on Robotics* 30.5 (2014), pp. 1123–1136.

[105] Huang Huang, Michael Danielczuk, Chung Min Kim, Letian Fu, Zachary Tam, Jeffrey Ichnowski, Anelia Angelova, Brian Ichter, and Ken Goldberg. "Mechanical search on shelves using a novel "bluction" tool". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 6158–6164.

[106] Jeffrey Mahler, Florian T Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James Kuffner, and Ken Goldberg. "Dex-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2016, pp. 1957–1964.

[107] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-dof graspnet: Variational grasp generation for object manipulation". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2901–2910.

[108] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Learning robust, real-time, reactive robotic grasping". In: *Int. Journal of Robotics Research (IJRR)* 39.2-3 (2020), pp. 183–201.

[109] Jeffrey Mahler and Ken Goldberg. "Learning deep policies for robot bin picking by simulating robust grasping sequences". In: *Conference on Robot Learning (CoRL)*. 2017, pp. 515–524.

[110] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. "Mechanical search: Multi-step retrieval of a target object occluded by clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 1614–1621.

[111] Michael Danielczuk, Anelia Angelova, Vincent Vanhoucke, and Ken Goldberg. "X-ray: Mechanical search for an occluded object by minimizing support of learned occupancy distributions". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[112] Huang Huang, Marcus Dominguez-Kuhne, Jeffrey Ichnowski, Vishal Satish, Michael Danielczuk, Kate Sanders, Andrew Lee, Anelia Angelova, Vincent Vanhoucke, and Ken Goldberg. "Mechanical Search on Shelves using Lateral Access X-RAY". In: *arXiv preprint arXiv:2011.11696* (2020).

[113] Jue Kun Li, David Hsu, and Wee Sun Lee. "Act to see and see to act: POMDP planning for objects search in clutter". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2016, pp. 5701–5707.

[114] Andrew Price, Linyi Jin, and Dmitry Berenson. "Inferring occluded geometry improves performance when retrieving an object from dense clutter". In: *Int. S. Robotics Research (ISRR)*. 2019.

[115] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. "6-dof grasping for target-driven object manipulation in clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 6232–6238.

[116] Enrique Dunn and Jan-Michael Frahm. "Next Best View Planning for Active Model Improvement." In: *BMVC*. 2009, pp. 1–11.

[117] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. "Active recognition through next view planning: a survey". In: *Pattern Recognition* 37.3 (2004), pp. 429–446.

[118] Ruzena Bajcsy. "Active perception". In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[119] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. "Revisiting active perception". In: *Autonomous Robots* 42.2 (2018), pp. 177–196.

[120] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees". In: *Autonomous robots* 34.3 (2013), pp. 189–206.

[121] Dan Halperin. "Robust geometric computing in motion". In: *Int. Journal of Robotics Research (IJRR)* 21.3 (2002), pp. 219–232.

[122] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. "Online planning for target object search in clutter under partial observability". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 8241–8247.

[123] Tara Boroushaki, Junshan Leng, Ian Clester, Alberto Rodriguez, and Fadel Adib. "Robotic Grasping of Fully-Occluded Objects using RF Perception". In: *arXiv preprint arXiv:2012.15436* (2020).

[124] Yiannis Aloimonos. *Active perception*. Psychology Press, 2013.

[125] John Aloimonos. "Purposive and qualitative active vision". In: *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. 1990, pp. 346–360.

[126] Richard Pito. "A solution to the next best view problem for automated surface acquisition". In: *IEEE Transactions on pattern analysis and machine intelligence* 21.10 (1999), pp. 1016–1030.

[127] Henry Carrillo, Ian Reid, and José A Castellanos. "On the comparison of uncertainty criteria for active SLAM". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2012, pp. 2080–2087.

[128] Gregory Kahn, Peter Sujan, Sachin Patil, Shaunak Bopardikar, Julian Ryde, Ken Goldberg, and Pieter Abbeel. "Active exploration using trajectory optimization for robotic grasping in the presence of occlusions". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2015, pp. 4783–4790.

[129] Douglas Morrison, Peter Corke, and Jürgen Leitner. "Multi-view picking: Next-best-view reaching for improved grasping in clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 8762–8768.

[130] Tonci Novkovic, Remi Pautrat, Fadri Furrer, Michel Breyer, Roland Siegwart, and Juan Nieto. "Object finding in cluttered scenes using interactive perception". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020, pp. 8338–8344.

[131] João Santos, Miguel Oliveira, Rafael Arrais, and Germano Veiga. "Autonomous Scene Exploration for Robotics: A Conditional Random View-Sampling and Evaluation Using a Voxel-Sorting Mechanism for Efficient Ray Casting". In: *Sensors* 20.15 (2020), p. 4331.

[132] Megha Gupta, Thomas Rühr, Michael Beetz, and Gaurav S Sukhatme. "Interactive environment exploration in clutter". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2013, pp. 5265–5272.

[133] Pierre Soille. "Erosion and dilation". In: *Morphological Image Analysis*. Springer, 2004, pp. 63–103.

[134] Tomas Lozano-Perez, Matthew T Mason, and Russell H Taylor. "Automatic synthesis of fine-motion strategies for robots". In: *Int. Journal of Robotics Research (IJRR)* 3.1 (1984), pp. 3–24.

[135] Kentaro Wada. *Octomap-python*. `https://github.com/wkentaro/octomap-python`. 2013.

[136] Michael Danielczuk, Matthew Matl, Saurabh Gupta, Andrew Li, Andrew Lee, Jeffrey Mahler, and Ken Goldberg. "Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2019, pp. 7283–7290.

[137] *Thingiverse online 3D object database.* Thingiverse. URL: `https://www.thingiverse.com/` (visited on 03/09/2021).

[138] Berk Calli, Aaron Walsman, Arjun Singh, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. "Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols". In: *arXiv preprint arXiv:1502.03143* (2015).

[139] Johannes Lutz Schönberger and Jan-Michael Frahm. "Structure-from-Motion Revisited". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[140] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. "Pixelwise View Selection for Unstructured Multi-View Stereo". In: *European Conference on Computer Vision (ECCV)*. 2016.

[141] Po-Jen Lai and Chiou-Shann Fuh. "Transparent object detection using regions with convolutional neural network". In: *IPPR Conference on Computer Vision, Graphics, and Image Processing*. Vol. 2. 2015.

[142] May Phyo Khaing and Mukunoki Masayuki. "Transparent object detection using convolutional neural network". In: *International Conference on Big Data Analysis and Deep Learning Applications*. Springer. 2018, pp. 86–93.

[143] Enze Xie, Wenjia Wang, Wenhai Wang, Peize Sun, Hang Xu, Ding Liang, and Ping Luo. "Segmenting transparent object in the wild with transformer". In: *arXiv preprint arXiv:2101.08461* (2021).

[144] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[145] Cody J Phillips, Matthieu Lecce, and Kostas Daniilidis. "Seeing Glassware: from Edge Detection to Pose Estimation and Shape Recovery." In: *Robotics: Science and Systems*. Vol. 3. 2016.

[146] Chi Xu, Jiale Chen, Mengyang Yao, Jun Zhou, Lijun Zhang, and Yi Liu. "6DoF Pose Estimation of Transparent Object from a Single RGB-D Image". In: *Sensors* 20.23 (2020), p. 6790.

[147] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. "Densefusion: 6d object pose estimation by iterative dense fusion". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 3343–3352.

[148] Shreeyak Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. "Clear Grasp: 3D Shape Estimation of Transparent Objects for Manipulation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3634–3642.

[149] Zheming Zhou, Tianyang Pan, Shiyu Wu, Haonan Chang, and Odest Chadwicke Jenkins. "Glassloc: Plenoptic grasp pose detection in transparent clutter". In: *arXiv preprint arXiv:1909.04269* (2019).

[150] Zheming Zhou, Xiaotong Chen, and Odest Chadwicke Jenkins. "LIT: Light-field Inference of Transparency for Refractive Object Localization". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4548–4555.

[151] Luyang Zhu, Arsalan Mousavian, Yu Xiang, Hammad Mazhar, Jozef van Eenbergen, Shoubhik Debnath, and Dieter Fox. "RGB-D Local Implicit Function for Depth Completion of Transparent Objects". In: *arXiv preprint arXiv:2104.00622* (2021).

[152] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. "Occupancy networks: Learning 3d reconstruction in function space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470.

[153] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. "DeepSDF: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174.

[154] Zhiqin Chen and Hao Zhang. "Learning implicit fields for generative shape modeling". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948.

[155] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. "Scene representation networks: Continuous 3d-structure-aware neural scene representations". In: *arXiv preprint arXiv:1906.01618* (2019).

[156] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. "PlenOctrees for real-time rendering of neural radiance fields". In: *arXiv preprint arXiv:2103.14024* (2021).

[157] Boyang Deng, Jonathan T. Barron, and Pratul P. Srinivasan. *JaxNeRF: an efficient JAX implementation of NeRF*. 2020.

[158] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. "Depth-supervised NeRF: Fewer Views and Faster Training for Free". In: *arXiv e-prints* (2021), arXiv–2107.

[159] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. "Vision-Only Robot Navigation in a Neural Radiance World". In: *arXiv preprint arXiv:2110.00168* (2021).

[160] Kilian Kleeberger, Richard Bormann, Werner Kraus, and Marco F Huber. "A survey on learning-based robotic grasping". In: *Current Robotics Reports* (2020), pp. 1–11.

[161] Antonio Bicchi and Vijay Kumar. "Robotic grasping and contact: A review". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. Vol. 1. IEEE. 2000, pp. 348–353.

[162] Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[163] Daniel Kappler, Jeannette Bohg, and Stefan Schaal. "Leveraging big data for grasp planning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 4304–4311.

[164] Domenico Prattichizzo, Jeffrey C Trinkle, Bruno Siciliano, and Oussama Khatib. "Springer handbook of robotics". In: *Grasping; Springer: Berlin/Heidelberg, Germany* (2008), pp. 671–700.

[165] Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. "Grasp2vec: Learning object representations from self-supervised grasping". In: *arXiv preprint arXiv:18-11.06964* (2018).

[166] Gang Peng, Zhenyu Ren, Hao Wang, and Xinde Li. "A self-supervised learning-based 6-DOF grasp planning method for manipulator". In: *arXiv preprint arXiv:2102.00205* (2021).

[167] Ian Lenz, Honglak Lee, and Ashutosh Saxena. "Deep learning for detecting robotic grasps". In: *International Journal of Robotics Research (IJRR)* 34.4-5 (2015), pp. 705–724.

[168] Joseph Redmon and Anelia Angelova. "Real-time grasp detection using convolutional neural networks". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 1316–1322.

[169] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. "6-DOF GraspNet: Variational grasp generation for object manipulation". In: 2019, pp. 2901–2910.

[170] Yuzhe Qin, Rui Chen, Hao Zhu, Meng Song, Jing Xu, and Hao Su. "S4G: Amodal single-view single-shot SE(3) grasp detection in cluttered scenes". In: *Conference on robot learning*. PMLR. 2020, pp. 53–65.

[171] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. "Contact-GraspNet: Efficient 6-DoF Grasp Generation in Cluttered Scenes". In: *arXiv preprint arXiv:2103.14127* (2021).

[172] Yahav Avigal, Vishal Satish, Zachary Tam, Huang Huang, Harry Zhang, Michael Danielczuk, Jeffrey Ichnowski, and Ken Goldberg. "AVPLUG: Approach Vector PLanning for Unicontact Grasping amid Clutter". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2021, pp. 1140–1147.

[173] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan Nieto. "Volumetric grasping network: Real-time 6 DOF grasp detection in clutter". In: *arXiv preprint arXiv:2101.01132* (2021).

[174] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. "Grasping in the wild: Learning 6 DOF closed-loop grasping from low-cost demonstrations". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4978–4985.

[175] Shreeyak S. Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. "ClearGrasp: 3D Shape Estimation of Transparent Objects for Manipulation". In: *CoRR* abs/1910.02550 (2019). arXiv: 1910.02550.

[176] Jeffrey Ichnowski, Yahav Avigal, Justin Kerr, and Ken Goldberg. "Dex-NeRF: Using a Neural Radiance Field to Grasp Transparent Objects". In: *5th Annual Conference on Robot Learning*. 2021.

[177] Cheng Sun, Min Sun, and Hwann-Tzong Chen. "Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5459–5469.

[178] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. "Plenoxels: Radiance Fields Without Neural Networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5501–5510.

[179] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding". In: *ACM Trans. Graph.* (2022).

[180] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. "Nerfies: Deformable neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5865–5874.

[181] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. "Neural scene flow fields for space-time view synthesis of dynamic scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6498–6508.

[182] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. "Graf: Generative radiance fields for 3D-aware image synthesis". In: *Advances in Neural Information Processing Systems (NeurIPS)*. Vol. 33. 2020.

[183] Eric R Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. "pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 5799–5809.

[184] Jiatao Gu, Lingjie Liu, Peng Wang, and Christian Theobalt. "Stylenerf: A style-based 3d-aware generator for high-resolution image synthesis". In: *arXiv preprint arXiv:2110.08985* (2021).

[185] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. "inerf: Inverting neural radiance fields for pose estimation". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 1323–1330.

[186] Quan Meng, Anpei Chen, Haimin Luo, Minye Wu, Hao Su, Lan Xu, Xuming He, and Jingyi Yu. "Gnerf: Gan-based neural radiance field without posed camera". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6351–6361.

[187] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. "Barf: Bundle-adjusting neural radiance fields". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5741–5751.

[188] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. "TensoRF: Tensorial Radiance Fields". In: *arXiv preprint arXiv:2203.09517* (2022).

[189] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. "IBRNet: Learning Multi-View Image-Based Rendering". In: *CVPR*. 2021.

[190] Alex Trevithick and Bo Yang. "Grf: Learning a general radiance field for 3d representation and rendering". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 15182–15192.

[191] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 14124–14133.

[192] Julian Chibane, Aayush Bansal, Verica Lazova, and Gerard Pons-Moll. "Stereo radiance fields (srf): Learning view synthesis for sparse views of novel scenes". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 7911–7920.

[193] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. "pixelnerf: Neural radiance fields from one or few images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4578–4587.

[194] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P Srinivasan, Jonathan T Barron, and Ren Ng. "Learned initializations for optimizing coordinate-based neural representations". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 2846–2855.

[195] Chen Gao, Yichang Shih, Wei-Sheng Lai, Chia-Kai Liang, and Jia-Bin Huang. "Portrait neural radiance fields from a single image". In: *arXiv preprint arXiv:2012.05903* (2020).

[196] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew J Davison. "imap: Implicit mapping and positioning in real-time". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 6229–6238.

[197] Michal Adamkiewicz, Timothy Chen, Adam Caccavale, Rachel Gardner, Preston Culbertson, Jeannette Bohg, and Mac Schwager. "Vision-only robot navigation in a neural radiance world". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4606–4613.

[198] Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. "Implicit Object Mapping With Noisy Data". In: *arXiv preprint arXiv:2204.10516* (2022).

[199] Yunzhu Li, Shuang Li, Vincent Sitzmann, Pulkit Agrawal, and Antonio Torralba. "3d neural scene representations for visuomotor control". In: *Conference on Robot Learning*. PMLR. 2022, pp. 112–123.

[200] Lin Yen-Chen, Pete Florence, Jonathan T Barron, Tsung-Yi Lin, Alberto Rodriguez, and Phillip Isola. "NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields". In: *arXiv preprint arXiv:2203.01913* (2022).

[201] Wei-Cheng Tseng, Hung-Ju Liao, Lin Yen-Chen, and Min Sun. "CLA-NeRF: Category-Level Articulated Neural Radiance Field". In: *arXiv preprint arXiv:2202.00181* (2022).

[202] Danny Driess, Zhiao Huang, Yunzhu Li, Russ Tedrake, and Marc Toussaint. "Learning multi-object dynamics with compositional neural radiance fields". In: *arXiv preprint arXiv:2202.11855* (2022).

[203] Haoping Xu, Yi Ru Wang, Sagi Eppel, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. "Seeing Glass: Joint Point Cloud and Depth Completion for Transparent Objects". In: *CoRR* abs/2110.00087 (2021). arXiv: 2110.00087.

[204] Thomas Weng, Amith Pallankize, Yimin Tang, Oliver Kroemer, and David Held. "Multi-modal Transfer Learning for Grasping Transparent and Specular Objects". In: *CoRR* abs/2006.00028 (2020). arXiv: 2006.00028.

[205] Qiyu Dai, Yan Zhu, Yiran Geng, Ciyu Ruan, Jiazhao Zhang, and He Wang. *GraspNeRF: Multiview-based 6-DoF Grasp Detection for Transparent and Specular Objects Using Generalizable NeRF*. 2022.

[206] Abhishek Kar, Christian Häne, and Jitendra Malik. "Learning a Multi-View Stereo Machine". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017.

[207] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. "NICE-SLAM: Neural Implicit Scalable Encoding for SLAM". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022.

[208] Alex Yu, Sara Fridovich-Keil, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. "Plenoxels: Radiance Fields without Neural Networks". In: *CoRR* abs/2112.05131 (2021). arXiv: `2112.05131`.

[209] Vishal Satish, Jeffrey Mahler, and Ken Goldberg. "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks". In: 4.2 (2019), pp. 1357–1364.

[210] Xupeng Zhu, Dian Wang, Ondrej Biza, Guanang Su, Robin Walters, and Robert Platt. "Sample Efficient Grasp Learning Using Equivariant Models". In: *arXiv preprint arXiv:2202.09468* (2022).

[211] Yu Zheng and Wen-Han Qian. "Coping with the grasping uncertainties in force-closure analysis". In: *The international journal of robotics research* 24.4 (2005), pp. 311–327.

[212] Jeffrey Mahler, Sachin Patil, Ben Kehoe, Jur Van Den Berg, Matei Ciocarlie, Pieter Abbeel, and Ken Goldberg. "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)*. IEEE. 2015, pp. 4919–4926.

[213] Chung Min Kim, Michael Danielczuk, Isabella Huang, and Ken Goldberg. "Simulation of Parallel-Jaw Grasping using Incremental Potential Contact Models". In: *arXiv preprint arXiv:2111.01391* (2021).

[214] Matthew Matl. *Pyrender*. `https://github.com/mmatl/pyrender`. 2019.

[215] Michel Breyer, Jen Jen Chung, Lionel Ott, Roland Siegwart, and Juan I. Nieto. "Volumetric Grasping Network: Real-time 6 DOF Grasp Detection in Clutter". In: *CoRR* abs/2101.01132 (2021). arXiv: `2101.01132`.

[216] Hermann Mayer, Faustino Gomez, Daan Wierstra, Istvan Nagy, Alois Knoll, and Jürgen Schmidhuber. "A system for robotic heart surgery that learns to tie knots using recurrent neural networks". In: *Advanced Robotics* 22.13-14 (2008), pp. 1521–1537.

[217] Jose Sanchez, Juan-Antonio Corrales, Belhassen-Chedli Bouzgarrou, and Youcef Mezouar. "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey". In: *The International Journal of Robotics Research* 37.7 (2018), pp. 688–716.

[218] Jur Van Den Berg, Stephen Miller, Daniel Duckworth, Humphrey Hu, Andrew Wan, Xiao-Yu Fu, Ken Goldberg, and Pieter Abbeel. "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 2074–2081.

[219] Vainavi Viswanath, Kaushik Shivakumar, Justin Kerr, Brijen Thananjeyan, Ellen Novoseller, Jeffrey Ichnowski, Alejandro Escontrela, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. "Autonomously Untangling Long Cables". In: *Robotics: Science and Systems (RSS)* (2022).

[220] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[221] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, et al. "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[222] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021.

[223] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. "Combining self-supervised learning and imitation for vision-based rope manipulation". In: *2017 IEEE Int. Conf. on Robotics and Automation (ICRA)*. IEEE. 2017, pp. 2146–2153.

[224] Jan Matas, Stephen James, and Andrew J Davison. "Sim-to-real reinforcement learning for deformable object manipulation". In: *Conference on Robot Learning*. PMLR. 2018, pp. 734–743.

[225] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and Pieter Abbeel. "Learning to manipulate deformable objects without demonstrations". In: *Robotics: Science and Systems (RSS)* (2020).

[226] Robert Lee, Daniel Ward, Akansel Cosgun, Vibhavari Dasagi, Peter Corke, and Jurgen Leitner. "Learning arbitrary-goal fabric folding with one hour of real robot experience". In: *Conference on Robot Learning* (2020).

[227] Daniel Seita, Aditya Ganapathi, Ryan Hoque, Minho Hwang, Edward Cen, Ajay Kumar Tanwani, Ashwin Balakrishna, Brijen Thananjeyan, Jeffrey Ichnowski, Nawid Jamali, et al. "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 9651–9658.

[228] Daniel Seita, Pete Florence, Jonathan Tompson, Erwin Coumans, Vikas Sindhwani, Ken Goldberg, and Andy Zeng. "Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4568–4575.

[229] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. "Learning predictive representations for deformable objects using contrastive estimation". In: *Conference on Robot Learning*. PMLR. 2021, pp. 564–574.

[230] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph E Gonzalez, and Ken Goldberg. "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9411–9418.

[231] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph E Gonzalez, Nawid Jamali, et al. "Learning dense visual correspondences in simulation to smooth and fold real fabrics". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11515–11522.

[232] Harry Zhang, Jeffrey Ichnowski, Daniel Seita, Jonathan Wang, Huang Huang, and Ken Goldberg. "Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 4560–4567.

[233] Vincent Lim, Huang Huang, Lawrence Yunliang Chen, Jonathan Wang, Jeffrey Ichnowski, Daniel Seita, Michael Laskey, and Ken Goldberg. "Real2Sim2Real: Self-Supervised Learning of Physical Single-Step Dynamic Actions for Planar Robot Casting". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8282–8289.

[234] Yahav Avigal, Lars Berscheid, Tamim Asfour, Torsten Kröger, and Ken Goldberg. "SpeedFolding: Learning Efficient Bimanual Folding of Garments". In: *International Conference on Intelligent Robots and Systems (IROS) 2022* (2022).

[235] Lawrence Yunliang Chen, Huang Huang, Ellen Novoseller, Daniel Seita, Jeffrey Ichnowski, Michael Laskey, Richard Cheng, Thomas Kollar, and Ken Goldberg. "Efficiently Learning Single-Arm Fling Motions to Smooth Garments". In: *International Symposium on Robotics Research*. 2022.

[236] Peter R Florence, Lucas Manuelli, and Russ Tedrake. "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation". In: *Conf. on Robot Learning (CoRL)*. 2018.

[237] Aditya Ganapathi, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Daniel Seita, Jennifer Grannen, Minho Hwang, Ryan Hoque, Joseph E Gonzalez, Nawid Jamali, et al. "Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021.

[238] Angelina Wang, Thanard Kurutach, Kara Liu, Pieter Abbeel, and Aviv Tamar. "Learning robotic manipulation through visual planning and acting". In: *Robotics: Science and Systems (RSS)* (2019).

[239] Vainavi Viswanath, Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, Jeffrey Ichnowski, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. "Disentangling Dense Multi-Cable Knots". In: *IEEE- /RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021).

[240] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Minho Hwang, Vainavi Viswanath, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. "Untangling dense knots by learning task-relevant keypoints". In: *Conference on Robot Learning* (2020).

[241] Wen Hao Lui and Ashutosh Saxena. "Tangled: Learning to untangle ropes with RGB-D perception". In: *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE. 2013, pp. 837–844.

[242] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen Novoseller, Minho Hwang, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. "Untangling dense non-planar knots by learning manipulation features and recovery policies". In: *Proc. Robotics: Science and Systems (RSS)* (2021).

[243] Kenneth Y Goldberg and Ruzena Bajcsy. "Active touch and robot perception". In: *Cognition and Brain Theory* 7.2 (1984), pp. 199–214.

[244] Tonci Novkovic, Remi Pautrat, Fadri Furrer, Michel Breyer, Roland Siegwart, and Juan Nieto. "Object finding in cluttered scenes using interactive perception". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8338–8344.

[245] Constantine J Tsikos and Ruzena K Bajcsy. "Segmentation via manipulation". In: *Technical Reports (CIS)* (1988), p. 694.

[246] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. "Mechanical search: Multi-step retrieval of a target object occluded by clutter". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 1614–1621.

[247] Bryan Willimon, Stan Birchfield, and Ian Walker. "Classification of clothing using interactive perception". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1862–1868.

[248] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[249] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CVPR* (2015).

[250] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2011.

[251] Jihong Zhu, Andrea Cherubini, Claire Dune, David Navarro-Alarcon, Farshid Alambeigi, D Berenson, Fanny Ficuciello, Kensuke Harada, Jens Kober, Xiang Li, et al. "Challenges and Outlook in Robotic Manipulation of Deformable Objects". In: *IEEE Robotics and Automation Magazine* (2021).

[252] Andreas Doumanoglou, Jan Stria, Georgia Peleka, Ioannis Mariolis, Vladimir Petrik, Andreas Kargakos, Libor Wagner, Václav Hlaváč, Tae-Kyun Kim, and Sotiris Malassiotis. "Folding clothes autonomously: A complete pipeline". In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1461–1478.

[253] Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding". In: *IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 2308–2315.

[254] Lawrence Yunliang Chen, Huang Huang, Ellen Novoseller, Daniel Seita, Jeffrey Ichnowski, Michael Laskey, Richard Cheng, Thomas Kollar, and Ken Goldberg. "Efficiently Learning Single-Arm Fling Motions to Smooth Garments". In: *arXiv e-prints* (2022), arXiv–2206.

[255] Christian Bersch, Benjamin Pitzer, and Sören Kammel. "Bimanual robotic cloth manipulation for laundry folding". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 1413–1419.

[256] Christian Smith, Yiannis Karayiannidis, Lazaros Nalpantidis, Xavi Gratal, Peng Qi, Dimos V Dimarogonas, and Danica Kragic. "Dual arm manipulation—A survey". In: *Robotics and Autonomous systems* 60.10 (2012), pp. 1340–1353.

[257] Aaron Edsinger and Charles C Kemp. "Two arms are better than one: A behavior based control system for assistive bimanual manipulation". In: *Recent progress in robotics: Viable robotic service to human*. Springer, 2007, pp. 345–355.

[258] Irene Garcia-Camacho, Martina Lippi, Michael C Welle, Hang Yin, Rika Antonova, Anastasiia Varava, Julia Borras, Carme Torras, Alessandro Marino, Guillem Alenya, et al. "Benchmarking bimanual cloth manipulation". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1111–1118.

[259] Li Sun, Gerarado Aragon-Camarasa, Paul Cockshott, Simon Rogers, and J Paul Siebert. "A heuristic-based approach for flattening wrinkled clothes". In: *Conference Towards Autonomous Robotic Systems*. Springer. 2013, pp. 148–160.

[260] Bryan Willimon, Stan Birchfield, and Ian Walker. "Model for unfolding laundry using interactive perception". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 4871–4876.

[261] Devin J Balkcom and Matthew T Mason. "Robotic origami folding". In: *The International Journal of Robotics Research* 27.5 (2008), pp. 613–627.

[262] Kenta Tanaka, Yusuke Kamotani, and Yasuyoshi Yokokohji. "Origami folding by a robotic hand". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2007, pp. 2540–2547.

[263] Daisuke Tanaka, Solvi Arnold, and Kimitoshi Yamazaki. "Emd net: An encode–manipulate–decode network for cloth manipulation". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 1771–1778.

[264] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4238–4245.

[265] Lars Berscheid, Pascal Meißner, and Torsten Kröger. "Robot learning of shifting objects for grasping in cluttered environments". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 612–618.

[266] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Vainavi Viswanath, Michael Laskey, Joseph Gonzalez, and Ken Goldberg. "Untangling Dense Knots by Learning Task-Relevant Keypoints". In: *Conference on Robot Learning*. PMLR. 2021, pp. 782–800.

[267] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[268] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[269] Lars Berscheid, Pascal Meißner, and Torsten Kröger. "Self-supervised learning for precise pick-and-place without object model". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4828–4835.

[270] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. "Transporter Networks: Rearranging the Visual World for Robotic Manipulation". In: *Conference on Robot Learning*. PMLR. 2021, pp. 726–747.

[271] *How to Fold a T-Shirt in Two Seconds*. `https://www.wikihow.com/Fold-a-T-Shirt-in-Two-Seconds`. Accessed: 2022-03-01.

[272] S Gliessman, M Altieri, et al. "Polyculture cropping has advantages". In: *California Agriculture* 36.7 (1982), pp. 14–16.

[273] Matt Liebman. "Polyculture cropping systems". In: *Agroecology*. CRC Press, 2018, pp. 205–218.

[274] Timothy E Crews, Wim Carton, and Lennart Olsson. "Is the future of agriculture perennial? Imperatives and opportunities to reinvent agriculture by shifting from annual monocultures to perennial polycultures". In: *Global Sustainability* 1 (2018).

[275] James W Jones, Gerrit Hoogenboom, Cheryl H Porter, Ken J Boote, William D Batchelor, LA Hunt, Paul W Wilkens, Upendra Singh, Arjan J Gijsman, and Joe T Ritchie. "The DSSAT cropping system model". In: *European journal of agronomy* 18.3-4 (2003), pp. 235–265.

[276] Pasquale Steduto, Theodore C Hsiao, Dirk Raes, and Elias Fereres. "AquaCrop—The FAO crop model to simulate yield response to water: I. Concepts and underlying principles". In: *Agronomy Journal* 101.3 (2009), pp. 426–437.

[277] Kelly R Thorp and Kevin F Bronson. "A model-independent open-source geospatial tool for managing point-based environmental model simulations at multiple spatial locations". In: *Environmental modelling & software* 50 (2013), pp. 25–36.

[278] Nived Chebrolu, Thomas Läbe, and Cyrill Stachniss. "Spatio-Temporal Non-Rigid Registration of 3D Point Clouds of Plants". In: ().

[279] Christian Damgaard, Jacob Weiner, and Hisae Nagashima. "Modelling individual growth and competition in plant populations: growth curves of Chenopodium album at two densities". In: *Journal of Ecology* 90.4 (2002), pp. 666–671.

[280] William J Price, Bahman Shafii, and Donald C Thill. "An individual-plant growth simulation model for quantifying plant competition". In: (1994).

[281] Uta Berger, Cyril Piou, Katja Schiffers, and Volker Grimm. "Competition among plants: concepts, individual-based modelling approaches, and a proposal for a future research strategy". In: *Perspectives in Plant Ecology, Evolution and Systematics* 9.3-4 (2008), pp. 121–135.

[282] T Czárán and S Bartha. "The effect of spatial pattern on community dynamics; a comparison of simulated and field data". In: *Progress in theoretical vegetation science*. Springer, 1990, pp. 229–239.

[283] Marius Wiggert, Leela Amladi, Ron Berenstein, Stefano Carpin, Joshua Viers, Stavros Vougioukas, and Ken Goldberg. "RAPID-MOLT: A Meso-scale, Open-source, Low-cost Testbed for Robot Assisted Precision Irrigation and Delivery". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2019, pp. 1489–1496.

[284] Novian Habibie, Aditya Murda Nugraha, Ahmad Zaki Anshori, M. Anwar Ma'sum, and Wisnu Jatmiko. "Fruit mapping mobile robot on simulated agricultural area in Gazebo simulator using simultaneous localization and mapping (SLAM)". In: (2017).

[285] Eric Rohmer, Surya PN Singh, and Marc Freese. "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework". In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. 2013.

[286] K. R. Aravind and P. Raja. "Design and Simulation of Crop Monitoring Robot for Green House". In: (2016).

[287] Hans Peter Koelewijn. "Rapid change in relative growth rate between the vegetative and reproductive stage of the life cycle in Plantago coronopus". In: *New phytologist* 163.1 (2004), pp. 67–76.

[288] Andrew Keller. "Evapotranspiration and crop water productivity: making sense of the yield-ET relationship". In: *Impacts of Global Climate Change*. 2005, pp. 1–11.

[289] Theodore C Hsiao. "Effects of drought and elevated CO 2 on plant water use efficiency and productivity". In: *Interacting stresses on plants in a changing climate*. Springer, 1993, pp. 435–465.

[290] LG Firbank and AR Watkinson. "A model of interference within plant monocultures". In: *Journal of Theoretical Biology* 116.2 (1985), pp. 291–311.

[291] Anupama Goyal, Elizabeth Karayekov, Vinicius Costa Galvão, Hong Ren, Jorge J Casal, and Christian Fankhauser. "Shade promotes phototropism through phytochrome B-controlled auxin production". In: *Current Biology* 26.24 (2016), pp. 3280–3287.

[292] Moritoshi Iino, Chen Long, and Xiaojing Wang. "Auxin-and abscisic acid-dependent osmoregulation in protoplasts of Phaseolus vulgaris pulvini". In: *Plant and Cell Physiology* 42.11 (2001), pp. 1219–1227.

[293] Argyris Zardilis, Alastair Hume, and Andrew J Millar. "A multi-model framework for the Arabidopsis life cycle". In: *Journal of experimental botany* 70.9 (2019), pp. 2463–2477.

[294] Tadaki Hirose, Toshihiko Kinugasa, and Yukinori Shitaka. "Time of flowering, costs of reproduction, and reproductive output in annuals". In: *Reproductive allocation in plants*. Elsevier, 2005, pp. 159–188.

[295] Peter John Lumsden and Andrew J Millar. *Biological rhythms and photoperiodism in plants*. Bios Scientific Publishers, 1998.

[296] J C Van Dam, J Huygen, JG Wesseling, RA Feddes, P Kabat, PEV Van Walsum, P Groenendijk, and CA Van Diepen. *Theory of SWAP version 2.0; Simulation of water flow, solute transport and plant growth in the soil-water-atmosphere-plant environment*. Tech. rep. DLO Winand Staring Centre, 1997.

[297] David Tseng, David Wang, Carolyn Chen, Lauren Miller, William Song, Joshua Viers, Stavros Vougioukas, Stefano Carpin, Juan Aparicio Ojea, and Ken Goldberg. "Towards automating precision irrigation: Deep learning to infer local soil moisture conditions from synthetic aerial agricultural images". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2018, pp. 284–291.

[298] Stephen J Risch. "Intercropping as cultural pest control: prospects and limitations". In: *Environmental Management* 7.1 (1983), pp. 9–14.

[299] Sven Erik Jorgensen and Brian D Fath. *Encyclopedia of ecology*. Newnes, 2014.

[300] Fusuo Zhang and Long Li. "Using competitive and facilitative interactions in intercropping systems enhances crop productivity and nutrient-use efficiency". In: *Plant and soil* 248.1-2 (2003), pp. 305–312.

[301] NA Bogie, R Bayala, I Diedhiou, RP Dick, and TA Ghezzehei. "Intercropping with two native woody shrubs improves water status and development of interplanted groundnut and pearl millet in the Sahel". In: *Plant and soil* 435.1-2 (2019), pp. 143–159.

[302] Teja Tscharntke, Yann Clough, Shonil A Bhagwat, Damayanti Buchori, Heiko Faust, Dietrich Hertel, Dirk Hölscher, Jana Juhrbandt, Michael Kessler, Ivette Perfecto, et al. "Multifunctional shade-tree management in tropical agroforestry landscapes–a review". In: *Journal of Applied Ecology* 48.3 (2011), pp. 619–629.

[303] Todd S Rosenstock, Daniel Liptzin, Kristin Dzurella, Anna Fryjoff-Hung, Allan Hollander, Vivian Jensen, Aaron King, George Kourakos, Alison McNally, G Stuart Pettygrove, et al. "Agriculture's contribution to nitrate contamination of Californian groundwater (1945–2005)". In: *Journal of Environmental Quality* 43.3 (2014), pp. 895–907.

[304] Alberto Mantovani. *Pesticide risk assessment: European framework shows need for safer alternatives*. 2019. URL: `https://www.openaccessgovernment.org/pesticide-risk-assessment/79226/` (visited on 12/11/2019).

[305] David Tilman, Kenneth G Cassman, Pamela A Matson, Rosamond Naylor, and Stephen Polasky. "Agricultural sustainability and intensive production practices". In: *Nature* 418.6898 (2002), pp. 671–677.

[306] Yahav Avigal, Jensen Gao, William Wong, Kevin Li, Grady Pierroz, Fang Shuo Deng, Mark Theis, Mark Presten, and Ken Goldberg. "Simulating Polyculture Farming to Tune Automation Policies for Plant Diversity and Precision Irrigation". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE. 2020, pp. 238–245.

[307] Jonathan Minchin. *ROMI: Robotics for Microfarming*. 2020. URL: `https://www.openaccessgovernment.org/romi-robotics-for-microfarming/80533/` (visited on 01/14/2020).

[308] BOWERY FARMING INC. *Bowery farming*. 2020. URL: `https://boweryfarming.com/` (visited on 10/15/2020).

[309] Joseph Santarromana Ken Goldberg. *The Telegarden*. 1995. URL: `https://goldberg.berkeley.edu/garden/Ars/` (visited on 12/11/2019).

[310] Ken Goldberg. *The Robot in the Garden: Telerobotics and Telepistemology in the Age of the Internet*. Mit Press, 2001.

[311] Sandunika Fernando, Ranusha Nethmi, Ashen Silva, Ayesh Perera, Rajitha De Silva, and Pradeep WK Abeygunawardhana. "AI Based Greenhouse Farming Support System with Robotic Monitoring". In: *2020 IEEE REGION 10 CONFERENCE (TENCON)*. IEEE. 2020, pp. 1368–1373.

[312] TjeerdJan Stomph, Christos Dordas, Alain Baranger, Joshua de Rijk, Bei Dong, Jochem Evers, Chunfeng Gu, Long Li, Johan Simon, Erik Steen Jensen, et al. "Designing intercrops for high yield, yield stability and efficient use of resources: Are there principles?" In: *Advances in Agronomy*. Vol. 160. 1. Elsevier, 2020, pp. 1–50.

[313] Joshua E Whitman, Harshal Maske, Hassan A Kingravi, and Girish Chowdhary. "Evolving Gaussian Processes and Kernel Observers for Learning and Control in Spatiotemporally Varying Domains: With Applications in Agriculture, Weather Monitoring, and Fluid Dynamics". In: *IEEE Control Systems Magazine* 41.1 (2021), pp. 30–69.

[314] Fang Gou, Martin K van Ittersum, and Wopke van der Werf. "Simulating potential growth in a relay-strip intercropping system: model description, calibration and testing". In: *Field Crops Research* 200 (2017), pp. 122–142.

[315] Meixiu Tan, Fang Gou, Tjeerd Jan Stomph, Jing Wang, Wen Yin, Lizhen Zhang, Qiang Chai, and Wopke van der Werf. "Dynamic process-based modelling of crop growth and competitive water extraction in relay strip intercropping: Model development and application to wheat-maize intercropping". In: *Field Crops Research* 246 (2020), p. 107613.

[316] Yang Yu. "Crop yields in intercropping: meta-analysis and virtual plant modelling". PhD thesis. Wageningen University, 2016.

[317] FarmBot. *FarmBot*. 2020. URL: `https://farm.bot/` (visited on 10/26/2020).

[318] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[319] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[320] Craig W Whippo and Roger P Hangarter. "Phototropism: bending towards enlightenment". In: *The Plant Cell* 18.5 (2006), pp. 1110–1119.

[321] Daniela Dietrich. "Hydrotropism: how roots search for water". In: *Journal of experimental botany* 69.11 (2018), pp. 2759–2771.

[322] *METER Environment*. METER. URL: https://www.metergroup.com/environment/ (visited on 10/29/2020).

[323] Brenda B. Lin, Monika H. Egerer, Heidi Liere, Shalene Jha, and Stacy M. Philpott. "Soil management is key to maintaining soil moisture in urban gardens facing changing climatic conditions". In: *Scientific Reports* 8.1 (Dec. 3, 2018). Number: 1 Publisher: Nature Publishing Group, p. 17565.

[324] Pinetree Garden Seeds. *Pinetree Garden Seeds - Vegetable Collections*. 2020. URL: https://www.superseeds.com/ (visited on 10/15/2020).

[325] Katarzyna Adamczewska-Sowińska and Józef Sowiński. "Polyculture Management: A Crucial System for Sustainable Agriculture Development". In: *Soil Health Restoration and Management*. Springer, 2020, pp. 279–319.

[326] J.E. Parker, W.E. Snyder, G.C. Hamilton, and Cesar Rodriguez-Saona. "Companion planting and insect pest control". In: *Weed and Pest Control - Conventional and New Challenges* (Jan. 2013), pp. 1–30.

[327] Aaron L. Iverson, Linda E. Marín, Katherine K. Ennis, David J. Gonthier, Benjamin T. Connor-Barrie, Jane L. Remfert, Bradley J. Cardinale, and Ivette Perfecto. "REVIEW: Do polycultures promote win-wins or trade-offs in agricultural ecosystem services? A meta-analysis". In: *Journal of Applied Ecology* 51.6 (2014), pp. 1593–1602. eprint: https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/1365-2664.12334.

[328] S Yogesh, SI Halikatti, SM Hiremath, MP Potdar, SI Harlapur, H Venkatesh, et al. "Light use efficiency, productivity and profitability of maize and soybean intercropping as influenced by planting geometry and row proportion." In: *Karnataka Journal of Agricultural Sciences* 27.1 (2014), pp. 1–4.

[329] Weizheng Ren, Liangliang Hu, Jian Zhang, Cuiping Sun, Jianjun Tang, Yongge Yuan, and Xin Chen. "Can positive interactions between cultivated species help to sustain modern agriculture?" In: *Frontiers in Ecology and the Environment* 12.9 (2014), pp. 507–514. eprint: https://esajournals.onlinelibrary.wiley.com/doi/pdf/10.1890/130162.

[330] David L. Dunn, ed. *Ethicon Wound Closure Manual*. Ethicon, Incorporated, 1994.

[331] Daegu Son and Aram Harijan. "Overview of surgical scar prevention and management". In: *Journal of Korean medical science* 29.6 (2014), p. 751.

[332] Nick Marsidi, Sofieke AM Vermeulen, Tim Horeman, and Roel E Genders. "Measuring forces in suture techniques for wound closure". In: *journal of surgical research* 255 (2020), pp. 135–143.

[333] Jean Gaston Descoux, Walley J Temple, Shirley A Huchcroft, Cyril B Frank, and Nigel G Shrive. "Linea alba closure: determination of ideal distance between sutures". In: *Journal of Investigative Surgery* 6.2 (1993), pp. 201–209.

[334] Uday Devgan. *Basic Principles of Ophthalmic Suturing*. https://cataractcoach.com/2018/07/29/basic-principles-of-ophthalmic-suturing/. Accessed: 2023-03-12.

[335] Florent Nageotte, Philippe Zanne, Christophe Doignon, and Michel De Mathelin. "Stitching planning in laparoscopic surgery: Towards robot-assisted suturing". In: *The International Journal of Robotics Research* 28.10 (2009), pp. 1303–1321.

[336] John Schulman, Jonathan Ho, Cameron Lee, and Pieter Abbeel. "Generalization in robotic manipulation through the use of non-rigid registration". In: *Proceedings of the 16th International Symposium on Robotics Research (ISRR)*. 2013.

[337] Siddarth Sen, Animesh Garg, David V Gealy, Stephen McKinley, Yiming Jen, and Ken Goldberg. "Automating multi-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization". In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 4178–4185.

[338] Azad Shademan, Ryan S Decker, Justin D Opfermann, Simon Leonard, Axel Krieger, and Peter CW Kim. "Supervised autonomous robotic soft tissue surgery". In: *Science translational medicine* 8.337 (2016), 337ra64–337ra64.

[339] Sahba Aghajani Pedram, Peter Ferguson, Ji Ma, Erik Dutson, and Jacob Rosen. "Autonomous suturing via surgical robot: An algorithm for optimal selection of needle diameter, shape, and path". In: *2017 IEEE International conference on robotics and automation (ICRA)*. IEEE. 2017, pp. 2391–2398.

[340] Russell C Jackson, Viraj Desai, Jean P Castillo, and M Cenk Çavuşoğlu. "Needle-tissue interaction force state estimation for robotic surgical suturing". In: *2016 IEEE-/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2016, pp. 3659–3664.

[341] Sahba Aghajani Pedram, Changyeob Shin, Peter Walker Ferguson, Ji Ma, Erik P Dutson, and Jacob Rosen. "Autonomous suturing framework and quantification using a cable-driven surgical robot". In: *IEEE Transactions on Robotics* 37.2 (2020), pp. 404–417.

[342] Brijen Thananjeyan, Ajay Tanwani, Jessica Ji, Danyal Fer, Vatsal Patel, Sanjay Krishnan, and Ken Goldberg. "Optimizing robot-assisted surgery suture plans to avoid joint limits and singularities". In: *2019 International Symposium on Medical Robotics (ISMR)*. IEEE. 2019, pp. 1–7.

[343] Oxford Medical Education. *Suturing techniques*. Apr. 2016. URL: https://oxfordme-%20dicaleducation.com/clinical-skills/procedures/suturing-techniques/.

[344] Dieter Kraft. "A software package for sequential quadratic programming". In: *Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt* (19-88).

[345] Brijen Thananjeyan, Justin Kerr, Huang Huang, Joseph E Gonzalez, and Ken Goldberg. "All you need is luv: Unsupervised collection of labeled images using invisible uv fluorescent indicators". In: *arXiv preprint arXiv:2203.04566* (2022).

[346] Russell C Jackson, Rick Yuan, Der-Lin Chow, Wyatt S Newman, and M Cenk Çavuşoğlu. "Real-time visual tracking of dynamic surgical suture threads". In: *IEEE Transactions on Automation science and Engineering* 15.3 (2017), pp. 1078–1090.

[347] Kaushik Shivakumar, Vainavi Viswanath, Anrui Gu, Yahav Avigal, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Ken Goldberg. "Sgtm 2.0: Autonomously untangling long cables using interactive perception". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 5837–5843.

[348] Nicolas Padoy and Gregory D Hager. "3D thread tracking for robotic assistance in tele-surgery". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 2102–2107.

[349] Neelay Joglekar, Fei Liu, Ryan Orosco, and Michael Yip. "Suture thread spline reconstruction from endoscopic images for robotic surgery with reliability-driven keypoint detection". In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2023, pp. 4747–4753.

[350] Bo Lu, Bin Li, Wei Chen, Yueming Jin, Zixu Zhao, Qi Dou, Pheng-Ann Heng, and Yunhui Liu. "Toward image-guided automated suture grasping under complex environments: A learning-enabled and optimization-based holistic framework". In: *IEEE Transactions on Automation Science and Engineering* 19.4 (2021), pp. 3794–3808.

[351] Roberto Martín-Martín and Oliver Brock. "Coupled recursive estimation for online interactive perception of articulated objects". In: *The International Journal of Robotics Research* 41.8 (2022), pp. 741–777.

[352] Prajval Kumar Murali, Anirvan Dutta, Michael Gentner, Etienne Burdet, Ravinder Dahiya, and Mohsen Kaboli. "Active visuo-tactile interactive robotic perception for accurate object pose estimation in dense clutter". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4686–4693.

[353] Peter Kazanzides, Zihan Chen, Anton Deguet, Gregory S Fischer, Russell H Taylor, and Simon P DiMaio. "An open-source research kit for the da Vinci® Surgical System". In: *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2014, pp. 6434–6439.

[354] Vainavi Viswanath, Kaushik Shivakumar, Jainil Ajmera, Mallika Parulekar, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Ken Goldberg. "Learning to Trace and Untangle Semi-planar Knots (TUSK)". In: *arXiv preprint arXiv:2303-.08975* (2023).

[355] Azarakhsh Keipour, Maryam Bandari, and Stefan Schaal. "Deformable one-dimensional object detection for routing and manipulation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 4329–4336.

[356] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. "3d gaussian splatting for real-time radiance field rendering". In: *ACM Transactions on Graphics* 42.4 (2023), pp. 1–14.