# Contact-based Perception and Planning for Robotic Manipulation in Novel Environments

by

Sheng Zhong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2024

Doctoral Committee:

Associate Professor Dmitry Berenson, Chair
Assistant Professor Nima Fazeli
Assistant Professor Maani Ghaffari
Professor Ken Goldberg

Sheng Zhong

zhsh@umich.edu

ORCID iD: 0000-0002-8658-3061

# TABLE OF CONTENTS

**LIST OF FIGURES** . . . . . v

**LIST OF TABLES** . . . . . viii

**ABSTRACT** . . . . . ix

**CHAPTER**

**I. Introduction** . . . . . 1

1.1 Introduction . . . . . 1
1.1.1 Challenges for Novel Environments . . . . . 2
1.1.2 Challenges for Using Contact . . . . . 2
1.2 Related Work . . . . . 3
1.3 Contributions . . . . . 4

**II. Adaptation to Novel Environments and Escaping Traps** . . . 5

2.1 Introduction . . . . . 5
2.2 Problem Statement . . . . . 6
2.3 Related Work . . . . . 8
2.4 Methods . . . . . 10
2.4.1 Offline: Invariant Representation for Dynamics . . . 10
2.4.2 Online: Trap-Aware MPC . . . . . 11
2.5 Experiments . . . . . 15
2.5.1 Experiment Environments . . . . . 15
2.5.2 Offline: Learning Invariant Representations . . . . . 16
2.5.3 Online: Tasks in Novel Environments . . . . . 17
2.6 Discussion . . . . . 19
2.6.1 Task Performance Analysis . . . . . 19
2.6.2 Ablation Studies . . . . . 20
2.7 Additional Details . . . . . 20
2.7.1 Environment details . . . . . 20
2.7.2 Representation learning & GP . . . . . 21
2.8 Conclusion . . . . . 21

**III. Tracking Contact Points in Cluttered Environments** . . . . . 25

3.1 Introduction . . . . . 26
3.2 Related Work . . . . . 27
3.3 Problem Statement . . . . . 29
3.4 Method . . . . . 29
3.4.1 Contact Detection and Isolation . . . . . 30
3.4.2 Soft Tracking . . . . . 30
3.4.3 Segmenting into Objects . . . . . 33
3.5 Results . . . . . 34
3.5.1 Baselines . . . . . 35
3.5.2 Training Set for Tuning . . . . . 35
3.5.3 Blind Object Retrieval . . . . . 37
3.5.4 Blind Object Retrieval in Simulation . . . . . 38
3.5.5 Real Robot Blind Object Retrieval . . . . . 40
3.6 Discussion . . . . . 41
3.6.1 Generalizing Beyond Object Translation . . . . . 41
3.6.2 Generalization Beyond Single Point Contacts . . . . 41
3.6.3 Rummaging Policy . . . . . 41
3.7 Conclusion . . . . . 42

**IV. Diverse Plausible Object Registration** . . . . . 43

4.1 Introduction . . . . . 43
4.2 Related Work . . . . . 46
4.3 Problem Statement . . . . . 47
4.4 Method . . . . . 48
4.4.1 Relaxation of Semantic Constraints . . . . . 49
4.4.2 SDF Query Improvements . . . . . 51
4.4.3 Quality Diversity Optimization . . . . . 51
4.4.4 Online Updates to $\hat{\mathcal{T}}$ . . . . . 53
4.5 Results . . . . . 53
4.5.1 Simulated Environment . . . . . 56
4.5.2 Real Environment . . . . . 57
4.5.3 Computing Plausible Set . . . . . 58
4.5.4 Baselines . . . . . 58
4.5.5 Ablations . . . . . 59
4.5.6 Probing Experiments . . . . . 59
4.5.7 QD Method Comparison . . . . . 60
4.5.8 QD Objective Comparison . . . . . 61
4.5.9 CVO Performance . . . . . 63
4.6 Discussion . . . . . 63
4.7 Conclusion . . . . . 64

**V. Rummaging Using Mutual Information** . . . . . 65

5.1 Introduction . . . . . 65
5.2 Related Work . . . . . 67
5.2.1 Representing Belief . . . . . 68
5.2.2 Information Gain . . . . . 68
5.2.3 Planning . . . . . 69
5.3 Problem Statement . . . . . 70
5.4 Method . . . . . 72
5.4.1 Representing Pose Posterior . . . . . 73
5.4.2 Mutual Information Surrogate . . . . . 74
5.4.3 Illustrative Example . . . . . 77
5.4.4 Posterior Update Process . . . . . 78
5.4.5 Planning Problem . . . . . 81
5.4.6 Information Gain Cost . . . . . 83
5.4.7 Dynamics Model . . . . . 84
5.4.8 Reachability Cost . . . . . 85
5.4.9 Kernel Interpolated MPPI . . . . . 88
5.4.10 Termination Condition . . . . . 89
5.5 Experiments . . . . . 90
5.5.1 Sim Environment . . . . . 90
5.5.2 Sim Tasks . . . . . 91
5.5.3 Real Environment . . . . . 93
5.5.4 Real Task . . . . . 93
5.5.5 Sensor Model . . . . . 94
5.5.6 KMPPI Parameters . . . . . 95
5.5.7 Evaluation . . . . . 95
5.5.8 Baselines and Ablations . . . . . 97
5.5.9 Results . . . . . 100
5.5.10 Runtime Comparison . . . . . 101
5.6 Discussion and Future Work . . . . . 103
5.6.1 Single Object Assumption . . . . . 103
5.6.2 Unknown Object Shape . . . . . 104
5.7 Conclusion . . . . . 105

**BIBLIOGRAPHY** . . . . . 105

# LIST OF FIGURES

**Figure**


| | | |
|---|---|---|
| 2.1 | TAMPC on peg-in-hole tasks with obstacles near the goal | 7 |
| 2.2 | Architecture for learning (left) and using (right) an invariant representation $\hat{f}$ | 10 |
| 2.3 | High-level control architecture. | 12 |
| 2.4 | Annotated simulation environments of (left) planar pushing, and (right) peg-in-hole. | 15 |
| 2.5 | Learning curves on validation and OOD data sets for planar pushing representation. | 17 |
| 2.6 | (top) Initial condition and (bottom) typical traps for planar pushing and peg-in-hole tasks. | 17 |
| 2.7 | Minimum distance to goal after 500 steps (300 for Real Peg-T) accounting for walls (computed with Djikstra's algorithm). | 18 |
| 3.1 | Initial (top) and after rummaging (bot) cluttered environment with STUCCO allowing us to successfully estimate the pose of the cracker box and grasp it without visual perception. | 26 |
| 3.2 | Prediction (a-c) and update (d) step of one particle. | 30 |
| 3.3 | Training environments in simulation. | 36 |
| 3.4 | Simulated BOR task in 4 different environments | 37 |
| 3.5 | Tracking metrics evaluated on the training set | 38 |
| 3.6 | Iterative closest point pose estimates from 30 random starts plotted in green. | 39 |
| 3.7 | Steps during a BOR run with (left) initially wrong associations of contact points to objects, (mid) moving to right before entering the gap between the objects, and (right) resolving the previous ambiguity from moving through the gap and penalizing particles with points in between. | 39 |
| 4.1 | Left: Set up of a real-world probing experiment where the goal is to estimate the drill's pose. | 44 |
| 4.2 | $\hat{C}(\mathcal{X}, \mathbf{T})$ negative gradients with respect to sampled points shown for a X-Z cross section of the YCB drill | 48 |
| 4.3 | Real probing configurations for the drill and mustard. | 55 |
| 4.4 | Simulated probing configurations for multiple YCB objects. | 55 |
| 4.5 | Unreliable visual perception | 56 |

4.6 Plausible Diversity for real drill (left) and mustard (right) probing experiments across 2 configurations and 6 trials each . . . . . 56
4.7 Plausible Diversity for simulated probing experiments across different YCB objects . . . . . 57
4.8 Reducing uncertainty in estimated pose as a result of additional free space points for selected methods . . . . . 60
4.9 Average time per registration of 30 transforms on the real probing experiments . . . . . 61
4.10 Comparison of QD optimization progress using $\hat{C}$ on the real drill experiment . . . . . 62
4.11 Plausible Diversity on the two real mustard bottle experiments with a focus on the improvement the Medial Constraint baseline receives from QD optimization. . . . . 62
4.12 Plausible Diversity on the two real mustard bottle experiments with a focus on the improvement the CVO baseline receives from ignoring $\mathcal{X}_f$. . . . . 63
5.1 (a) Set up of a real-world active exploration experiment where the goal is to estimate the pose of a movable target object. . . . . 66
5.2 Example sensor model that gives a probability of observing each semantics class given a SDF value. . . . . 72
5.3 Flow chart showing one time step of RUMI's approach for solving Eq. 5.1. . . . . 72
5.4 (left) Example mug object with (middle) $\mathcal{X}$ rendered from a pinhole camera on one side . . . . . 76
5.5 Computed information gain field $\tilde{I}(\mathbf{x}|\mathcal{X})$ from Fig. 5.4 with (left) $N = 30$, (middle) $N = 100$, and (right) $N = 1000$. . . . . 77
5.6 Resampling during a pybullet simulated mug task with partial initial observation like in Fig. 5.4. . . . . 78
5.7 (left) Reachability of workspace positions for the (right) simulated KUKA arm and workspace . . . . . 87
5.8 Planned trajectories on a toy 2D linear integrator environment . . . 89
5.9 Robot interior and information gathering points . . . . . 90
5.10 Starting poses for labelled sim tasks. . . . . 91
5.11 Comparison of the rendered $\mathcal{X}_0$ given to (left) the sim mug 0 task and (right) sim box 2 task. . . . . 92
5.12 Initial configuration of the real mug and box tasks (top) and example initialized pose particles (bottom). . . . . 94
5.13 Convergence of pose particles measured by $APC(\mathbf{T}_{1..N})$ for the sim mug 0 and sim mug 1 tasks . . . . . 97
5.14 (left) Gaussian Process fit to initial observations $\mathcal{X}_0$ of the sim mug 0 task . . . . . 98
5.15 Comparison of the fields to plan over evaluated at each $\mathbf{x} \in \mathcal{W}$ for (left) our method using $\tilde{I}(\mathbf{x}|\mathcal{X})$ against the (right) GPVR baseline using $\mathbf{var}(\mathbf{x}|\mathcal{X})$. . . . . 101

5.16 $nll(\mathcal{X})$ after each execution step for simulation tasks depicted in Fig. 5.10. . . . . 102
5.17 $nll(\mathcal{X})$ after each execution step for the (left) real mug and (right) real box tasks depicted in Fig. 5.12. . . . . 103
5.18 Average computation time per execution step of the (top) planar sim mug 0 task and the (bot) 3D sim box 0 task with one standard deviation as the error bar . . . . . 104

# LIST OF TABLES

**Table**


| | | |
|---|---|---|
| 2.1 | Success counts across all tasks | 23 |
| 2.2 | MPPI parameters for different environments. | 23 |
| 2.3 | TAMPC parameters across environments. | 23 |
| 2.4 | Goal cost parameters for each environment. | 24 |
| 3.1 | STUCCO parameters for Blind Object Retrieval. | 36 |
| 3.2 | Quantitative comparison of our method against baselines on 20 runs of blind object retrieval in different simulated cluttered environments and 5 runs on the real environment in Fig. 3.1. | 40 |
| 4.1 | $\epsilon$ used to generate the plausible set for each objects. | 58 |
| 5.1 | Task setup for different objects. | 94 |
| 5.2 | Linear correlation between $nll(\mathcal{X})$ and $APC(\mathbf{T}_{1..N})$ across all sim tasks | 96 |
| 5.3 | Maximum $nll$ threshold for success for each task. | 96 |
| 5.4 | Our full method parameters across the different tasks. | 99 |
| 5.5 | Number of successful trials after 10 runs of active rummaging for pose estimation in different tasks in Fig. 5.10 and Fig. 5.12. | 106 |
| 5.6 | Median cumulative $nll(\mathcal{X})$ for 10 runs of active rummaging for pose estimation in different tasks in Fig. 5.10 and Fig. 5.12. | 106 |

# ABSTRACT

This thesis proposes a framework for autonomous robotic perception and planning for manipulation tasks in unknown environments by leveraging information from purposeful contacts and explicitly reasoning about uncertainty. The high-level goal is to enhance the amount of information that can be extracted from contacts, enabling greater utilization of this sensing modality. We focus on challenging tasks where objects to be manipulated are occluded by the environment, other objects, or themselves, which limits the applicability of purely visual sensing and necessitates contact-based information gathering.

Each chapter of our work tackles a specific challenge arising from collecting information through contact, with the goal of enabling robots to explore autonomously. The first challenge we considered is the limited applicability of long-horizon planning when global perception is lacking. Traps may arise where the state remains in a cycle without accomplishing the goal, and we develop a hierarchical control scheme to detect and escape from traps.

Contact-based exploration is also challenging due to the ambiguity of associating contact points to specific objects in multi-object environments. To resolve this, we present a method that maintains a belief over both current and past contact points without relying on rigid associations. This flexibility allows for the correction of erroneous estimates.

Building on these contact point estimates, we infer the plausible poses of known objects. A key component in our method is the use of negative information—data indicating observed free space—which constrains possible object poses by measuring the discrepancy between these potential poses and the observed point clouds. This approach is especially effective in highly-occluded environments where visual object segmentation often fails.

To integrate our pose estimates into real-time decision-making, we formulate a conditional probability on object poses given the disparity with observed point clouds.

We derive a cost function from the mutual information between the object's pose and the occupancy of the workspace points, facilitating its application in closed-loop model predictive control (MPC). Our method also includes a reachability cost function to prevent objects from being pushed out of the robot's workspace and incorporates a stochastic dynamics model to predict information gain changes as the object is manipulated.

The algorithms developed in this thesis emphasize efficient parallel computation and are evaluated using both simulated and real experiments. All implementations are made publicly available as open-source libraries.

# CHAPTER I

# Introduction

## 1.1 Introduction

A major trend in modern robotics is to move away from static, controlled environments such as factories towards dynamic, unknown environments such as homes. This is exemplified in the recent Amazon Picking Challenge, where robots competed to perceive, pick, and place household items from well-lit shelves. Despite having fore-knowledge of the environment and the relatively controlled conditions, competitors noted that occlusion was a major limitation to their visual perception *Morrison et al.* (2018) *Zeng et al.* (2022). In home environments, occlusion and poor lighting conditions, as well as the presence of unknown objects confound vision systems (e.g. cameras and LiDARs). To overcome visual limitations, robots can use contact-rich interactions to gather information.

This thesis focuses on how to effectively use the information gathered through contacts to perceive objects and to plan for their manipulation. We first distinguish contact from tactile information. We use *tactile* to refer to any information collected from touch, including the presence of contact at a point or patch (contact information), texture, and temperature. As most robots are not equipped with specialized tactile sensors, this thesis focuses on using only contact information. We consider tasks where visual perception is significantly occluded, missing, or can only be used as a weak prior and where contact is an essential source of information. For example, rummaging inside a cabinet cluttered with objects presents many occlusions that limit visual perception and necessitates making contact to determine the location of a target object. We restrict our consideration to rigid household objects, such as those from the YCB dataset *Calli et al.* (2017). Knowing the position of contact points for a rigid object can be sufficient to localize the object, but this is not true for deformable

objects due to their increased degrees of freedom. Thus the methods in this thesis are designed to apply to rigid-body localization and manipulation tasks.

### 1.1.1 Challenges for Novel Environments

Deploying a robot to novel environments requires adapting to new dynamics and an overall more reactive approach. Rather than generating start-to-goal trajectories formed using classic motion planning techniques, creating finite horizon trajectories with model predictive control may be more appropriate. However, dynamics accuracy, particularly in contact-rich tasks, is often limited outside of the experienced trajectory. A limited control horizon combined with unknown dynamics can often lead to poor local minima, intuitively termed "traps" that are developed more rigorously in Chapter II.

### 1.1.2 Challenges for Using Contact

There are many challenges to using contact information, contributing to vision as the preferred and predominant sensor modality. These include the coupling of perception and manipulation, and the sparse and local nature of the information collected from contact.

The coupling refers to how each contact potentially changes the environment as objects are moved. Apart from the object directly in contact, other objects may be indirectly moved. Sensors may not be sensitive enough to detect contact without simultaneously moving the object.

Despite advances in tactile sensors such as the Soft-bubble *Alspach et al.* (2019) and the Gelsight *Yuan et al.* (2017) sensors, by nature the robot only makes contact with one part of an object at a time. Conversely, if the robot makes contact with more than one object, as is common in cluttered environments, it is challenging to associate contacts with objects, and to further track this association through extended interaction. Being able to perceive only a local part of an object is a significant challenge for traditional pose estimation algorithms such as iterative closest point (ICP) and its variants *Bouaziz et al.* (2013).

The challenges from having sparse and local information is amplified by the fact that gathering information via contact may move the object. To address these challenges, we explicitly reason about and represent the uncertainty of quantities we estimate, including contact point positions and object poses. We develop efficient methods to evaluate, predict, and update our beliefs.

We assume the robot can accurate detect external contacts and localize them to positions on the surface of the robot. This is either sensed directly such as through the Soft-bubble, or from estimated applied external wrench at the end effector origin and applying the Contact Particle Filter *Koval et al.* (2015). Applied external wrench can be measured directly with a wrist-mounted force-torque sensor, or estimated by comparing the measured joint torques against expected torques.

## 1.2 Related Work

In this section, I broadly review recent work related to the high level goal of my thesis. More detailed related works are reviewed in each chapter for the specific challenges and methods therein.

Much work has been done on contact modelling and learning specific tactile tasks, such as in-hand manipulation *Andrychowicz et al.* (2020), and planar-pushing *Bauza and Rodriguez* (2017). However, in these cases the robot or object to be manipulated is well known, and the interaction occurs in isolation. Perception is often given (and performed through vision) while the focus is on the manipulation task.

For works more closely related to our goal of contact-based object perception, they often assume the object is stationary throughout the interaction *Driess et al.* (2019); *Xu et al.* (2022); *Suresh et al.* (2022). This critically enables their methods to trivially combine observations from multiple contacts. On the task of object recognition, many work assume additional sensor capabilities, such as detecting texture and temperature *Kaboli and Cheng* (2016); *Liu et al.* (2017). With another assumption of having a controller that can slide along the object's surface, work has been done on shape exploration while implicitly modelling the shape as a Gaussian Process *Driess et al.* (2017); *Suresh et al.* (2021).

On the other hand, work has also been done on relaxing assumptions. In particular, for robots that do not have specialized tactile sensors, the Contact Particle Filter *Manuelli and Tedrake* (2016) localizes contact to points on the robot surface from experienced external wrench. In a situation with an unknown number of objects and unknown association of contact to objects, the probability hypothesis density (PHD) filter *Vo and Ma* (2006) has been developed to track objects across time, albeit with dense sensor information such as vision and sonar.

The methods we develop most closely align with the interactive perception *Bohg et al.* (2017) problem, where perception is intrinsically tied to manipulating the environment itself.

## 1.3 Contributions

This thesis will make the following contributions, each focused on a specific challenge presented above and in their own chapter:

- [Completed] Chapter II TAMPC: a hierarchical controller that detects traps, escapes traps, and prevents re-entry of traps on novel environments.
- [Completed] Chapter III STUCCO: an algorithm for propagating the belief of contact point positions over across a sequence of contacts.
- [Completed] Chapter IV CHSEL: an algorithm for estimating object poses given free space information that characterizes the pose uncertainty.
- [Completed] Chapter V RUMI: an information theoretic rummaging method to estimate object pose based on the mutual information between the pose and the robot trajectory.

Collectively, these contributions provide a robust approach to overcoming the limitations of visual perception and contact perception in dynamic environments, enhancing the robot's ability to interact with and manipulate movable objects.

# CHAPTER II

# Adaptation to Novel Environments and Escaping Traps

This chapter considers the problem of adapting dynamics to novel environments for contact-rich tasks. It first considers how a state representation can generalize outside of its training data by extracting invariances inherent to the dynamics. Then it considers "traps" that inevitably arises from finite-horizon planning with partially-known dynamics. Specifically, how to detect traps, escape traps, and avoid re-entering them in the future.

## 2.1 Introduction

In this chapter, we study the problem of controlling robots in environments with unforeseen **traps**. Informally, traps are states in which the robot's controller fails to make progress towards its goal and gets "stuck". Traps are common in robotics and can arise due to many factors including geometric constraints imposed by obstacles, frictional locking effects, and nonholonomic dynamics leading to dropped degrees of freedom *Borenstein et al.* (2005); *Fantoni and Lozano* (2012); *Koditschek et al.* (2004). In this chapter, we consider instances of trap dynamics in planar pushing with walls and peg-in-hole with unmodeled obstructions to the goal.

Developing generalizable algorithms that rapidly adapt to handle the wide variety of traps encountered by robots is important to their deployment in the real-world. Two central challenges in online adaptation to environments with traps are the data-efficiency requirements and the lack of progress towards the goal for actions inside of traps. In this chapter, our key insight is that we can address these challenges by explicitly reasoning over different dynamic modes, in particular traps, together with contingent recovery policies, organized as a hierarchical controller. We introduce an

online modeling and controls method that balances naive optimism and pessimism when encountering novel dynamics. Our method learns a dynamics representation that infers underlying invariances and exploits it when possible (optimism) while treading carefully to escape and avoid potential traps in non-nominal dynamics (pessimism). Specifically, we:

1. Introduce a novel representation learning approach that effectively generalizes dynamics and show its efficacy for task execution on out-of-distribution data when used with our proposed controller;

2. Introduce Trap-Aware Model Predictive Control (TAMPC), a novel control algorithm that reasons about non-nominal dynamics and traps to reach goals in novel environments with traps;

3. Evaluate our method on real robot and simulated peg-in-hole, and simulated planar pushing tasks with traps where adaptive control and reinforcement learning baselines achieve 0% success rate. These include difficult tasks where trap-handling baselines achieve less than 50% success, while our method achieves at least 60% success on all tasks.

We show that state-of-the-art techniques *Fu et al.* (2016); *Haarnoja et al.* (2018), while capable of adapting to novel dynamics, are insufficient for escaping traps that our approach handles by their explicit consideration. Additionally, our method performs well on tasks that prior trap-handling methods struggle on.

## 2.2 Problem Statement

Let $\mathbf{x} \in \mathcal{X}$ and $\mathbf{u} \in \mathcal{U}$ denote the $N_x$ dimensional state and $N_u$ dimensional control. Under test conditions the system follows novel dynamics $\Delta\mathbf{x} = f_v(\mathbf{x}, \mathbf{u})$. The objective of our method is to reach a goal $\mathbf{x} \in \mathcal{G} \subset \mathcal{X}$ as quickly as possible:

$$\begin{aligned} \underset{\mathbf{u}_0,\ldots,\mathbf{u}_{T-1}}{\arg\min} \quad & T \\ \text{s.t.} \quad & \mathbf{x}_{t+1} = \mathbf{x}_t + f_v(\mathbf{x}_t, \mathbf{u}_t),\ t = 0, ..., T-1 \\ & \mathbf{x}_T \in \mathcal{G} \end{aligned} \tag{2.1}$$

This problem is difficult because the novel dynamics $f_v$ are not known. Instead, we assume we have access to a dataset of sampled transitions with random actions, which can be used to learn an approximate dynamics model $\hat{f}$ of the system under

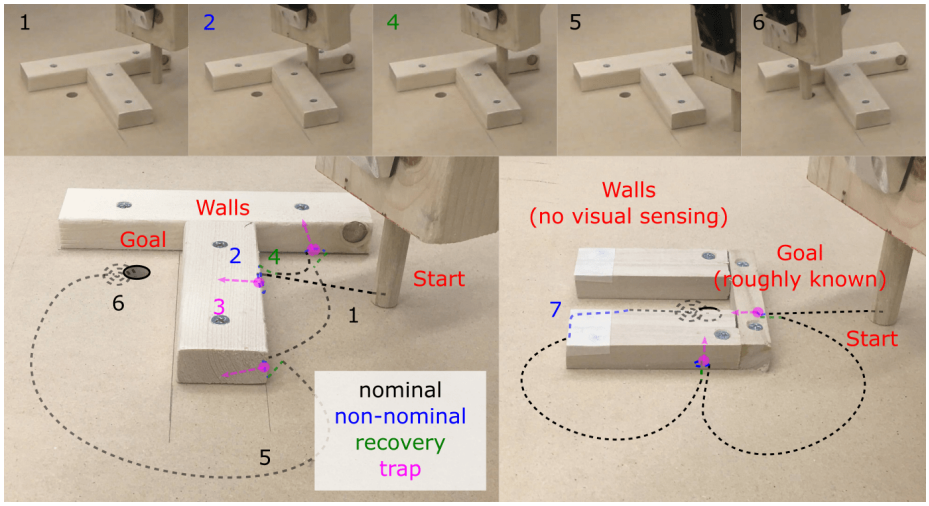


Figure 2.1: TAMPC on peg-in-hole tasks with obstacles near the goal. The robot has no visual sensing (cannot anticipate walls) and has not encountered walls in the training data. Path segments show (1) the initial direct trajectory to goal, (2) detecting non-nominal dynamics from reaction forces and exploration of it by sliding along the wall, (3) detecting a trap due to the inability to make progress using non-nominal dynamics, (4) recovery to nominal dynamics, (5) going around seen traps to goal, (6) spiraling to find the precise location of the hole, and (7) sliding against the wall (non-nominal) towards the goal.

*nominal* dynamics $f$ where:

$$f_v(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}, \mathbf{u}) + e(\mathbf{x}, \mathbf{u}) \tag{2.2}$$

We assume the error dynamics $e$ are relatively small (w.r.t. nominal) except for non-nominal regions $\bar{\mathcal{X}} \subset \mathcal{X}$ for which:

$$|e(\mathbf{x}, \mathbf{u})| \sim |f(\mathbf{x}, \mathbf{u})|, \qquad \forall \mathbf{x} \in \bar{\mathcal{X}}, \;\; \exists\, \mathbf{u} \in \mathcal{U} \tag{2.3}$$

In nominal regions where $\hat{f} \approx f \approx f_v$ a Model Predictive Controller (MPC) can provide high quality solutions without direct access to $f_v$. Using a specified cost function $C : \mathcal{X} \times \mathcal{U} \to [0, \infty)$, following the MPC policy $\mathbf{u} = \texttt{MPC}(\mathbf{x}, \hat{f}, C)$ creates the dynamical system:

$$\Delta \mathbf{x} = f_v(\mathbf{x}, \texttt{MPC}(\mathbf{x}, \hat{f}, C)) \tag{2.4}$$

This dynamical system may have attractors *Milnor* (1985), which are subsets $A \subseteq \mathcal{X}$ where:

- $\mathbf{x}_{t_0} \in A \implies \mathbf{x}_t \in A \quad \forall t \geq t_0$

- $A$ has a basin of attraction
  $B(A) \subseteq \mathcal{X} = \{\mathbf{x} \mid \mathbf{x}_0 = \mathbf{x},\ \exists\ t \geq 0,\ \mathbf{x}_t \in A\}$
- $A$ has no non-empty subset with the first two properties

We define a *trap* as an attractor $A$ such that $A \cap \mathcal{G} = \emptyset$, and the *trap set* $\mathcal{T} \subseteq \mathcal{X}$ as the union of all traps. Escaping a trap requires altering the dynamical system (Eq. 2.4) by altering $\hat{f}$, $C$, or the `MPC` function. Traps can be avoided in nominal regions with a sufficiently-powerful and long-horizon MPC, and a sufficiently-accurate dynamics approximation $\hat{f}$. This work addresses detecting, recovering from, and avoiding traps caused by non-nominal dynamics. To aid in trap detection, recovery, and avoidance we assume a state distance function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ and a control similarity function $s : \mathcal{U} \times \mathcal{U} \to [0, 1]$ are given.

## 2.3 Related Work

In this section, we review related work to the two main components of this chapter: handling traps and generalizing models to out-of-distribution (OOD) dynamics.

**Handling Traps:** Traps can arise due to many factors including nonholonomic dynamics, frictional locking, and geometric constraints *Borenstein et al.* (2005); *Fantoni and Lozano* (2012); *Koditschek et al.* (2004). In particular, they can occur when the environment is partially unknown, as in the case of online path planning.

Traps have been considered in methods based on Artificial Potential Fields (APF) *Khatib* (1986); *Lee and Park* (2003). Traps are local minima in the potential field under the policy of following the gradient of the field. In the case where the environment is only partially known *a priori*, local minima escape (LME) methods such as *Lee and Park* (2003); *Fedele et al.* (2018) can be used to first detect then escape local minima. Our method uses similar ideas to virtual obstacles (VO) *Lee and Park* (2003) for detecting traps and avoiding revisits while addressing weaknesses common to APF-LME methods. Specifically, we are able to handle traps near goals by associating actions with trap states (using $s$) to penalize similar actions to the one previously entering the trap while near it, rather than penalize being near the trap altogether. We also avoid blocking off paths to the goal with virtual obstacles by shrinking their effect over time. Lastly, having an explicit recovery policy and using a controller that plans ahead lets us more efficiently escape traps with "deep" basins (many actions are required to leave the basin). We compare against two APF-LME methods in our experiments.

Another way to handle traps is with exploration, such as through intrinsic curiosity (based on model predictability) *Pathak et al.* (2017); *Burda et al.* (2018), state counting *Bellemare et al.* (2016), or stochastic policies *Osband et al.* (2016). However, trap dynamics can be difficult to escape from and can require returning to dynamics the model predicts well (so receives little exploration reward). We show in an ablation test how random actions are insufficient for escaping traps in some tasks we consider. Similar to *Ecoffet et al.* (2019), we remember interesting past states. While they design domain-specific state interest scores, we effectively allow for online adaptation of the state score based on how much movement the induced policy generates while inside a trap. We use this score to direct our recovery policy.

Adapting to trap dynamics is another possible approach. Actor-critic methods have been used to control nonlinear systems with unknown dynamics online *Dierks and Jagannathan* (2012), and we evaluate this approach on our tasks. Another approach is with locally-fitted models which *Fu et al.* (2016) showed could be mixed with a global model and used in MPC. Similar to this approach, our method adapts a nominal model to local dynamics; however, we do not always exploit the dynamics to reach the goal.

**Generalizing models to OOD Dynamics:** One goal of our method is to generalize the nominal dynamics to OOD novel environments. A popular approach for doing this is explicitly learning to be robust to expected variations across training and test environments. This includes methods such as meta-learning *Finn et al.* (2017); *Li et al.* (2018), domain randomization *Tobin et al.* (2017); *Pan et al.* (2010), Sim-to-real *Peng et al.* (2018), and other transfer learning *Zhang et al.* (2018) methods. These methods are unsuitable for this problem because our training data contains only nominal dynamics, whereas they need a diverse set of non-nominal dynamics. Instead, we learn a robust, or "disentangled" representation *Chen et al.* (2016) of the system under which models can generalize. This idea is active in computer vision, where learning based on invariance has become popular *Krueger et al.* (2020). Using similar ideas, we present a novel architecture for learning invariant representations for dynamics models.

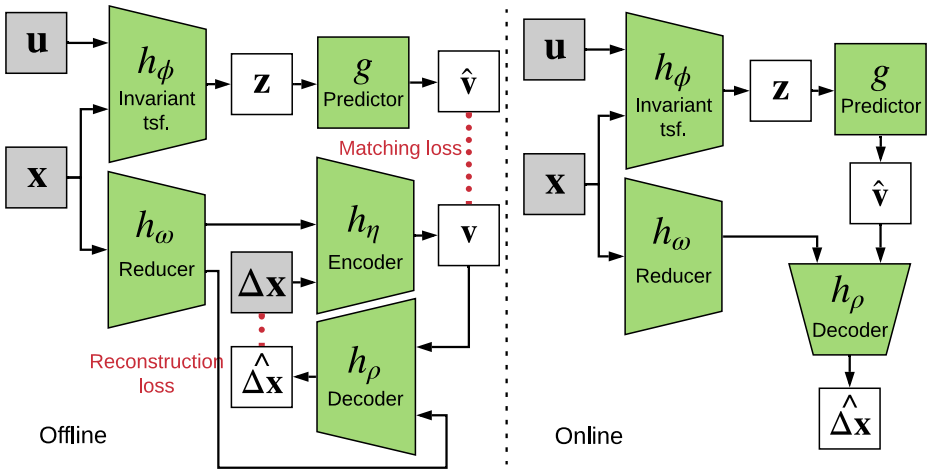


Figure 2.2: Architecture for learning (left) and using (right) an invariant representation $\hat{f}$. Grey = given data, green = parameterized transforms, white = computed values, and red dotted lines = losses.

## 2.4 Methods

Our approach is composed of two components: offline representation learning and online control. First, we present how we learn a representation that allows for generalization by exploiting inherent invariances inferred from the nominal data, shown in Fig. 2.2. Second, we present Trap-Aware Model Predictive Control (**TAMPC**), a two-level hierarchical MPC method shown in Fig. 2.3. The high-level controller explicitly reasons about non-nominal dynamics and traps, deciding when to exploit the dynamics and when to recover to familiar ones by outputting the model and cost function the low-level controller uses to compute control signals.

### 2.4.1 Offline: Invariant Representation for Dynamics

In this section, our objective is to learn $\hat{f}$ while exploiting potential underlying invariances in $\mathcal{X} \times \mathcal{U}$ to achieve better generalization to unseen data. More formally, our representation consists of an invariant transform $h_\phi$ and a predictive module $g$, shown in Fig. 2.2. $h_\phi$ maps $\mathcal{X} \times \mathcal{U}$ to a latent space ($\mathbf{z} \in \mathbb{R}^{N_z}$) that $g$ maps to latent output ($\mathbf{v} \in \mathbb{R}^{N_v}$) that is then mapped back to $\mathcal{X}$ using $h_\rho$. We parameterize the transforms with neural networks and build in two mechanisms to promote meaningful latent spaces:

First, we impose $N_z < N_x + N_u$ to create an information bottleneck which encourages $\mathbf{z}$ to ignore information not relevant for predicting dynamics. Typically, $N_z$ can be iteratively decreased until predictive performance on $\boldsymbol{\tau}$ drops significantly com-

pared to a model in the original space. Further, we limit the size of $g$ to be smaller than that of $h_\phi$ such that the dynamics take on a simple form.

Second, to reduce compounding errors when $\mathbf{x}, \mathbf{u}$ is OOD, we partially decouple $\mathbf{z}$ and $\mathbf{v}$ by learning $\mathbf{v}$ in an autoencoder fashion from $\mathbf{x}, \Delta\mathbf{x}$ with encoder $h_\eta$ and decoder $h_\rho$. Our innovation is to match the encoded $\mathbf{v}$ with the $\hat{\mathbf{v}}$ output from the dynamics predictor. To further improve generalization, we restrict information passed from $\mathbf{x}$ to $\mathbf{v}$ with a dimension reducing transform $h_\omega : \mathcal{X} \to \mathbb{R}^n_\omega$. These two mechanisms yield the following expressions:

$$\Delta\mathbf{x} \approx \hat{\Delta\mathbf{x}} = h_\rho(\mathbf{v}, h_\omega(\mathbf{x})) \qquad \mathbf{v} = h_\eta(\Delta\mathbf{x}, h_\omega(\mathbf{x})) \approx \hat{\mathbf{v}} = g(\mathbf{z})$$

and their associated batch reconstruction and matching loss:

$$\mathcal{L}_r = \frac{\mathbb{E}\left\|\Delta\mathbf{x} - h_\rho(\mathbf{v}, h_\omega(\mathbf{x}))\right\|_2}{\mathbb{E}\left\|\Delta\mathbf{x}\right\|_2} \qquad \mathcal{L}_m = \frac{\mathbb{E}\left\|\mathbf{v} - \hat{\mathbf{v}}\right\|_2}{\mathbb{E}\left\|\mathbf{v}\right\|_2}$$
$$\mathcal{L}_b(\boldsymbol{\tau}_i) = \lambda_r \mathcal{L}_r(\boldsymbol{\tau}_i) + \lambda_m \mathcal{L}_m(\boldsymbol{\tau}_i)$$

These losses are ratios relative to the norm of the quantity we are trying to match to avoid decreasing loss by scaling the representation. In addition to these two losses, we apply Variance Risk Extrapolation (V-REx *Krueger et al.* (2020)) to explicitly penalize the variance in loss across the $M$ trajectories:

$$\mathcal{L}(\boldsymbol{\tau}) = \beta\, \mathbf{var}\left\{\mathcal{L}_b(\boldsymbol{\tau}_1), ..., \mathcal{L}_b(\boldsymbol{\tau}_M)\right\} + \sum_{i=1}^{M} \mathcal{L}_b(\boldsymbol{\tau}_i) \tag{2.5}$$

We train on Eq. (2.5) using gradient descent.

After learning the transforms, we replace $g$ with a higher capacity model and fine-tune it on the nominal data with just $\mathcal{L}_m$. For further details, please see App. C. Since we have no access to $\Delta\mathbf{x}$ online, we pass $\hat{\mathbf{v}}$ to $h_\rho$ instead of $\mathbf{v}$:

$$\hat{f}(\mathbf{x}, \mathbf{u}) = h_\rho(g(h_\phi(\mathbf{x}, \mathbf{u})), h_\omega(\mathbf{x})) \tag{2.6}$$

### 2.4.2 Online: Trap-Aware MPC

Online, we require a controller that has two important properties. First, it should incorporate strategies to escape from and avoid detected traps. Second, it should iteratively improve its dynamics representation, in particular when encountering previously unseen modes. To address these challenges, our approach uses a two-level

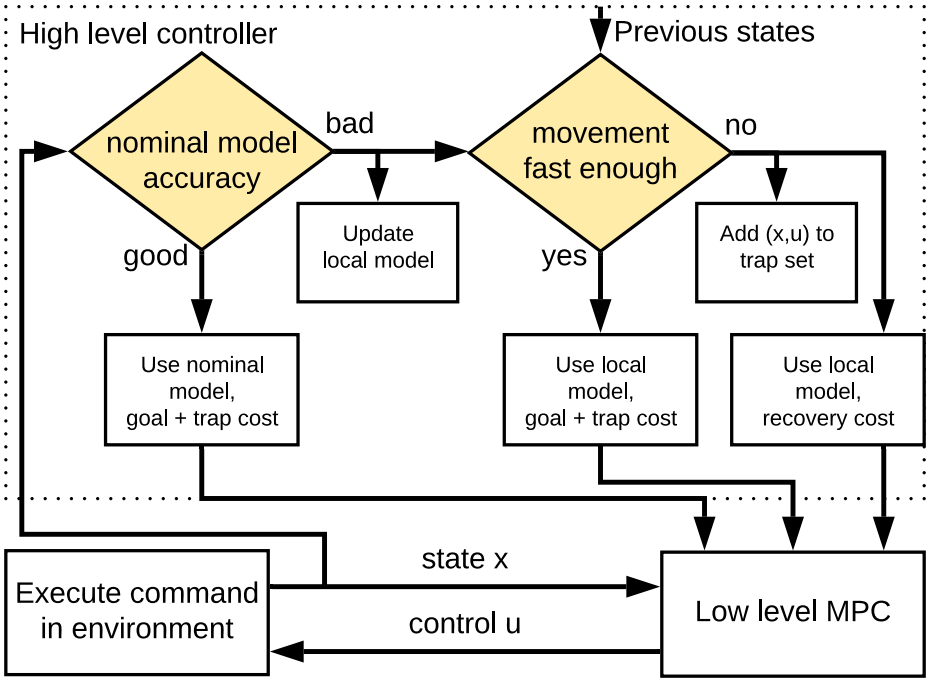


Figure 2.3: High-level control architecture.

hierarchical controller where the high-level controller is described in Alg. 1.

TAMPC operates in either an exploit or recovery mode, depending on dynamics encountered. When in nominal dynamics, the algorithm exploits its confidence in predictions. When encountering non-nominal dynamics, it attempts to exploit a local approximation built online until it detects entering a trap, at which time recovery is triggered. This approach attempts to balance between a potentially overly-conservative treatment of all non-nominal dynamics as traps and an overly-optimistic approach of exploiting all non-nominal dynamics assuming goal progress is always possible.

The first step in striking this balance is identifying non-nominal dynamics. Here, we evaluate the nominal model prediction error against observed states ("nominal model accuracy" block in Fig. 2.3 and lines 5 and 13 from Alg. 1):

$$\|(\Delta \mathbf{x} - \hat{f}(\mathbf{x}, \mathbf{u}))/E\|_2 \; > \; \epsilon_N \tag{2.7}$$

where $\epsilon_N$ is a designed tolerance threshold and $E$ is the expected model error per dimension computed on the training data. To handle jitter, we consider transitions from $N_n$ consecutive time steps.

When in non-nominal dynamics, the controller needs to differentiate between dynamics it can navigate to reach the goal vs. traps and adapt its dynamics models

accordingly. Similar to *Lee and Park* (2003), we detect this as when we are unable to make progress towards the goal by considering the latest window of states. To be robust to cost function shape, we consider the state distance instead of cost. Specifically, we monitor the maximum one-step state distance $d(\mathbf{x}_t, \mathbf{x}_{t-1})$ in nominal dynamics, $d_0$, and compare it against the average state distance to recent states: $d(\mathbf{x}_t, \mathbf{x}_a)/(t-a) < \epsilon_T d_0$ (depicted in "movement fast enough" block of Fig. 2.3). Here, $t$ is the time from task start. We consider $a = t_0, ..., t - N_d$, where $t_0$ is the start of non-nominal dynamics or the end of last recovery, whichever is more recent. We ensure state distances are measured over windows of at least size $N_d$ to handle jitter. $\epsilon_T$ is how much slower the controller tolerates moving in non-nominal dynamics. For more details see Alg. 2.

Our model adaptation strategy, for both non-nominal dynamics and traps, is to mix the nominal model with an online fitted local model. Rather than the linear models considered in prior work *Fu et al.* (2016), we add an estimate of the error dynamics $\hat{e}$ represented as a Gaussian Process (GP) to the output of the nominal model. Using a GP provides a sample-efficient model that captures non-linear dynamics. To mitigate over-generalizing local non-nominal dynamics to where nominal dynamics holds, we fit it to only the last $N_e$ points since entering non-nominal dynamics. We also avoid over-generalizing the invariance that holds in nominal dynamics by constructing the GP model in the original state-control space. Our total dynamics is then

$$\hat{f}_v(\mathbf{x}, \mathbf{u}) = \hat{f}(\mathbf{x}, \mathbf{u}) + \hat{e}(\mathbf{x}, \mathbf{u}) \tag{2.8}$$

$$f_v(\mathbf{x}, \mathbf{u}) \approx \mathbb{E}[\hat{f}_v(\mathbf{x}, \mathbf{u})] \tag{2.9}$$

When exploiting dynamics to navigate towards the goal, we regularize the goal-directed cost $C$ with a trap set cost $C_T$ to avoid previously seen traps (line 27 from Alg. 1). This trap set $\mathcal{T}_c$ is expanded whenever we detect entering a trap. We add to it the transition with the lowest ratio of actual to expected movement (from one-step prediction, $\hat{\mathbf{x}}$) since the end of last recovery:

$$b = \arg\min_a \frac{d(\mathbf{x}_a, \mathbf{x}_{a+1})}{d(\mathbf{x}_a, \hat{\mathbf{x}}_{a+1})}, \mathcal{T}_c \leftarrow \mathcal{T}_c \cup \{(\mathbf{x}_b, \mathbf{u}_b)\} \tag{2.10}$$

To handle traps close to the goal, we only penalize revisiting trap states if similar actions are to be taken. With the control similarity function $s : \mathcal{U} \times \mathcal{U} \rightarrow [0, 1]$ we

formulate the cost, similar to the virtual obstacles of *Lee and Park* (2003):

$$C_T(\mathbf{x}, \mathbf{u}) = \sum_{\mathbf{x}', \mathbf{u}' \in \mathcal{T}_c} \frac{s(\mathbf{u}, \mathbf{u}')}{d(\mathbf{x}, \mathbf{x}')^2} \tag{2.11}$$

The costs are combined as $C + \alpha C_T$ (line 27 from Alg. 1), where $\alpha \in (0, \infty)$ is annealed by $\alpha_a \in (0, 1)$ each step in nominal dynamics (line 9 from Alg. 1). Annealing the cost avoids assigning a fixed radius to the traps, which if small results in inefficiency as we encounter many adjacent traps, and if large results in removing many paths to the goal set.

We switch from exploit to recovery mode when detecting a trap, but it is not obvious what the recovery policy should be. Driven by the online setting and our objective of data-efficiency: First, we restrict the recovery policy to be one induced by running the low-level MPC on some cost function other than one used in exploit mode. Second, we propose hypothesis cost functions and consider only convex combinations of them. Without domain knowledge, one hypothesis is to return to one of the last visited nominal states. However, the dynamics may not always allow this. Another hypothesis is to return to a state that allowed for the most one-step movement. Both of these are implemented in terms of the following cost, where $S$ is a state set and we pass in either $\mathcal{X}_0$, the set of last visited nominal states, or $\mathcal{X}_f$, the set of $N_f$ states that allowed for the most single step movement since entering non-nominal dynamics:

$$C_R(\mathbf{x}, \mathbf{u}, S) = \min_{\mathbf{x}' \in S} d(\mathbf{x}, \mathbf{x}')^2 \tag{2.12}$$

Third, we formulate learning the recovery policy online as a non-stationary multi-arm bandit (MAB) problem. We initialize $N_a$ bandit arms, each a random convex combination of our hypothesis cost functions. Every $N_m$ steps in recovery mode, we pull an arm to select and execute a recovery policy. After executing $N_m$ control steps, we update that arm's estimated value with a movement reward: $d(\mathbf{x}_t, \mathbf{x}_{t-N_m})/N_m d_0$. When in a trap, we assume any movement is good, even away from the goal. The normalization makes tuning easier across environments. To accelerate learning, we exploit the correlation between arms, calculated as the cosine similarity between the cost weights. Our formulation fits the problem from *McConachie and Berenson* (2020) and we implement their framework for non-stationary correlated multi-arm bandits.

Finally, we return to exploit mode after a fixed number of steps $N_R$, if we returned to nominal dynamics, or if we stopped moving after leaving the initial trap state. For details see Alg. 3.

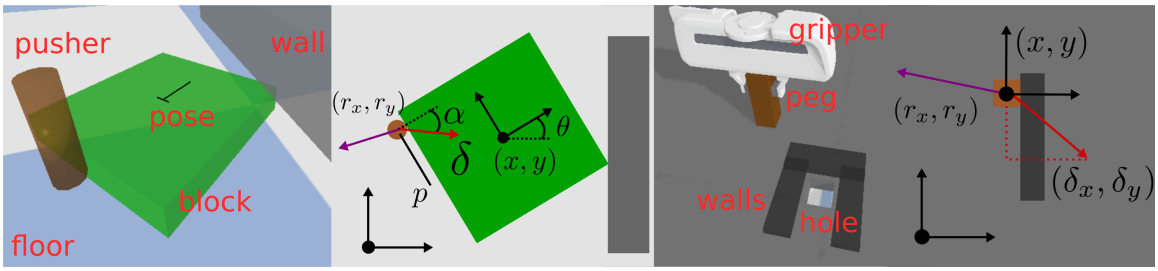


Figure 2.4: Annotated simulation environments of (left) planar pushing, and (right) peg-in-hole.

## 2.5 Experiments

In this section, we first evaluate our dynamics representation learning approach, in particular how well it generalizes out-of-distribution. Second, we compare TAMPC against baselines on tasks with traps in two environments.

### 2.5.1 Experiment Environments

Our two tasks are quasi-static planar pushing and peg-in-hole. Both tasks are evaluated in simulation using PyBullet *Coumans and Bai* (2016–2021) and the latter is additionally evaluated empirically using a 7DoF KUKA LBR iiwa arm depicted in Fig. 2.1. Our simulation time step is 1/240s, however each control step waits until reaction forces are stable. The action size for each control step is described in App B. In planar pushing, the goal is to push a block to a known desired position. In peg-in-hole, the goal is to place the peg into a hole with approximately known location. In both environments, the robot has access to its own pose and senses the reaction force at the end-effector. **Thus the robot cannot perceive the obstacle geometry visually, it only perceives contact through reaction force. During offline learning of nominal dynamics, there are no obstacles or traps**. During online task completion, obstacles are introduced in the environment, inducing unforeseen traps. See Fig. 2.4 for a depiction of the environments and Fig. 2.6 for typical traps from tasks in these environments, and App. B for environment details.

In planar pushing, the robot controls a cylindrical pusher restricted to push a square block from a fixed side. Fig. 2.6 shows traps introduced by walls. Frictional contact with a wall limits sliding along it and causes most actions to rotate the block into the wall. State is $\mathbf{x} = (x, y, \theta, r_x, r_y)$ where $(x, y, \theta)$ is the block pose, and $(r_x, r_y)$ is the reaction force the pusher feels, both in world frame. Control is $\mathbf{u} = (p, \delta, \alpha)$, where $p$ is where along the side to push, $\delta$ is the push distance, and $\alpha$

is the push direction relative to side normal. The state distance is the 2-norm of the pose, with yaw normalized by the block's radius of gyration. The control similarity is $s(\mathbf{u}_1, \mathbf{u}_2) = \max(0, \text{cossim}((p_1, \alpha_1), (p_2, \alpha_2)))$ where cossim is cosine similarity.

In peg-in-hole, we control a gripper holding a peg (square in simulation and circular on the real robot) that is constrained to slide along a surface. Traps in this environment geometrically block the shortest path to the hole. The state is $\mathbf{x} = (x, y, z, r_x, r_y)$ and control is $\mathbf{u} = (\delta_x, \delta_y)$, the distance to move in $x$ and $y$. We execute these on the real robot using a Cartesian impedance controller. The state distance is the 2-norm of the position and the control similarity is $s(\mathbf{u}_1, \mathbf{u}_2) = \max(0, \text{cossim}(\mathbf{u}_1, \mathbf{u}_2))$. The goal-directed cost for both environments is in the form $C(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u}$. The MPC assigns a terminal multiplier of 50 at the end of the horizon on the state cost. See Tab. 2.4 for the cost parameters of each environment.

$\boldsymbol{\tau}$ for simulated environments consists of $M = 200$ trajectories with $T = 50$ transitions (all collision-free). For the real robot, we use $M = 20$, $T = 30$. We generate them by uniform randomly sampling starts from $[-1, 1] \times [-1, 1]$ ($\theta$ for planar pushing is also uniformly sampled; for the real robot we start each randomly inside the robot workspace) and applying actions uniformly sampled from $\mathcal{U}$.

### 2.5.2 Offline: Learning Invariant Representations

In this section we evaluate if our representation can learn useful invariances from offline training on $\boldsymbol{\tau}$. We expect nominal dynamics in freespace in our environments to be invariant to translation. Since $\boldsymbol{\tau}$ has positions around $[-1, 1] \times [-1, 1]$, we evaluate translational invariance translating the validation set by $(10, 10)$. We evaluate relative MSE $\|\hat{\Delta\mathbf{x}} - \Delta\mathbf{x}\|_2 / \mathbb{E}\|\Delta\mathbf{x}\|_2$ (we do not directly optimize this) against a fully connected baseline of comparable size mapping $\mathbf{x}, \mathbf{u}$ to $\hat{\Delta\mathbf{x}}$ learned on relative MSE. As Fig. 2.5b shows, our performance on the translated set is better than the baseline, and trends toward the original set's performance. Note that we expect our method to have higher validation MSE since V-REx sacrifices in-distribution loss for lower variance across trajectories. We use $N_z = 5, N_v = 5, n_\omega = 2$, and implement the transforms with fully connected networks. For network sizes and learning details see App. C.

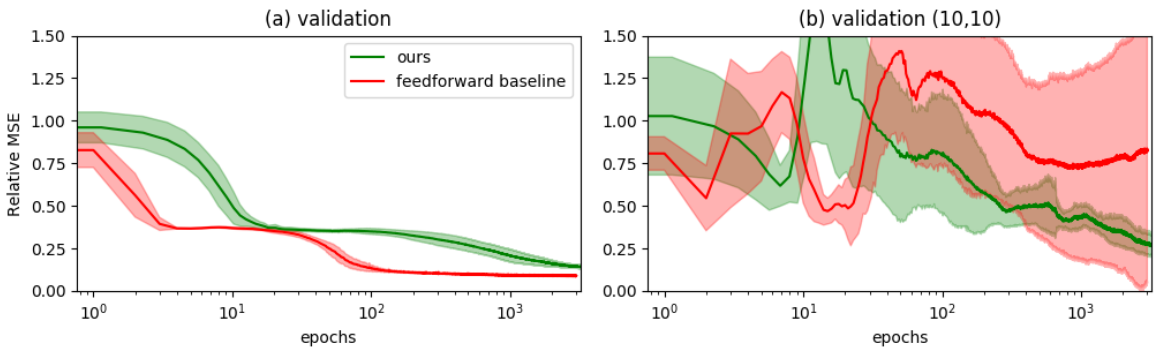


Figure 2.5: Learning curves on validation and OOD data sets for planar pushing representation. Mean across 10 runs is solid while 1 std. is shaded.

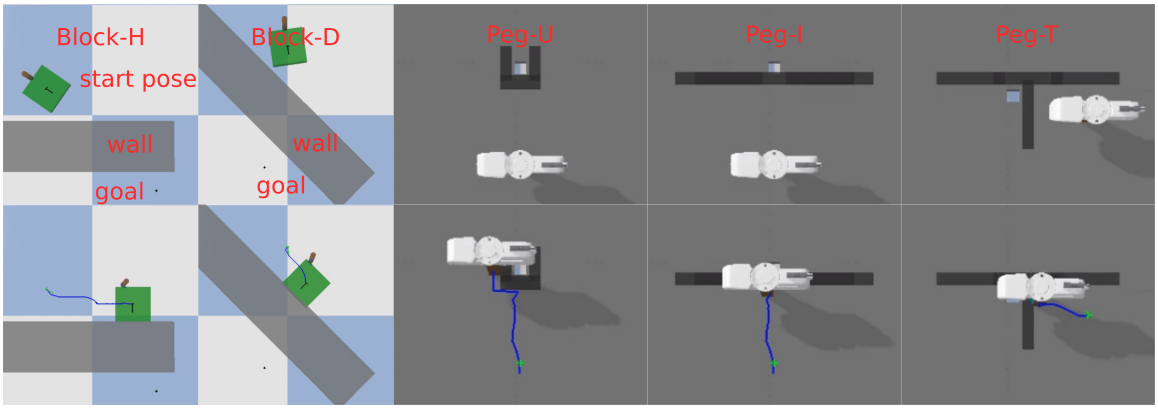


Figure 2.6: (top) Initial condition and (bottom) typical traps for planar pushing and peg-in-hole tasks. Our method has no visual sensing and is pre-trained only on environments with no obstacles.

### 2.5.3 Online: Tasks in Novel Environments

We evaluate TAMPC against baselines and ablations on the tasks shown in Fig. 2.1 and Fig. 2.6. For TAMPC's low-level MPC, we use a modified model predictive path integral controller (MPPI) *Williams et al.* (2017b) where we take the expected cost across $R = 10$ rollouts for each control trajectory sample to account for stochastic dynamics. See Alg. 4 for our modifications to MPPI. TAMPC takes less than 1s to compute each control step for all tasks. We run for 500 steps (300 for Real Peg-T).

**Baselines:** We compare against five baselines. The first is the APF-VO method from *Lee and Park* (2003), which uses the gradient on an APF to select actions. The potential field is populated with repulsive balls in $\mathcal{X}$ based on $d$ as we encounter local minima. We estimate the gradient by sampling 5000 single-step actions and feeding through $\hat{f}$. Second is an APF-LME method from *Fedele et al.* (2018) (APF-SP) using switched potentials between the attractive global potential, and a helicoid obstacle

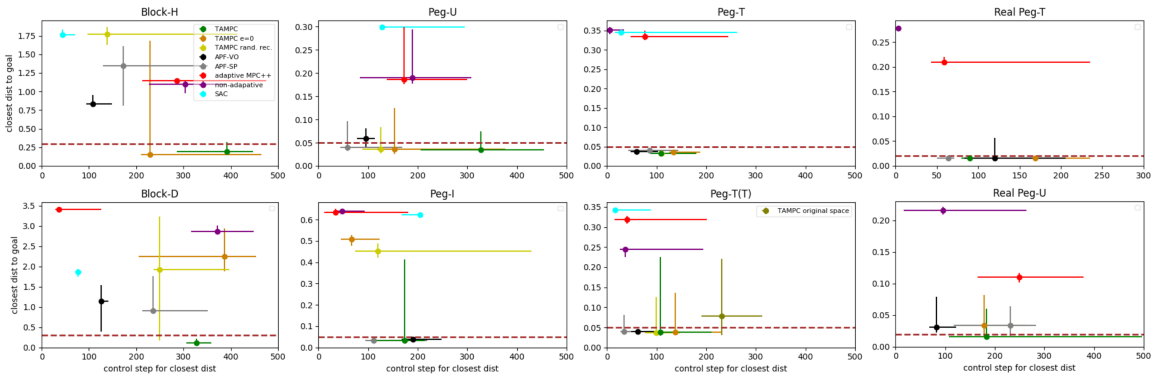


Figure 2.7: Minimum distance to goal after 500 steps (300 for Real Peg-T) accounting for walls (computed with Djikstra's algorithm). Median across 10 runs is plotted with error bars representing 20–80$^{\text{th}}$ percentile. Task success is achieving lower distance than the dotted red line.

potential, suitable for 2D obstacles. The APF methods use the $\hat{f}$ learned with our proposed method (Section 2.4.1) for next-state prediction. Third is online adaptive MPC from *Fu et al.* (2016) ("adaptive MPC++"), which does MPC on a linearized global model mixed with a locally-fitted linear model. iLQR (code provided by *Fu et al.* (2016)'s author) performs poorly in freespace of the planar pushing environment. We instead use MPPI with a locally-fitted GP model (effectively an ablation of TAMPC with control mode fixed to `NONNOM`). Next is model-free reinforcement learning with Soft Actor-Critic (SAC) *Haarnoja et al.* (2018). Here, a nominal policy is learned offline for 1000000 steps on the nominal environment, which is used to initialize the policy at test time. Online, the policy is retrained after every control on the dense environment reward. Lastly, our "non-adaptive" baseline runs MPPI on the nominal model.

We also evaluated ablations to demonstrate the value of TAMPC components. "TAMPC rand. rec." takes uniform random actions until dynamics is nominal instead of using our recovery policy. "TAMPC original space" uses a dynamics model learned in the original $\mathcal{X} \times \mathcal{U}$ (only for Peg-T(T)). Lastly, "TAMPC $e = 0$" does not estimate error dynamics.

## 2.6 Discussion

### 2.6.1 Task Performance Analysis

Fig. 2.7 and Tab. 2.1 summarizes the results. For Fig. 2.7, ideal controller performance would approach the lower left corners. We see that TAMPC outperforms all baselines on the block pushing tasks and Peg-U, with slightly worse performance on the other peg tasks compared to APF baselines. APF baselines struggled with block pushing since turning when the goal is behind the block is also a trap for APF methods, because any action increases immediate cost and they do not plan ahead. On the real robot, joint limits were handled naturally as traps by TAMPC and APF-VO.

Note that the APF baselines were tuned to each task, thus it is fair to compare against tuning TAMPC to each task. However, we highlight that our method is empirically robust to parameter value choices, as we achieve high success even when using the same parameter values across tasks and environments, listed in Tab. 2.3. Peg-U and Peg-I were difficult tasks that benefited from independently tuning *only three* important parameters, which we give intuition for: We control the exploration of non-nominal dynamics with $N_d$. For cases like Peg-U where the goal is surrounded by non-nominal dynamics, we increase exploration by increasing $N_d$ with the trade-off of staying longer in traps. Independently, we control the expected trap basin depth (steps required to escape traps) with the MPC horizon $H$. Intuitively, we increase $H$ to match deeper basins, as in Peg-I, at the cost of more computation. Lastly, $\alpha_a \in (0, 1)$ controls the trap cost annealing rate. Too low a value prevents escape from difficult traps while values close to 1 leads to waiting longer in cost function local minima. We used $N_d = 15, H = 15$ for Peg-U, and $H = 20, \alpha_a = 0.95$ for Peg-I.

For APF-VO, Peg-U was difficult as the narrow top of the U could be blocked off if the square peg caught on a corner at the entrance. In these cases, TAMPC was able to revisit close to the trap state by applying dissimilar actions to before. This was less an issue in Real Peg-U as we used a round peg, but a different complicating factor is that the walls are thinner (compared to simulation) relative to single-step action size. This meant that virtual obstacles were placed even closer to the goal. APF-SP often oscillated in Peg-U due to traps on either side of the U while inside it.

The non-TAMPC and non-APF baselines tend to cluster around the top left corner in Fig. 2.7, indicating that they entered a trap quickly and never escaped. Indeed, we see that they all never succeed. For adaptive MPC, this may be due to a combination of insufficient knowledge of dynamics around traps, over-generalization of trap dynamics, and using too short of a MPC horizon. SAC likely stays inside of traps

because no action immediately decreases cost, and action sequences that eventually decrease cost have high short-term cost and are thus unlikely to be sampled in 500 steps.

### 2.6.2 Ablation Studies

Pushing task performance degraded on both TAMPC rand. rec. and TAMPC $e = 0$, suggesting value from both the recovery policy and local error estimation. This is likely because trap escape requires long action sequences exploiting the local non-nominal dynamics to rotate the block against the wall. This is unlike the peg environment where the gripper can directly move away from the wall and where TAMPC rand. rec. performs well. Note that ablations used the parameter values in Tab. 2.3 for Peg-I and Peg-U, instead of the tuned parameters from Section 2.6.1. This may explain their decreased performance on them. The Peg-T(T) task (copy of Peg-T translated 10 units in $x, y$) highlights our learned dynamics representation. Using our representation, we maintain closer performance to Peg-T than TAMPC original space (3 successes). This is because annealing the trap set cost requires being in recognized nominal dynamics, without which it is easy to get stuck in local minima.

## 2.7 Additional Details

Making U-turns in planar pushing requires $H \geq 25$. We shorten the horizon to 5 and remove the terminal state cost of 50 when in recovery mode to encourage immediate progress.

$\rho \in (0, \infty)$ depends on how accurately we need to model trap dynamics to escape them. Increasing $\rho$ leads to selecting only the best sampled action while a lower value leads to more exploration by taking sub-optimal actions.

Nominal error tolerance $\epsilon_N$ depends on the variance of the model prediction error in nominal dynamics. A higher variance requires a higher $\epsilon_N$. We use a higher value for peg-in-hole because of simulation quirks in measuring reaction force from the two fingers gripping the peg.

### 2.7.1 Environment details

The planar pusher is a cylinder with radius 0.02m to push a square with side length $a = 0.3$m. We have $\theta \in [-\pi, \pi]$, $p \in [-a/2, a/2]$, $\delta \in [0, a/8]$, and $\alpha \in [-\pi/4, \pi/4]$.

All distances are in meters and all angles are in radians. The state distance function is the 2-norm of the pose, where yaw is normalized by the block's radius of gyration. $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + \sqrt{a^2/6}(\theta_1 - \theta_2)^2}$. The sim peg-in-hole peg is square with side length 0.03m, and control is limited to $\delta_x, \delta_y \in [0, 0.03]$. These values are internally normalized so MPPI outputs control in the range $[-1, 1]$.

### 2.7.2 Representation learning & GP

Each of the transforms is represented by 2 hidden layer multilayer perceptrons (MLP) activated by LeakyReLU and implemented in PyTorch. They each have (16, 32) hidden units except for the simple dynamics $g$ which has (16, 16) hidden units, which is replaced with (32, 32) for fine-tuning. The feedforward baseline has (16, 32, 32, 32, 16, 32) hidden units to have comparable capacity. We optimize for 3000 epochs using Adam with default settings (learning rate 0.001), $\lambda_r = \lambda_m = 1$, and $\beta = 1$. For training with V-REx, we use a batch size of 2048, and a batch size of 500 otherwise. We use the GP implementation of gpytorch with an RBF kernel, zero mean, and independent output dimensions. For the GP, on every transition, we retrain for 15 iterations on the last 50 transitions since entering non-nominal dynamics to only fit non-nominal data.

## 2.8 Conclusion

This chapter presented TAMPC, a controller that escapes traps in novel environments, and showed that it performs well on a variety of tasks with traps in simulation and on a real robot. Specifically, it is capable of handling cases where traps are close to the goal, and when the system dynamics require many control steps to escape traps. In contrast, we showed that trap-handling baselines struggle in these scenarios. Additionally, we presented and validated a novel approach to learning an invariant representation. Through the ablation studies, we demonstrated the individual value of TAMPC components: learning an invariant representation of dynamics to generalize to out-of-distribution data, estimating error dynamics online with a local model, and executing trap recovery with a multi-arm-bandit based policy. Finally, the failure of adaptive control and reinforcement learning baselines on our tasks suggests that it is beneficial to explicitly consider traps. Future work will explore higher-dimensional problems where the state space could be computationally challenging for the local GP model.

**Algorithm 1:** TAMPC high-level control loop

**Given** : $C(\mathbf{x}, \mathbf{u})$ cost, $\mathbf{x}_0$, MPC, parameters from Tab. 2.3

$s \leftarrow$ NOM, $t \leftarrow 0$, $d_0 \leftarrow 0$, $\mathbf{x} \leftarrow \mathbf{x}_0$, $\mathcal{T}_c \leftarrow \{\}$, $\alpha \leftarrow 1$, $w \leftarrow \mathbf{0}$

MAB arms $\leftarrow N_a$ random convex combs.

**while** $C(\boldsymbol{x}, 0) >$ *acceptable threshold* **do**

  **if** *s is NOM* **then**

    **if** *not nominal from Eq. 2.7* **then**

      $s \leftarrow$ NONNOM

      initialize GP $\hat{e}$ with $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \Delta\mathbf{x}_{t-1})$

    **else**

      $\alpha \leftarrow \alpha \cdot \alpha_a$ // anneal

      $d_0 \leftarrow \max(d_0, d(\mathbf{x}_{t-N_d}, \mathbf{x})/N_d)$

  **else**

    fit $\hat{e}$ to include $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}, \Delta\mathbf{x}_{t-1})$

    $n \leftarrow$ was nominal last $N_n$ steps // Eq. 2.7

    **if** EnteringTrap$(d_0)$ **then**

      $s \leftarrow$ REC

      expand $\mathcal{T}_c$ according to Eq. 2.10

    **if** *s is REC* **then**

      **if** *n* ***or*** Recovered$(d_0)$ **then**

        $s \leftarrow$ NONNOM

        $\alpha \leftarrow$ normalize $\alpha$ so $|C_T| \sim |C|$

      **else if** $N_m$ *steps since last arm pull* **then**

        reward last arm pulled with $d(\mathbf{x}_t, \mathbf{x}_{t-N_m})/N_m d_0$

        $w \leftarrow$ Thompson sample an arm

    **if** *n* **then**

      $s \leftarrow$ NOM

  MPC.model $\leftarrow \hat{f}$ **if** s is NOM **else** $\hat{f} + \hat{e}$

  MPC.cost $\leftarrow \rho \cdot (w_1 C_R(\mathcal{X}_0) + w_2 C_R(\mathcal{X}_f))$ **if** $s$ is REC **else** $C + \alpha C_T$ // Eq. 2.12 and 2.11

  $\mathbf{u} \leftarrow$ MPC$(\mathbf{x})$, $t \leftarrow t + 1$

  $\mathbf{x} \leftarrow$ apply $\mathbf{u}$ and observe from env

**Algorithm 2:** EnteringTrap

**Given:** $t_0$ time since end of last recovery or start of local dynamics, whichever is more recent, $\mathbf{x}_{t_0}, ..., \mathbf{x}_t$, $d_0$, $N_d$, $\epsilon_T$

**for** $a \leftarrow t_0$ **to** $t - N_d$ **do**

  **if** $d(\boldsymbol{x}_t, \boldsymbol{x}_a)/(t-a) < \epsilon_T d_0$ **then**

    **return** True

**return** False

Table 2.1: Success counts across all tasks, as defined by achieving distance below the thresholds in Fig. 2.7.

| Method | Success over 10 trials | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | B-H | B-D | P-U | P-I | P-T | P-T(T) | RP-T | RP-U |
| **TAMPC** | 8 | 9 | 7 | 7 | 9 | 6 | 10 | 6 |
| TAMPC e=0 | 6 | 1 | 5 | 0 | 9 | 7 | 9 | 3 |
| TAMPC rand. rec. | 1 | 4 | 5 | 0 | 9 | 7 | - | - |
| APF-VO | 0 | 1 | 4 | 10 | 10 | 10 | 8 | 2 |
| APF-SP | 1 | 0 | 5 | 10 | 10 | 8 | 10 | 4 |
| adaptive MPC++ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| non-adaptive | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SAC | 0 | 0 | 0 | 0 | 0 | 0 | - | - |

Table 2.2: MPPI parameters for different environments.

| Parameter | block | peg | real peg |
|---|---|---|---|
| $K$ samples | 500 | 500 | 500 |
| $H$ horizon | 40 | 10 | 15 |
| $R$ rollouts | 10 | 10 | 10 |
| $\lambda$ | 0.01 | 0.01 | 0.01 |
| $\mathbf{u}'$ | $[0, 0.5, 0]$ | $[0, 0]$ | $[0, 0]$ |
| $\mu$ | $[0, 0.1, 0]$ | $[0, 0]$ | $[0, 0]$ |
| $\Sigma$ | $\mathrm{diag}\,[0.2, 0.4, 0.7]$ | $\mathrm{diag}\,[0.2, 0.2]$ | $\mathrm{diag}\,[0.2, 0.2]$ |

Table 2.3: TAMPC parameters across environments.

| Parameter | block | peg | real peg |
|---|---|---|---|
| $\alpha_a$ trap cost annealing rate | 0.97 | 0.9 | 0.8 |
| $\rho$ recovery cost weight | 2000 | 1 | 1 |
| $\epsilon_N$ nominal error tolerance | 8.77 | 12.3 | 15 |
| $\epsilon_T$ trap tolerance | 0.6 | 0.6 | 0.6 |
| $N_d$ min dynamics window | 5 | 5 | 2 |
| $N_n$ nominal window | 3 | 3 | 3 |
| $N_m$ steps for bandit arm pulls | 3 | 3 | 3 |
| $N_a$ number of bandit arms | 100 | 100 | 100 |
| $N_R$ max steps for recovery | 20 | 20 | 20 |
| $N_e$ local model window | 50 | 50 | 50 |
| $\epsilon_c$ converged threshold | $0.05\epsilon_T$ | $0.05\epsilon_T$ | $0.05\epsilon_T$ |
| $\epsilon_m$ move threshold | 1 | 1 | 1 |

Table 2.4: Goal cost parameters for each environment.

| Term | block | peg | real peg |
|---|---|---|---|
| $Q$ | $\text{diag}\,[10, 10, 0, 0, 0]$ | $\text{diag}\,[1, 1, 0, 0, 0]$ | $\text{diag}\,[1, 1, 0, 0, 0]$ |
| $R$ | $\text{diag}\,[0.1, 0.1, 0.1]$ | $\text{diag}\,[1, 1]$ | $\text{diag}\,[1, 1]$ |

**Algorithm 3:** `Recovered`

**Given:** $\mathbf{x}_0, ..., \mathbf{x}_t$ since start recovery, $d_0$, parameters from Tab.2.3

**if** $t < N_d$ **then**
  **return** False
**else if** $t > N_R$ **then**
  **return** True
converged $\leftarrow d(\mathbf{x}_t, \mathbf{x}_{t-N_d})/N_d < \epsilon_c d_0$
away $\leftarrow d(\mathbf{x}_t, \mathbf{x}_0) > \epsilon_m d_0$
**return** converged **and** away

**Algorithm 4:** MPC Implementation: multi-rollout MPPI. Differences from MPPI *Zhong and Power* (2019) are highlighted.

**Given:** cost, model, $\mathbf{x}$, $\tilde{\mathbf{U}}$, Tab. 2.2 parameters

$\epsilon \leftarrow \mathcal{N}(\mu, \Sigma)$ `// u perturbation for` $H$ `steps`
$\mathbf{U}, \epsilon \leftarrow$ clip $\tilde{\mathbf{U}}$ to control bounds
$\mathbf{X}_0 \leftarrow \mathbf{x}$
$c \leftarrow 0$ `//` $R \times K$
**for** $r \leftarrow 0$ **to** $R - 1$ **do**
  **for** $t \leftarrow 0$ **to** $H - 1$ **do**
    $\mathbf{X}_{t+1} \leftarrow \text{model}(\mathbf{X}_t, \mathbf{U}_t)$ `// sample rollout`
    $c_{r,t} \leftarrow \text{cost}(\mathbf{X}_{t+1}, \mathbf{U}_t)$
    $c_r \leftarrow c_r + c_{r,t}$
$c \leftarrow$ mean $c$ across $R$
$\mathbf{U} \leftarrow$ softmax mix perturbations
**return** $\mathbf{U}$

# CHAPTER III

# Tracking Contact Points in Cluttered Environments

The previous chapter presented a hierarchical controller suitable for contact-rich tasks in novel environments. However, it is only applicable for static environments since traps are associated with state-action pairs. Environments with movable obstacles and objects are much more difficult. As introduced in Chapter 1.1.2, a significant part of the challenge in using contact information in these environments is the difficulty of associating contact points and objects when there are an unknown number of objects in the workspace.

This chapter addresses this problem by proposing to track the belief of the contact point positions without assignment of association. We make the assumption that contact points closer together are more likely to come from the same object. Based on this, we update the belief by sampling associations to any newly detected contact and applying a given dynamics model on the associated contact points. To correct for error that may arise from incorrect associations and inaccurate dynamics modelling, we use an observation model based on the assumption that contact can only occur on object-robot surfaces to weigh particles.

Compared to methods that explicitly label object-contact associations through clustering, our probabilistic approach allows for corrections in hindsight. Empirically in simulation and in a real robot experiment, this translated to significantly better performance for our method on object retrieval tasks in clutter, where the robot must recognize an object and estimate its pose amongst other objects.

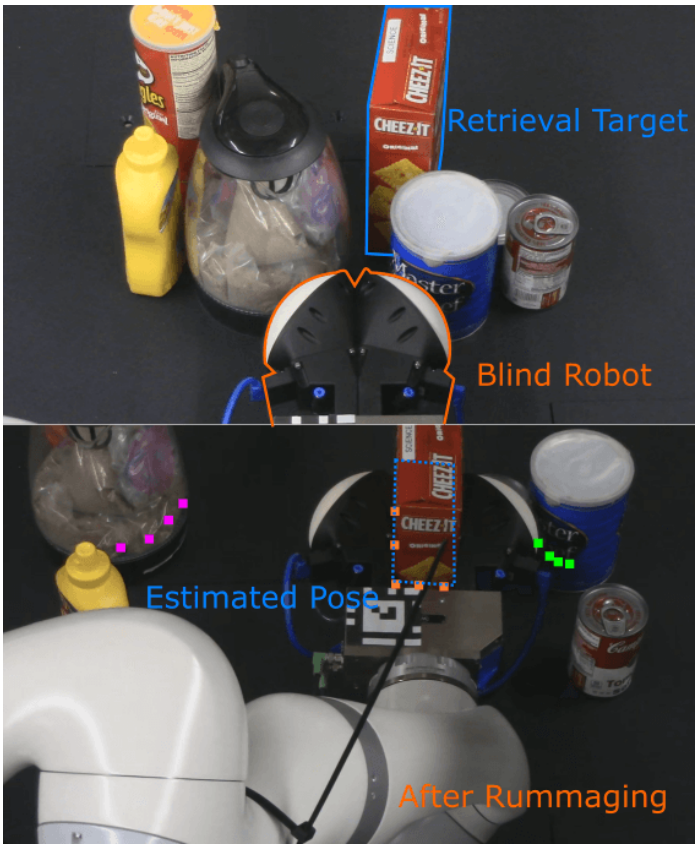


Figure 3.1: Initial (top) and after rummaging (bot) cluttered environment with STUCCO allowing us to successfully estimate the pose of the cracker box and grasp it without visual perception. The segmented tracked contact points are shown in different colors.

## 3.1 Introduction

This chapter considers the problem of tracking objects in cluttered environments without visual feedback. Applications such as rummaging through a cupboard, refrigerator, or bin for a target object require tracking objects to estimate the pose of the target with limited or no visual sensing. In these scenarios contact feedback is necessary to estimate the poses of objects. A key difficulty is that the objects in these scenarios are movable, requiring the robot to estimate the poses of objects as they move. This is especially challenging because we do not assume we know the shapes of, or even the number of, objects in the environment *a priori*. Thus when two nearby contacts are detected, it is not clear if we have contacted two objects once or one object twice. This ambiguous data association makes tracking much more difficult, as we may need to change the association of past contacts with objects when we observe new data.

Previous work in this area has focused either on single target tracking from contact *Koval et al.* (2015) or on visual tracking of objects *Betke et al.* (2007); *Schwarz et al.* (2018). Work on tracking multiple targets with lidar/sonar/visual data is relevant, but relies on receiving long-range information at high frequency to be effective, which is not the case for contact. To our knowledge, this is the first work to address the problem of tracking multiple objects using only contact feedback.

The key insight that allows us to tackle this problem is that we can efficiently propagate a belief over contact points without explicit object assignments. We can then sample from that belief to generate hypotheses of contact points and associations. We term this approach "soft tracking" to emphasize its difference with tracking explicit "hard" associations.

Given a model of the pushing dynamics (which can be very simplistic) and an existing method for localizing contact on the surface of the robot *Manuelli and Tedrake* (2016) (contact isolation), our method, which we call Soft Tracking Using Contacts for Cluttered Objects (STUCCO), tracks the belief over contact point locations and implicit associations using a particle filter. We propagate the belief by sampling whether each contact point moved with probability inversely proportional to its distance to the latest contact point, then updating the particles to enforce that contact points could not have occurred inside the robot. The best estimate of contact points and a hard association of them to objects, useful for downstream tasks, can be extracted from the belief through our segmentation process.

To show the utility of STUCCO, we demonstrate how it can be used to solve the Blind Object Retrieval (BOR) problem, where a target object of known shape must be retrieved from a planar cluttered environment. We evaluate our method and baselines on both simulated and real (Fig. 3.1) instances of this type of problem and find that our method achieves at least 65% grasp success on all environments while no baseline achieves more than 5% grasp success on all of them.

## 3.2 Related Work

When we know the number of objects in the environment and the mapping between sensing and object is unambiguous, single target tracking methods can be used, such as ones from *Wu et al.* (2013) when vision is available, or the Manifold Particle Filter *Koval et al.* (2015) when contact feedback is available. For single isolated objects, pose and shape estimation has been demonstrated using tactile feedback *Yu and Rodriguez* (2018); *Suresh et al.* (2020). Here, we focus on the much more difficult

problem when data association is ambiguous and there are an unknown number of objects.

For this problem, computer vision methods have traditionally been used. The relevant problem is termed Multiple Object Tracking, with *Luo et al.* (2020) providing a comprehensive survey of modern methods. In cases where vision is available, its information density makes it attractive as the primary method for object detection and tracking. Our method could be used in conjunction to resolve ambiguities and provide information around occlusions. Indeed, often the robot will occlude the target as it approaches for manipulation.

Outside of computer vision, Multiple Target Tracking is a more common term to refer to the problem and is associated with methods that are agnostic to the information source *Stone et al.* (2013). Classically, Multiple Hypothesis Tracking (MHT) *Blackman* (2004) propagates hypotheses on associations of observations (contacts) to specific targets (objects). A relaxation of allowing association probabilities, instead of fixed associations, is Joint Probabilistic Data Association (JPDA) *Fortmann et al.* (1983). While there are ways to limit the combinatorial number of hypotheses to make these methods tractable, in the context of unknown object shapes, the explicit association of contact to objects is difficult. Often only much later do we have sufficient data to discriminate previous associations, so many hypotheses must be kept.

An alternative to explicitly considering associations is propagating the intensity (first-order moment) of the posterior on the number of targets and their states. This class of methods is called intensity filters *Stone et al.* (2013), with the Probability Hypothesis Density (PHD) filter *Vo and Ma* (2006) being a notable special case. We compare against an implementation of the PHD filter as a baseline. All these methods were designed with dense information sources in mind (radar, sonar, or cameras), and their assumptions are problematic in the context of blind manipulation. Most significantly, their observation models assume each target generates an observation at each step with some state-independent probability. This is clearly not the case for contact, since we can only observe contact from objects close to the robot. Our method takes inspiration from intensity filters and propagates a belief without explicit associations that exploits the local nature of contact.

Several methods have been proposed for manipulation in cluttered environments. An RGB-D approach *Schwarz et al.* (2018) demonstrated success in visually segmenting then retrieving objects in clutter. However, the environments they showed allow immediate segmentation of the target object without needing to rummage; addition-

ally, the objects were often well separated from each other. A haptic approach with whole-arm tactile sensing was demonstrated in *Jain et al.* (2013) to successfully reach in clutter. In contrast to their focus on robot control for navigation, allowing them to push movable objects out of the way, we focus on perceiving the objects themselves for downstream manipulation tasks. Similar to *Jain et al.* (2013), our approach benefits from making numerous contacts, as each contact gives us information. While we do not have accurate localization of contact points from whole-arm tactile feedback (which is limited to very few current robots) we are able to perform our tasks using only the estimate of the external wrench at the end-effector.

## 3.3 Problem Statement

Let $\mathbf{x} \in \mathrm{SE}(3)$ denote the robot end-effector pose. We are given a trajectory $\mathbf{x}_0, ..., \mathbf{x}_T$, during which the robot has made contacts with some objects. We assume that the robot is the only agent in the environment, so objects only move in direct or indirect contact with the robot. Additionally we assume that the robot moves rigidly with no compliance, the robot's geometry is known, the clutter is rigid, and that we are given a dynamics model of how objects transform for some robot motion. Our objective is to track the contact points such that they stay close to object surfaces and are segmented corresponding to the objects they belong to.

Concretely, we define *contact error* (CE) on a contact point to be the smallest euclidean distance from the tracked point to any object surface. The contact error on the trajectory is the average over all contact points. Additionally, we evaluate the segmentation quality using the Fowlkes-Mallows index (FMI) *Fowlkes and Mallows* (1983), which approaches 0 for random assignments (with increasing number of points) and 1 for perfect assignments.

## 3.4 Method

At a high level, our approach enables downstream tasks such as object tracking and retrieval for robots "rummaging" in environments with only tactile feedback. To this end, our approach takes as input the robot trajectory $\mathbf{x}_0, ..., \mathbf{x}_T$ and a set of contact points (each one denoted $\mathbf{p} \in \mathbb{R}^3$) detected during motion. The output is the tracked set of contact points segmented based on object motions. Our method is composed of three elements: contact detection and isolation, soft tracking, and contact point segmentation, of which contact detection and isolation uses prior work

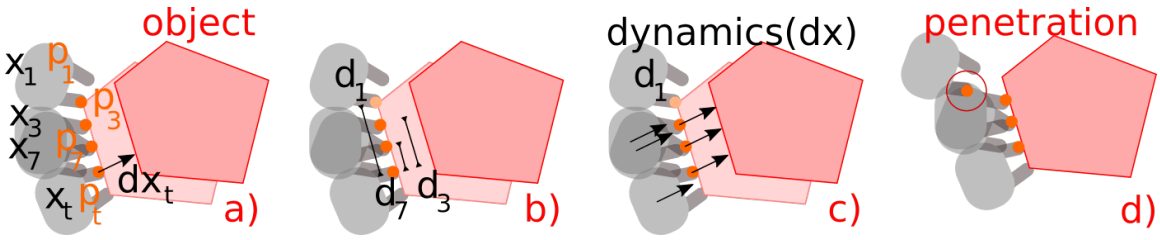


Figure 3.2: Prediction (a-c) and update (d) step of one particle. (a) Initial observation of the latest contact point, end-effector pose, and change in end-effector pose. (b) Compute connection probability based on distance to $\mathbf{p}_t$ and sample that connection; point 1 was not connected. (c) Apply dynamics to all connected points. (d) Update step assigns this particle low probability due to $\mathbf{p}_1$ penetrating $\mathbf{x}_3$ and $\mathbf{x}_7$.

while the rest are our contributions. In the following, we provide the details of each component.

### 3.4.1 Contact Detection and Isolation

To detect contact, we utilize the momentum observer *De Luca and Mattone* (2005). This observer estimates the external wrench applied to the robot ($\gamma$) using the robot's joint torques and dynamics model. We detect contact if a specified threshold $\bar{\epsilon}$ is exceeded, similar to *Manuelli and Tedrake* (2016); *Haddadin et al.* (2017):

$$\gamma^T \Sigma_{meas}^{-1} \gamma > \bar{\epsilon} \tag{3.1}$$

where $\Sigma_{meas}^{-1}$ is the precision matrix of the residual, measured by executing random actions in free space.

Once detected, we localize contact on the robot's surface using the Contact Particle Filter (CPF) *Manuelli and Tedrake* (2016). This filter iteratively solves for the contact location on the robot's surface assuming a point contact that can transmit forces but no torques – commonly referred to as the Hard Finger approximation *Prattichizzo and Trinkle* (2016). We note that the remainder of our method does not depend on the details of the contact detection and isolation algorithm. As such, advances in this area can be used to extend the functionality of our approach.

### 3.4.2 Soft Tracking

STUCCO maintains a belief over the positions of all contact points. One possibility is to track each contact point independently (e.g. a Kalman Filter to estimate each contact point's position); however, this approach ignores the dependence be-

**Algorithm 5:** Soft Tracking Using Contacts for Cluttered Objects

**Given:** $N$ number of particles, $l$ characteristic length, $l_p$ penetration length, `pxpen` point to robot penetration, `pxdyn` object dynamics for change in end-effector pose

$P_{1..N} \leftarrow \{\}$ `// particles`
$t \leftarrow 0$
**while** *robot is in execution* **do**
  robot executes action $\mathbf{u}_t$
  $t \leftarrow t + 1$
  observe $\mathbf{x}_t$ and $d\mathbf{x}_t$, change in end-effector pose while in contact
  **if** *in contact from Eq. 3.1* **then**
    $\mathbf{p}_t \leftarrow$ get latest contact point
    **for** $n \leftarrow 1$ **to** $N$ **do**
      $P_n \leftarrow P_n \cup \{(\mathbf{p}_t, \mathbf{x}_t)\}$
      $d_{0..t} \leftarrow \|\mathbf{p}_{n,0..t} - \mathbf{p}_t\|_2$
      $p_{connect,0..t} = e^{-d_{0..t}^2/l}$
      $p_{sample,0..t} \sim \mathcal{U}(0,1)$ indep.
      $adj \leftarrow p_{sample,0..t} < p_{connect,0..t}$
      `// predict step`
      $P_{n,adj} \leftarrow \texttt{pxdyn}(P_{n,adj}, d\mathbf{x}_t)$
      `// update step`
      $\epsilon \leftarrow \sum_{i=1}^{|P_n|} \sum_{j=1}^{|P_n|} \texttt{pxpen}(\mathbf{x}_{n,i}, \mathbf{p}_{n,j})$
      $p_{obs,n} \leftarrow e^{-\epsilon^2/l_p}$;
  **else**
    **for** $n \leftarrow 1$ **to** $N$ **do**
      $\epsilon \leftarrow \sum_{j=1}^{|P_n|} \texttt{pxpen}(\mathbf{x}_t, \mathbf{p}_{n,j})$
      $p_{obs,n} \leftarrow e^{-\epsilon^2/l_p}$;
  `ImportanceResample`$(P, p_{obs})$
  `ReplaceInconsistentPoints`$(P,$ `pxpen`$)$

tween contact points that stems from the connectivity between points that belong to the same object. To utilize this basic assumption and represent the belief, we use a particle filter where each particle represents the set of all contact point positions and associated end-effector poses for those contacts. For convenience, we refer to the pair $(\mathbf{p}, \mathbf{x})$ as a point. Alg. 5 shows how we propagate this belief while Fig. 3.2 depicts one step of our method for a single particle.

Our algorithm is structured in alternating prediction and update steps typical of Bayesian filters. Each particle is propagated independently, thus for simplicity we describe the process in terms of a single particle. However, in practice the process

can be parallelized across particles and points.

Our method does not explicitly track point to object associations, like the iFilter from *Stone et al.* (2013) and PHD filter from *Vo and Ma* (2006). Instead, at each step we sample associations to predict motion, the source of the "soft tracking" name. Incorrect associations are propagated forward, resulting in low likelihood for the particle when the update step detects inconsistencies arising from some past mistake.

In detail, the prediction step (lines 11 to 15 in Alg. 5) estimates how each contact point moves for an observed change in robot end-effector pose $d\mathbf{x}$. Since we assume that the robot is the only agent in the environment, we only predict motion when in contact. Contact points belong to objects; however, the likelihood of two contact points belonging to the same object scales inversely w.r.t. their relative distance. Line 12 from Alg. 5 encodes this using a characteristic length $l$ parameter. To determine each contact point's adjacency (belonging to the same object) to the most recently encountered one, we randomly sample proportional to the likelihood provided by their relative distance. Contact points on the same object and their associated end-effector poses move together according to the given dynamics function `pxdyn` on line 15. Thus a single contact point $\mathbf{p}_{n,i}$ at time $i < t$ would only move if it is adjacent to $\mathbf{p}_t$.

The update step in lines 21 and 17 evaluates the likelihood of each particle in realizing the most recent observation, $p_{obs,n}$. We utilize the fact that contact can only occur on the robot surface to evaluate each particle. To this effect, we define the function $\texttt{pxpen}(\mathbf{x}, \mathbf{p})$ which outputs 0 if $\mathbf{p}$ is outside the robot when the end-effector is in pose $\mathbf{x}$, and otherwise $\min_{\mathbf{p}_s \in S(\mathbf{x})} \|\mathbf{p} - \mathbf{p}_s\|_2$, where $S(\mathbf{x})$ is the set of points on the robot surface at $\mathbf{x}$.

When in contact, the predicted movement of the contact points may result in penetration between any pair of $\mathbf{x}$ and $\mathbf{p}$. Thus in line 16 we sum the penetration between all pairs in the particle. In contrast, in line 20 when out of contact, we only need to evaluate the observed $\mathbf{x}_t$ against all contact points since there is no predicted movement.

In both cases, the computation of $p_{obs,n}$ parallels our computation for adjacency during the prediction step, with a separate length parameter $l_p$ scaling with the expected contact isolation error (actual contact point's distance to the estimated contact point). A lower value will result in more false positives of penetration while a higher value will result in more false negatives. With $p_{obs}$, we perform the standard particle filter importance resampling (line 22 of Alg. 5).

Even after resampling, particles may still have penetration inconsistencies. This

**Algorithm 6:** `ReplaceInconsistentPoints`

**Given:** $P_{1..N}$ particles, `pxpen` point to robot penetration

1 **for** $n \leftarrow 1$ **to** $N$ **do**
2  **for** $j \leftarrow 1$ **to** $|P_n|$ **do**
  `// how inconsistent each point is`
3  $\epsilon_j \leftarrow \sum_{i=1}^{|P_n|} \texttt{pxpen}(\mathbf{x}_{n,i}, \mathbf{p}_{n,j})$
4  $incon \leftarrow \epsilon > 0$ `// indices`
5  **for** $j \in incon$ **do**
6  $k \leftarrow \arg\min_{i \in \neg incon} \|\mathbf{p}_{n,i}, \mathbf{p}_{n,j}\|_2$
7  $P_{n,j} \leftarrow \{\mathbf{p}_{n,k}, \mathbf{x}_{n,k}\}$ `// as well as weight`

could be due to none of the particles sampling a consistent prediction, or from errors in the contact isolation or contact dynamics. To address this, we call `ReplaceInconsistentPoints` after resampling, detailed in Alg. 6. A point is inconsistent and discarded if its $\mathbf{p}$ incurs any penetration, and replaced with the closest consistent point in terms of $\mathbf{p}$ Euclidean distance in lines 6 and 7 of Alg. 6.

### 3.4.3 Segmenting into Objects

Many useful applications of tracking require a single estimate of the contact points as well as hard assignments to objects. To achieve this, our method selects the most likely particle (MAP) according to the particle weights (updated each step with $p_{obs}$). Alg. 7 details how the MAP particle is segmented into groups of contact points that are estimated to belong to the same object.

Similar to line 12 from Alg. 5, we compute the connection probability and compare it against a threshold $\alpha$ to determine if an edge between two points exists. The resulting adjacency matrix $A$ describes a graph over all the points, from which we find connected components. Each connected component is an object. Our segmentation is a form of agglomerative clustering (such as with BIRCH *Zhang et al.* (1996) or DBSCAN *Schubert et al.* (2017)), which is well suited for irregular and elongated shapes, such as the set of points belonging to surfaces of objects. A common weakness of these methods is combining two clusters when noise or an error creates a data point between them. Our update process mitigates this weakness when the robot's configuration overlaps with the erroneous contact point (depicted in Fig. 3.7) and it is deemed inconsistent, but it remains an issue if our robot does not explore that location.

**Algorithm 7:** Segment a particle into objects

**Given:** $P_n$ a single particle, $l$ characteristic length, $\alpha$ probability threshold for each edge

**for** $i \leftarrow 1$ **to** $|P_n|$ **do**
  **for** $j \leftarrow 1$ **to** $|P_n|$ **do**
    $d \leftarrow \|\mathbf{p}_{n,i} - \mathbf{p}_{n,j}\|_2$
    $A_{i,j} = e^{-d^2/l} > \alpha$
**return** connected components of adjacency matrix $A$

## 3.5 Results

In this section, we evaluate and benchmark the performance of our approach on: i) tracking and segmentation of contact points under a "blind rummaging" policy; and ii) a downstream task of Blind Object Retrieval (BOR) – both in cluttered environments. To this end, we first describe our baselines. Next, we describe the robot environment and training data. Then, we formalize the downstream BOR task that uses contact tracking. Lastly, we quantitatively evaluate our method and baselines on BOR in simulated and real-world cluttered environments.

For all tasks, we used the following `pxdyn`:

$$\texttt{pxdyn}(P_n, d\mathbf{x}) = \{(\mathbf{p} + F(d\mathbf{x}), \mathbf{x} + d\mathbf{x}) | (\mathbf{p}, \mathbf{x}) \in P_n\} \tag{3.2}$$

where $F$ extracts the linear translation of the pose change. This motion model implicitly assumes that objects translate together with the robot when in contact without rotation. A more sophisticated motion model may be used if object properties such as size, shape, or pressure distribution are known *a priori*; however, this is not the case in our experiments. Here, we demonstrate that our method is able to partially mitigate errors from this approximation since some of its predictions result in contact point penetration.

To speed up our method, we implemented Alg. 5 to process each particle in parallel. In particular, `pxpen` was implemented as a parallel lookup of a pre-computed discretized (resolution 1mm) signed distance field of the end-effector in link frame. The transform of contact points from world to link frame was also implemented to be parallel.

For contact isolation, we used the Single-CPF from *Manuelli and Tedrake* (2016) which assumes each detected contact occurred at only one contact point. We note that while there are inherent ambiguities in isolating contact from externally-applied

wrenches, our method is robust to these errors. On the real robot, we additionally consider contacts detected by each of the two soft-bubble sensors *Kuppuswamy et al.* (2020) (seen in Fig. 3.1). This distributed tactile sensing modality significantly mitigates ambiguities from using only wrench estimates.

### 3.5.1 Baselines

We compare against baselines that maintain a single estimate of all $\mathbf{p}$, clustering on $\mathbf{p}$ at each step in contact, and applying the dynamics function to all points in the same cluster as $\mathbf{p}_t$. The baselines differ in their clustering methods, with BIRCH *Zhang et al.* (1996) and DBSCAN *Schubert et al.* (2017) by default not needing to specify the number of clusters. A k-means baseline was implemented that starts with a single cluster and increases the cluster number by 1 if doing so reduces the inertia (what k-means minimizes) sufficiently. Additionally, we consider a Gaussian Mixture (GM) implementation of the PHD filter *Vo and Ma* (2006). As introduced in Section 3.2, this method propagates the intensity (first-order moment) of the posterior on the number of objects and their positions. The intensity is integrated over to extract discrete targets (objects), which we clustered the contact points to using nearest neighbours then propagated in the same way as the clustering methods.

### 3.5.2 Training Set for Tuning

For simulation, we use a floating Franka Emika (FE) gripper from the PANDA arm (see Fig. 3.4) with a fixed height and constrained orientation. The gripper is simulated in PyBullet *Coumans and Bai* (2016–2021) and takes discrete action steps in the form of desired $dx, dy$, with a maximum per step movement of 0.03m along each dimension. Each simulation time step is 1/240s, and we moved slowly to avoid bouncing objects off the robot. The residual $\gamma$ used for contact detection and isolation here is the measured force torque on the gripper provided by the simulator.

Note that our method takes contact points as input and is not limited to planar systems. However, restricting to planar motion simplifies the data collection, contact isolation, and the downstream task of BOR.

Our training set consists of 40 trials of randomized start and goal positions for each of the 4 environments depicted in Fig. 3.3. We generated trajectories using a greedy controller that entered a random walk of length 6 upon contact. Trajectories that were in contact less than 5% of the time were discarded without replacement, yielding a total of 129 valid trajectories.

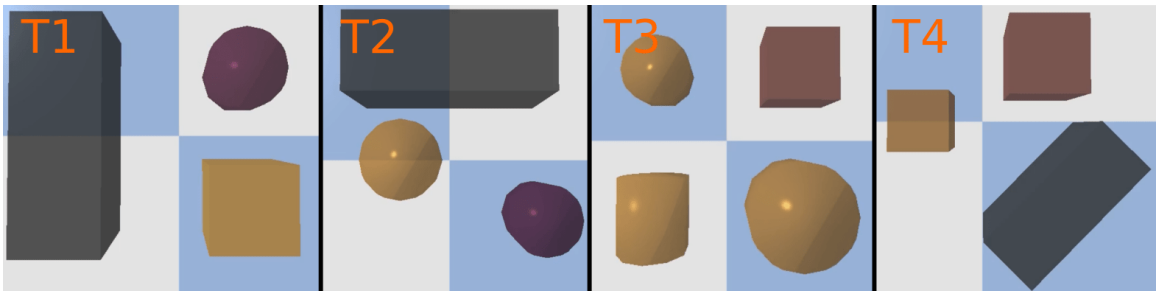


Figure 3.3: Training environments in simulation. Immovable walls are colored grey while the darker the movable object is, the more massive it is.

Table 3.1: STUCCO parameters for Blind Object Retrieval.

| Parameter | sim | real |
|---|---|---|
| $l$ characteristic length | 0.02 | 0.006 |
| $l_p$ penetration length | 0.002 | 0.002 |
| $\bar{\epsilon}$ residual threshold | 1 | 5 |
| $N$ number of particles | 100 | 100 |
| $\alpha$ connection threshold | 0.4 | 0.4 |

Tuning consisted of parameter sweeping to maximize median FMI and minimize median CE on the whole training set. For our method, the primary parameter to tune was the characteristic length $l$, which was larger on the real robot to handle a large kettle. See Tab. 3.1 for our tuned parameters. BIRCH was tuned to have threshold 0.08, DBSCAN was tuned to have eps 0.05 and minimum neighbourhood size of 1, k-means was tuned to need an inertia improvement of 5 times to increase the number of clusters, and the GMPHD filter was tuned to have birth probability of 0.001, spawn probability of 0, and detection probability of 0.3.

The tuned performances of all methods on the training set are shown in Fig. 3.5 (top), where our method outperforms all baselines in CE. Since we are interested in manipulation in clutter, Fig. 3.5 (bottom) shows the performance on runs that had ambiguous contact assignments. For each step, this was computed using the minimum distance from the robot to the second closest object, with an ambiguity score of 1 corresponding to a distance of 0 and a score of 0 corresponding to a distance of 0.15m or more. The bottom figure shows runs with an average ambiguity of at least 0.3. On these, our method outperforms the baselines in both FMI and CE by an even larger margin, demonstrating that our method is well suited for clutter.

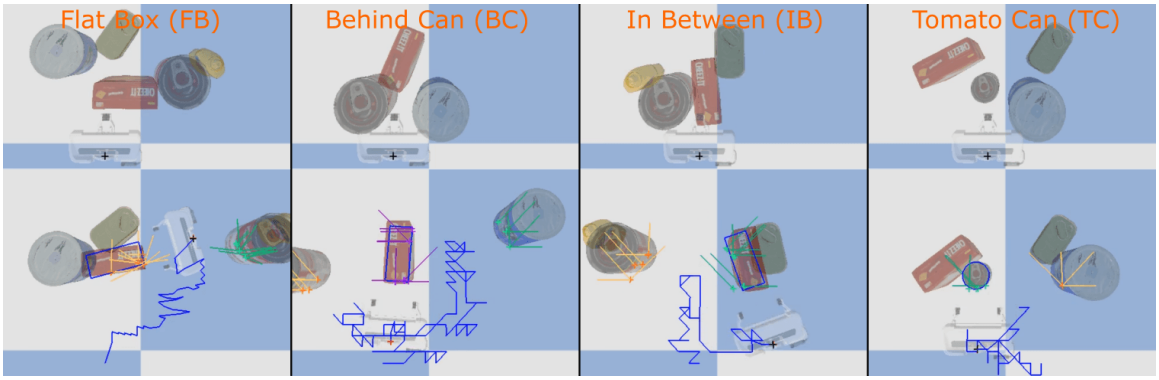


Figure 3.4: Simulated BOR task in 4 different environments, with each starting condition (top) and after executing actions (bottom), with the trail in blue. Overlaid is STUCCO's best estimate of segmented objects, with propagated contact points as crosses and associated actions taken with a different color for each object. The pose estimate of the target object is represented by a blue outline.

### 3.5.3 Blind Object Retrieval

We present the problem of Blind Object Retrieval: pose estimation and grasping of a target object with known geometry using no visual perception. To perform this task, the robot rummages in clutter to collect contact points that it can segment into objects using our method. Using the segmented objects, the robot runs iterative closest point (ICP) *Arun et al.* (1987) between the set of contact points of each object and model points sampled from the known surface of the target object. ICP is run 30 times from random initial poses and the object with the lowest variance in position estimation is selected (see Fig. 3.6). An important source of variance is uncertainty in orientation due to contacts being unevenly distributed across the object surface. From the ICP estimates of the selected object, we further select the one that penetrates $\mathbf{x}_t$ the least, and on ties choose the lowest ICP matching error.

To evaluate success, we attempt a grasp at the estimated pose after executing a given rummaging policy. Grasp success is an important metric to evaluate on for two reasons. First, it avoids the need to combine position and orientation errors. These metrics are typically combined using radius of gyrations which may not be available during run time. Second, it does not penalize small pose errors that may not be relevant to the task and can safely be ignored while penalizing those beyond a gripper dependent threshold that will always result in grasp failure.

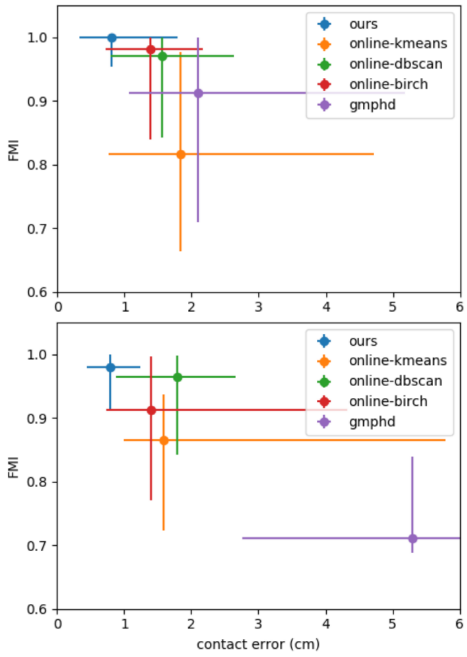


Figure 3.5: Tracking metrics evaluated on the training set, with the median plotted and error bars indicating 20-80$^{th}$ percentile. (top) Results for the whole data set, and (bot) results for only runs with an ambiguity score of at least 0.3. Ideal performance is an FMI of 1 and CE of 0 corresponding to points in the upper left.

### 3.5.4 Blind Object Retrieval in Simulation

In simulation, we designed 4 cluttered environments (see Fig. 3.4) with YCB objects *Calli et al.* (2017), with the target being the cracker box in FB, BC, and IB, and the tomato can in TC. A grasp was successful if it closed on the two long sides of the box and not on a corner, or around more than half of the tomato can. The control sequence was manually created to make contact with multiple objects that were initially close together, while making sufficient contacts to identify the target. When replayed, each action was perturbed with uniformly random noise of up to $\pm$0.5mm (compared to max action step of 30mm).

We performed 20 runs of each task (same random seed used for each baseline so they are evaluated under the same actions), with the statistical comparison of our method against baselines summarized in Tab. 3.2. Our method was the only one that achieved 65% grasp success or higher on all tasks, and achieved significantly lower contact error than all baselines.

Our method also achieved better FMI than baselines. The overall lower FMI scores compared to the training set seen in Fig. 3.5 attests to the difficulty of the BOR tasks. Importantly, the lowered FMI (indicating more assignment errors) for

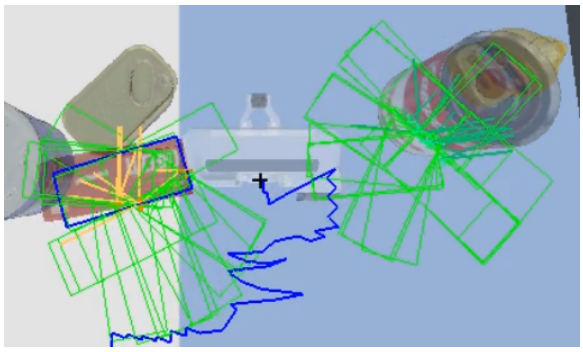

Figure 3.6: Iterative closest point pose estimates from 30 random starts plotted in green. ICP run on the left set of points (corresponding to the cracker box) result in lower positional variance in the estimate than the ICP results run on the set of points corresponding to the can (right).

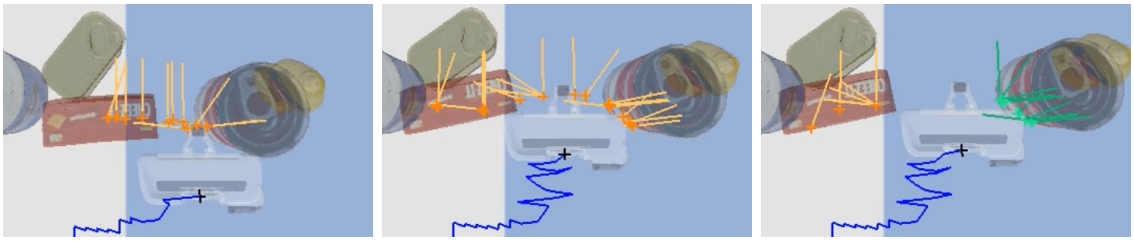

Figure 3.7: Steps during a BOR run with (left) initially wrong associations of contact points to objects, (mid) moving to right before entering the gap between the objects, and (right) resolving the previous ambiguity from moving through the gap and penalizing particles with points in between. Tracked contact points are in orange for the first object and green for the second one.

our method did not translate to increased CE. This is a key strength of our method and can be attributed to the particle update penalizing contact penetration and also the `ReplaceInconsistentPoints` function. Specifically, the action sequences often moved back and forth between two objects, eventually opening a gap and moving through it, as captured in Fig. 3.7. The oscillatory motion initially left particles with contact points between the gap, but after moving through it, the update process assigned high likelihood to particles that separated the contact points to either side. Additionally, `ReplaceInconsistentPoints` replaced remaining points in between with points on a side while occasionally making an assignment error.

The baselines have no mechanisms for correcting associations in hindsight, so the lower FMI translated to higher CE and lower grasp success. However, despite the high errors in CE, the baselines sometimes achieved grasp success due to the ICP eliminating many wrong pose estimates.

Table 3.2: Quantitative comparison of our method against baselines on 20 runs of blind object retrieval in different simulated cluttered environments and 5 runs on the real environment in Fig. 3.1. GS is grasp success (%), and CE is contact error (cm). Top values per category are in bold while standard deviations are in parentheses. FB, BC, IB, and TC are depicted in Fig. 3.4.

| task | | ours | BIRCH | DBSCAN | k-means | GMPHD |
|---|---|---|---|---|---|---|
| FB | GS | **70** | 5 | 5 | 45 | 25 |
| | FMI | **0.69 (0.08)** | 0.62 (0.07) | 0.66 (0.10) | 0.63 (0.05) | 0.56 (0.05) |
| | CE | **0.72 (0.25)** | 4.63 (0.53) | 3.52 (1.54) | 1.49 (0.40) | 3.90 (0.61) |
| BC | GS | **80** | 0 | 10 | 5 | 0 |
| | FMI | **0.92 (0.05)** | 0.84 (0.09) | 0.89 (0.02) | 0.83 (0.04) | 0.57 (0.04) |
| | CE | **1.33 (0.13)** | 6.49 (0.36) | 6.56 (0.31) | 7.04 (0.41) | 7.33 (0.44) |
| IB | GS | **65** | 0 | 60 | 10 | 0 |
| | FMI | **0.78 (0.07)** | 0.71 (0.05) | 0.75 (0.06) | 0.47 (0.23) | 0.46 (0.06) |
| | CE | **1.04 (0.33)** | 3.27 (0.36) | 2.31 (0.58) | 4.84 (1.81) | 5.62 (0.85) |
| TC | GS | **85** | 0 | 0 | 25 | 20 |
| | FMI | **1.00 (0.00)** | 0.79 (0.07) | **1.00 (0.00)** | 0.54 (0.17) | 0.54 (0.17) |
| | CE | **0.31 (0.90)** | 4.20 (0.22) | 4.14 (0.09) | 6.23 (0.72) | 8.09 (0.22) |
| Real | GS | **100** | 20 | 0 | 20 | 0 |

### 3.5.5 Real Robot Blind Object Retrieval

We applied our method on a real 7DoF KUKA LBR iiwa arm with two soft-bubble tactile sensors *Kuppuswamy et al.* (2020) on the gripper for the BOR task depicted in Fig. 3.1. Similar to simulation, we restricted our motion to be planar, with a max step of 20mm implemented using a Cartesian impedance controller. KUKA's on-board software estimated the externally applied wrench at the end-effector using the measured joint torques. To accommodate the limited sensitivity of this measurement, We filled the YCB objects to increase their mass and reduce the effect of measurement noise.

Contact isolation was performed independently by the left and right soft-bubble sensors, then by the CPF if neither of them detected contact. Each bubble had a depth camera inside that measured surface deformation. Pixels with deformations greater than 4mm were considered deformed. We then averaged all deformed pixel coordinates and projected that point to the camera frame then rigidly transformed it to produce a contact point in the world frame. Due to the deformable nature of the sensors, we adjusted `pxpen` to ignore the first 10mm of penetration.

To extend Alg. 5 to multiple contact points per step, each contact point's distance in line 11 is measured against their closest new contact point, and dynamics in line 15

is performed with their associated $d\mathbf{x}$. Note that $d\mathbf{x}$ for each new contact point may be distinct due to making contact at different times during the action.

## 3.6 Discussion

Proprioceptive and tactile driven object state-estimation is an important functionality for autonomous robotic systems in highly unstructured environments. Here, we discuss important extensions that can further generalize our method to more challenging instances of "blind" object state-estimation for downstream tasks such as object retrieval. These extensions include generalizing beyond point contacts and generating the rummaging policies.

### 3.6.1 Generalizing Beyond Object Translation

Our update step can use inconsistency in object motion to correct for errors in the dynamics function, such as assuming that objects translate without rotation. However, it is unable to handle higher dimensional pose changes such as those induced by toppling or deforming. To address this, more rich information beyond point contacts can be extracted from each contact (e.g., incipient slip from the soft-bubbles) together with more sophisticated object dynamics models.

### 3.6.2 Generalization Beyond Single Point Contacts

Alg. 5 can generalize beyond single point contacts without major changes. Indeed, as shown in our real robot experiment with essentially three contact detectors, we can easily generalize to multiple contacts per step. The soft-bubble sensors provide rich contact information that we hope to exploit in future work. Advanced representations of contact patches such as meshes and non-uniform rational B-splines (NURBS) *Piegl and Tiller* (1996) could directly replace or exist alongside contact points in Alg. 5 as long as we have efficient pairwise distance functions between them.

### 3.6.3 Rummaging Policy

In this chapter, our method assumed a prescribed action sequence that makes sufficient contact with our target object to uniquely identify it. Chapter V generates this policy using active perception.

## 3.7 Conclusion

We presented STUCCO, a contact tracking method that maintains a belief over contact point locations to enable corrections in hindsight. The method is based on the basic assumptions that points closer together are more likely to be on the same object, and that contact only occurs on the surface of the robot. We showed that it performs well on a variety of Blind Object Retrieval tasks in clutter and demonstrated its application on a real robot. Specifically, it is capable of handling cases where contact is initially made on different objects close together, and later correct their tracking when they are moved apart. In contrast, we showed that clustering and the PHD filter baselines struggle in these scenarios. Finally, the failure of baselines that maintain a single estimate of the contact points on our tasks suggests that it is beneficial to maintain a belief over them to allow corrections in hindsight. Future work will focus on representing rich contact patches and generating the rummaging policy.

# CHAPTER IV

# Diverse Plausible Object Registration

The previous chapter introduced the blind object retrieval task (BOR), which involves tracking the contact points through a contact-rich trajectory, segmenting the contact points into objects, the performing object pose estimation with each object's segmented points. The target object was identified as the set of contact points that had the least matching error in the ICP process for pose estimation. We verified it in simulation and on a real world set-up, but both problems were planar and required many contacts before we had a sufficiently good pose estimation. Compared to vision, contact provides few surface points, and those are localized to the same part of the object. Additionally, most methods produce single, best estimates. However, the comparative lack of information from contacts results in a lack of constraints on the set of plausible poses for the object. In such cases, estimating the set of plausible poses is more useful than producing a single best estimate.

This chapter addresses this through 1) using volumetric information, such as whether a point is known to be outside the object (free space) through the virtue of being swept by the robot or from vision, and 2) leveraging Quality Diversity (QD) optimization *Pugh et al.* (2016); *Fontaine and Nikolaidis* (2021) to simultaneously search for a diverse set of plausible poses. To our knowledge, it is the first application of QD to pose estimation.

## 4.1 Introduction

Pose registration—the process of estimating the pose of a given rigid object from sensor data, is a fundamental problem in robotics, as it is necessary for manipulation and reasoning. Much research has been done in estimating object pose from visual data, especially laser-range data *Collet et al.* (2009); *Cheng et al.* (2018). However, a clear view of the object may not always be available (e.g. an object in a cupboard,

as in Fig. 4.1, or grocery bag) or the material properties of the object may make it difficult to perceive visually (e.g. transparency).

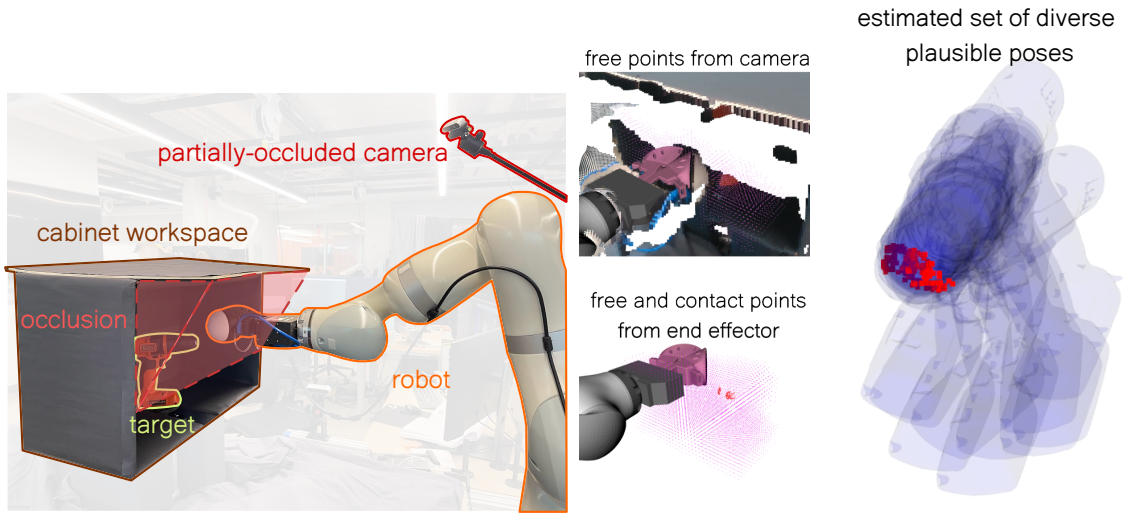


Figure 4.1: Left: Set up of a real-world probing experiment where the goal is to estimate the drill's pose. Middle: input to CHSEL, made of known free points in pink (from the camera and swept robot volume) and known surface points in red (from contact). Right: CHSEL uses these points and the object model to estimate a diverse set of plausible poses.

Partial occlusion in manipulation tasks motivates for rummaging, and researchers have investigated the use of tactile and force feedback for pose registration *Sipos and Fazeli* (2022); *Dikhale et al.* (2022). However, the nature of this data is quite different from the point-clouds produced by laser-scanners. While point-cloud data is information-dense (e.g. many points on the surface of the object), tactile and force data arising from contact contain much less information (e.g. one contact point per motion) in addition to being time-consuming to collect. This lack of information can be partially mitigated by assuming that a contact sensor moves along the surface of the object *Suresh et al.* (2022); *Driess et al.* (2017). However, creating controllers that can do this without moving the object is challenging.

In the context of pose registration problems, the lack of informative data results in a lack of constraints on the set of plausible poses of the object. In such cases, producing an accurate estimate of the true pose is very unlikely, and it is more useful to estimate the set of plausible poses. Especially towards the beginning of contact-based tasks, uncertainty in the object pose is high due to insufficient data, sensor noise, and inherent object symmetries. Characterizing this uncertainty, such as in the form of a set of plausible poses, is useful for object recognition *Xu et al.* (2022), active perception *Eidenberger and Scharinger* (2010), and simultaneous localization and mapping (SLAM) *Arras et al.* (2003); *Fu et al.* (2021).

The most common methods for pose registration are based on the Iterative Closest Point (ICP) algorithm *Segal et al.* (2009); *Wang and Zhao* (2017), which outputs a single pose estimate for a given initial pose. These methods can be effective for point-cloud data, but producing a set of estimates from random initialization does not yield good coverage of the set of plausible poses for contact data. Bayesian methods that aim to capture the full distribution of plausible object poses use approximation techniques such as Markov Chain Monte Carlo (MCMC) and variational inference *Maken et al.* (2021). However, such variational methods depend heavily on informative priors, which we do not assume are available.

To overcome the above limitations, we present Constrained pose Hypothesis Set Elimination (**CHSEL**), which has three key attributes: First, we go beyond only considering points on the surface of the object, considering volumetric information instead (similar to *Slavcheva et al.* (2016) and *Haugo and Stahl* (2020)). This allows us to infer more data (and thus more constraints on the pose) from robot motion. For example, when a robot moves into contact with an object, we observe contact points, as well as all the free space the robot traversed before and during contact. Note that this representation can also include free space and object surface points observed by a visual sensor.

Second, to take advantage of powerful gradient-based optimization tools, we construct a differentiable cost function that can be used to efficiently optimize a given pose based on volumetric information. Finally, and most importantly, to estimate a diverse set of poses simultaneously, we adapt methods from the Quality Diversity (QD) optimization literature. To our knowledge, this work is the first application of QD methods to the problem of pose registration. QD methods explicitly optimize for a set of solutions which are both diverse and high-quality, making them a natural choice for pose registration problems that seek to capture the set of plausible poses. We also show how to update our set of estimates online as more data is gathered by the robot.

Our experiments suggest that CHSEL has large performance improvements over several baseline methods for both simulated and real-world contact data. Additionally, we compare against alternatives and show that our cost function is a good QD objective. We also show that real-world visual data can be incorporated seamlessly into our cost function while demonstrating similar performance improvements.

## 4.2 Related Work

While our work is the first to use known free space to produce a diverse set of pose registrations, prior work has been done separately in using free space in registration and diverse set (also known as multi-hypothesis) registration. Geometric registration has been extensively studied in robotics and computer vision (see *Tam et al.* (2012) for an overview). In particular, the distinction between free space and surface points can be framed as point semantics or features, and methods such as the 3D Normal Distribution Transform (3D NDT) *Magnusson et al.* (2007) and its continuous generalization Continuous Visual Odometry (CVO) *Zhang et al.* (2021) have been designed with them in mind. We compare against CVO as a baseline. *Haugo and Stahl* (2020) considers free space explicitly, filling it with balls via the medial axis transformation. They then formulate a cost penalizing object-ball penetration while requiring points to lie on the surface. This is a baseline in our experiments.

Specific to SE(2) pose estimation in planar contact problems, the Manifold Particle Filter *Koval et al.* (2015) exploits a robot's contact manifold to estimate an pose. However, it struggles to scale to full SE(3) pose estimation as it is expected to require exponentially more particles.

Deep learning based methods such as SegICP *Wong et al.* (2017) and MHPE *Fu et al.* (2021) have been developed to produce a plausible set of pose estimates. However, they can only use points from the object surface and require relatively dense information.

Related to registration is the problem of object reconstruction. SDF-2-SDF-*Slavcheva et al.* (2016) minimizes the difference between pairs of signed distance fields (SDFs). They construct an SDF using observed RGBD images and match it against the target SDF. In cases where a dense view of surface points is not available, such as when the camera is occluded or if the sensing is performed via contact, the constructed SDF will be invalid. Similar to them, we directly work with the target object's SDF, but importantly do not assign SDF values to observed free points. Instead, we only require that known free points be outside the surface (SDF 0-level set).

Diversity in registration has mainly been explored as characterizing the pose uncertainty. *Censi* (2007) provides a closed form estimate for ICP based methods, but require that the initial point-correspondences are correct and that the minimization procedure does not get caught in local minima. This is unlikely to be valid with partial information, and does not utilize known free space. *Buch et al.* (2017) produces

high quality uncertainty estimates through MCMC simulation of a depth camera, which is computationally expensive while being restricted in input modality. *Maken et al.* (2021) performs Stein Variational Gradient Descent (SVGD) on a differentiable formulation of the ICP objective. They approximate the distribution of poses by running ICP from different starts, which in general is not the distribution of poses consistent with the data. We compare against SVGD optimization as a baseline.

Generating diverse sets of high quality solutions has been explored explicitly in recent research on evolutionary optimization techniques. In particular, Quality Diversity (QD) *Pugh et al.* (2016) techniques such as MAP-Elites *Mouret and Clune* (2015) and CMA-MEGA *Fontaine and Nikolaidis* (2021) have been developed to optimize objectives while enforcing diversity in some aspect of the solutions. We leverage QD optimization methods with our proposed differentiable cost function to estimate a set of plausible diverse transforms.

## 4.3 Problem Statement

For a target object, we have its precomputed object frame signed distance function (SDF) derived from its 3D model, $\texttt{sdf} : \mathbb{R}^3 \rightarrow \mathbb{R}$ , and are given a set of points $\mathcal{X} = \{(\mathbf{x}_1, s_1), ..., (\mathbf{x}_N, s_N)\}$ with known world positions $\mathbf{x}_n \in \mathbb{R}^3$ and semantics $s_n$ (described below). $\mathcal{X}$ is produced from sensor data. Object registration is the problem of finding transforms $\mathbf{T} \in \mathrm{SE}(3)$ that satisfy constraints imposed by $\mathcal{X}$. Let $\mathbf{T}^*$ be the true object transform, then the semantics are

$$s_n = \begin{cases} \texttt{free} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) > 0 \\ \texttt{occupied} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) < 0 \\ v_n & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) = v_n \end{cases} \tag{4.1}$$

We quantify the degree to which the constraints of $\mathcal{X}$ are satisfied by using a cost function (lower is better) $C(\mathcal{X}, \mathbf{T}) = \sum_{n=1}^{N} c(\mathbf{T}\mathbf{x}_n, s_n)$ where $c_m \gg 0$ and

$$c(\mathbf{x}, s) = \begin{cases} c_m \mathbf{1}(\texttt{sdf}(\mathbf{x}) \leq 0) & \text{if } s = \texttt{free} \\ c_m \mathbf{1}(\texttt{sdf}(\mathbf{x}) \geq 0) & \text{if } s = \texttt{occupied} \\ |v - \texttt{sdf}(\mathbf{x})| & \text{else} \end{cases} \tag{4.2}$$

$\mathbf{1}$ is the indicator function that evaluates to 1 if the argument is true and otherwise

evaluates to 0. We then define the plausible set $\mathcal{T}_\epsilon = \{\mathbf{T} \mid C(\mathcal{X}, \mathbf{T}) - C(\mathcal{X}, \mathbf{T}^*) < \epsilon\}$ where $\epsilon > 0$ is the degree of violation we allow in considering the constraints satisfied. Our goal is to produce a hypothesis transform set $\hat{\mathcal{T}}$ that covers $\mathcal{T}_\epsilon$. To quantify how well we cover this set, we use the Plausible Diversity metric *Saund and Berenson* (2021) *Saund and Berenson* (2022):

$$M_c = \frac{1}{|\mathcal{T}_\epsilon|} \sum_{\mathbf{T} \in \mathcal{T}_\epsilon} \min_{\hat{\mathbf{T}} \in \hat{\mathcal{T}}} d(\mathbf{T}, \hat{\mathbf{T}}) \qquad \text{coverage} \tag{4.3}$$

$$M_p = \frac{1}{|\hat{\mathcal{T}}|} \sum_{\hat{\mathbf{T}} \in \hat{\mathcal{T}}} \min_{\mathbf{T} \in \mathcal{T}_\epsilon} d(\mathbf{T}, \hat{\mathbf{T}}) \qquad \text{plausibility} \tag{4.4}$$

$$M_{pd} = M_c + M_p \qquad \text{plausible diversity} \tag{4.5}$$

where $d$ is a distance function on transforms, such as Chamfer Distance between the resulting transformed objects. This metric penalizes $\hat{\mathcal{T}}$ if 1) it does not include a transform that is close to each transform in the plausible set (a lack of coverage); or 2) includes transforms that are far from any transform in the plausible set (such transforms are implausible).

## 4.4 Method

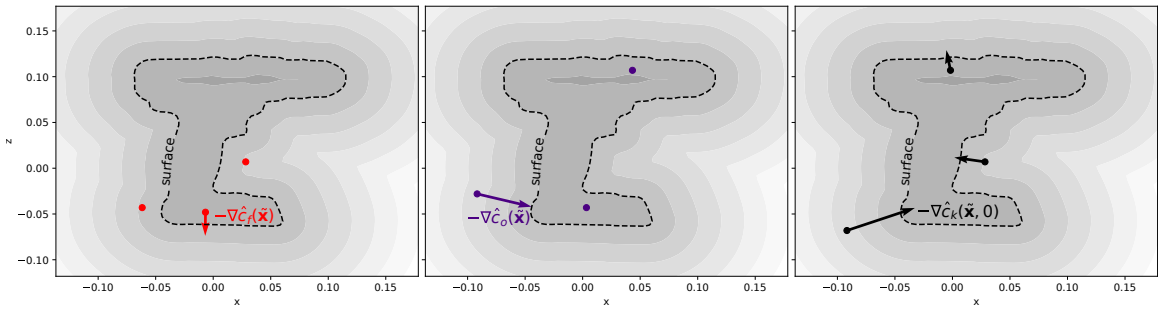


Figure 4.2: $\hat{C}(\mathcal{X}, \mathbf{T})$ negative gradients with respect to sampled points shown for a X-Z cross section of the YCB drill. Red points are known free space and $-\nabla\hat{c}_f$ pushes them outside the object. Purple points are known occupied and $-\nabla\hat{c}_o$ pushes them inside the object. Black points have known SDF values (here they are known surface points, $\texttt{sdf}(\tilde{\mathbf{x}}) = 0$) and $-\nabla\hat{c}_k$ pushes them towards the corresponding SDF level set.

This section presents CHSEL, which consists of a differentiable cost function (relaxation of Eq. 4.2) that enables gradient-based methods to reduce a transform's cost, and a quality diversity optimization scheme, which uses that cost to estimate the set

of plausible transforms. We also show how to update CHSEL's $\hat{\mathcal{T}}$ estimates online as more points are perceived.

### 4.4.1 Relaxation of Semantic Constraints

Eq. 4.2 has discrete components and is not differentiable. We would like a relaxation $\hat{C}(\mathcal{X}, \mathbf{T})$ of $C(\mathcal{X}, \mathbf{T})$ that is differentiable to be more amenable to optimization. For convenience, when the $\mathbf{T}$ used is unambiguous, we denote point positions in the world frame transformed to an estimated object frame as $\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x}$ in homogeneous coordinates (append 1 to $\tilde{\mathbf{x}}$ and $\mathbf{x}$). Specifically, we want to efficiently compute the gradient $\nabla_{\mathbf{T}}\hat{C}(\mathcal{X}, \mathbf{T})$. For better geometric intuition, we consider the gradient contributed by each known point:

$$\nabla_{\mathbf{T}}\hat{C}(\mathcal{X}, \mathbf{T}) = \sum_{n=1}^{N} \nabla_{\mathbf{T}}\hat{c}(\mathbf{T}\mathbf{x}_n, s_n) = \sum_{n=1}^{N} \nabla_{\tilde{\mathbf{x}}}\hat{c}(\tilde{\mathbf{x}}, s_n) \tag{4.6}$$

This gradient is spatial and with respect to the transformed point. Intuitively, gradient descent will move the points spatially along their negative gradients through adjusting $\mathbf{T}$. This is visualized in Fig. 4.2. As it is clear what each gradient is with respect to, we drop the subscript in future usage.

The separate semantic classes motivate us to consider each case separately. We partition $\mathcal{X}$ into $\mathcal{X}_f = \{(\mathbf{x}, s) \mid s = \texttt{free}\}$, $\mathcal{X}_o = \{(\mathbf{x}, s) \mid s = \texttt{occupied}\}$, and $\mathcal{X}_k = \{(\mathbf{x}, s) \mid s \in \mathbb{R}\}$. We then decompose the gradient:

$$\nabla\hat{C}(\mathcal{X}, \mathbf{T}) = \sum_{\mathbf{x}, s\in\mathcal{X}_f} \nabla\hat{c}_f(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s\in\mathcal{X}_o} \nabla\hat{c}_o(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s\in\mathcal{X}_k} \nabla\hat{c}_k(\tilde{\mathbf{x}}, s) \tag{4.7}$$

At each point, the cost arises from an SDF value mismatch and thus the gradient must be along the direction of greatest SDF value change. This is provided precisely by $\nabla_{\tilde{\mathbf{x}}}\texttt{sdf}(\tilde{\mathbf{x}})$, the gradient (normalized such that $\|\nabla\texttt{sdf}(\tilde{\mathbf{x}})\|_2 = 1$) of the SDF at that point. Thus all cost gradients must be parallel or anti-parallel to the SDF gradient. See Section 4.4.2 for how we achieve efficient lookup of SDF values and gradients. Fig. 4.2 shows our cost applied to points of each semantic class. Arrows indicate the negative cost gradient experienced by that point, which is the spatial direction the points will move along when we perform gradient descent on the cost. We define the gradients directly and assign its magnitude as the cost value.

#### 4.4.1.1 Free space cost

From Eq. 4.2, points in $\mathcal{X}_f$ achieve 0 cost when $\texttt{sdf}(\tilde{\mathbf{x}}) > 0$. When $\texttt{sdf}(\tilde{\mathbf{x}}) \leq 0$ the negative gradient points towards the SDF 0-level set (surface of the object). To tolerate small degrees of violation due to uncertainty in the point positions, we aim for the $\alpha$-level set where $\alpha < 0$. We define the magnitude of free space violation as $\max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}}))$. This has the effect of only giving non-zero gradients to violations beyond $\alpha$. Thus we have

$$\nabla \hat{c}_f(\tilde{\mathbf{x}}) = -C \max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}})) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{4.8}$$

where $C > 0$ is a scaling parameter. In a sense, it controls the degree of relaxation since using smaller values will lead to a smoother optimization path, particularly near the start of the optimization, while a higher value is needed to enforce the high cost from Eq. 4.2. This scaling parameter can be annealed during the optimization process, i.e. starting with a small value and increasing over optimization iterations.

#### 4.4.1.2 Occupied space cost

Symmetric to the free space cost, violating occupied points moves along $-\nabla \hat{c}_o$ to the $-\alpha$-level set. In this case, violation occurs when $\texttt{sdf}(\tilde{\mathbf{x}}) > -\alpha$ and has magnitude $-\min(0, -\alpha - \texttt{sdf}(\tilde{\mathbf{x}}))$:

$$\begin{aligned} \nabla \hat{c}_o(\tilde{\mathbf{x}}) &= -C \min(0, -\alpha - \texttt{sdf}(\tilde{\mathbf{x}})) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \\ &= C \max(0, \alpha + \texttt{sdf}(\tilde{\mathbf{x}})) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \end{aligned} \tag{4.9}$$

#### 4.4.1.3 Known SDF cost

This cost is a generalization of surface matching present in many registration methods. Known surface points are a special case of $s = 0$, and is commonly perceived through contact and visual perception. The cost's structure is similar to the previous costs, with the difference being that instead of $\alpha$ and $-\alpha$, each point has a separate desired level set given by its semantic value:

$$\nabla \hat{c}_k(\tilde{\mathbf{x}}, s) = (\texttt{sdf}(\tilde{\mathbf{x}}) - s) \nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{4.10}$$

### 4.4.2 SDF Query Improvements

Evaluating $\nabla\hat{c}(\mathcal{X}, \mathbf{T})$ requires computing $\texttt{sdf}(\tilde{\mathbf{x}})$ and $\nabla\texttt{sdf}(\tilde{\mathbf{x}})$ for $N$ known points. Regardless of the structure and efficiency of the given `sdf`, we precompute a voxel-grid approximation of it to enable fast parallel lookup. Each voxel reports the SDF value and gradient at the center of it. Each voxel is cubic with side length (resolution) $r_t$, with the whole grid being the object's bounding box padded by $\gamma > 0$ on all sides. Queries of points outside the voxel-grid are deferred to the original `sdf`, with $\texttt{sdf}(\tilde{\mathbf{x}}) = ||\tilde{\mathbf{x}} - \mathbf{x}'||_2$ and $\nabla_{\tilde{\mathbf{x}}}(\tilde{\mathbf{x}}) = (1 - 2\mathbf{1}(\tilde{\mathbf{x}} \text{ inside } \mathcal{M}))(\tilde{\mathbf{x}} - \mathbf{x}')$. Where $\mathbf{x}' = \arg\min_{\mathbf{x}\in\mathcal{M}} ||\tilde{\mathbf{x}} - \mathbf{x}||_2$ is the closest point on the mesh to $\tilde{\mathbf{x}}$, and a ray is traced from the inside of the object (assuming object-centered origin is interior) to $\tilde{\mathbf{x}}$, with an even number of mesh surface crossings indicating it is inside. Lower $r_t$ (a denser grid) trades higher memory usage for more accurate representation.

Another challenge to the efficiency of evaluating $\nabla\hat{C}(\mathcal{X}, \mathbf{T})$ is the representation of known free points $\mathcal{X}_f$. This is typically a volume, such as the space swept out by a robot's motion or derived from visual data. Representing this volume as a dense set of points makes $\nabla\hat{c}_f$ prohibitively expensive to evaluate. Similar to the 3D Normal Distribution Transform *Magnusson* (2009), we discretize the free space into a voxel-grid. The voxel-grid has resolution $r_f$, and the whole grid expands to the range of free points. $r_f$ allows us to set the maximum point density.

### 4.4.3 Quality Diversity Optimization

With Eq. 4.7 we can optimize an initial $\mathbf{T}$ using stochastic gradient descent (SGD). The optimized $\mathbf{T}$ depends on the starting $\mathbf{T}$ and will achieve a local minima of $\hat{C}$. A naive approach to creating the estimated plausible transform set $\hat{\mathcal{T}}$ is to start with a set of transforms $\hat{\mathcal{T}}_0$ and perform SGD on each $\mathbf{T} \in \hat{\mathcal{T}}_0$ separately. We compare to this approach as an ablation in our experiments, where we find that this method often produces $\hat{\mathcal{T}}$ with poor plausible diversity as it relies only on the diversity of local minima for coverage.

Instead, we turn to Quality Diversity (QD) optimization. At a high level, in addition to the $\mathbb{R}^m$ solution space to search over to maximize an objective $f : \mathbb{R}^m \to \mathbb{R}$, there are $k$ behavior (also known as measure) functions $B_i : \mathbb{R}^m \to \mathbb{R}$, jointly $\mathbf{B} : \mathbb{R}^m \to \mathbb{R}^k$. For the behavior space $\mathcal{B} = \mathbf{B}(\mathbb{R}^m)$ (image of $\mathbf{B}$), the QD objective is to find for each $\mathbf{b} \in \mathcal{B}$ a solution $\boldsymbol{\theta} \in \mathbb{R}^m$ such that $\mathbf{B}(\boldsymbol{\theta}) = \mathbf{b}$ and $f(\boldsymbol{\theta})$ is maximized. See *Pugh et al.* (2016) for an overview of the field.

For our problem, $f(\boldsymbol{\theta}) = -\hat{C}(\mathcal{X}, \mathbf{T})$, and we search over the transforms repre-

sented in $\mathbb{R}^9$, with 3 translational components and the 6 dimensional representation of rotation suggested by *Zhou et al.* (2019). Our $\mathbf{B}$ extracts the translational components of the pose. In a sense, we are searching for the best rotation given some translation to minimize $\hat{C}$. Intuitively, QD's enforced diversity over $\mathcal{B}$ will prevent the collapse of $\hat{\mathcal{T}}$ when $\mathcal{X}$ does not sufficiently constrain our estimation.

In particular, we use CMA-MEGA *Fontaine and Nikolaidis* (2021) optimization to take advantage of our cost's differentiability to more efficiently search for good solutions. $\mathcal{B}$ is discretized into a regularly spaced grid, called the archive, with each cell holding the best solution for that cell. Diversity in $\hat{\mathcal{T}}$ is enforced by requiring each $\mathbf{T} \in \hat{\mathcal{T}}$ to come from a different cell in $\mathcal{B}$. This is an evolutionary method in that the lowest cost transforms from different cells are iteratively combined to generate new transforms. Thus, it is valuable to populate the archive with low cost transforms $\hat{\mathcal{T}}_l$ to initialize the search. If no prior estimate is available, $\hat{\mathcal{T}}_l = \{\}$, but when we run CHSEL iteratively, $\hat{\mathcal{T}}_l$ contains the estimates from the previous iteration (see Sec. 4.4.4).

Algorithm 8 describes how we use QD optimization. First, we run SGD on the given initial transform set $\hat{\mathcal{T}}_0$ using Eq. 4.7 to create an $\hat{\mathcal{T}}'$. We extract its translation components $\mathbf{B}(\hat{\mathcal{T}}) = \mathcal{P}$. Using the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of $\mathcal{P}$ along each dimension, we size the grid $\mathcal{B}$ between $[\boldsymbol{\mu} - b_\sigma \boldsymbol{\sigma}, \boldsymbol{\mu} + b_\sigma \boldsymbol{\sigma}]$. The grid is centered on the mean $\boldsymbol{\mu}$ with extents scaled by the standard deviation $\boldsymbol{\sigma}$ along each dimension. A large $\boldsymbol{\sigma}$ suggests that there are low cost solutions with very different values along that dimension, motivating a wider search range. The parameter $b_\sigma > 0$ adjusts how many standard deviations out we search for solutions. We initialize $\mathcal{B}$ with known low cost transforms from $\hat{\mathcal{T}}_l$, along with the SGD solutions $\hat{\mathcal{T}}'$ to seed the QD optimization. Note that sizing the archive defines the region of the behavior space to search over while initializing it populates some grid cells with transform values. We then run CMA-MEGA on $\mathcal{B}$ for $n_o$ iterations to populate $\mathcal{B}$ with the lowest cost $\mathbf{T}$ for each cell. Finally, we select the $\mathbf{T}$ from the $|\hat{\mathcal{T}}_0|$ lowest cost cells as $\hat{\mathcal{T}}$.

Since we initialize $\mathcal{B}$ with $\hat{\mathcal{T}}' \cup \hat{\mathcal{T}}_l$, the QD optimization can be seen as a fine-tuning process. Initially, each $\mathbf{T} \in \hat{\mathcal{T}}' \cup \hat{\mathcal{T}}_l$ is the best solution for their respective cells (assuming they fall into different cells of $\mathcal{B}$). If a lower cost transform exists for a cell, QD optimization will eventually find it and replace the original $\mathbf{T}$ from $\hat{\mathcal{T}}' \cup \hat{\mathcal{T}}_l$.

**Algorithm 8:** CHSEL: QD optimization for $\hat{\mathcal{T}}$

**Given:** $\mathcal{X}$ known points, $\hat{\mathcal{T}}_0$ initial transform set, $\hat{\mathcal{T}}_l$ low cost transform set, $b_\sigma$ number of standard deviations to consider, $n_o$ number of QD iterations

1 $\hat{\mathcal{T}}' \leftarrow$ SGD on $\hat{\mathcal{T}}_0$ with $\hat{C}(\mathcal{X}, \mathbf{T})$
2 $\mathcal{P} \leftarrow \mathbf{B}(\hat{\mathcal{T}}')$
3 $\boldsymbol{\mu} \leftarrow \text{mean}(\mathcal{P})$, $\boldsymbol{\sigma} \leftarrow \text{std}(\mathcal{P})$
4 $\mathcal{B} \leftarrow$ grid with dimensions $[\boldsymbol{\mu} - b_\sigma \boldsymbol{\sigma}, \boldsymbol{\mu} + b_\sigma \boldsymbol{\sigma}]$
5 $\mathcal{B} \leftarrow \text{UpdateCells}(\mathcal{B}, \hat{\mathcal{T}}' \cup \hat{\mathcal{T}}_l)$ `// initialize the search with low cost transforms`
6 $\mathcal{B} \leftarrow \text{CMA-MEGA}(\mathcal{B}, \hat{C}, n_o)$
7 $\hat{\mathcal{T}} \leftarrow \{\mathbf{T} | \mathbf{T} \in$ cells from $\mathcal{B}$ with $|\hat{\mathcal{T}}_0|$ lowest costs $\}$

### 4.4.4 Online Updates to $\hat{\mathcal{T}}$

Registration can be performed iteratively as new sensor data are added to $\mathcal{X}$. Information from the previous registration allows us to more efficiently search for $\hat{\mathcal{T}}$. Our update process is described in Algorithm 9. First, we consider the generation and update of the initial transform set $\hat{\mathcal{T}}_0$. Before any registration, we sample uniformly at random (both position and rotation), within the given workspace $\mathcal{W}$ where the object could possibly be. Assuming the object has not moved, we update $\hat{\mathcal{T}}_0$ as $|\hat{\mathcal{T}}_0|$ perturbations around the best $\mathbf{T}$ of the previous estimated set, $\mathbf{T}' = \arg\min_{\mathbf{T} \in \hat{\mathcal{T}}} \hat{C}(\mathcal{X}, \mathbf{T})$. We perturb its translational components with Gaussian noise $\sigma_t > 0$ along each dimension. Then, in line 12, we uniformly sample a rotation axis, scaled with an angle sampled as Gaussian noise with $\sigma_R > 0$. We take the exponential map of this axis-angle representation and multiply it by the rotational component of $\mathbf{T}'$. This sampling based update of $\hat{\mathcal{T}}_0$ helps with escaping bad local minima.

Secondly, the data update changes $\mathcal{X}$ and invalidates the previously computed $\mathcal{B}$, but the solutions in each cell of the $\mathcal{B}$ may still have low cost with respect to the updated $\mathcal{X}$. We select them as $\hat{\mathcal{T}}_l$ and use them to initialize the next QD optimization in line 5 of Algorithm 8.

## 4.5 Results

In this section, we first describe our simulated and real robot environments, and how we estimate $\mathcal{X}$ from sensor data. Next, we describe how we generate the plausible set. Then, we describe our baselines and ablations and how we quantitatively evaluate each method on probing experiments of objects in simulation and in the real world.

**Algorithm 9:** Online update of $\hat{\mathcal{T}}$

**Given:** $\mathcal{W}$ workspace dimension, $\sigma_t$ translation noise, $\sigma_R$ rotation noise

$\hat{\mathcal{T}}_0 \leftarrow \mathcal{P} \sim U(\mathcal{W}) \times U(\mathrm{SO}(3))$
$\hat{\mathcal{T}}_l \leftarrow \{\}$
**while** *register* **do**
  $\mathcal{X} \leftarrow$ perceived from environment
  $\hat{\mathcal{T}} \leftarrow \mathrm{CHSEL}(\mathcal{X}, \hat{\mathcal{T}}_0, \hat{\mathcal{T}}_l)$
  $\hat{\mathcal{T}}_l \leftarrow \hat{\mathcal{T}}$
  $\mathbf{T}' \leftarrow \arg\min_{\mathbf{T} \in \hat{\mathcal{T}}} \hat{C}(\mathcal{X}, \mathbf{T})$
  $\hat{\mathcal{T}}_0' \leftarrow \{\}$
  **for** $i \leftarrow 1$ **to** $|\hat{\mathcal{T}}_0|$ **do**
    $\boldsymbol{\Delta t} \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}([\sigma_t, \sigma_t, \sigma_t])$
    $\theta \sim \mathcal{N}(0, \sigma_R)$
    $\mathbf{e} \sim U(\{\mathbf{x} \mid \|\mathbf{x}\|_2 = 1, \mathbf{x} \in \mathbb{R}^3\})$
    $\boldsymbol{\Delta R} \leftarrow e^{\theta \mathbf{e}}$ // axis angle to matrix
    $\hat{\mathcal{T}}_0' \leftarrow \hat{\mathcal{T}}_0' \cup \{(\boldsymbol{\Delta t} + \mathrm{trans}(\mathbf{T}'), \boldsymbol{\Delta R} \cdot \mathrm{rot}(\mathbf{T}'))\}$
  $\hat{\mathcal{T}}_0 \leftarrow \hat{\mathcal{T}}_0'$

Lastly, we evaluate the value of our cost as a QD objective.

In these experiments, a target object is in a fixed pose inside an occluded cabinet, and we estimate its pose through a fixed sequence of probing actions by a robot (some of which will result in contact). We run all methods after each probe, updating our set of pose estimates using Algorithm 9. Note that all methods receive the same known points $\mathcal{X}$ after each probe. Each probe extends the robot straight into the cabinet for a fixed distance or until contact. The probing locations are configuration specific and designed such that at least some contacts are made. We use YCB objects *Calli et al.* (2017) for both simulated and real experiments as their meshes are readily available. We precompute the object-frame SDF from these meshes as described in Section 4.4.2. In all cases, we are estimating a set of 30 transforms ($|\hat{\mathcal{T}}| = 30$), and use parameters $\alpha = -10mm, b_\sigma = 3, C = 20, \sigma_t = 0.05m, \sigma_R = 0.3$. In the sim experiments we use $n_o = 100$ and in the real world $n_o = 500$. For the Plausible Diversity distance function $d(\mathbf{T}, \hat{\mathbf{T}})$, we sample 200 points on the object surface and evaluate the Chamfer Distance between them after being transformed. Note that the 200 points sampled are different per trial. We extract the $x$ and $y$ components of the pose using $\mathbf{B}$ - we found no significant difference in performance from also extracting $z$.

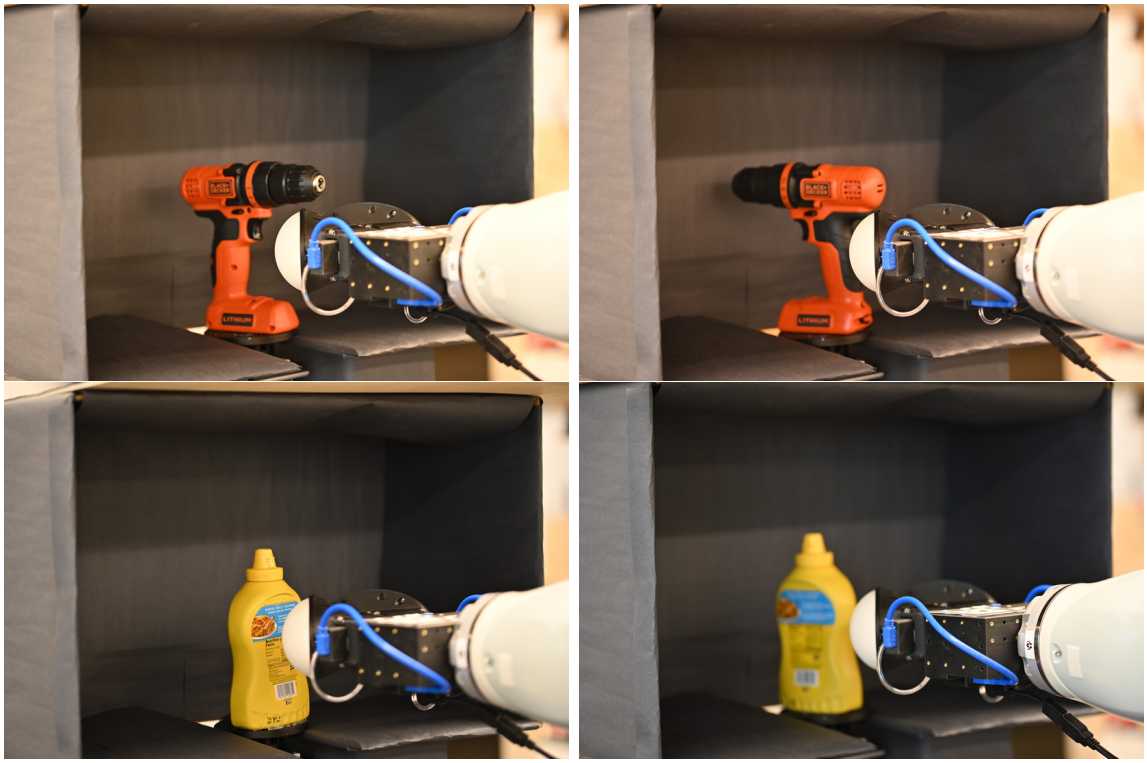

Figure 4.3: Real probing configurations for the drill and mustard.

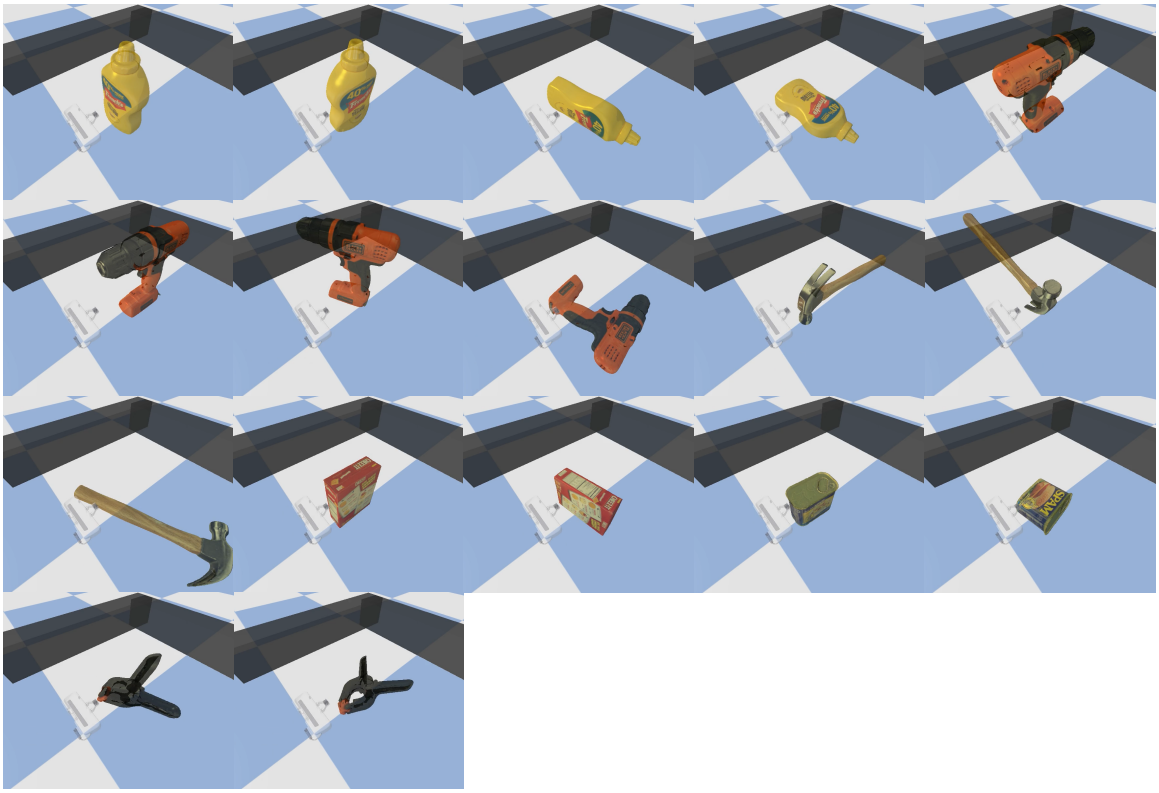

Figure 4.4: Simulated probing configurations for multiple YCB objects.

Figure 4.5: Unreliable RGBD readings inside the partially occluded cabinet, viewed from both sides, with an approximate pose of the mustard bottle in purple.

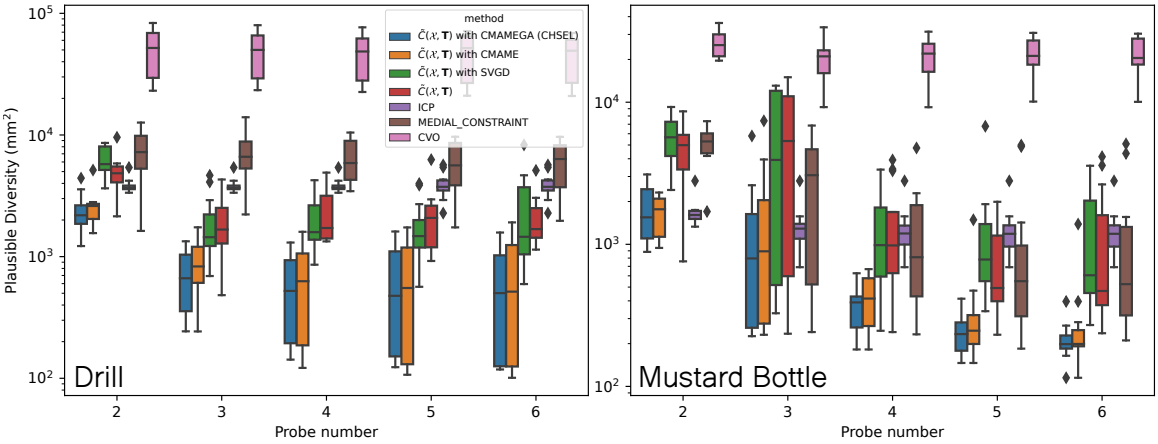


Figure 4.6: Plausible Diversity for real drill (left) and mustard (right) probing experiments across 2 configurations and 6 trials each. The bars indicate the 25 to 75 percentile while the whiskers are the min and max with outliers as diamonds. Lower is better.

### 4.5.1 Simulated Environment

We use PyBullet *Coumans and Bai* (2016–2021) to simulate a Franka Emika (FE) gripper (see Fig. 4.4) that is position controlled. The workspace is voxelized with resolution $r_f = 25mm$ and spans $[-0.1, 0.5] \times [-0.3, 0.3] \times [-0.075, 0.625]$ in meters. We label the boundary of the workspace as free space. The SDF is voxelized with resolution $r_t = 10mm$ with padding $\gamma = 50mm$. The robot sweeps out voxels in the workspace grid during its probing motions, and $\mathcal{X}_f$ is given by the center of the swept voxels. $\mathcal{X}_o$ is empty as we have have no sensors that detect non-surface occupancy, though such information can be added if available. $\mathcal{X}_k$ is given by the contact points, with each having semantics $s = 0$ since contact can only occur on

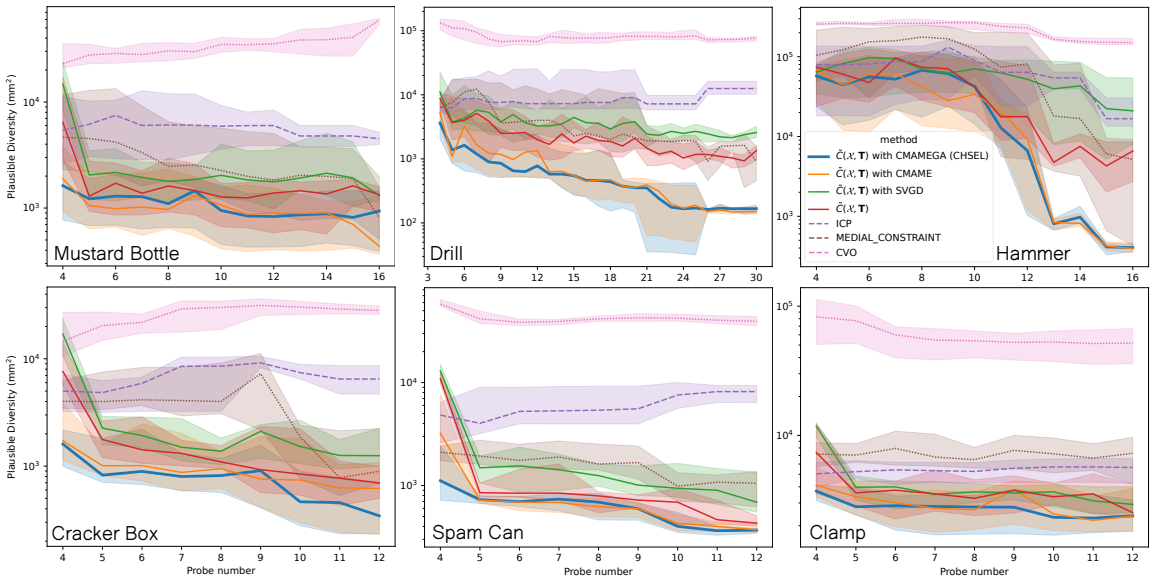


Figure 4.7: Plausible Diversity for simulated probing experiments across different YCB objects, with 2 to 4 configurations from Fig. 4.4 over 10 trials each. The median is in bold while the shaded region represents 25 to 75 percentile. Lower is better.

the object surface. Both the gripper and object are rigid and so only make single-point contacts which we retrieve from the simulator. For different trials, we seed the random number generator with different values.

### 4.5.2 Real Environment

For our real world experiment, we equip a 7DoF KUKA LBR iiwa arm with two soft-bubble tactile sensors *Kuppuswamy et al.* (2020) on the gripper (see Fig. 4.1 and Fig. 4.3). The soft-bubble sensors allow us to detect patch contact, which we consider as any point with deformation beyond $4mm$ and being in the top $10^{th}$ percentile of deformations. We use a mean filter to remove noise and downsample such that each contact produces at most 50 surface points.

The workspace is a physical cabinet mock-up and is voxelized with resolution $r_f = 10mm$, spanning $[0.7, 1.1] \times [-0.2, 0.2] \times [0.31, 0.6]$ meters. The SDF is voxelized with resolution $r_t = 5mm$ with padding $\gamma = 50mm$. In addition to populating $\mathcal{X}_f$ with robot swept volume, we utilize a RealSense RGBD camera, partially occluded by the cabinet. The camera is unreliable near occluding edges (see Fig. 4.5), thus we do not assume the object can be reliably segmented from the camera view, so we only use the free space information derived from the depth data. To that effect, we trace rays from the camera to 95% of each pixel's detected depth and add them to $\mathcal{X}_f$.

| Object | $\epsilon$ |
|---|---|
| Real Drill | 0.001 |
| Real Mustard Bottle | 0.0003 |
| Sim Drill | 0.001 |
| Sim Mustard Bottle | 0.0003 |
| Sim Hammer | 0.001 |
| Sim Cracker Box | 0.0005 |
| Sim Spam Can | 0.0003 |
| Sim Clamp | 0.0007 |

Table 4.1: $\epsilon$ used to generate the plausible set for each objects.

### 4.5.3 Computing Plausible Set

In order to evaluate our method, we need to compute $\mathcal{T}_\epsilon$, which is very computationally intensive. We compute $\mathcal{T}_\epsilon$ by densely sampling transforms around $\mathbf{T}^*$ and evaluate each using Eq. 4.2 with $c_m = 100000$. Specifically, we search over a grid spanning $[-0.1, 0.15] \times [-0.2, 0.2] \times [0, 0.1]$ meters with 15 cells along each dimension. We also uniformly random sample 10000 rotations which we combine with each translation cell. See Table 4.1 for the $\epsilon$ used to generate the plausible set of each object. They were selected such that most probe trials have around 30 members in $\mathcal{T}_\epsilon$ halfway through.

In simulation, we retrieve $\mathbf{T}^*$ from the simulator, while on the real robot we first manually specify an approximate pose, then search in two passes. The first pass searches around the specified pose to find the optimal transform, which is then used as $\mathbf{T}^*$ for the second pass.

### 4.5.4 Baselines

We compare against **ICP** as a weak baseline that does not use free space information. ICP registers the known surface points against another point set, which we provide as 500 points randomly sampled from the object surface. Note that these points are different for each trial. ICP is run until convergence.

Secondly, we compare against Continuous Visual Odometry (**CVO**) *Zhang et al.* (2021), the state of the art in semantic point set registration, and a continuous generalization of 3D NDT. We use 2 dimensional semantics to represent free points as $[0.9, 0.1]$ and surface points as $[0.1, 0.9]$. CVO registers the free and surface points against another semantic point set, which we provide as the center of the precomputed SDF voxels. Voxels with SDF value between $[-r_t, r_t]$ are labelled with surface

semantics, and voxels with SDF value greater than $r_t$ are labelled as free. Note that there are many more free points than surface points ($\approx 125:1$).

Next, we consider using Stein Variational Gradient Descent (**SVGD**) *Liu and Wang* (2016) of Eq. 4.7 to enforce diversity. We formulate $p(\mathbf{T}|\mathcal{X}) \propto e^{-\beta \hat{C}(\mathcal{X},\mathbf{T})}$, and select $\beta = 5$ as a scaling term for how peaked the distribution is. We have $|\hat{\mathcal{T}}_0|$ stein particles, each one initalized with a separate $\mathbf{T} \in \hat{\mathcal{T}}_0$, implicitly defining the prior. We use an RBF kernel with scale 0.01.

Lastly, we compare against *Haugo and Stahl* (2020), which forms free space constraints by covering the free space using balls along the volume's medial axis (we refer to this baseline as **Medial Constraint**). For each ball we have cost $\max(0, B_r - \texttt{sdf}(\tilde{B}_c))^2$ where $B_r$ is its radius and $B_c$ is the center position of the ball. For each surface point we have cost $\texttt{sdf}(\tilde{\mathbf{x}})^2$. The total cost is the sum of the mean ball cost and the mean point costs. We optimize this cost using CMA-ES *Hansen et al.* (2003), a gradient-free evolutionary optimization technique.

### 4.5.5 Ablations

We ablate components of our method starting with how useful the gradient is for accelerating QD optimization. Instead of CMA-MEGA, we use **CMA-ME** *Fontaine et al.* (2020) which does not explore using gradients.

We also consider just gradient descent on Eq. 4.7 to evaluate the value of additional optimization. We run Adam for 500 iterations with learning rate 0.01, reset to 0.01 every 50 iterations. These are also the parameters used for initializing CMA-ME and CMA-MEGA.

### 4.5.6 Probing Experiments

Qualitatively, we see the progress of a probing experiment and the elimination of hypothesis transforms through gaining known free space points in Fig. 4.8. From the initial probes along the back of the drill, it could take on many possible upright orientations. Note that after the probe in the second row, the contact points constrain the pose such that the contacts must lie on the back of the drill. As we probe the left side of the drill, without making contact, we eliminate transforms that would conflict with the new free space points. Probing the other side further narrows down the plausible transforms. Note the lack of diversity from the Medial Constraint baseline and the poor estimation from ICP since it cannot use free space information.

Fig. 4.6 summarizes the results of the real probing experiments on the YCB drill

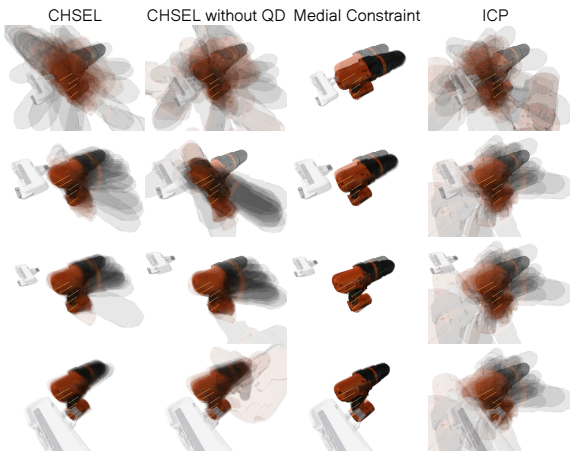


Figure 4.8: Reducing uncertainty in estimated pose as a result of additional free space points for selected methods, obtained by probing to the sides of the YCB drill. $\hat{\mathcal{T}}$ is represented as transformed copies of the mesh while contact points are drawn in orange, with the line indicating the direction of the probe.

and mustard bottle, each in two different configurations (see Fig. 4.3) over 6 trials. Fig. 4.7 summarizes the results of the simulation probing experiments on the YCB drill, mustard bottle, hammer, cracker box, spam can, and clamp. Additionally, we show the average time it takes for each method to perform registration on the real experiments in Fig. 4.9. This involves producing 30 transforms with $|\mathcal{X}| \in [13000, 21000]$, and $|\mathcal{X}_k| \in [0, 150]$. Note that all methods apart from CVO use parallelized implementations. Computations were performed on a NVIDIA GeForce RTX 2080 Ti with 11GB of VRAM.

From Fig. 4.6 and Fig. 4.7, we see that applying QD optimization to $\hat{C}$ in general outperforms baselines and the ablations. This is particularly true on more irregular objects such as the drill and hammer, and when we have noisy data in the real experiment. Even without QD optimization, gradient descent on $\hat{C}$ outperforms the Medial Constraint baseline. This may be due to the ability of CMA-ES to escape local minima, leading to low coverage, as seen in Fig. 4.8. All methods, including ICP, outperform CVO. We suspect this is due to the large imbalance of $|\mathcal{X}_f|$ to $|\mathcal{X}_k|$ ($\approx 125:1$). See Appendix 4.5.9 for an investigation of CVO's performance.

### 4.5.7 QD Method Comparison

We investigate the value of our formulated cost's differentiability by considering the QD optimization process in further detail. In Fig. 4.10, we compare the $\hat{\mathcal{T}}$ performance of using CMA-MEGA and CMA-ME as we increase the number of QD optimization iterations. The results are from the front-facing real drill experiment

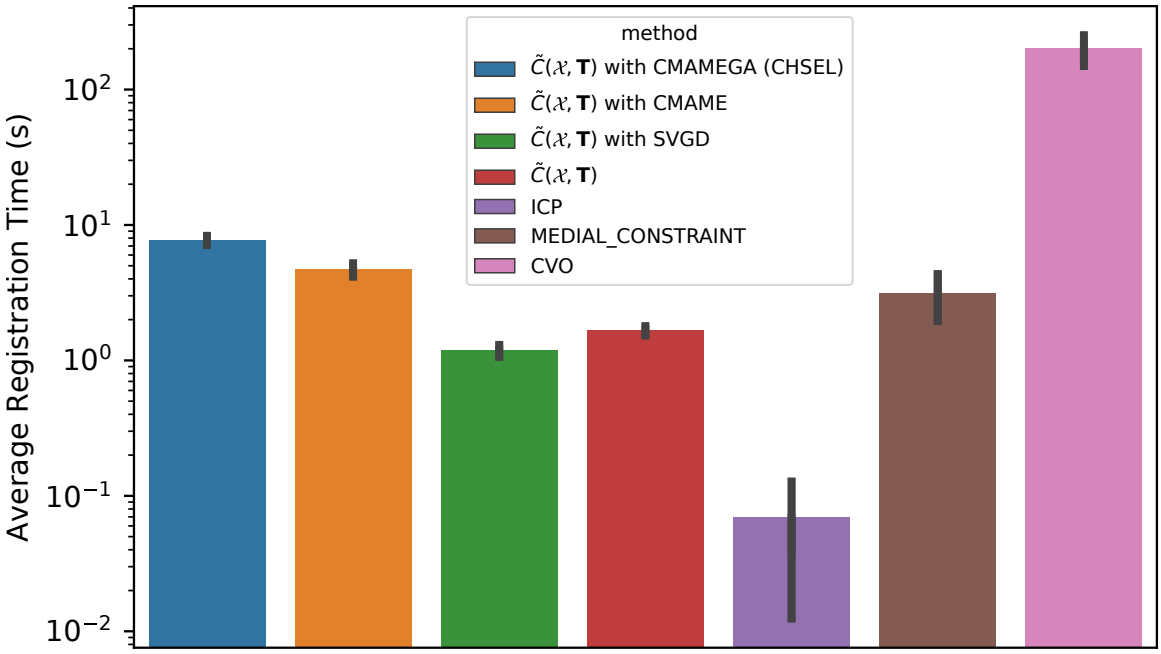


Figure 4.9: Average time per registration of 30 transforms on the real probing experiments. Error bars indicate one standard deviation.

(Fig. 4.3 top left), averaged over probes 5 and 6, and across the 6 trials. Both methods are initialized with the same $\hat{\mathcal{T}}_0$ and $\hat{\mathcal{T}}_l$ each trial and probe (see Algorithm 8). We see that CMA-MEGA is able to use our gradients to reach lower Plausible Diversity and average cost of the best cells in fewer iterations, and that they converge and reach parity after around 500 iterations (fewer in simulation due to lack of noise). In Fig. 4.6 and Fig. 4.7, both methods have run for enough iterations to converge. On average, each CMA-ME iteration takes 8.37ms while each CMA-MEGA iteration takes 11.5ms.

### 4.5.8 QD Objective Comparison

Lastly, we investigate how well QD optimization works with other objectives. We perform CMA-ME optimization using the Medial Constraint objective, with $\mathcal{B}$ initialized and sized from the $\hat{\mathcal{T}}$ estimated by Medial Constraint using CMA-ES. Fig. 4.11 shows results on the real mustard bottle experiments, where we see that while QD optimization improves the Medial Constraint performance, our method still significantly outperforms it. This demonstrates the value of $\hat{C}$ as a QD objective.

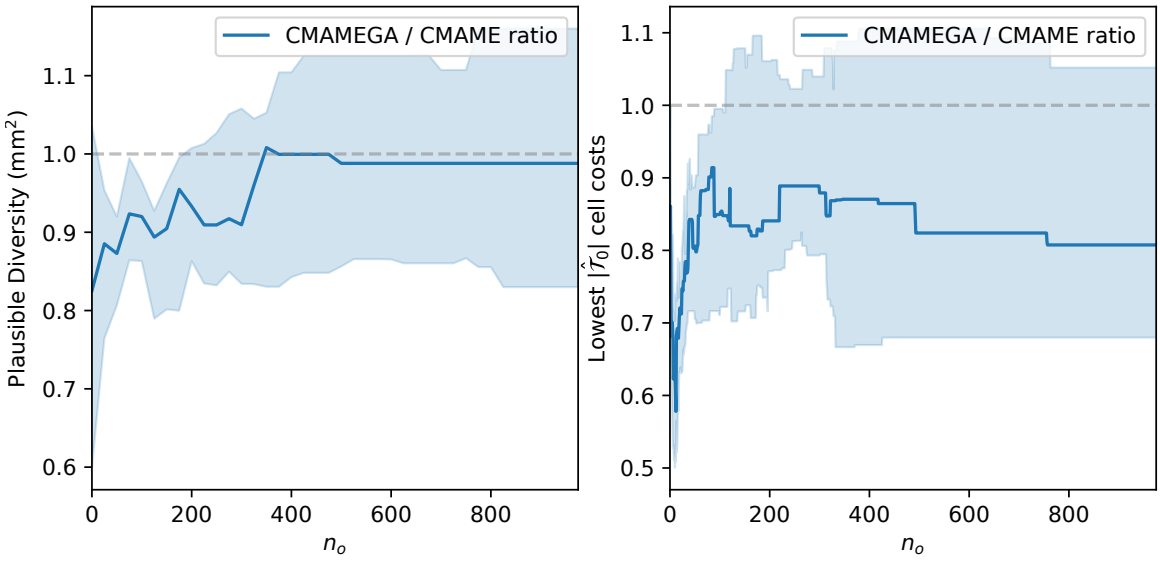


Figure 4.10: Comparison of QD optimization progress using $\hat{C}$ on the real drill experiment. Results are averaged across 6 trials and probe numbers 5 and 6. Median is in bold while the shaded region represents 25 to 75 percentile.

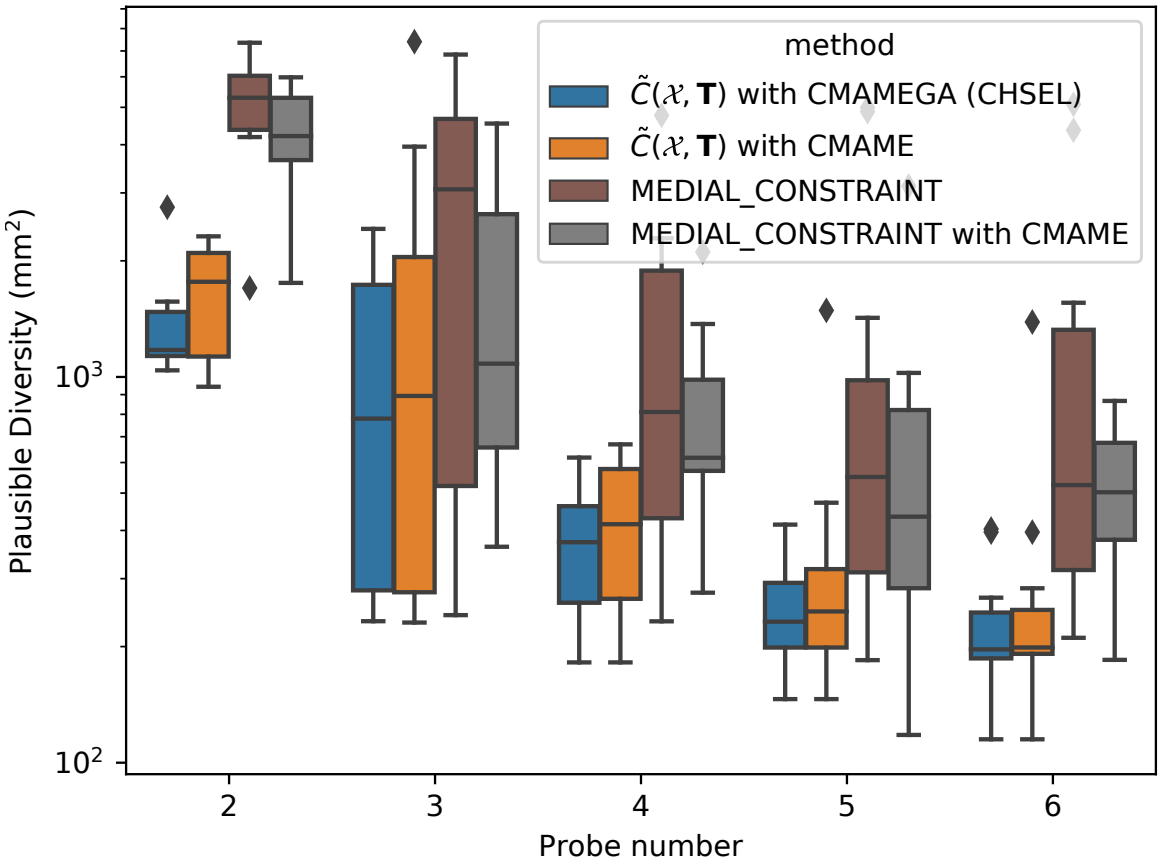


Figure 4.11: Plausible Diversity on the two real mustard bottle experiments with a focus on the improvement the Medial Constraint baseline receives from QD optimization.

### 4.5.9 CVO Performance

Qualitatively, we noticed that CVO's estimated transforms tend to place the object such that $\mathcal{X}_f$ is in concentrated regions of observed free points. To check if this free/surface imbalance was the cause of CVO's poor performance, we ran CVO while ignoring $\mathcal{X}_f$ on the real mustard bottle experiments (results shown in Fig. 4.12). We see that CVO performs comparably to ICP when ignoring $\mathcal{X}_f$. This is significantly better than when considering $\mathcal{X}_f$, suggesting that CVO is not able to effectively use the free space information.

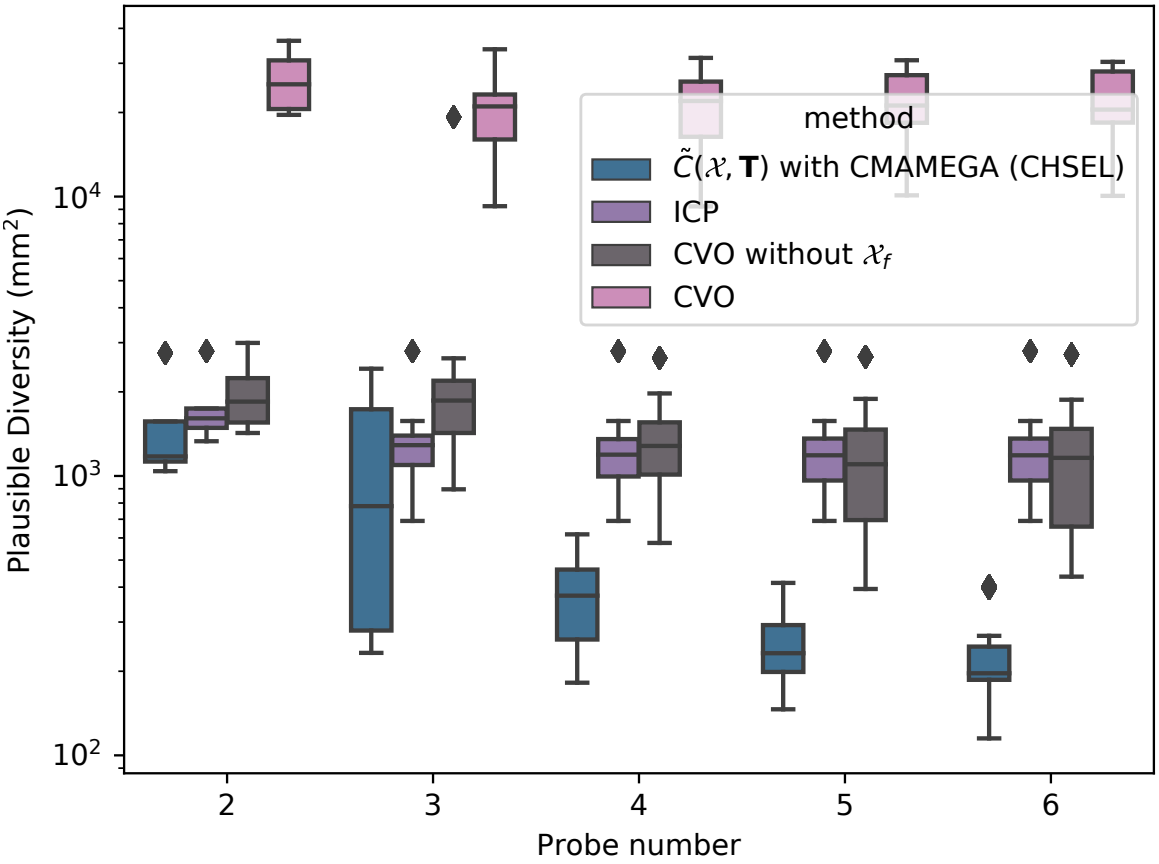


Figure 4.12: Plausible Diversity on the two real mustard bottle experiments with a focus on the improvement the CVO baseline receives from ignoring $\mathcal{X}_f$. Lower is better.

## 4.6 Discussion

In sequential registration problems such as our probing experiments, we assume that the object is stationary and that the updated semantic points are given. However, keeping the object still while probing it is not trivial, as every contact has the potential

to move the object. Rapid force and tactile feedback could minimize this issue. Contact could also be made with other objects during the probing motions. Contact point tracking and reasoning over object-contact associations is not within the scope of this chapter. However, in future work we will explore using a method such as STUCCO *Zhong et al.* (2022) to estimate object-contact associations and add the proper contacts to $\mathcal{X}$.

The experiments in this chapter used a fixed sequence of probing motions. This makes for a fair comparison between methods, since the sequence is not dependent on any method's pose estimates. However, in practice, the next probing motion should depend on the current pose estimate. In Chapter V we explore how to reason over the plausible set of poses and plan trajectories that efficiently disambiguate between them, so as to localize the object with as few probing motions as possible.

## 4.7 Conclusion

We presented CHSEL, a pose registration method that utilizes point semantics, such as whether a point is in free space or on the object surface, to impose additional constraints and reduce pose ambiguity. Rather than a single best estimate, it produces a set of diverse plausible estimates given the observed data. We showed that it performs well on both simulated and real data collected from robot probing experiments. In particular, we separately demonstrated the value of performing Quality Diversity (QD) optimization for registration, and the strength of our proposed differentiable cost function as a QD objective. Additionally, we showed how to update the estimated transform set online with updated data, that CHSEL performs well on data with few contact points, and that it is seamless to integrate vision as an input modality.

# CHAPTER V

# Rummaging Using Mutual Information

The previous chapter introduced a method for estimating an object pose and characterizing its uncertainty from a point cloud with volumetric semantics information. In the experiments, the probing trajectory was predetermined, and this work is to create a method to generating these probing trajectories. The goal is to do so in a way that gives the most information about the object's pose as possible within a certain number of actions. Concretely, it is to maximize the mutual information between a fixed-length trajectory and the object's pose distribution.

## 5.1 Introduction

Active exploration, the process of autonomously planning actions to gather more information about a target quantity, is a core problem in robotics, particularly when dealing with unknown environments *Bajcsy et al.* (2018). This problem encompasses a range of scenarios, differentiated by the type of robot (e.g., mobile vs. stationary), the primary sensor modality (often vision), and the specific quantity to be estimated.

As robotics applications have transitioned from known, structured environments like factories to the unknown, dynamic environments of homes, new challenges have emerged. One critical application area is object manipulation, where visual perception is often hindered by occlusions caused by both the environment and the objects themselves *Zhong et al.* (2023). To address these challenges, we focus on actively exploring to estimate the pose of a movable object with a known shape through contact-rich interactions, commonly referred to as rummaging.

Occlusions of the target object, both from itself and from other objects, motivate the need to use contact to determine the object's pose. Our prior work has investigated how to track the position of contact points during rummaging with an

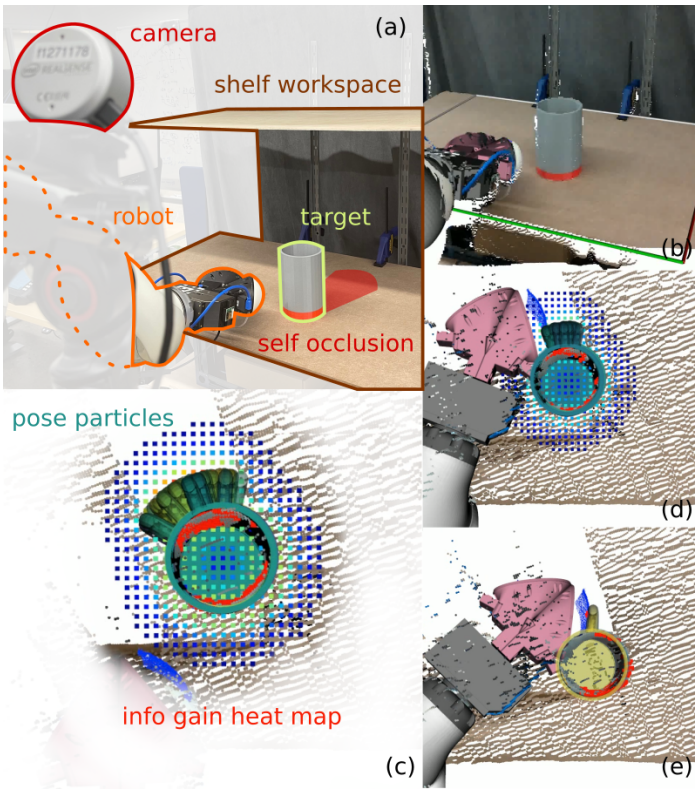


Figure 5.1: (a) Set up of a real-world active exploration experiment where the goal is to estimate the pose of a movable target object. The object pose remains ambiguous due to self occlusion. (b) The initial point cloud view of the scene from the camera perspective. (c) RUMI maintains a belief over object pose using a particle filter, where the pose particles are shown as overlaid objects with how red they are indicating likelihood. Observed surface points are in red. From the pose particles and observations, RUMI generates an information gain field to plan over, shown as a heat map. Only the most interesting workspace points are shown. (d) Even without making contact with the handle, the robot sweeps out free space that constrains the pose. (e) Finally making contact accurately estimates the mug pose.

unknown number of objects *Zhong et al.* (2022), and how to estimate the plausible set of object poses given observed contact and free space points *Zhong et al.* (2023). However, the problem of how to plan information-gathering trajectories to estimate a movable object's pose is still under-explored. A primary challenge is the object's mobility, coupled with the requirement for contact-based information collection. Without careful planning, making contact can inadvertently push the object out of the robot's workspace, as evidenced in our experiments.

Active exploration is often framed from an information-theoretic perspective, where the quantity to be estimated is treated as a random variable, and actions are selected to minimize its uncertainty. This approach can be computationally expen-

sive, necessitating a trade-off between accuracy and speed or limiting the exploration to a single next best action. Additionally, some methods restrict the action space to movements along the object's surface *Suresh et al.* (2022), *Driess et al.* (2017). While this restriction simplifies the problem, it also limits the robot's capabilities. Instead, we aim to enable robots to make and break contact dynamically throughout the rummaging process, enhancing their exploratory capabilities.

To address the above challenges, we present Rummaging Using Mutual Information (**RUMI**), an active exploration method. Specifically, our contributions are:

1. a framework for creating and updating a belief over poses given observed point clouds, augmented with volumetric semantics such as whether each point is in free space or on the surface of the object, based on the discrepancy formulated in CHSEL *Zhong et al.* (2023)

2. a measure of information gain based on the mutual information between the object pose and volumetric semantics at the positions that the robot trajectory will cover, and show that it can be efficiently computed in parallel for dense workspace points in real time

3. a closed loop MPC planning framework using cost functions based on the information gain and maintaining object reachability, and a stochastic object dynamics model

In our experiments, we show that RUMI is the only method to achieve consistent success in simulated and real robot rummaging tasks across various objects.

## 5.2 Related Work

In a broad sense, we focus on the problem of actively exploring an unknown environment to reduce the uncertainty of some quantity. There are many variants and names for the problem, including active sensing *Ryan and Hedrick* (2010), sensor path planning *Cai and Ferrari* (2009), active perception *Bajcsy et al.* (2018), and interactive perception *Bohg et al.* (2017). The variants differ primarily by the robot type (mobile vs stationary base), sensing modality, and by the quantity to be estimated; e.g. the map of the environment *Lluvia et al.* (2021), *Popović et al.* (2020), *Jadidi et al.* (2015), the shape of an object *Driess et al.* (2017), *Yi et al.* (2016), the pose of an object (object localization) *Danielczuk et al.* (2019), *Andreopoulos et al.* (2010), or the pose of effective grasps for objects *Kahn et al.* (2015), *Ottenhaus et al.* (2019).

In the case of unknown object shape, or reconstruction from a set of objects, the problem is also known as active shape completion *Rustler et al.* (2022). This paper focuses on estimating the pose of a movable rigid object with a known shape.

In general, active exploration is the iterative process of:

1. forming a belief over state given observations
2. computing expected information gain over a workspace
3. planning an action sequence
4. executing some of the action sequence and collecting observations

### 5.2.1 Representing Belief

Representations suitable for active exploration have been studied extensively. In many cases, parametric filters like the extended Kalman Filter (EKF) *Sim and Roy* (2005), *Leung et al.* (2006) may be used when the posterior of the quantity of measure should be approximately Gaussian. Otherwise, non-parametric methods like particle filters *Deng et al.* (2021), *Koval et al.* (2015) are often used. Occupancy grids have also been popular, e.g. used in the simultaneous localization and mapping (SLAM) variant of active exploration *Meyer-Delius et al.* (2012), *Vespa et al.* (2018), *Carrillo et al.* (2015). In particular, when assuming each grid cell is independent, information gain based on the entropy of all the cells may be efficiently computed on an occupancy grid. We make a similar assumption that enables efficient computation of our information gain.

Recently, Gaussian processes (GPs) *Jadidi et al.* (2015) have also been used for estimating object shape. GP implicit surfaces (GPIS) have shown strong representation power *Dragiev et al.* (2011), *Driess et al.* (2017). GPIS uses a GP to output a field in which the 0-level set represents the surface of the object. In our method, we do not need the full representation power of a GP since we have a known object shape. Instead, we use a particle filter to represent the pose distribution, and present a novel way to evaluate the particle probabilities given an observed point cloud.

### 5.2.2 Information Gain

The information gain can be formulated in many ways, often depending on the belief representation. For GPIS the variance of the GP *Driess et al.* (2017), or the differential entropy of the GP for adding a new data point *Driess et al.* (2019) can

be evaluated directly and used. However, despite work on geometric shape priors for GPIS *Martens et al.* (2016), there remains no satisfactory way to condition a GP on a known shape with unknown pose. We implement a GPIS baseline and condition it on the shape by augmenting the input data. Mutual information between observations and the estimated quantity is also common *Jadidi et al.* (2015), *MacDonald and Smith* (2019), which measures the reduction in uncertainty of the estimated quantity given the observations. Thus, we formulate our information gain function based on the mutual information between the object pose and the occupancy at points a robot trajectory would sweep out.

### 5.2.3 Planning

Searching for an optimally-informative trajectory is usually computationally intensive. GP-based methods in particular are limited by inference times that grow rapidly with increasing number of data points, often addressed by using sparse GPs or downsampling to trade off accuracy *Snelson and Ghahramani* (2005). Some methods greedily selects the optimal next configuration, and additionally constrain the action space to slide along the surface of the object *Suresh et al.* (2022), *Driess et al.* (2017). Our formulation of the information gain allows us to efficiently evaluate it for many query trajectories in parallel, enabling us to use longer-horizon planning methods such as sampling-based model predictive control in a closed loop. We consider difficult tasks which necessitate long horizon planning.

Active exploration problems also differs by sensing modality. In the context of object shape and pose estimation, the most common modality is visual perception, with the common framing of the problem as finding the next best view *Krainin et al.* (2011). Tactile approaches have also demonstrated success *Yi et al.* (2016), *Driess et al.* (2017), as well as hybrid approaches *Rustler et al.* (2022), *Smith et al.* (2021). Tightly coupled with sensing modality is the distinction of whether the robot is passively observing the environment or actively interacting with and changing the environment as in the interactive perception problem *Bohg et al.* (2017). RUMI is a hybrid approach for interactive perception, primarily relying on contact-rich interactions using tactile sensors, but also leveraging visual perception to initialize pose estimates. Visual perception in our case is weakened by environmental occlusion or object self-occlusion. Unlike most other methods for object pose or shape estimation, we do not assume the object is stationary, which accounts for a large part of the difficulty. The closest method to ours is Act-VH *Rustler et al.* (2022), which trains an implicit surface neural network to output hypothesis voxel grids of seen objects given

a partially observed point cloud and selects the best point to probe next. One major weakness of this method is the need to either retrain their network on all candidate objects whenever there is a new target object, or to train a network per object and assume object identity is known. Our method can be applied to new known objects without any training. Additionally, their object is in between the robot and visual perception, meaning that the visually-occluded region is highly reachable, bypassing a major challenge that we address. Lastly, we consider the information gain from full robot trajectories rather than a single next point to probe.

## 5.3 Problem Statement

Let $\mathbf{q} \in \mathbb{R}^{N_q}$ denote the robot configuration, and $\mathbf{u} \in \mathbb{R}^{N_u}$ denote control. We study a single robot exploring an unmodeled environment, using limited visual perception and contact-heavy rummaging to estimate the pose of a single movable rigid target object of known shape. A rigid object's configuration is defined by its pose, a transform $\mathbf{T} \in \mathrm{SE}(3)$. Every $\mathbf{T}$ can be identified with a $\mathbb{R}^{4\times4}$ homogeneous transformation matrix, and for convenience, we use $\tilde{\mathbf{x}} = \mathbf{T}\mathbf{x}$ to denote the homogeneous transform of point $\mathbf{x} \in \mathbb{R}^3$ from world frame coordinates to the object frame of $\mathbf{T}$ (homogeneous coordinates have 1 appended). There is an underlying dynamics function $f : \mathbb{R}^{N_q} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_q}$ that we do not know, but are given the free space dynamics function $f_f : \mathbb{R}^{N_q} \times \mathbb{R}^{N_u} \to \mathbb{R}^{N_q}$. The difference in dynamics is primarily due to contact between the robot and the target object. We are interested in generating a fixed length trajectory of $T$ actions, $\mathbf{u}_1, ..., \mathbf{u}_T$ to actively explore and estimate the target object's pose.

Specifically, we have the target object's precomputed object frame signed distance function (SDF) derived from its 3D model, $\texttt{sdf} : \mathbb{R}^3 \to \mathbb{R}$. After each action, sensors observe a set of points at time $t$ : $\mathcal{X}'_t = \{(\mathbf{x}_1, s_1), ..., (\mathbf{x}_N, s_N)\}_t$ with observed world positions $\mathbf{x}_n \in \mathbb{R}^3$ and semantics $s_n$ (described below). For convenience, we refer to a pair of position and semantics as a *geometric feature*. Let $\mathcal{X}_t$ denote the accumulated set of geometric features up to and including time $t$. Sensors may include but are not limited to robot proprioception, end-effector mounted tactile sensors, and external cameras.

We treat the pose of the target object as a random variable and define $p(\mathbf{T}|\mathcal{X}_t)$ as the posterior probability distribution over poses given $\mathcal{X}_t$. Observation noise, object symmetry, and the partial nature of $\mathcal{X}_t$ results in pose uncertainty.

Let $\mathbf{T}^*$ be the true object transform, then the observed semantics are

$$s_n = \begin{cases} \texttt{free} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) > 0 \\ \texttt{occupied} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) < 0 \\ \texttt{surface} & \text{implies } \texttt{sdf}(\mathbf{T}^*\mathbf{x}_n) = 0 \end{cases}$$

For a workspace point $\mathbf{x}$ that we have not observed, its semantics is a discrete random variable $S_\mathbf{x}$ with the shorthand $p(S_\mathbf{x}) = p(s|\mathbf{x})$. We are given a sensor model $p(S_\mathbf{x}|\mathbf{T}) = p(S_\mathbf{x}|\texttt{sdf}(\mathbf{T}\mathbf{x}))$ such as in Fig. 5.2 that gives the probability of observing each $s$ value given an SDF value. The sensor model does not consider uncertainty over the position, and we assume we are given exact positions with only uncertainty over semantics $S_\mathbf{x}$.

Given a prior $p(\mathbf{T})$, and starting at $\mathbf{q}_1$, our goal is to estimate the pose of the object by maximizing the expected information gain after $T$ actions:

$$\begin{aligned} \underset{\mathbf{u}_1,\ldots,\mathbf{u}_T}{\arg\max} \quad & \mathbb{E}_{\mathcal{X}_T}[D_{KL}(p(\boldsymbol{\theta}|\mathcal{X}_T)||p(\boldsymbol{\theta}))] \\ \text{s.t.} \quad & \mathbf{q}_{t+1} = f(\mathbf{q}_t, \mathbf{u}_t),\ t = 1, ..., T \end{aligned} \tag{5.1}$$

The expectation is over the semantics of each point in $\mathcal{X}_T$. Note that this is equivalent to the mutual information between $\mathbf{T}$ and $\mathcal{X}_T$, $I(\mathbf{T}; \mathcal{X}_T)$ *Murphy* (2012).

The challenge of this problem comes from the need for contact-based perception due to limited sensing capabilities, coupled with the fact that the target object is movable. Moreover, an ineffective action sequence can result in undesirable contacts, potentially pushing the object out of the robot's reachable workspace.

We evaluate the quality of the estimated pose distribution by evaluating the likelihood of the ground truth pose $\mathcal{L}(\mathbf{T}^*|\mathcal{X}_t)$, or equivalently its negative log likelihood (NLL). Low NLL indicates both certainty and correctness of the pose distribution. We do so by sampling a set of surface points in the object frame and transforming them by $\mathbf{T}^*$ to produce world positions $\mathbf{X}$. We then evaluate the NLL of all of the points having `surface` semantics:

$$nll(\mathcal{X}) = -\log p(\bigcap_{\mathbf{x}\in\mathbf{X}} S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{5.2}$$

We use this metric as well as computational efficiency to evaluate our method against baselines and ablations.

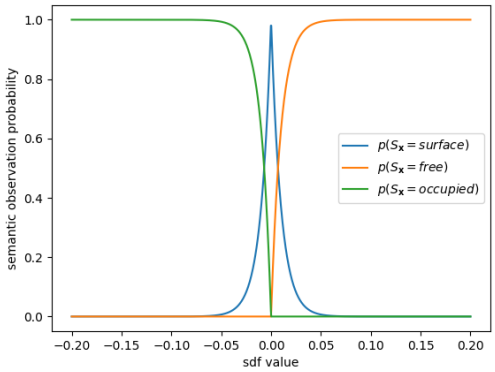


Figure 5.2: Example sensor model that gives a probability of observing each semantics class given a SDF value.

## 5.4 Method

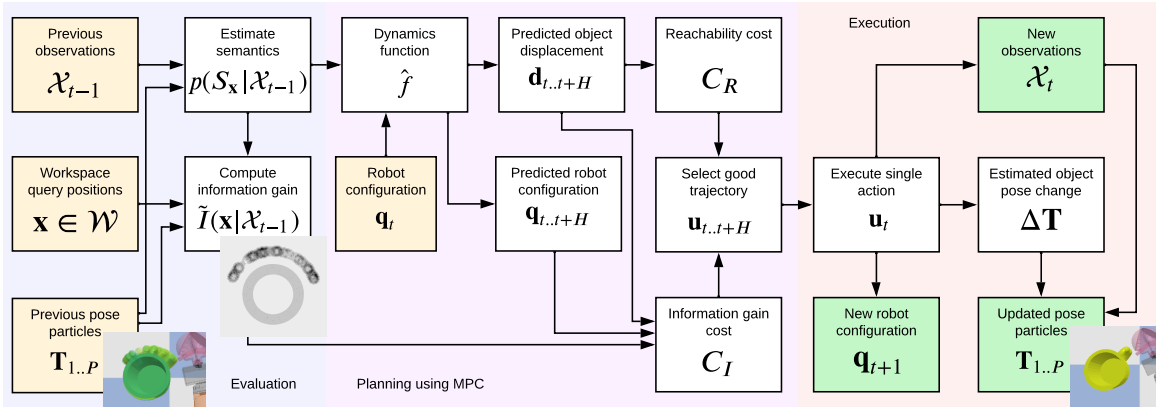


Figure 5.3: Flow chart showing one time step of RUMI's approach for solving Eq. 5.1. Beige blocks are inputs to this time step while green ones are outputs of this time step. The process is also into evaluating current information gain, planning into the future using the information gain, and executing one step of the plan and updating observations.

Our high level approach to addressing the problem in Eq. 5.1 is depicted in Fig. 5.3. We represent the pose posterior $p(\mathbf{T}|\mathcal{X})$ with a particle filter and describe how to evaluate $p(\mathbf{T}|\mathcal{X})$. Next, we present a tractable surrogate for information gain that we develop into a cost function for model predictive control (MPC). To discourage trajectories that move the target object out of the robot's reachable area, we develop an additional reachability cost function. Furthermore, to estimate the displacement of the target object given an action trajectory, we implement a stochastic dynamics model $\hat{f}$. We use the cost functions and the dynamics function inside MPC, which

executes in a closed loop for $T$ steps. During this process, we detail how to merge current observations with previous ones and update the pose posterior $p(\mathbf{T}|\mathcal{X})$.

### 5.4.1 Representing Pose Posterior

We maintain a belief over the pose posterior $p(\mathbf{T}|\mathcal{X}_t)$ using a particle filter, where each particle is a pose. We have $N$ particles $\mathbf{T}_{1..N}$, with weights $w_{1..N}$ such that $\sum_{i=1}^{N} w_i = 1$. Our choice of a particle filter over alternative representations is motivated by the potential multi-modality of the posterior and the ability to process each particle in parallel.

A major obstacle to the tractability of solving Eq. 5.1 is the information correlation between geometric features. Observing one decreases the information gain from others in a non-trivial manner, and it is a common long-standing assumption to consider the information gain from each point independently *Cao et al.* (2013), *Stachniss and Burgard* (2003). Thus, we assume the conditional mutual independence of $S_{\mathbf{x}}$ for all query positions $\mathbf{x}$ given observed $\mathcal{X}_t$.

Critical to our method is a way to evaluate the posterior $p(\mathbf{T}|\mathcal{X})$. Our prior work CHSEL *Zhong et al.* (2023) formulated a differentiable cost function $\hat{C}(\mathcal{X}, \mathbf{T})$ that evaluates the discrepancy between $\mathcal{X}$ and $\mathbf{T}$. It bears similarity to hydroelastic, or pressure field contact modelling *Elandt et al.* (2019), *Masterjohn et al.* (2022), except in addition to the pressure field penalizing object penetration, there are pressure fields that penalize semantics violation, such as observed free space geometric features being inside objects.

We simplify the third semantics class from CHSEL, which represented known SDF of any value. We restrict it to $s = 0$, which refers to surface points. The cost is formulated by first partitioning the observed $\mathcal{X}$ into $\mathcal{X}_f = \{(\mathbf{x}, s) \mid s = \texttt{free}\}$, $\mathcal{X}_o = \{(\mathbf{x}, s) \mid s = \texttt{occupied}\}$, and $\mathcal{X}_s = \{(\mathbf{x}, s) \mid s = \texttt{surface}\}$.

$$\hat{C}(\mathcal{X}, \mathbf{T}) = \sum_{\mathbf{x}, s \in \mathcal{X}_f} \hat{c}_f(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s \in \mathcal{X}_o} \hat{c}_o(\tilde{\mathbf{x}}) + \sum_{\mathbf{x}, s \in \mathcal{X}_s} \hat{c}_k(\tilde{\mathbf{x}}, 0) \tag{5.3}$$

$$\hat{c}_f(\tilde{\mathbf{x}}) = C \max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}})) \tag{5.4}$$

$$\hat{c}_o(\tilde{\mathbf{x}}) = C \max(0, \alpha + \texttt{sdf}(\tilde{\mathbf{x}})) \tag{5.5}$$

$$\hat{c}_k(\tilde{\mathbf{x}}, s) = |\texttt{sdf}(\tilde{\mathbf{x}}) - s| \tag{5.6}$$

where $C > 0$ is a scaling parameter and $\alpha > 0$ allows for small degrees of violation due to uncertainty in the positions.

Their gradients are defined as

$$\nabla \hat{c}_f(\tilde{\mathbf{x}}) = C \max(0, \alpha - \texttt{sdf}(\tilde{\mathbf{x}}))(-\nabla \texttt{sdf}(\tilde{\mathbf{x}})) \tag{5.7}$$

$$\nabla \hat{c}_o(\tilde{\mathbf{x}}) = C \max(0, \alpha + \texttt{sdf}(\tilde{\mathbf{x}}))\nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{5.8}$$

$$\nabla \hat{c}_k(\tilde{\mathbf{x}}, s) = (\texttt{sdf}(\tilde{\mathbf{x}}) - s)\nabla \texttt{sdf}(\tilde{\mathbf{x}}) \tag{5.9}$$

where $\nabla \texttt{sdf}(\tilde{\mathbf{x}})$ is the object SDF gradient with respect to an object-frame position $\tilde{\mathbf{x}}$ and normalized such that $\|\nabla \texttt{sdf}(\tilde{\mathbf{x}})\|_2 = 1$.

Similar to energy-based methods, we use the Boltzmann distribution *Haarnoja et al.* (2017), *Teh et al.* (2003) to interpret Eq. 5.3 as the posterior pose probability:

$$p(\mathbf{T}|\mathcal{X}) = \eta e^{-\lambda \hat{C}(\mathcal{X}, \boldsymbol{\theta})} \tag{5.10}$$

where $\lambda > 0$ selects how peaky the distribution should be and $\eta$ is the normalization constant such that $\int \eta e^{-\lambda \hat{C}(\mathcal{X}, \boldsymbol{\theta})} d\mathbf{T} = 1$.

We observe that the cost in Eq. 5.3 is additive in the sense

$$\hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \mathbf{T}) = \hat{C}(\mathcal{X}, \mathbf{T}) + \hat{C}((\mathbf{x}, s), \mathbf{T}) \tag{5.11}$$

This is an important property that enables us to efficiently evaluate information gain of all workspace positions in parallel.

### 5.4.2 Mutual Information Surrogate

Our conditional mutual independence assumption of $p(S_{\mathbf{x}}|\mathcal{X})$ lets us consider the information gain from knowing the semantics at a single new position, which we denote the *information gain field* $\tilde{I}(\mathbf{x}|\mathcal{X})$. This is much simpler than considering the information gain of a robot trajectory directly because there is no time component or correlation between the semantics of neighbouring geometric features. Suppose we have observed $\mathcal{X}$ and want to evaluate the information gain from observing some new geometric feature $(\mathbf{x}, s)$. Note that here we are querying a specific given value of $\mathbf{x}$, but $S_{\mathbf{x}}$ is still a random variable, so the expectation is over $p(s|\mathcal{X}, \mathbf{x}) = p(S_{\mathbf{x}}|\mathcal{X})$:

$$I_f(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s \sim p(S_{\mathbf{x}}|\mathcal{X})} [D_{KL}(p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))||p(\boldsymbol{\theta}|\mathcal{X}))] \tag{5.12}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))} [\log \frac{p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))}{p(\boldsymbol{\theta}|\mathcal{X})}]] \tag{5.13}$$

The forward KL divergence results in an expectation over $p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))$. Since we need to evaluate the information gain for many positions in the workspace, this becomes intractable.

To address this challenge, we use the reverse KL divergence, since the expectation is over $p(\boldsymbol{\theta}|\mathcal{X})$ for all queried positions. In general, KL divergence is not symmetric. However, when two distributions are close together the KL divergence is approximately symmetric *Zhang et al.* (2024), *Kullback* (1997). In our case the KL divergence is between $p(\boldsymbol{\theta}|\mathcal{X})$ and $p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))$ with all having SE(3) support, avoiding infinite divergences. As we increase $|\mathcal{X}|$ during exploration, we expect the two distributions to become closer and the reverse KL to better approximate the forward KL divergence.

Intuitively, a geometric feature has high reverse KL divergence if it has high $p(\boldsymbol{\theta}|\mathcal{X})$ and low $p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))$. These correspond to geometric features that would invalidate currently high-probability poses i.e. these are positions we would like to explore.

Using reverse KL, We now have

$$I_r(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s}[D_{KL}(p(\boldsymbol{\theta}|\mathcal{X})||p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s)))] \tag{5.14}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{X})}[\log \frac{p(\boldsymbol{\theta}|\mathcal{X})}{p(\boldsymbol{\theta}|\mathcal{X} \cup (\mathbf{x}, s))}]] \tag{5.15}$$

Substituting Eq. 5.10 in

$$I_r(\mathbf{x}|\mathcal{X}) = \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\log \frac{p(\mathcal{X}|\mathbf{T})}{p(\mathcal{X} \cup (\mathbf{x}, s)|\mathbf{T})}]] \tag{5.16}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta}}[\log \frac{\eta_1 e^{-\lambda \hat{C}(\mathcal{X}, \boldsymbol{\theta})}}{\eta_2 e^{-\lambda \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \boldsymbol{\theta})}}]] \tag{5.17}$$

$$= \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta}}[\log \frac{e^{-\lambda \hat{C}(\mathcal{X}, \boldsymbol{\theta})}}{e^{-\lambda \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \boldsymbol{\theta})}}]] + \log \frac{\eta_1}{\eta_2} \tag{5.18}$$

$$= \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta}}[-\hat{C}(\mathcal{X}, \boldsymbol{\theta}) + \hat{C}(\mathcal{X} \cup (\mathbf{x}, s), \boldsymbol{\theta}))] + \log \frac{\eta_1}{\eta_2} \tag{5.19}$$

where $\eta_1$ and $\eta_2$ are the normalizing constants for $p(\mathbf{T}|\mathcal{X})$ and $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$, respectively. First we simplify using the additive property of $\hat{C}$ (Eq. 5.11) then consider the normalizing constants,

$$I_r(\mathbf{x}|\mathcal{X}) = \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\mathbf{T}}[\hat{C}((\mathbf{x}, s), \boldsymbol{\theta})]] + \log \frac{\eta_1}{\eta_2} \tag{5.20}$$

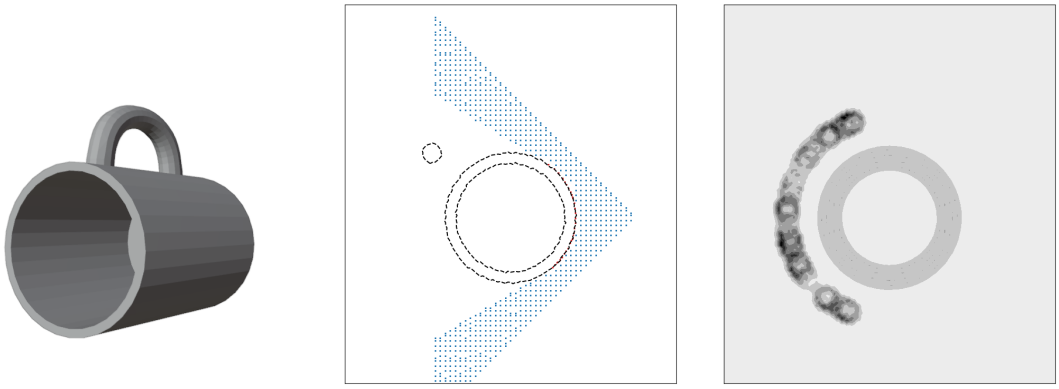

Figure 5.4: (left) Example mug object with (middle) $\mathcal{X}$ rendered from a pinhole camera on one side, not seeing where the handle is. The ground truth object surface is outlined in dotted black, observed surface geometric features are in red, and observed free space is in blue. (right) The information gain field estimated with $N = 100$ is darker where there is more information, where the handles could be.

We note that $\eta_2$ depends on the querying position $\mathbf{x}$ because each $\mathbf{x}$ induces a different $p(\mathbf{T}|\mathcal{X} \cup (\mathbf{x}, s))$. This normalizing constant is intractable to compute because it involves an integral over $\mathbf{T}$, so we instead optimize the approximation

$$\tilde{I}(\mathbf{x}|\mathcal{X}) = \lambda \mathop{\mathbb{E}}_{s}[\mathop{\mathbb{E}}_{\boldsymbol{\theta}}[\hat{C}((\mathbf{x}, s), \boldsymbol{\theta})]] \tag{5.21}$$

$$= \lambda \sum_{s} p(S_{\mathbf{x}} = s|\mathcal{X}) \mathop{\mathbb{E}}_{\boldsymbol{\theta}}[\hat{C}((\mathbf{x}, s), \boldsymbol{\theta})] \tag{5.22}$$

Selecting $\lambda$ too high leads to the pose particle weights dominated by a few, causing particle degeneracy.

We approximate the expectation over the posterior by taking the weighted sum over the pose particles

$$\tilde{I}(\mathbf{x}|\mathcal{X}) \approx \lambda \sum_{s} \sum_{i=1}^{N} p(S_{\mathbf{x}} = s|\mathcal{X}) w_i \hat{C}((\mathbf{x}, s), \mathbf{T}_i) \tag{5.23}$$

Finally, we consider how we can approximate the conditional semantics distribution $p(S_{\mathbf{x}}|\mathcal{X})$ which is the last term required for fully computing $\tilde{I}(\mathbf{x}|\mathcal{X})$. We use the law of total probability

$$p(S_{\mathbf{x}}|\mathcal{X}) = \int p(S_{\mathbf{x}}|\boldsymbol{\theta}, \mathcal{X}) p(\mathbf{T}|\mathcal{X}) d\mathbf{T} \tag{5.24}$$

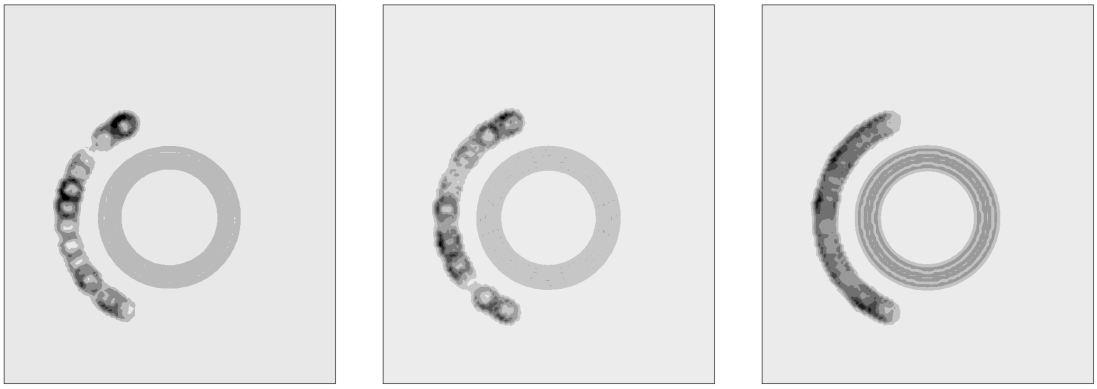

Figure 5.5: Computed information gain field $\tilde{I}(\mathbf{x}|\mathcal{X})$ from Fig. 5.4 with (left) $N = 30$, (middle) $N = 100$, and (right) $N = 1000$.

Here again we approximate the expectation over the posterior by taking the weighted sum over the pose particles

$$p(S_{\mathbf{x}}|\mathcal{X}) \approx \sum_{i=1}^{N} w_i p(S_{\mathbf{x}}|\mathbf{T}_i, \mathcal{X}) \tag{5.25}$$

we assume the conditional independence of $S_{\mathbf{x}}$ and $\mathcal{X}$ when given $\mathbf{T}$, so

$$p(S_{\mathbf{x}}|\mathcal{X}) \approx \sum_{i=1}^{N} w_i p(S_{\mathbf{x}}|\mathbf{T}_i)) \tag{5.26}$$

$$= \sum_{i=1}^{N} w_i p(S_{\mathbf{x}}|\texttt{sdf}(\mathbf{T}_i\mathbf{x})) \tag{5.27}$$

where $p(S_{\mathbf{x}}|\texttt{sdf}(\mathbf{T}_i\mathbf{x}))$ is given by the sensor model.

Note that all the terms in Eq. 5.23 only query $\mathbf{x}$ and $\mathbf{T}_{1..N}$, without needing to directly consider $\mathcal{X} \cup (\mathbf{x}, s)$. This enables us to evaluate $\tilde{I}(\mathbf{x}|\mathcal{X})$ for all positions inside a workspace $\mathbf{x} \in \mathcal{W} \subset \mathbb{R}^3$ in parallel.

### 5.4.3 Illustrative Example

To develop intuition, we consider a mug as the target object, depicted in Fig. 5.4 (left). Initially, a camera observes one side of the mug, narrowing down its position. However, since it cannot observe the handle and there is partial rotational symmetry, there is uncertainty in the orientation of the object. Fig. 5.4 (right) is the computed information gain $\tilde{I}(\mathbf{x}|\mathcal{X})$ over the entire workspace, showing that most of the

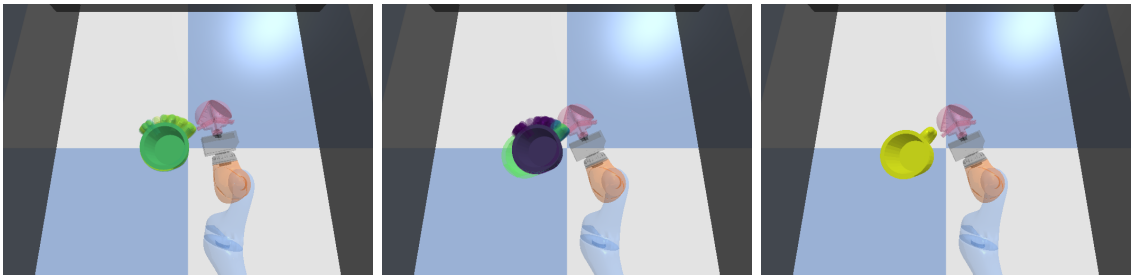

Figure 5.6: Resampling during a pybullet simulated mug task with partial initial observation like in Fig. 5.4. (left) Before contact pose particles, and (middle) pose particles after contact with many receiving low likelihood indicated by the dark color due to discrepancy with the newly observed surface geometric features. This leads to resampling, and (right) resampled particles that are all high likelihood and centered around the ground truth pose.

information gain is concentrated where the handle could be.

Intuitively, we expect a smooth dark band where the handles could be but observe unevenness. This is due to the approximation error of $p(\boldsymbol{\theta}|\mathcal{X})$ being represented by finitely many pose particles and the approximation of $I_r$ with $\tilde{I}$. This is illustrated by Fig. 5.5. With larger $N$, the trade off for gaining a more accurate approximation of $p(\boldsymbol{\theta}|\mathcal{X})$ is increased memory usage. Since we process the particles and query positions in parallel, memory becomes the bottleneck and they have to be processed in batches, turning the memory trade off into a runtime one.

### 5.4.4 Posterior Update Process

So far, we have developed the information gain field given some observed $\mathcal{X}$ at one time step. We now describe the active rummaging process in Algorithm 10 to update the posterior.

Before any actions, we are given the pose prior $p(\mathbf{T})$ in the form of $N$ initial poses $\mathbf{T}_{0,1..N}$. Note that given a fixed set of geometric features $\mathcal{X}$, the posterior probability of poses can be compared using Eq. 5.10. With the relative posterior probability and samples from the prior, we can theoretically draw samples from the posterior using techniques such as Markov Chain Monte Carlo (MCMC) *Geyer* (1992), *Casella and George* (1992). However, MCMC tend to struggle with the high dimensionality of poses ($\mathbf{T} \in \mathrm{SE}(3)$). With the interpretation of Eq. 5.3 as the posterior (Eq. 5.10), optimization of Eq. 5.3 on prior pose particles can naturally be interpreted as approximately sampling from the posterior. Thus we apply CHSEL (Algorithm 1 from *Zhong et al.* (2023)) to produce the initial pose particles in Algorithm 10 line 2. CHSEL performs Quality Diversity (QD) optimization *Pugh et al.* (2016) on Eq. 5.3 to find

**Algorithm 10:** Particle filter posterior update

**Given:** $N$ number of particles,
$\mathbf{T}_{0,1..N}$ initial poses,
$\mathbf{q}_1$ initial robot configuration,
$\sigma_t$ translation noise,
$\sigma_R$ rotation noise,
$l_r$ resample discrepancy threshold,
$\mathcal{W}$ workspace set of query positions

1 $\mathcal{X}_0 \leftarrow$ sensors observe at $\mathbf{q}_1$
2 $\mathbf{T}_{1..N} \leftarrow \texttt{CHSEL}(\mathbf{T}_{0,1..N}, \mathcal{X}_0)$
3 $w_{1..N} \leftarrow \texttt{WeighParticles}(\mathbf{T}_{1..N}, \mathcal{X}_0)$
4 **for** $t \leftarrow 1$ **to** $T$ **do**
5 compute $p(S_\mathbf{x}|\mathcal{X}_{t-1})$ using Eq. 5.27 and $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ using Eq. 5.23 for $\mathbf{x} \in \mathcal{W}$ and cache in voxel grids
6 $\mathbf{u}_t \leftarrow \texttt{Plan}(\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1}), p(S_\mathbf{x}|\mathcal{X}_{t-1}), \mathbf{q}_t)$
7 robot executes action $\mathbf{u}_t$ to arrive at $\mathbf{q}_{t+1}$
8 $\mathcal{X}'_t \leftarrow$ sensors observe at $\mathbf{q}_{t+1}$
9 $\Delta\mathbf{T}_r \leftarrow$ sensors observe change in robot end-effector pose while in contact
10 $\Delta\mathbf{T}, \Delta\mathbf{T}_w \leftarrow \texttt{ObjMove}(\mathcal{X}_{t-1}, \mathcal{X}'_t, \mathbf{T}_{1..N}, \Delta\mathbf{T}_r)$
11 **if** $\Delta\boldsymbol{T}$ *not* $\mathbf{0}$ **then**
// predict step
12 **for** $i \leftarrow 1$ **to** $N$ **do**
13 $\Delta\mathbf{T}_\sigma \leftarrow \texttt{PerturbTransform}(\sigma_t, \sigma_R)$ $\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$
14 $\mathcal{X}_t \leftarrow \texttt{MergeObs}(\mathcal{X}_{t-1}, \mathcal{X}'_t, \mathbf{T}_{1..N}, \Delta\mathbf{T}_w)$
// update step, even when not in contact
15 $l_{1..N} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$
16 **if** $\max(l_{1..N}) > l_r$ **then**
17 $\mathbf{T}_{1..N} \leftarrow \texttt{Resample}(\mathbf{T}_{1..N}, w_{1..N}, \mathcal{X}_t)$
18 $w_{1..N} \leftarrow 1/N$
19 **else**
20 $w_{1..N} \leftarrow \texttt{WeighParticles}(\mathbf{T}_{1..N}, \mathcal{X}_t)$

poses that have low discrepancy while maintaining diversity across some measure of pose space. We use the orientation component of $\mathbf{T}$, or just the yaw when restricting the pose search space to SE(2) as the measure.

We then assign weights to each pose particle as described in Algorithm 11. These weights represent the relative posterior probability of each particle. We normalize the weights so that $\sum_{i=1}^{N} w_i = 1$. Normalizing is important so that the use of weights in approximating expectations over the pose posterior in Eq. 5.23 and Eq. 5.25 remain valid. A side benefit of normalization is that we can omit the normalizing constant $\eta$ from Eq. 5.10 in Algorithm 11 line 2.

**Algorithm 11:** `WeighParticles` assign pose particle weights

**Given:** $\mathbf{T}_{1..N}$ pose particles,
$\mathcal{X}$ set of observed geometric features,
$\lambda$ peakiness

1 $l_{1..N} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$
2 $w_{1..N} \leftarrow e^{-\lambda l_{1..N}}$
3 $w_{1..N} \leftarrow w_{1..N} / \sum_{i=1}^{N} w_i$ `// normalize sum to 1`

Then for each time step $t$, we first compute $p(S_{\mathbf{x}}|\mathcal{X}_{t-1})$ and $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ using Eq. 5.27 and Eq. 5.23, respectively, for $\mathbf{x} \in \mathcal{W}$ and cache the results in voxel grids. These voxel grids allow linear interpolation querying and return 0 for $\tilde{I}(\mathbf{x}|\mathcal{X}_{t-1})$ and `free` for $p(S_{\mathbf{x}}|\mathcal{X}_{t-1})$ when $\mathbf{x}$ is outside $\mathcal{W}$. They are used to plan a robot trajectory, as described in Subsection 5.4.5. The robot executes the first action in the planned trajectory and sensors observe both a new set of geometric features $\mathcal{X}'_t$ and the change in robot end effector pose while in contact $\Delta\mathbf{T}_r$.

In Algorithm 10 line 10 we estimate the change in pose $\Delta\mathbf{T}$ of the target object given $\mathcal{X}_{t-1}$, $\mathcal{X}'_t$, $\mathbf{T}_{1..N}$, and $\Delta\mathbf{T}_r$. Some end-effectors can either enforce sticking contact *Jaiswal and Kumar* (2017) or measure slip (such as in *Melchiorri* (2000), *Romeo and Zollo* (2020)) to estimate $\Delta\mathbf{T}$ directly. Not all robots have these sensors, so we present an optimization based method in Algorithm 14. The main idea is to find a $\Delta\mathbf{T}$ that transforms $\mathcal{X}_{t-1}$ such that it is consistent with the most recently observed $\mathcal{X}'_t$. Our prior is that contact was sticking; that is $\Delta\mathbf{T} = \Delta\mathbf{T}_r$ in Algorithm 14 line 1. We select a representative pose particle $\mathbf{T}_i$ with the lowest discrepancy to apply $\Delta\mathbf{T}$ to. For $N_o$ optimization steps, we evaluate $\hat{C}$ on $\mathcal{X}'$ and the hypothesis new pose $\Delta\mathbf{T} \cdot \mathbf{T}_i$. $\hat{C}$ is differentiable with respect to $\Delta\mathbf{T} \cdot \mathbf{T}_i$, and we back propagate gradients to $\Delta\mathbf{T}$ and perform stochastic gradient descent (SGD). We then produce the world frame change in pose $\Delta\mathbf{T}_w = \mathbf{T}_i^{-1} \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$ that can be applied to world frame positions.

Typically in particle filters we update the posterior via alternating prediction (via forward dynamics) and correction (from sensor data) steps. If the object did not move, then we also predict the pose particles remain stationary. If the object did move ($\Delta\mathbf{T}$ not $\mathbf{0}$) then our forward dynamics predicts movement $\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$, where $\Delta\mathbf{T}_\sigma$ is a transform perturbation sampled with the process in Algorithm 12 that adds diversity to the particles.

Before we can perform the correction step, we first merge the previous observations $\mathcal{X}_{t-1}$ with the current observations $\mathcal{X}'_t$, as described in Algorithm 15. The object geometric features are transformed by $\Delta\mathbf{T}_w$ while the free geometric features remain

stationary. However, the move might have invalidated some previous free ones and so we check whether $\texttt{sdf}(\mathbf{T}_i\mathbf{x}) > 0,\ \forall i = 1..N$ for each $(\mathbf{x}, \texttt{free}) \in \mathcal{X}$ in Algorithm 15 line 3. We then take the union of the transformed $\mathcal{X}_o$, validated $\mathcal{X}'_f$, and newly observed $\mathcal{X}'$. To avoid duplicate data, we voxel downsample by creating voxel grids, one per semantics value, that spans the range of the positions with resolution $r_d$. We assign the voxel grids with the positions then extract the center of voxel cells that received any assignment as new positions. We denote this downsampling process as $D : \mathbb{R}^3 \times \mathbb{R}^+ \to \mathbb{R}^3$.

With updated observations $\mathcal{X}_t$, we can update the weights of the pose particles. Importantly, we update even when not making contact because observing $s = \texttt{free}$ geometric features provides information about where the object is not. This process is described in Algorithm 11, where Eq. 5.3 is applied to get discrepancies $l_{1..N}$. We then apply Eq. 5.10 to convert it to an unnormalized probability. For numerical stability, we subtract the minimum $l$ from all of them to get relative discrepancy. This is without loss of generality since the normalization forces the weights to sum to 1.

In addition to the update step, we resample the pose particles to avoid degeneracy and maintain diversity as is typical of particle filters. Many heuristics exist for deciding when to resample *Li et al.* (2015) based mostly on removing low weight particles. However, the particle weights only represent their relative probability with respect to other particles, and we have a more direct signal in the discrepancy $l_{1..N} = \hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$ to evaluate when the pose particles have low likelihood. We use this in Algorithm 10 line 16 by comparing the maximum discrepancy of the particles to a threshold $l_r > 0$. For better robustness against outlier pose samples, a percentile of the discrepancy instead of the max can be used. This process is visualized in Fig. 5.6, where a contact made with the handle at the back of the mug forces a resample due to the previous pose particles' discrepancy with the observed surface geometric features.

Finally, the resampling process is described in Algorithm 13. We first perform the well known sampling importance resampling *Li et al.* (2015), then like in the prediction step we perturb the pose particles to generate diversity. We then ensure the pose particles have high probability by performing SGD on $\hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$.

### 5.4.5 Planning Problem

We use model predictive path integral (MPPI) control *Williams et al.* (2017a) to plan a $H$ horizon length trajectory and execute the first step of it in Algorithm 10 line 6. $H$ may be less than $T$ due to computation limitations. Without loss of gen-

**Algorithm 12:** `PerturbTransform` generate a random delta transformation

**Given:** $\sigma_t$ translation noise, $\sigma_R$ rotation noise
**Output:** $\Delta\mathbf{T}_\sigma$ delta transformation

// sample process noise
$\boldsymbol{\Delta t} \sim \mathcal{N}(\mathbf{0}, \text{diag}([\sigma_t, \sigma_t, \sigma_t])$
$\theta \sim \mathcal{N}(0, \sigma_R)$
$\mathbf{e} \sim U(\{\mathbf{x} \mid \|\mathbf{x}\|_2 = 1, \mathbf{x} \in \mathbb{R}^3\})$
$\boldsymbol{\Delta R} \leftarrow e^{\theta \mathbf{e}}$ // axis angle to matrix
$\Delta\mathbf{T}_\sigma \leftarrow \begin{bmatrix} \boldsymbol{\Delta t} & \boldsymbol{\Delta R} \\ \mathbf{0} & 1 \end{bmatrix}$

**Algorithm 13:** `Resample` pose particles to avoid degeneracy and high discrepancy

**Given:** $\mathbf{T}_{1..N}$ pose particles,
$w_{1..N}$ particle weights,
$\mathcal{X}$ set of observed geometric features,
$\sigma_t$ translation noise,
$\sigma_R$ rotation noise
$N_o$ resample optimization steps

// well known sampling importance resampling
$\mathbf{T}_{1..N} \leftarrow \texttt{ImportanceResample}(\mathbf{T}_{1..N}, w_{1..N})$
**for** $i \leftarrow 1$ **to** $N$ **do**
  $\Delta\mathbf{T}_\sigma \leftarrow \texttt{PerturbTransform}(\sigma_t, \sigma_R)$ $\mathbf{T}_i \leftarrow \Delta\mathbf{T}_\sigma \cdot \mathbf{T}_i$
**for** $j \leftarrow 1$ **to** $N_o$ **do**
  differentiate $\hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$ to get $\mathbf{T}_{1..N}$ gradients
  SGD to optimize $\mathbf{T}_{1..N}$

**Algorithm 14:** `ObjMove` estimate change in object pose while in contact

**Given:** $\mathcal{X}$ set of previously observed geometric features
$\mathcal{X}'$ set of just observed geometric features
$\mathbf{T}_{1..N}$ pose particles,
$\Delta\mathbf{T}_r$ change in end-effector pose during contact
$N_o$ number of optimization steps
$\Delta\mathbf{T} \leftarrow \Delta\mathbf{T}_r$ // sticking contact prior
$l_{1..N} \leftarrow \hat{C}(\mathcal{X}, \mathbf{T}_{1..N})$
$i \leftarrow \arg\min l_{1..N}$
**for** $j \leftarrow 1$ **to** $N_o$ **do**
  // hypothesis moved object pose
  differentiate $\hat{C}(\mathcal{X}', \Delta\mathbf{T} \cdot \mathbf{T}_i)$ to get $\Delta\mathbf{T}$ gradients
  SGD to optimize $\Delta\mathbf{T}$
$\Delta\mathbf{T}_w \leftarrow \mathbf{T}_i^{-1} \cdot \Delta\mathbf{T} \cdot \mathbf{T}_i$

**Algorithm 15:** `MergeObs` merge observations

**Given:** $\mathcal{X}$ set of previously observed geometric features,
$\mathcal{X}'$ set of just observed geometric features,
$\mathbf{T}_{1..N}$ pose particles,
$\Delta\mathbf{T}_w$ world frame change in object pose
$r_d$ downsample resolution

1 $\mathcal{X}_o \leftarrow \{(\Delta\mathbf{T}_w\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}, s \neq \texttt{free}\}$
2 $\mathcal{X}_f \leftarrow \{(\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}, s = \texttt{free}\}$ `// stationary`
`// remove all that may be occupied now`
3 $\mathcal{X}'_f \leftarrow \{(\mathbf{x}, s) \mid (\mathbf{x}, s) \in \mathcal{X}_f, \texttt{sdf}(\mathbf{T}_i\mathbf{x}) > 0,\ \forall i = 1..N\}$
4 $\mathcal{X} \leftarrow \mathcal{X}_o \cup \mathcal{X}'_f \cup \mathcal{X}'$
5 voxel downsample $\mathcal{X}$ with resolution $r_d$

erality, consider $t = 1$ at planning time for notation simplification. MPPI samples many Gaussian action perturbations around a nominal action trajectory to produce $\mathbf{u}_{1..H}$, rolls out the robot configuration from $\mathbf{q}_0$ to get $\mathbf{q}_{1..H}$ with a dynamics function, and evaluates each configuration trajectory with a cost function to weigh how the action trajectories should be combined. We initialize the nominal trajectory with noise, warm start it by running MPPI without actually executing the planned trajectory for several iterations. Then when executing $\mathbf{u}_1$, we use $\mathbf{u}_{2..H}, \mathbf{0}$ as the nominal trajectory for the next step. By convention, MPPI minimizes cost, and so we present costs where lower values are better.

### 5.4.6 Information Gain Cost

We assume we have the robot model such that we can map $h(\mathbf{q}) \to \mathcal{R}$ where $\mathcal{R}$ is the set of world coordinate points inside or on the surface of the robot. Note that when observing $\mathcal{X}'_t$ in Algorithm 10 line 8, $\{(\mathbf{x}, \texttt{free}) \mid \mathbf{x} \in h(\mathbf{q}_{t+1})\}$ should at least be in $\mathcal{X}'_t$ since the object cannot be inside the robot. Additionally, we assume we can identify $h_I(\mathbf{q}) \subset h(\mathbf{q})$ that selects the points of the robot that can observe information through contact. For example, the wrist of the end effector may be much less effective at reliably localizing contact than the tactile sensor. We only consider $h_I$ for gathering information but the full $h$ for the dynamics model.

For a rolled-out configuration trajectory $\mathbf{q}_{1..H}$ we define the information gain cost

$$C'_I(\mathbf{q}_{1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} h_I(\mathbf{q}_i), r_d)} -\tilde{I}(\mathbf{x}|\mathcal{X}) \tag{5.28}$$

which is the information gain field at every robot interior point in the rolled out

trajectory, downsampled to avoid double-counting.

This cost function develops naturally from $\tilde{I}(\mathbf{x}|\mathcal{X})$, however it does not take into account that the object can move, and in doing so, can change $\tilde{I}(\mathbf{x}|\mathcal{X})$. Consider a trajectory where a robot moves into contact with the target object then continues in a straight line with the target object remaining in sticking contact. While it would traverse the workspace and gather high $C_I'$ as a result, relative to the object it has not moved after coming into contact, and so should collect no new information. Indeed, $\tilde{I}(\mathbf{x}|\mathcal{X})$ is better seen as an object frame field, as only motion relative to the object should collect information.

To address this, we introduce predicted object displacement $\mathbf{d} \in \mathbb{R}^3$, and define the adjusted information gain cost

$$C_I(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} [h_I(\mathbf{q}_i) - \mathbf{d}_i], r_d)} -\tilde{I}(\mathbf{x}|\mathcal{X}) \tag{5.29}$$

where $h(\mathbf{q}_i) - \mathbf{d}_i \; \forall i = 1..H$ transforms the world query positions to be in the displaced object frame.

### 5.4.7 Dynamics Model

We predict the displacement $\mathbf{d}$ in our dynamics model $\hat{f}$ in addition to $\mathbf{q}$. We assume the difference of the true dynamics $f$ from the given free space dynamics $f_f$ is only due to making contact with the target object, and use the precomputed $p(S_\mathbf{x}|\mathcal{X})$ voxel grid to predict when that occurs. One step of $\hat{f}$ is described in Algorithm 16 and below:

First we apply free space dynamics to get candidate configuration $\mathbf{q}'$. We then sample if this configuration leads to contact by considering the least likely to be `free` position $\mathbf{x}_i$ from $h(\mathbf{q}') - \mathbf{d}_t$ in Algorithm 16 line 3. We randomly sample from the categorical distribution $s \sim p(S_{\mathbf{x}_i}|\mathcal{X})$. If we sample $s =$ `free`, then the candidate configuration is used as the next one and the object is not displaced. Otherwise, we need to consider if it is a pushing contact. We compute this action's displacement $\mathbf{d}'$ by considering the change in position from where $\mathbf{x}_i$ was before the action. Then, we estimate the surface normal $\hat{\mathbf{n}}$ at this point in line 11 by taking the weighted sum of the SDF gradient of the contact position transformed by each of the pose particles. If the angle between $\hat{\mathbf{n}}$ and $-\mathbf{d}'$ is less than some threshold $\theta_p$ based on an estimation of the friction cone between the robot and the object, then it is considered pushing. If it is a pushing contact, we increase object displacement and move the robot normally.

Otherwise, the robot is predicted to remain in its previous configuration to discourage non-pushing contacts, and no further object displacement is produced.

Note that $\mathbf{q}_{t+1}, \mathbf{d}_{t+1} \sim \hat{f}(\mathbf{q}_t, \mathbf{d}_t; ...)$ is stochastic since we sample contacts. To reduce variance, for a single action trajectory $\mathbf{u}_{1..H}$ we roll out multiple configuration trajectories by applying $\hat{f}$ on copies of the starting configuration $\mathbf{q}_1$ and $\mathbf{u}_{1..H}$. The cost of $\mathbf{u}_{1..H}$ is the average cost across the multiple $\mathbf{q}_{1..H+1}$. Practically, if the object has thin walls relative to the distance a single action could move the robot, as in the case of mugs, each action could be divided up and applied sequentially to avoid dynamics predicting the robot penetrating the object walls.

### 5.4.8 Reachability Cost

For manipulator arms with immobile bases, it is important to explicitly penalize when actions could move the object outside of its reachable region. Under just the information gain cost from Eq. 5.29, an action trajectory pushing the object out of reach will evaluate to have equal or better cost than a trajectory doing nothing. If the object is at the edge of the robot's reachability, such as a mug with sides that are within reach but the occluded handle at the back being out of reach, sampling a $H$ step trajectory that first displaces the mug then collects the high information gain at the back of the mug is very unlikely. $H$ may also be too short to allow such a trajectory to exist.

To address this, we introduce *reachability* $r(\mathbf{x}) \in [0, 1]$ and the reachability cost $C_R(\mathbf{d}_{1..H})$ which encodes the desired behaviour of pushing object frame points $\mathbf{x}$ with high $\tilde{I}(\mathbf{x}|\mathcal{X})$ to where they are reachable.

Reachability $r(\mathbf{x})$ represents the capability of the robot to gather information at $\mathbf{x}$, similar to checking $\exists \mathbf{q}$ s.t. $\mathbf{x} \in h(\mathbf{q})$. This can be approximated by performing inverse kinematics (IK) with $\mathbf{x}$ set as the goal position relative to the robot end effector frame. We also consider how robust $\mathbf{x}$ is to reach with different configurations, and evaluate the average IK performance with a fixed set of goal orientations $R_i \in R \subset \mathrm{SO}(3)$ in addition to the goal position $\mathbf{x}$. Let $e_{\mathbf{x}}(\mathbf{x}, R_i)$ and $e_R(\mathbf{x}, R_i)$ be the position and rotation errors from running IK with the goal set to $(\mathbf{x}, R_i)$. We weigh $e_R$ against $e_{\mathbf{x}}$ with $\alpha_R \geq 0$ and define an error tolerance threshold $e_m$ such that any error at or above this value receives $r(\mathbf{x}) = 0$. Thus we define

$$r(\mathbf{x}) = \frac{1}{e_m} \max(0, e_m - \frac{1}{|R|} \sum_{R_i \in R} [e_{\mathbf{x}}(\mathbf{x}, R_i) + \alpha_R e_R(\mathbf{x}, R_i)]) \tag{5.30}$$

**Algorithm 16:** $\hat{f}$ estimated robot and object displacement dynamics

**Given:** $\mathbf{q}_t$ current robot configuration,
$\mathbf{u}_t$ action,
$\mathbf{d}_t$ current object displacement,
$\mathbf{T}_{1..N}$ pose particles, $w_{1..N}$ particle weights
$p(S_{\mathbf{x}}|\mathcal{X})$ semantics probability voxel grid,
$f_f$ given free space dynamics,
$\nabla\texttt{sdf}$ object frame SDF gradient,
$h$ robot interior points model,
$\theta_p$ pushing angle threshold
**Output:** $\mathbf{q}_{t+1}$ new robot configuration,
$\mathbf{d}_{t+1}$ new object displacement

$\mathbf{q}' \leftarrow f_f(\mathbf{q}_t, \mathbf{u}_t)$ `// candidate config`
$\mathcal{R} \leftarrow h(\mathbf{q}') - \mathbf{d}_t$
`// find most likely contact`
$i \leftarrow \arg\min_{\mathbf{x}_i \in \mathcal{R}} p(S_{\mathbf{x}_i} = \texttt{free}|\mathcal{X})$
$s \sim p(S_{\mathbf{x}_i}|\mathcal{X})$ `// sample semantics`
**if** $s = \mathit{free}$ **then**
    $\mathbf{q}_{t+1} \leftarrow \mathbf{q}'$
    $\mathbf{d}_{t+1} \leftarrow \mathbf{d}_t$
**else**
    `// determine displacement`
    $\mathcal{R}_b \leftarrow h(\mathbf{q}_t)$
    $\mathbf{d}' \leftarrow \mathbf{x}_i - \mathcal{R}_b[i]$`// corresponding ith position`
    `// estimate object surface normal`
    $\hat{\mathbf{n}} \leftarrow \sum_{j=1}^{N} w_j \nabla\texttt{sdf}(\mathbf{T}_j \mathbf{x}_i)$
    **if** *angle between* $\hat{\boldsymbol{n}}$ *and* $-\boldsymbol{d}' < \theta_p$ **then**
        `// pushing or not`
        $\mathbf{q}_{t+1} \leftarrow \mathbf{q}'$
        $\mathbf{d}_{t+1} \leftarrow \mathbf{d}_t + \mathbf{d}'$
    **else**
        `// discourage non-pushing contact`
        $\mathbf{q}_{t+1} \leftarrow \mathbf{q}_t$
        $\mathbf{d}_{t+1} \leftarrow \mathbf{d}_t$

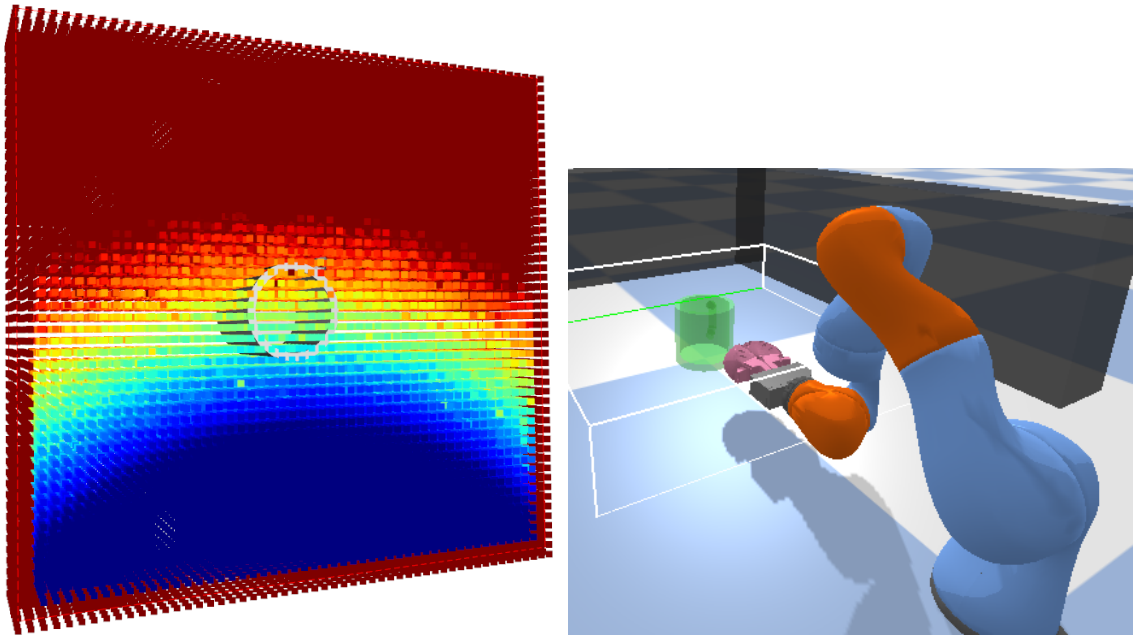

Figure 5.7: (left) Reachability of workspace positions for the (right) simulated KUKA arm and workspace. Red corresponds to $r(\mathbf{x}) = 0$ and dark blue corresponds to $r(\mathbf{x}) = 1$. The workspace in sim is drawn as a box around the object.

We precompute this for $\mathbf{x} \in \mathcal{W}$ and store the results in a voxel grid that allows linear interpolation. This only has to be done once per robot and workspace combination. See Fig. 5.7 for an example visualization of $r(\mathcal{W})$.

The reachability cost $C_R(\mathbf{d}_{1..H})$ is then the total reachable information within the workspace after displacement. We compute it according to Algorithm 17. First we compute $\bar{\tilde{I}}(\mathbf{x})$, the average information gain at every displaced workspace position over the planning horizon. Note that $\tilde{I}(\mathbf{x}|\mathcal{X})$ can be interpreted as the object frame information gain field at the time of planning, and so stationary workspace positions are effectively displaced by $-\mathbf{d}_{1..H}$ during planning. The reachable information is just the product $\bar{\tilde{I}}(\mathbf{x})r(\mathbf{x})$ which we sum across all the workspace points. This is then compared against the total information in the workspace to produce a negative ratio $C_R \in [-1, 0]$. Because $C_I$ and $C_R$ are in different units, having $C_R$ be a ratio allows easier tuning of the total trajectory cost:

$$C(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) = \beta_I C_I(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) + \beta_R C_R(\mathbf{d}_{1..H}) \tag{5.31}$$

**Algorithm 17:** $C_R$ reachability cost

**Given:** $\mathbf{d}_{1..H}$ object displacement trajectory,
$\mathcal{W}$ workspace,
$\tilde{I}(\mathbf{x}|\mathcal{X})$ information gain voxel grid,

$\bar{\tilde{I}}(\mathbf{x}) \leftarrow \frac{1}{H}\sum_t^H \tilde{I}(\mathcal{W} - \mathbf{d}_t|\mathcal{X})$ `// average info`

$RI \leftarrow \sum_{\mathbf{x}\in\mathcal{W}} \bar{\tilde{I}}(\mathbf{x}) r(\mathbf{x})$ `// reachable info`

$RI_m \leftarrow \sum_{\mathbf{x}\in\mathcal{W}} \tilde{I}(\mathbf{x}|\mathcal{X})$ `// max info possible`

$C_R(\mathbf{d}_{1..H}) \leftarrow -RI/RI_m$

### 5.4.9 Kernel Interpolated MPPI

The total cost from Eq. 5.31 does not include any explicit smoothing terms. To improve the smoothness of produced trajectories, we perform interpolation similar to *Miura et al.* (2024). The idea is to sample $H_v < H$ control points $\mathbf{v}_i \in \mathbb{R}^{N_u}$, then use a kernel $K : \mathbb{R}^{N_u} \times \mathbb{R}^{N_u} \to \mathbb{R}$ to interpolate the $\mathbf{u}_{1..H}$ in between $\mathbf{v}_{1..H_v}$. We call this method Kernel Interpolated MPPI (KMPPI). This is more general than the B-spline interpolation of *Miura et al.* (2024) since it can be accomplished by using a B-spline kernel.

Let $\mathbf{H_u} = [0, 1, ..., H-1]$ denote the time coordinate of each $\mathbf{u}$ along the trajectory. We assume $\mathbf{v}_{1..H_v}$ are evenly spread out along the trajectory, and since there are $H_v$ of them, subsequent ones increase their time coordinate by $\frac{(H-1)}{(H_v-1)}$ to give $\mathbf{H_v} = [0, \frac{(H-1)}{(H_v-1)}, \frac{2(H-1)}{(H_v-1)}, ..., H-1]$. The even assignment of $\mathbf{H_v}$ is not necessary; any can be given as long as the first term is 0 and the last term is $H-1$. Given a control sequence $\mathbf{v}_{1..H_v}$ we then convert it to $\mathbf{u}_{1..H}$

$$\mathbf{u}_{1..H} = K(\mathbf{H_u}, \mathbf{H_v})K(\mathbf{H_v}, \mathbf{H_v})^{-1}\mathbf{v}_{1..H_v} \tag{5.32}$$

This allows smoothing in the action space, rather than in the robot configuration space, and we observe that it works well on our tasks. See Fig. 5.8 for a qualitative evaluation of the smoothing property on a toy 2D problem.

With the interpolated $\mathbf{u}_{1..H}$, KMPPI's subsequent steps are the same as MPPI's in that it generates configuration rollouts by applying the dynamics function $\mathbf{q}_{t+1}, \mathbf{d}_{t+1} \sim \hat{f}(\mathbf{q}_t, \mathbf{d}_t; ...), t = 1..H$, evaluates the cost of each $\mathbf{u}_{1..H}$ with Eq. 5.31, then combines the trajectory samples with a softmax based on the cost.

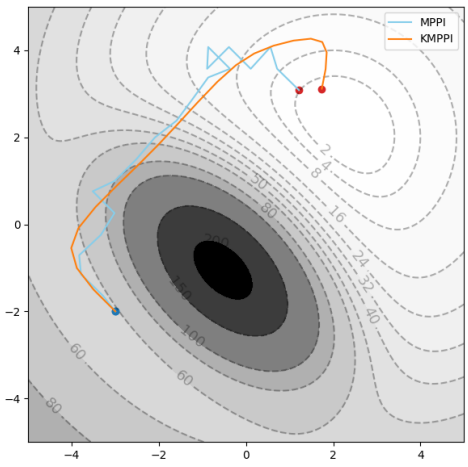


Figure 5.8: Planned trajectories on a toy 2D linear integrator environment. The cost contour map is represented, with the cost of the trajectory being the accumulated cost experienced at each state. There is additionally a quadratic action penalty at each step $\mathbf{u}^T 0.1\mathbf{I}\mathbf{u}$. Dark blue is the starting state. Each trajectory has $H = 20$, with our KMPPI having $H_v = 5$ and using the radial basis function kernel with scale 2.

### 5.4.10 Termination Condition

In actual execution, we do not have access to $\mathbf{T}^*$ to evaluate $nll(\mathcal{X})$ and need another signal to terminate execution. We use the convergence of the pose particles, with the hypothesis that pose particles likely only converges when $p(\mathbf{T}^*|\mathcal{X})$ is high, i.e. the pose particles do not randomly converge to an incorrect estimate. We evaluate convergence using the average square root pairwise Chamfer distance between the pose particles ($APC$). Similar to the $nll(\mathcal{X})$ evaluation, we evaluate this on a sampled set of object frame surface positions $\tilde{x} \in \tilde{\mathbf{X}}$.

$$APC(\mathbf{T}_{1..N}) = \frac{1}{N^2|\mathbf{X}_o|} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{\tilde{x} \in \tilde{\mathbf{X}}} |\texttt{sdf}(\mathbf{T}_i^{-1}\mathbf{T}_j\tilde{x})| \tag{5.33}$$

We terminate execution when $APC(\mathbf{T}_{1..N}) < \beta_t l_c$, where $l_c$ is the diagonal length of the object's bounding box, and $\beta_t$ is a ratio that selects for a desired level of pose particle convergence. A lower value means rummaging will continue for longer, but may produce a more accurate pose estimate.

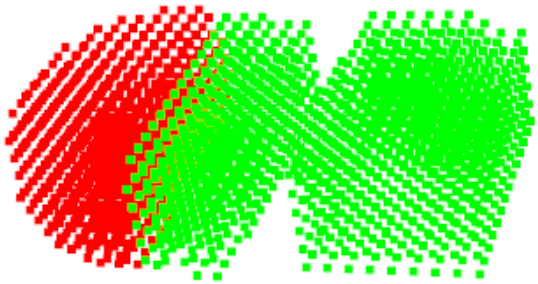

Figure 5.9: Visualization of interior robot points $h(\mathbf{q})$ as green and red points, and the information gathering subset $h_I(\mathbf{q})$ as just the red points for the robot gripper mounted with two soft-bubble tactile sensors seen in Fig. 5.1.

## 5.5 Experiments

In this section, we first describe our simulated and real robot environments. We then detail the experiments to estimate the pose of a movable target object. We introduce our baselines and ablations and how we quantitatively evaluate the methods on the experiment. Lastly, we present results that show RUMI is the only method to perform consistently well across all the experiments.

### 5.5.1 Sim Environment

Common to all the experiments, we have a single movable object on a flat surface starting within reach of a single 7DoF KUKA LBR iiwa arm with two soft-bubble tactile sensors *Kuppuswamy et al.* (2020). This is modelled in sim in Fig. 5.7. Due to the complexity of modelling deformable objects, we model the soft-bubble tactile sensors as rigid bodies and observe the surface points of any object penetrating them after each simulation step. We also include a fixed external depth camera to reduce the initial exploration required, but we also show that our method works without a good initial view of the object in some sim experiments.

For observing $s = \texttt{surface}$ points at $\mathbf{q}_t$, we select $\{\mathbf{x} | \ \mathbf{x} \in h_I(\mathbf{q}_t), |\texttt{sdf}(\mathbf{T}^*\mathbf{x})| < 3mm\}$. This simulates some observation noise which we show that RUMI is robust to, despite assuming no noise in the observed positions. We assume we can only gather contact information from the front of the gripper, where the two soft-bubble tactile sensors are mounted. In planning, this is the difference between $h(\mathbf{q})$ and $h_I(\mathbf{q})$ for our end effector shown in Fig. 5.9.

The $\mathcal{X}'$ provided by the depth camera includes $s = \texttt{free}$ points generated by

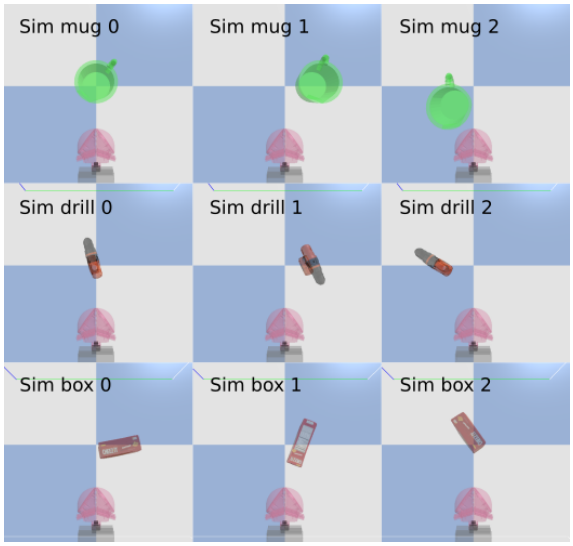


Figure 5.10: Starting poses for labelled sim tasks.

tracing rays from the camera to 95% of each pixel's detected depth, and $s = \texttt{surface}$ from segmented object surfaces. See Fig. 5.4 (middle), and Fig. 5.11 for example observation point clouds. We only use vision to provide the initial $\mathcal{X}_0$ to demonstrate the viability of tactile based rummaging.

To highlight the difference between other components of all methods, we directly observe $\Delta\mathbf{T}$ in Algorithm 10 line 9. Note that this also applies to all baselines and ablations, and so does not provide an unfair advantage to RUMI. This is equivalent to assuming we can accurately measure slip between the end effector and object.

### 5.5.2 Sim Tasks

In simulation, we experiment on 3 different objects: a mug, a YCB *Calli et al.* (2017) power drill, and a YCB cracker box, each with 3 different initial poses depicted in Fig. 5.10. For each, we perform 10 runs of $T = 40$ steps, using a different fixed random seed for each run that is shared across baselines and ablations. We terminated tasks early if the pose particles converged as measured by $APC(\mathbf{T}_{1..N}) < 0.03 l_c$, where $l_c$ is the diagonal distance of each object's bounding box.

In each experiment, the robot's end effector is position and yaw controlled, with the action space either being $\mathbf{u} = [dx, dy, d\theta]$ (planar) or $\mathbf{u} = [dx, dy, dz, d\theta]$ (3D) with ranges from [-1,1] for each dimension. The action spaces are scaled to allow the use of consistent KMPPI parameters across experiments. We scale these to physical units by translating a control value of 1 to $80mm$ or 0.5 radians, carried out in many mini steps. We perform inverse kinematics to convert these to joint position commands.

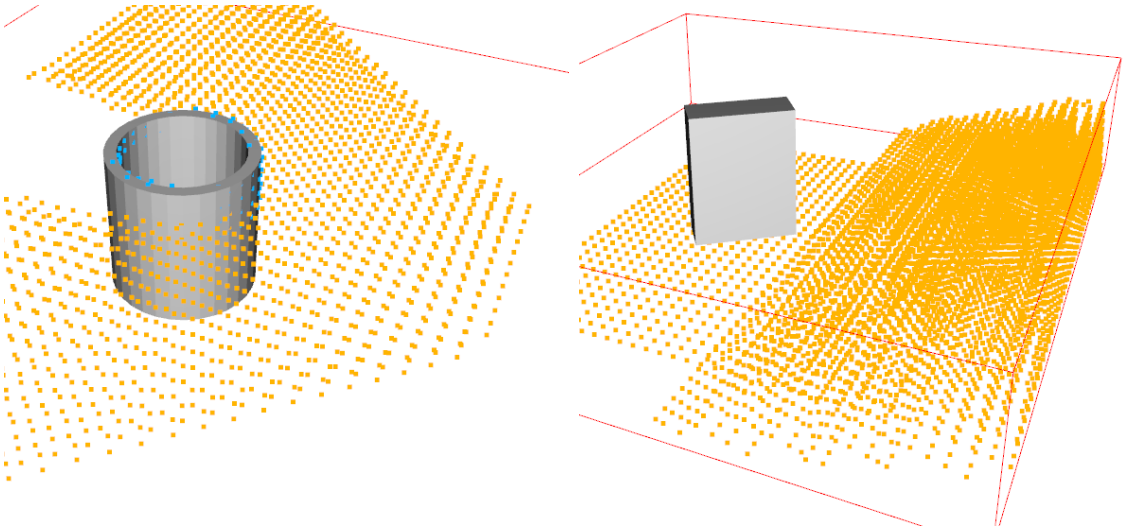

Figure 5.11: Comparison of the rendered $\mathcal{X}_0$ given to (left) the sim mug 0 task and (right) sim box 2 task. $s = \texttt{free}$ points are in orange, and $s = \texttt{surface}$ points are in blue. There are no initially observed surface points on the box.

We used regular grids with resolutions (grid square side length) $r_w$ as the workspaces. Note that other sets of worldspace points that are not necessarily regular grids could be used. We used $\mathcal{W} = [0, 0.8] \times [-0.4, 0.4]$ in meters for planar action spaces, and $\mathcal{W} = [0, 0.8] \times [-0.4, 0.4] \times [0, 0.2]$ for 3D action spaces. $r_w$ for each task can be found in Tab. 5.1.

Each object is intended to illustrate a different aspect of exploration. For the mug and power drill, we assume the object stays upright and search for their pose in SE(2) instead of SE(3). The mug tasks evaluates how well $\tilde{I}(\mathbf{x}|\mathcal{X})$ conforms to our intuition, since we expect the most information to be where the handle could be. The sim drill task evaluates how well our planner extends to objects with complex geometry. The sim box task tests how well the pose particles can represent full SE(3) and the necessary 3D exploration to identify the which side of the box is lying against the floor. Additionally, for the drill and box tasks, we increase the difficulty in terms of environmental occlusions by placing the camera at an angle such that it cannot directly observe the object. The camera configurations and the initial object pose are depicted in Fig. 5.11. For SE(3) pose search in the box experiments, we add free points where the floor is to avoid pose estimates that penetrate the floor. The different task setups are summarized in Tab. 5.1.

We use different $\mathbf{T}_{0,1..N}$, the prior pose particles, for the mug tasks where we initially observe the front of it, to the other tasks where we initially cannot see it. For mugs, we first estimate the position of the center of the mug, then $\mathbf{T}_{0,1..N}$ is sampled

with uniformly random yaw and the same center. For the other tasks, we sample $\mathbf{T}_{0,1..N}$ with random positions sampled from $\mathcal{N}(0, 0.05) \times \mathcal{N}(0, 0.05) \times 0$, and also uniformly random yaw (assuming upright).

### 5.5.3 Real Environment

The real robot setup is seen in Fig. 5.1 and Fig. 5.10, the same robot we modelled in simulation. The soft-bubble sensors are compliant to contact and have a depth camera inside to estimate dense contact patches. Similar to prior work *Zhong et al.* (2023), we consider points on the soft bubble surface with deformation beyond $4mm$ and being in the top $10^{th}$ percentile of all deformations to be in contact. We apply a mean filter to remove noise. We use a RealSense L515 lidar camera as the fixed external camera. For evaluating ground truth object pose, we have a RealSense D435 camera mounted looking top-down on the workspace.

The mug had distinct colors from the shelf and so we segmented it with a color filter. To improve segmentation, we used a robot self-filter and an edge filter to remove unreliable points, and used a temporal filter to only accept surface points that persists over a 0.4s window. See Fig. 5.1 for example observation point clouds. We re-observe the scene after each action. Due to self-occlusion and object symmetry, visual observations do not uniquely identify object pose. Same as for the simulated box, the real box task has occluded vision that prevented direct observation of it, seen in the top of Fig. 5.12.

For the real mug task, we do not assume we can accurately measure slip between the end effector and object. Instead, we estimate $\Delta\mathbf{T}$ with Algorithm 14 for all methods. For the real box task, we observe the change in object pose from the ground truth since we cannot directly observe the object to estimate $\Delta\mathbf{T}$ with Algorithm 14.

### 5.5.4 Real Task

On the real robot we estimate the pose of a mug and box starting in a single configuration depicted in Fig. 5.12 and execute $T = 15$ steps of each method. The robot's action space is seen in Tab. 5.1, and a control value of 1 corresponds to $50mm$ or 0.4 radians. The workspace was $\mathcal{W} = [0.55, 1.1] \times [-0.33, 0.33] \times [0.23, 0.47]$ in meters (for planar action space, a fixed height of 0.305m was used). The $\mathbf{T}_{0,1..N}$ initialization process is similar to sim for each corresponding task. $\mathbf{T}_{1..N}$ after sampling from CHSEL in Algorithm 10 line 2 is shown at the bottom of Fig. 5.12. Note that the box's initial $\mathbf{T}_{1..N}$ covers the workspace since vision was occluded. We terminated tasks early if

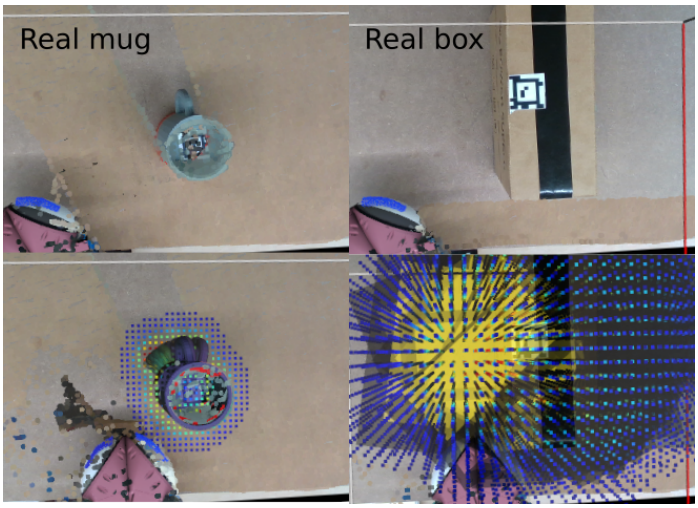


Figure 5.12: Initial configuration of the real mug and box tasks (top) and example initialized pose particles (bottom). In the box task, the workspace is occluded and the box cannot be directly observed. Point cloud observations from an external side view (accessible to the robot) are overlaid on a top-down raw camera view (inaccessible to the robot).

| Object | action space | pose search space | resolution $r_w$ (m) |
|---|---|---|---|
| Sim mug | planar | SE(2) | 0.01 |
| Sim drill | 3D | SE(2) | 0.02 |
| Sim box | 3D | SE(3) | 0.02 |
| Real mug | planar | SE(2) | 0.01 |
| Real box | 3D | SE(2) | 0.02 |

Table 5.1: Task setup for different objects.

the pose particles converged as measured by $APC(\mathbf{T}_{1..N}) < 0.05 l_c$, where $l_c$ is the diagonal distance of the mug's bounding box.

### 5.5.5 Sensor Model

We use the sensor model depicted in Fig. 5.2. Let $v = \texttt{sdf}(\mathbf{Tx})$ be the SDF value of a given query position $\mathbf{x}$, then $p(S_\mathbf{x}|\mathbf{T}) = p(S_\mathbf{x}|\texttt{sdf}(\mathbf{Tx})) = p(S_\mathbf{x}|v)$ is defined by

$$p(S_\mathbf{x}|v) = \begin{cases} \max(0, 1 - e^{-\alpha v}) & \texttt{free} \\ \max(0, 1 - e^{\alpha v}) & \texttt{occupied} \\ e^{-\alpha |v|} & \texttt{surface} \end{cases}$$

with $\alpha = 100$ where $v$ is in meters. This model represents some of the ambiguities of detecting contact with the soft-bubble and similar tactile sensors. Due to the

compliance of the membrane, even when a point is in free space, contact elsewhere could make it appear like this point is also in contact. Similarly, contact could also be missed, particularly around the edges of the soft-bubble. This sensor model performed well enough both in sim and on the real task that no calibration to the real soft-bubbles was needed.

### 5.5.6 KMPPI Parameters

We used a planning horizon of $H = 15$ and $H_v = 8$ number of control points. This is lower than the number of sim steps $T = 40$ because increasing horizon resulted in poorer-quality trajectories. This is due to the cost from Eq. 5.31 being a terminal cost for the whole trajectory, without distinguishing between steps inside the trajectory. We used the radial basis function (RBF) kernel with a scale of 2.

We planned using 500 action trajectory samples, each rolled out 5 times with $\hat{f}$ due to its stochastic nature. Additionally, to avoid contacts that penetrate the object, we split each action up into 4 sequentially applied actions that are 4 times lower in magnitude. We then use the average trajectory cost across the 5 rollouts.

For the inner MPPI parameters, we used $\lambda = 0.01$ for the temperature parameter from *Williams et al.* (2017a), with $\mathbf{0}$ noise mean and $1.5\mathbf{I}$ as the noise covariance.

### 5.5.7 Evaluation

We sample 500 positions $\tilde{x} \in \tilde{\mathbf{X}}$ uniformly on the surface of the object, and transform them to world positions with $\mathbf{T}^*$, the ground truth pose, to produce $\mathbf{X}$. We then evaluate the negative log likelihood of $\mathbf{x} \in \mathbf{X}$ being `surface` points from Eq. 5.2. Because we assume $S_\mathbf{x}$ is conditionally mutually independent to every other $S_\mathbf{x}$ given $\mathcal{X}$, we can simplify Eq. 5.2

$$nll(\mathcal{X}) = -\log p(\bigcap_{\mathbf{x}\in\mathbf{X}} S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{5.34}$$

$$= -\sum_{\mathbf{x}\in\mathbf{X}} \log p(S_\mathbf{x} = \texttt{surface}|\mathcal{X}) \tag{5.35}$$

We substitute Eq. 5.27 in for $p(S_\mathbf{x}|\mathcal{X})$ to approximate $nll$ with our pose particles

$$nll(\mathcal{X}) \approx -\sum_{\mathbf{x}\in\mathbf{X}} \log \sum_{i=1}^{N} w_i p(S_\mathbf{x} = \texttt{surface}|\texttt{sdf}(\mathbf{T}_i\mathbf{x})) \tag{5.36}$$

For the sim tasks, we have the ground truth object pose $\mathbf{T}^*$, while for the real

| Task | sim mug | | | sim drill | | | sim box | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 |
| $\text{cor}(nll, APC)$ | 0.93 | 0.82 | 0.84 | 0.74 | 0.94 | 0.94 | 0.58 | 0.80 | 0.49 |

Table 5.2: Linear correlation between $nll(\mathcal{X})$ and $APC(\mathbf{T}_{1..N})$ across all sim tasks. Runs from all methods were considered in its calculation.

| Object | success *nll* threshold |
|---|---|
| Sim mug | 20 |
| Sim drill | 100 |
| Sim box | 150 |
| Real mug | 35 |
| Real box | 250 |

Table 5.3: Maximum *nll* threshold for success for each task.

tasks, we observe $\mathbf{T}^*$ from a camera mounted above the workspace. We evaluated *nll* after each step as an effective exploration rate. Additionally, we specify a *nll* threshold below which we qualitatively observe to be a good enough quality to be considered a success, seen in Tab. 5.3. A run is counted a success if it achieves a minimum *nll* below the threshold at any step. This is typically, but not always, the last step. This is because, due to observation noise and moving the object outside of the observed region, the pose estimates could become less certain.

We also use the same $\tilde{x} \in \tilde{\mathbf{X}}$ to evaluate *APC* from Eq. 5.33. We used $\beta_t = 0.03$ for the sim tasks, meaning we terminated exploration when the average square root chamfer distance between all pairs of $\mathbf{T}_{1..N}$ is less than 3% of the object's bounding box diagonal length. For the real experiment we used $\beta_t = 0.05$.

Since we do not assume a ground-truth pose of the object is available in the real world, we needed a way to evaluate performance that does not rely on $nll(\mathcal{X})$ (which assumes knowledge of $\mathbf{T}^*$). We can, however, compute $APC(\mathbf{T}_{1..N})$ without privileged information.

First, we evaluated our hypothesis of $APC(\mathbf{T}_{1..N})$ as a good proxy for $nll(\mathcal{X})$ by computing the linear correlation between the two across all the tasks. The runs from all methods were used. This is shown in Tab. 5.2 and Fig. 5.13 for the sim mug 0 and sim mug 1 tasks, which can be compared to the $nll(\mathcal{X})$ shown in the top left and top middle of Fig. 5.16. We see that there is an especially strong positive correlation for SE(2) particles of the sim mug and sim drill tasks, averaging to a correlation of **0.87**. The correlation for the SE(3) sim box tasks is not as strong.

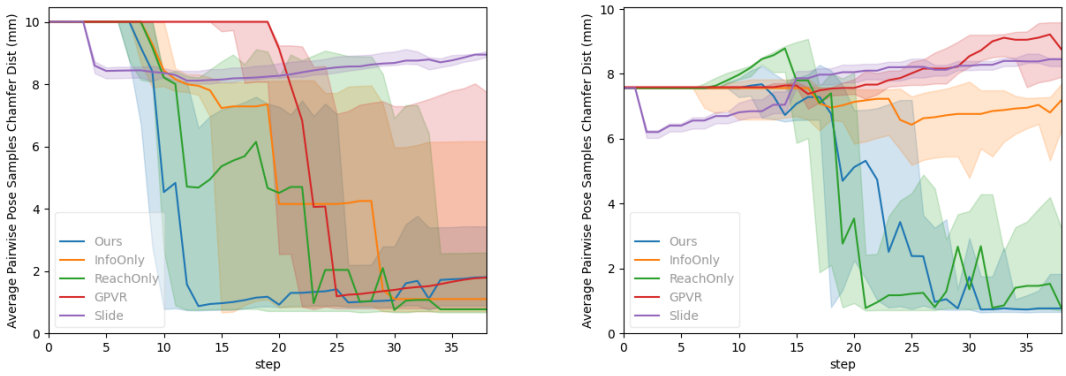


Figure 5.13: Convergence of pose particles measured by $APC(\mathbf{T}_{1..N})$ for the sim mug 0 and sim mug 1 tasks. The median over 10 runs is plotted, with the $25^{th}$ to $75^{th}$ percentile shaded. There is strong correlation with the $nll(\mathcal{X})$ in the top left and top middle of Fig. 5.16.

### 5.5.8 Baselines and Ablations

Our full method parameters are summarized in Tab. 5.4. These parameters were used for all simulated and real tasks (except for $\beta_t$ in deciding when to terminate), demonstrating the robustness of RUMI. For downsampling the observations in Algorithm. 15, we used different resolutions for the free space ($r_{d,f}$) and surface ($r_{d,s}$) points; we did not observe any occupied points. The baselines also required the creation and update of the $p(\mathbf{T}|\mathbf{X})$ pose particles, and we use the same parameters to do so.

We present two ablations to our full method, **InfoOnly** which sets $C_R$ to 0 and **ReachOnly** which sets $C_I$ to 0. They share all other parameters with the full method and evaluate the usefulness of each individual cost.

For baselines, we first present the **Slide** heuristic inspired by *Driess et al.* (2017). This method has two modes of operation - if it is currently in contact, then it moves tangentially to the estimated surface normal to slide along it. It moves parallel to the shelf, and for each run randomly decides at the start of the run whether to slide clockwise or counterclockwise around contact. If it is not in contact, then it moves towards the estimated center of the object. Estimating the object center requires our pose particles, so we still update $p(\mathbf{T}|\mathcal{X})$ using Algorithm 10.

We also consider a Gaussian Process Implicit Surface baseline (GPIS) *Caccamo et al.* (2016), *Driess et al.* (2017), *Lee et al.* (2019) that uses the variance of the GP as the exploration signal that we call GP Variance Reduction (**GPVR**). The GP is fit on $\{(\mathbf{x}, 0)|\ (\mathbf{x}, s) \in \mathcal{X}_t, s = \texttt{surface}\} \cup \{(\mathbf{x}, 1)|\ (\mathbf{x}, s) \in \mathcal{X}_t,\ s = \texttt{free}\}$. As typical

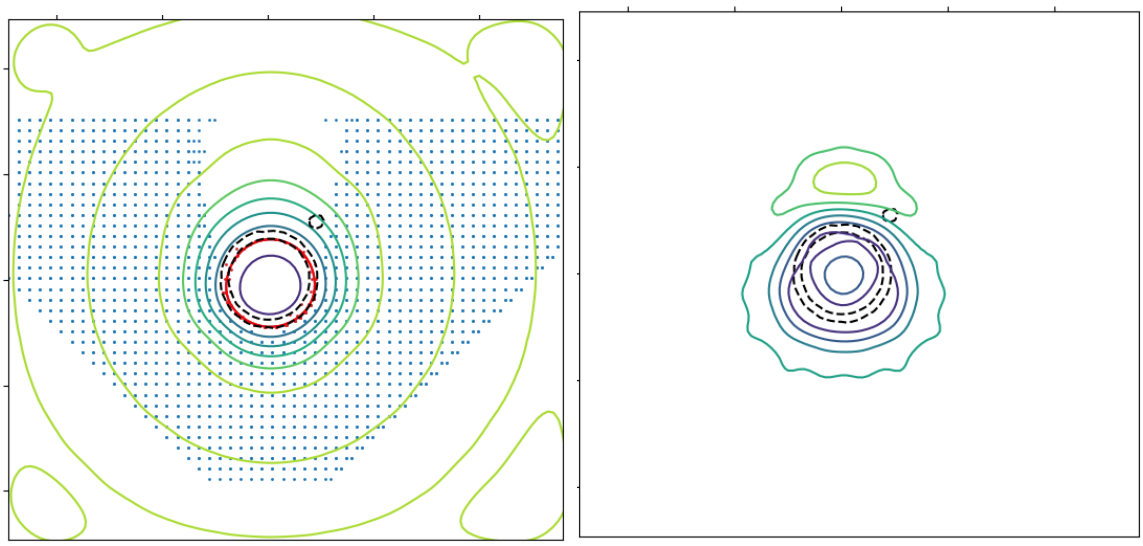

Figure 5.14: (left) Gaussian Process fit to initial observations $\mathcal{X}_0$ of the sim mug 0 task. Free space observed points are shown as blue dots, surface observed points as red dots, and the ground truth object surface as black dotted lines. The GP output is overlayed as a contour map, with the red line indicating the 0-level set, corresponding to where the GPIS surface is. (right) $\mathbf{var}(\mathbf{x}|\mathbf{X})$ contour map for the same GP, with green indicating higher variance.

for GPIS, surface points are labelled 0 and free points are labelled 1. It is refit on $\mathcal{X}_t$ until convergence after every robot execution step. We use the $\nu = 1.5$ Matern kernel as recommended by *Lee et al.* (2019). See Fig. 5.14 for a visualization of the fitted GP as well as its variance $\mathbf{var}(\mathbf{x}|\mathcal{X})$ on the sim mug 0 task given $\mathcal{X}_0$.

For GPVR to be competitive, we had to make several modifications. Firstly, we needed to encode object shape as that is given information to RUMI. This is non-trivial to do by modifying the kernel, so we instead augmented the input data with $\{(\mathbf{x}, 1)|\ \mathbf{x} \in \mathcal{W}, p(S_\mathbf{x} = \texttt{free}|\mathcal{X}_t) > 0.99\}$. For augmenting points, we used a version of each workspace with 7 times the resolution from Tab. 5.1 to avoid extremely slow inference from the number of data points. Again, this baseline requires the computation and maintenance of $p(\mathbf{T}|\mathcal{X})$ with the pose particles to enable the estimation of $p(S_\mathbf{x} = \texttt{free}|\mathcal{X}_t)$. Without the above data augmentation, GPVR explores the unobserved corners of the workspace, despite seeing parts of the object elsewhere. Secondly, we needed to plan further than just the next step. Otherwise, because we start in and are surrounded by free space, the method goes in initially random directions. Instead of the greedy policy of maximizing the GP variance $\mathbf{var}(\mathbf{x})$ at the next position from *Driess et al.* (2017), we formulated a cost function based on variance

| Parameter | value |
|---|---|
| $N$ number of pose particles | 100 |
| $\lambda$ peakiness | 2 |
| $l_r$ discrepancy resample threshold | 5 |
| $C$ CHSEL freespace discrepancy scale | 10 |
| $H$ planning horizon | 15 |
| $\theta_p$ pushing angle threshold | 45 degrees |
| $C_I$ information gain cost scale | 1 |
| $C_R$ reachability cost scale | 200 |
| $e_m$ reachability IK error threshold | 0.4 |
| $\alpha_R$ reachability IK rotation error scale | 0.1 |
| $r_{d,f}$ downsample resolution free space | 10mm |
| $r_{d,s}$ downsample resolution surface | 2mm |
| $\sigma_t$ pose translation noise | 10mm |
| $\sigma_R$ pose rotation noise | 0 |
| $\beta_t$ chamfer distance convergence ratio | 0.03 (0.05 for real) |
| $N_o$ number of optimization steps | 10 |

Table 5.4: Our full method parameters across the different tasks.

reduction similar in form to Eq. 5.29 for use inside KMPPI.

$$C_{GP}(\mathbf{q}_{1..H}, \mathbf{d}_{1..H}) = \sum_{\mathbf{x} \in D(\bigcup_{i=1}^{H} [h_I(\mathbf{q}_i) - \mathbf{d}_i], r_d)} - \mathbf{var}(\mathbf{x}|\mathcal{X}) \tag{5.37}$$

Instead of $\tilde{I}(\mathbf{x}|\mathcal{X})$, we used $\mathbf{var}(\mathbf{x}|\mathcal{X})$ which is the variance of the GP evaluated at $\mathbf{x}$. Similarly, before each planning step we precomputed $\mathbf{var}(\mathbf{x}|\mathcal{X})$ $\forall \mathbf{x} \in \mathcal{W}$ to store in a voxel grid for faster repeated lookup. We normalized $\mathbf{var}(\mathbf{x}|\mathcal{X})$ such that $\max_{\mathbf{x} \in \mathcal{W}} \mathbf{var}(\mathbf{x}|\mathcal{X}) = 1$.

See Fig. 5.15 for a comparison of $\tilde{I}(\mathbf{x}|\mathcal{X})$ against $\mathbf{var}(\mathbf{x}|\mathcal{X})$ to be planned over in a similar manner. From the figure, we see that $\mathbf{var}(\mathbf{x}|\mathcal{X})$ is low at where the handle could be. This is because $\mathcal{X}$ includes the inside back of the mug, and the Matern kernel does not directly encode object shape but is just based on the Euclidean distance between points. It cannot disambiguate the certainty of the back surface of the mug from the uncertainty of where the handle is, because it cannot know that a handle exists. Instead, $\mathbf{var}(\mathbf{x}|\mathcal{X})$ is highest farther behind the mug, where we have observed no data due to occlusion. This is contrasted with $\tilde{I}(\mathbf{x}|\mathcal{X})$, which is highest where the handle could be because those regions are where the pose particles disagree the most.

### 5.5.9 Results

The simulated task results are in Fig. 5.16 and the real task results are in Fig. 5.17. The number of successful trials out of 10 for each task is compared in Tab. 5.5. Additionally, the median over the cumulative $nll(\mathcal{X})$ of each run are in Tab. 5.6. For the sim and real tasks, cumulative $nll(\mathcal{X})$ over time is a good indicator of exploration speed; however, for the sim drill and box tasks, cumulative $nll(\mathcal{X})$ is dominated by the initial search for the first surface points of the object since they do not start with the object in view. Thus, for those tasks it is more a measure of how quickly the different methods make first contact with the object.

We observe that RUMI is the only method to achieve consistently good performance, if not the most number of successes, across all the sim and real tasks. On the sim mug tasks, it also had the lowest cumulative $nll(\mathcal{X})$, meaning it was the most efficient. The ablations show that both $C_I$ and $C_R$ are important for this task, although individually they can also perform well on certain tasks. For example on the sim mug task, ReachOnly achieved a high number of successes by itself. This was likely due to the handle being close to where the robot needed to push from to increase reachability. However even in this case, adding $C_I$ improves efficiency because the mug could be pushed into more reachable regions without contacting the handle. This explains the occasional failures of the ReachOnly method on the mug tasks. On the drill tasks, pushing the object to be more reachable did not reliably lead to contact that was informative about the pose, and it did much worse than our full method and the InfoOnly baseline.

The major failure case for all the methods was pushing the object to be outside the robot's reachable region. The performance gain of the full method against the InfoOnly ablation can be mostly attributed to preventing this. As long as the object was kept within reach and contacts kept being made with the object at different locations, the pose estimation was gradually improved. This is illustrated in ReachOnly's performance on the sim box tasks, where it is one of the slowest methods to reduce $nll$, but was still able to achieve a relatively high number of successful trials.

The Slide baseline exhibited behavior that in some ways was the opposite of ReachOnly's. It always pushed the object away from the robot, and it became a race of it gathering enough pose-identifying information from those contacts before the object moved out of reach. On the real robot, sometimes it did not register that a contact was made and would continue pushing forward. This strategy's success was highly configuration-dependent, seen in Tab. 5.5, where it can either achieve reliable success (since there is only randomness in the sliding direction), or no success. It

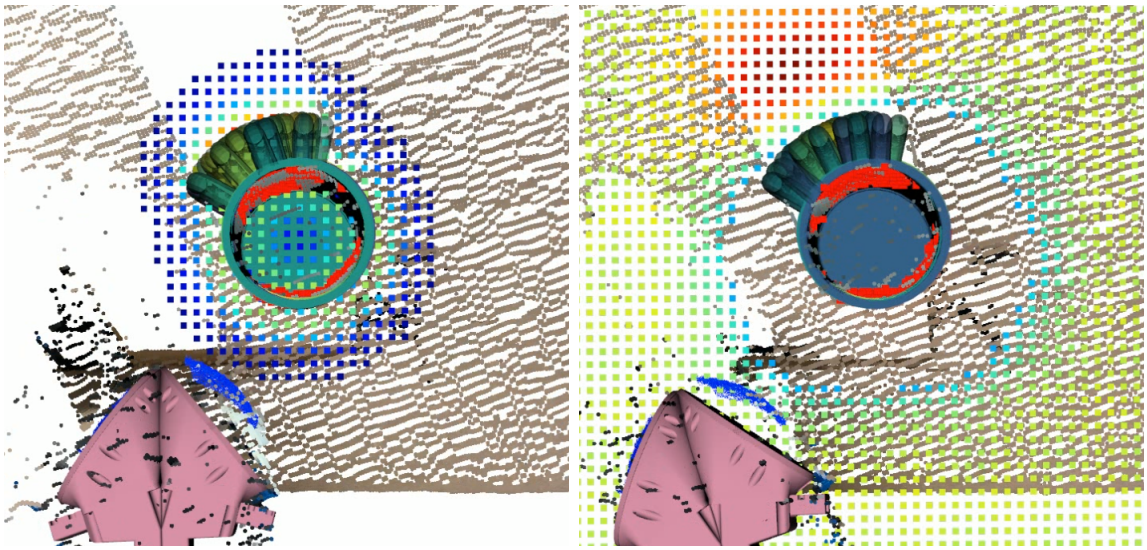

Figure 5.15: Comparison of the fields to plan over evaluated at each $\mathbf{x} \in \mathcal{W}$ for (left) our method using $\tilde{I}(\mathbf{x}|\mathcal{X})$ against the (right) GPVR baseline using $\mathbf{var}(\mathbf{x}|\mathcal{X})$. $\mathbf{x}$ with too low $\tilde{I}$ or $\mathbf{var}$ are omitted.

achieved 5 successes on sim drill because there was a 50% chance of it sliding clockwise or counterclockwise, with only one direction leading to success. This strategy however does often lead to it being the quickest method to make contact with the object, giving it low cumulative $nll(\mathcal{X})$ for the sim drill and box tasks.

The GPVR baseline's performance can be compared against the InfoOnly ablation's, as neither have an explicit cost for avoiding the object from being pushed out. As seen in Fig. 5.15, the highest $\mathbf{var}(\mathbf{x}|\mathcal{X})$, even when given points augmented using shape information, does not match where intuitively information about the shape might be held. This suggests that augmenting points is not a satisfactory way of conditioning on known object shape.

### 5.5.10 Runtime Comparison

We also recorded the average computation time per step in the sim mug task and sim box task to highlight RUMI's computational efficiency in Fig. 5.18. The activities are divided into ones that all methods performed (blue), those that only ours did (orange), and those that only the GPVR baseline did (green). The Slide baseline's runtime was negligible. All methods were implemented in PyTorch and accelerated by running on a modern computer with a NVIDIA RTX 4090 GPU. Computing $p(S_{\mathbf{x}}|\mathcal{X})$ and $\tilde{I}(\mathbf{x}|\mathcal{X})$ for $\mathbf{x} \in \mathcal{W}$ took around 0.01s per step, while evaluating our cost inside the MPC took around 0.1s for the sim mug. The time was dominated by

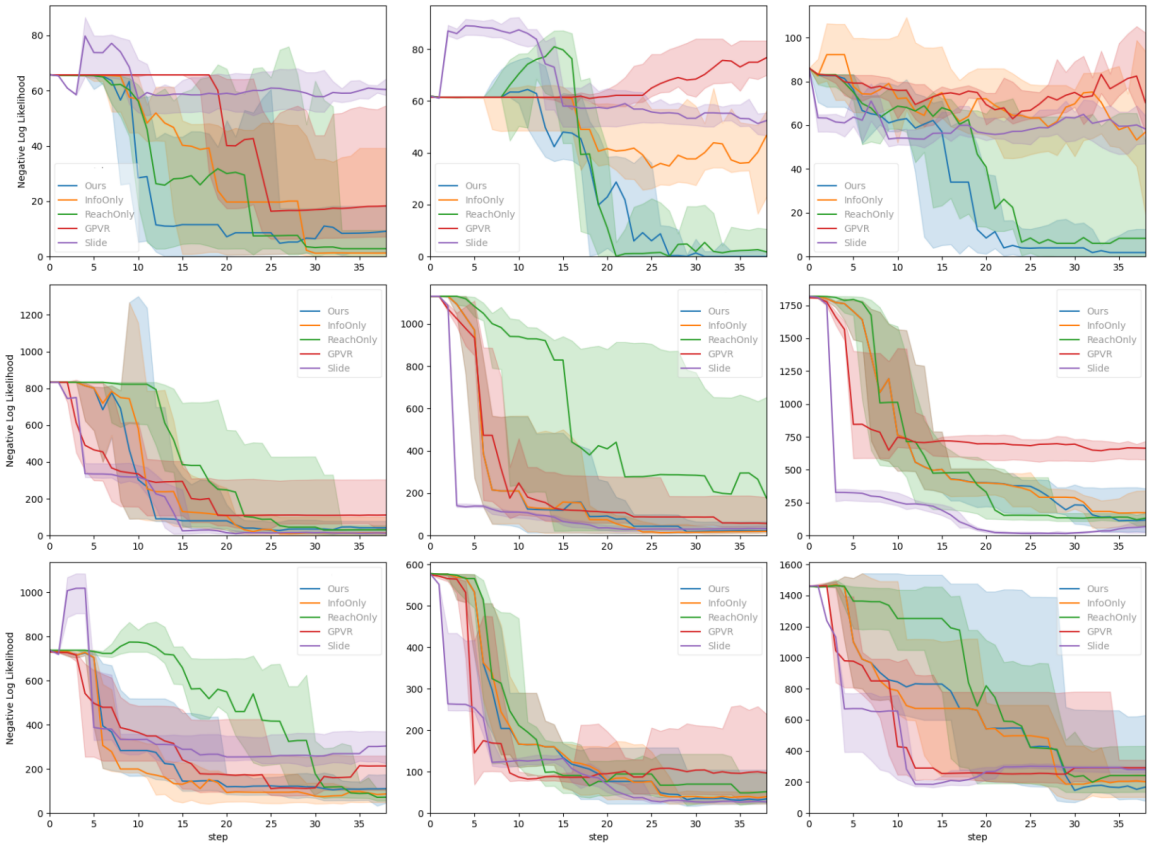


Figure 5.16: $nll(\mathcal{X})$ after each execution step for simulation tasks depicted in Fig. 5.10. The median over 10 runs is plotted, with the $25^{th}$ to $75^{th}$ percentile shaded. From top to bottom we have the sim mug, sim drill, and sim box tasks. From left to right we have configuration 0, 1, and 2.

evaluating $\hat{f}$ because it is stochastic and so benefited from sampling multiple state rollouts, in addition to dividing each step into 4 sequentially applied mini steps to avoid over-penetration. The GP fitting process took around 0.3s and involved caching $\mathbf{var}(\mathbf{x}|\mathcal{X})\ \forall \mathbf{x} \in \mathcal{W}$ in a voxel grid to speed up inference inside the cost.

In addition, we consider how well the methods scale to the full 3D sim box task. The main challenge is the increased $\mathcal{W}$ size, with approximately 2.24 times more total points. Caching $p(S_{\mathbf{x}}|\mathcal{X})$ and $\tilde{I}(\mathbf{x}|\mathcal{X})$ slowed down to 0.076s, or an increase of 7.6 times, while our cost evaluation increased around 4 times to 0.4s. The GPVR cost only doubled because we were down sampling the workspace by 7 (each voxel grid is 7 times larger) to augment the GP with.

Reducing the step size to no longer require dynamics mini steps would effectively improve the whole method's efficiency. Currently, RUMI can be run at around 1Hz, which was more than sufficient for quasi-static rummaging.

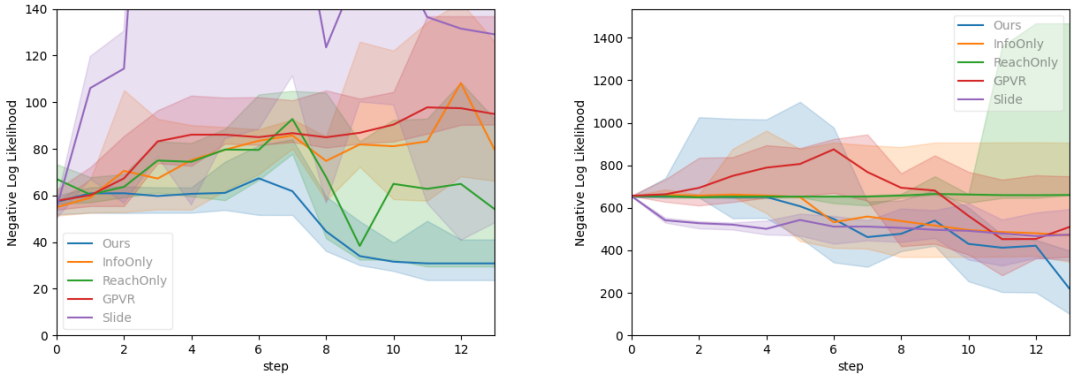


Figure 5.17: $nll(\mathcal{X})$ after each execution step for the (left) real mug and (right) real box tasks depicted in Fig. 5.12. The median over 10 runs is plotted, with the $25^{th}$ to $75^{th}$ percentile shaded.

## 5.6 Discussion and Future Work

### 5.6.1 Single Object Assumption

In this work we made the major assumption that there was only a single known movable target object in the workspace, with everything else being immovable and known. While this was necessary for us to tackle the other difficulties of the problem—namely limited visual perception, object symmetry, and gathering information by making contact with a movable object—realistic shelf environments are often cluttered with other movable objects. Additionally, we may not have the exact object mesh despite knowing its class (e.g. we are looking for some mug, but don't know it's exact shape). This introduces contact assignment ambiguity—any new contact points observed could belong to previously-observed objects, or a new object. Our prior work, STUCCO *Zhong et al.* (2022), tackled this problem by maintaining a belief over all the contact point positions, without any hard assignments to objects. The most likely estimate (MLE) of the contact positions is then passed to downstream tasks, including a process to segment the contact points into objects in a manner similar to agglomerative clustering. In future work, RUMI might be extended to handle multiple movable objects by evaluating $\tilde{I}(\mathbf{x}|\mathcal{X}) = \sum_i^O \tilde{I}(\mathbf{x}|\mathcal{X}_i)$ where O is the number of segmented objects, and $\mathcal{X}_i$ is the segmented $\mathcal{X}$ (free geometric features are shared across objects, in addition to other objects' surface positions being considered free for this object), treating all as candidate target objects.

On the sim tasks, we assumed we could accurately estimate slip between the object and the robot during contact. This is reasonable given known object shape,

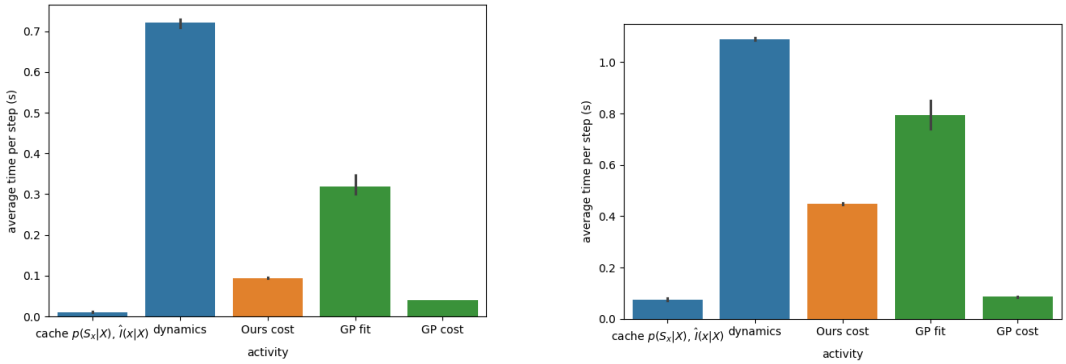


Figure 5.18: Average computation time per execution step of the (top) planar sim mug 0 task and the (bot) 3D sim box 0 task with one standard deviation as the error bar. Blue represents activities that were performed by all methods, orange represents just our method, and green represent activities that were only performed by the GPVR baseline.

mass distribution, and the coefficient of friction of interacting surfaces. However, this becomes unreasonable if the object is unknown. Additionally, Algorithm 14 for estimating slip on the real task requires knowledge of the object SDF. Thus one of the biggest challenges for extending RUMI to multiple, unknown objects in the scene will be estimating $\Delta\mathbf{T}$ from each contact. A reasonably accurate visual perception system could help significantly, particularly if it could identify which clustered $\mathcal{X}_i$ belongs to the target object.

### 5.6.2 Unknown Object Shape

The last point of improvement is to relax our knowledge of the object from having its SDF to just having a class label. One naive approach is to use a template SDF for each object class and absorb the SDF uncertainty into the sensor model $\mathbf{p}(S_\mathbf{x}|\texttt{sdf}(\mathbf{Tx}))$. However, this would fair poorly to classes that have high geometric variation inside each class, like mugs. One possible approach would be to extend the pose posterior particle filter to also represent object shape, such that each particle is both a pose and a shape. The shape could be parameterized by recent advances in 3D geometry representations, such as the Deformed Implicit Field *Lee et al.* (2022) that allows shape editing by constraining on surface points.

## 5.7 Conclusion

We presented RUMI, an active exploration method based on the mutual information between a movable target object's uncertain pose and the robot trajectory. It maintains an explicit belief over the object pose using a particle filter, updating it with observed point clouds augmented with semantics, such as whether a point is in free space or on the object surface. Given object SDF, we formulated an information gain cost function evaluating the expected KL divergence between the pose distribution before and after executing a robot trajectory. In addition, we implemented a reachability cost function and showed that it was necessary to prevent pushing the object outside the robot's reachable region. Through comparison with baselines in real and simulated experiments, we showed that RUMI could effectively and efficiently condition on object shape to explore and estimate object pose.

Table 5.5: Number of successful trials after 10 runs of active rummaging for pose estimation in different tasks in Fig. 5.10 and Fig. 5.12. Success is defined as the run achieving a minimum *nll* below the threshold defined in Tab. 5.3. The most and 1 success below the most successes in each section are bolded.

| Task | Ours | InfoOnly | ReachOnly | GPVR | Slide |
|---|---|---|---|---|---|
| sim mug 0 | **10** | 7 | 8 | 6 | 0 |
| sim mug 1 | **9** | 4 | **10** | 1 | 0 |
| sim mug 2 | **10** | 4 | 7 | 0 | 0 |
| mug total | **29** | 15 | 25 | 7 | 0 |
| sim drill 0 | **10** | 7 | 8 | 4 | 5 |
| sim drill 1 | **9** | **10** | 5 | 7 | **10** |
| sim drill 2 | 8 | 5 | 4 | 0 | **10** |
| drill total | **27** | 22 | 17 | 11 | 25 |
| sim box 0 | **8** | **9** | **8** | 6 | 0 |
| sim box 1 | **10** | **9** | **10** | **9** | **10** |
| sim box 2 | **7** | 5 | 4 | 2 | 1 |
| box total | **25** | 23 | 22 | 17 | 11 |
| real mug | **7** | 0 | 4 | 0 | 1 |
| real box | **7** | 3 | 0 | 2 | 3 |

Table 5.6: Median cumulative $nll(\mathcal{X})$ for 10 runs of active rummaging for pose estimation in different tasks in Fig. 5.10 and Fig. 5.12. The best and any 5% within the best are bolded.

| Task | Ours | InfoOnly | ReachOnly | GPVR | Slide |
|---|---|---|---|---|---|
| sim mug 0 | **823** | 1246 | 1162 | 1678 | 2478 |
| sim mug 1 | **1048** | 1918 | 1521 | 2459 | 2610 |
| sim mug 2 | **841** | 2859 | 1502 | 2973 | 2467 |
| mug total | **2712** | 6023 | 4185 | 7110 | 7555 |
| sim drill 0 | 9797 | 12860 | 16100 | 12516 | **6328** |
| sim drill 1 | 9637 | 8423 | 22666 | 13636 | **5617** |
| sim drill 2 | 27121 | 26761 | 25452 | 35090 | **10200** |
| drill total | 46555 | 48044 | 64218 | 61242 | **22145** |
| sim box 0 | 8776 | **8182** | 20537 | 11734 | 15497 |
| sim box 1 | 7004 | 7067 | 8075 | 7499 | **5129** |
| sim box 2 | 24028 | 24839 | 33705 | **15979** | 17653 |
| box total | 39808 | 40088 | 62317 | **35212** | 38279 |
| real mug | **640** | 1330 | 946 | 1145 | 4652 |
| real box | 9177 | 8341 | 8762 | 9994 | **7717** |

# BIBLIOGRAPHY


Alspach, A., K. Hashimoto, N. Kuppuswamy, and R. Tedrake (2019), Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation, in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pp. 597–604, IEEE.

Andreopoulos, A., S. Hasler, H. Wersing, H. Janssen, J. K. Tsotsos, and E. Korner (2010), Active 3d object localization using a humanoid robot, *IEEE Transactions on Robotics*, *27*(1), 47–64.

Andrychowicz, O. M., et al. (2020), Learning dexterous in-hand manipulation, *The International Journal of Robotics Research*, *39*(1), 3–20.

Arras, K. O., J. A. Castellanos, M. Schilt, and R. Siegwart (2003), Feature-based multi-hypothesis localization and tracking using geometric constraints, *Robotics and Autonomous Systems*, *44*(1), 41–53.

Arun, K. S., T. S. Huang, and S. D. Blostein (1987), Least-squares fitting of two 3-d point sets, *IEEE Transactions on pattern analysis and machine intelligence*, (5), 698–700.

Bajcsy, R., Y. Aloimonos, and J. K. Tsotsos (2018), Revisiting active perception, *Autonomous Robots*, *42*, 177–196.

Bauza, M., and A. Rodriguez (2017), A probabilistic data-driven model for planar pushing, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3008–3015, IEEE.

Bellemare, M., S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos (2016), Unifying count-based exploration and intrinsic motivation, in *NeurIPS*, pp. 1471–1479.

Betke, M., D. E. Hirsh, A. Bagchi, N. I. Hristov, N. C. Makris, and T. H. Kunz (2007), Tracking large variable numbers of objects in clutter, in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, IEEE.

Blackman, S. S. (2004), Multiple hypothesis tracking for multiple target tracking, *IEEE Aerospace and Electronic Systems Magazine*, *19*(1), 5–18.

Bohg, J., K. Hausman, B. Sankaran, O. Brock, D. Kragic, S. Schaal, and G. S. Sukhatme (2017), Interactive perception: Leveraging action in perception and perception in action, *IEEE Transactions on Robotics*, *33*(6), 1273–1291.

Borenstein, J., G. Granosik, and M. Hansen (2005), The omnitread serpentine robot: design and field performance, in *Unmanned Ground Vehicle Technology VII*, vol. 5804, pp. 324–332, SPIE.

Bouaziz, S., A. Tagliasacchi, and M. Pauly (2013), Sparse iterative closest point, in *Computer graphics forum*, vol. 32, pp. 113–123, Wiley Online Library.

Buch, A. G., D. Kraft, et al. (2017), Prediction of icp pose uncertainties using monte carlo simulation with synthetic depth images, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4640–4647, IEEE.

Burda, Y., H. Edwards, A. Storkey, and O. Klimov (2018), Exploration by random network distillation, *arXiv preprint arXiv:1810.12894*.

Caccamo, S., Y. Bekiroglu, C. H. Ek, and D. Kragic (2016), Active exploration using gaussian random fields and gaussian process implicit surfaces, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 582–589, IEEE.

Cai, C., and S. Ferrari (2009), Information-driven sensor path planning by approximate cell decomposition, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(3), 672–689.

Calli, B., A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar (2017), Yale-cmu-berkeley dataset for robotic manipulation research, *The International Journal of Robotics Research*, *36*(3), 261–268.

Cao, N., K. H. Low, and J. M. Dolan (2013), Multi-robot informative path planning for active sensing of environmental phenomena: a tale of two algorithms, in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 7–14.

Carrillo, H., P. Dames, V. Kumar, and J. A. Castellanos (2015), Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy, in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 487–494, IEEE.

Casella, G., and E. I. George (1992), Explaining the gibbs sampler, *The American Statistician*, *46*(3), 167–174.

Censi, A. (2007), An accurate closed-form estimate of icp's covariance, in *Proceedings 2007 IEEE international conference on robotics and automation*, pp. 3167–3172, IEEE.

Chen, X., Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel (2016), Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in *NeurIPS*.

Cheng, L., S. Chen, X. Liu, H. Xu, Y. Wu, M. Li, and Y. Chen (2018), Registration of laser scanning point clouds: A review, *Sensors*, *18*(5), 1641.

Collet, A., D. Berenson, S. S. Srinivasa, and D. Ferguson (2009), Object recognition and full pose registration from a single image for robotic manipulation, in *2009 IEEE International Conference on Robotics and Automation*, pp. 48–55, IEEE.

Coumans, E., and Y. Bai (2016–2021), Pybullet, a python module for physics simulation for games, robotics and machine learning, `http://pybullet.org`.

Danielczuk, M., A. Kurenkov, A. Balakrishna, M. Matl, D. Wang, R. Martín-Martín, A. Garg, S. Savarese, and K. Goldberg (2019), Mechanical search: Multi-step retrieval of a target object occluded by clutter, in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1614–1621, IEEE.

De Luca, A., and R. Mattone (2005), Sensorless robot collision detection and hybrid force/motion control, in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 999–1004, IEEE.

Deng, X., A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox (2021), Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking, *IEEE Transactions on Robotics*, *37*(5), 1328–1342.

Dierks, T., and S. Jagannathan (2012), Online optimal control of affine nonlinear discrete-time systems with unknown internal dynamics by using time-based policy update, *IEEE Trans Neural Netw Learn Syst*, *23*(7), 1118–1129.

Dikhale, S., K. Patel, D. Dhingra, I. Naramura, A. Hayashi, S. Iba, and N. Jamali (2022), Visuotactile 6d pose estimation of an in-hand object using vision and tactile sensor data, *IEEE Robotics and Automation Letters*, *7*(2), 2148–2155.

Dragiev, S., M. Toussaint, and M. Gienger (2011), Gaussian process implicit surfaces for shape estimation and grasping, in *2011 IEEE International Conference on Robotics and Automation*, pp. 2845–2850, IEEE.

Driess, D., P. Englert, and M. Toussaint (2017), Active learning with query paths for tactile object shape exploration, in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 65–72, IEEE.

Driess, D., D. Hennes, and M. Toussaint (2019), Active multi-contact continuous tactile exploration with gaussian process differential entropy, in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7844–7850, IEEE.

Ecoffet, A., J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune (2019), Go-explore: a new approach for hard-exploration problems, *arXiv preprint arXiv:1901.10995*.

Eidenberger, R., and J. Scharinger (2010), Active perception and scene modeling by planning with probabilistic 6d object poses, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1036–1043, IEEE.

Elandt, R., E. Drumwright, M. Sherman, and A. Ruina (2019), A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects, in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8238–8245, IEEE.

Fantoni, I., and R. Lozano (2012), *Non-linear control for underactuated mechanical systems*, Springer Science & Business Media.

Fedele, G., L. D'Alfonso, F. Chiaravalloti, and G. D'Aquila (2018), Obstacles avoidance based on switching potential functions, *Journal of Intelligent & Robotic Systems*, *90*(3-4), 387–405.

Finn, C., P. Abbeel, and S. Levine (2017), Model-agnostic meta-learning for fast adaptation of deep networks, in *ICML*.

Fontaine, M., and S. Nikolaidis (2021), Differentiable quality diversity, *Advances in Neural Information Processing Systems*, *34*, 10,040–10,052.

Fontaine, M. C., J. Togelius, S. Nikolaidis, and A. K. Hoover (2020), Covariance matrix adaptation for the rapid illumination of behavior space, in *Proceedings of the 2020 genetic and evolutionary computation conference*, pp. 94–102.

Fortmann, T., Y. Bar-Shalom, and M. Scheffe (1983), Sonar tracking of multiple targets using joint probabilistic data association, *IEEE journal of Oceanic Engineering*, *8*(3), 173–184.

Fowlkes, E. B., and C. L. Mallows (1983), A method for comparing two hierarchical clusterings, *Journal of the American statistical association*, *78*(383), 553–569.

Fu, J., S. Levine, and P. Abbeel (2016), One-shot learning of manipulation skills with online dynamics adaptation and neural network priors, in *IROS*.

Fu, J., Q. Huang, K. Doherty, Y. Wang, and J. J. Leonard (2021), A multi-hypothesis approach to pose ambiguity in object-based slam, in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7639–7646, IEEE.

Geyer, C. J. (1992), Practical markov chain monte carlo, *Statistical science*, pp. 473–483.

Haarnoja, T., H. Tang, P. Abbeel, and S. Levine (2017), Reinforcement learning with deep energy-based policies, in *International conference on machine learning*, pp. 1352–1361, PMLR.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine (2018), Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *ICML*, pp. 1861–1870.

Haddadin, S., A. De Luca, and A. Albu-Schäffer (2017), Robot collisions: A survey on detection, isolation, and identification, *IEEE Transactions on Robotics*, *33*(6), 1292–1312.

Hansen, N., S. D. Müller, and P. Koumoutsakos (2003), Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es), *Evolutionary computation*, *11*(1), 1–18.

Haugo, S., and A. Stahl (2020), Iterative closest point with minimal free space constraints, in *Advances in Visual Computing: 15th International Symposium, ISVC 2020, San Diego, CA, USA, October 5–7, 2020, Proceedings, Part II 15*, pp. 82–95, Springer.

Jadidi, M. G., J. V. Miro, and G. Dissanayake (2015), Mutual information-based exploration on continuous occupancy maps, in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6086–6092, IEEE.

Jain, A., M. D. Killpack, A. Edsinger, and C. C. Kemp (2013), Reaching in clutter with whole-arm tactile sensing, *The International Journal of Robotics Research*, *32*(4), 458–482.

Jaiswal, A., and B. Kumar (2017), Vacuum gripper-an important material handling tool, *Int J Sci Technol*, *7*, 1–8.

Kaboli, M., and G. Cheng (2016), Novel tactile descriptors and a tactile transfer learning technique for active in-hand object recognition via texture properties, in *IEE-RAS International Conference on Humanoid Robots-Workshop Tactile sensing for manipulation: new progress and challenges*.

Kahn, G., P. Sujan, S. Patil, S. Bopardikar, J. Ryde, K. Goldberg, and P. Abbeel (2015), Active exploration using trajectory optimization for robotic grasping in the presence of occlusions, in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4783–4790, IEEE.

Khatib, O. (1986), Real-time obstacle avoidance for manipulators and mobile robots, *The international journal of robotics research*, *5*(1), 90–98.

Koditschek, D. E., R. J. Full, and M. Buehler (2004), Mechanical aspects of legged locomotion control, *Arthropod structure & development*, *33*(3), 251–272.

Koval, M. C., N. S. Pollard, and S. S. Srinivasa (2015), Pose estimation for planar contact manipulation with manifold particle filters, *The International Journal of Robotics Research*, *34*(7), 922–945.

Krainin, M., B. Curless, and D. Fox (2011), Autonomous generation of complete 3d object models using next best view manipulation planning, in *2011 IEEE international conference on robotics and automation*, pp. 5031–5037, IEEE.

Krueger, D., E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, R. L. Priol, and A. Courville (2020), Out-of-distribution generalization via risk extrapolation (rex), *arXiv preprint arXiv:2003.00688*.

Kullback, S. (1997), *Information theory and statistics*, Courier Corporation.

Kuppuswamy, N., A. Alspach, A. Uttamchandani, S. Creasey, T. Ikeda, and R. Tedrake (2020), Soft-bubble grippers for robust and perceptive manipulation, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9917–9924, IEEE.

Lee, B., C. Zhang, Z. Huang, and D. D. Lee (2019), Online continuous mapping using gaussian process implicit surfaces, in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6884–6890, IEEE.

Lee, M. C., and M. G. Park (2003), Artificial potential field based path planning for mobile robots using a virtual obstacle concept, in *AIM*.

Lee, S., L. Chen, J. Wang, A. Liniger, S. Kumar, and F. Yu (2022), Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields, *IEEE Robotics and Automation Letters*, *7*(4), 12,070–12,077.

Leung, C., S. Huang, N. Kwok, and G. Dissanayake (2006), Planning under uncertainty using model predictive control for information gathering, *Robotics and Autonomous Systems*, *54*(11), 898–910.

Li, D., Y. Yang, Y.-Z. Song, and T. M. Hospedales (2018), Learning to generalize: Meta-learning for domain generalization, in *AAAI*.

Li, T., M. Bolic, and P. M. Djuric (2015), Resampling methods for particle filtering: classification, implementation, and strategies, *IEEE Signal processing magazine*, *32*(3), 70–86.

Liu, H., Y. Wu, F. Sun, and D. Guo (2017), Recent progress on tactile object recognition, *International Journal of Advanced Robotic Systems*, *14*(4), 1729881417717,056.

Liu, Q., and D. Wang (2016), Stein variational gradient descent: A general purpose bayesian inference algorithm, *Advances in neural information processing systems*, *29*.

Lluvia, I., E. Lazkano, and A. Ansuategi (2021), Active mapping and robot exploration: A survey, *Sensors*, *21*(7), 2445.

Luo, W., J. Xing, A. Milan, X. Zhang, W. Liu, and T.-K. Kim (2020), Multiple object tracking: A literature review, *Artificial Intelligence*, p. 103448.

MacDonald, R. A., and S. L. Smith (2019), Active sensing for motion planning in uncertain environments via mutual information policies, *The International Journal of Robotics Research*, *38*(2-3), 146–161.

Magnusson, M. (2009), The three-dimensional normal-distributions transform: an efficient representation for registration, surface analysis, and loop detection, Ph.D. thesis, Örebro universitet.

Magnusson, M., A. Lilienthal, and T. Duckett (2007), Scan registration for autonomous mining vehicles using 3d-ndt, *Journal of Field Robotics*, *24*(10), 803–827.

Maken, F. A., F. Ramos, and L. Ott (2021), Stein ICP for Uncertainty Estimation in Point Cloud Matching, *IEEE Robotics and Automation Letters*, *7*(2), 1063–1070.

Manuelli, L., and R. Tedrake (2016), Localizing external contact using proprioceptive sensors: The contact particle filter, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5062–5069, IEEE.

Martens, W., Y. Poffet, P. R. Soria, R. Fitch, and S. Sukkarieh (2016), Geometric priors for gaussian process implicit surfaces, *IEEE Robotics and Automation Letters*, *2*(2), 373–380.

Masterjohn, J., D. Guoy, J. Shepherd, and A. Castro (2022), Velocity level approximation of pressure field contact patches, *IEEE Robotics and Automation Letters*, *7*(4), 11,593–11,600.

McConachie, D., and D. Berenson (2020), Bandit-based model selection for deformable object manipulation, in *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pp. 704–719, Springer.

Melchiorri, C. (2000), Slip detection and control using tactile and force sensors, *IEEE/ASME transactions on mechatronics*, *5*(3), 235–243.

Meyer-Delius, D., M. Beinhofer, and W. Burgard (2012), Occupancy grid models for robot mapping in changing environments, in *Proceedings of the AAAI conference on artificial intelligence*, vol. 26, pp. 2024–2030.

Milnor, J. (1985), On the concept of attractor, *Communications in Mathematical Physics*, *99*, 177–195.

Miura, T., N. Akai, K. Honda, and S. Hara (2024), Spline-interpolated model predictive path integral control with stein variational inference for reactive navigation, *arXiv preprint arXiv:2404.10395*.

Morrison, D., et al. (2018), Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7757–7764, IEEE.

Mouret, J.-B., and J. Clune (2015), Illuminating search spaces by mapping elites, *arXiv preprint arXiv:1504.04909*.

Murphy, K. P. (2012), *Machine learning: a probabilistic perspective*, MIT press.

Osband, I., C. Blundell, A. Pritzel, and B. Van Roy (2016), Deep exploration via bootstrapped dqn, in *NeurIPS*, pp. 4026–4034.

Ottenhaus, S., D. Renninghoff, R. Grimm, F. Ferreira, and T. Asfour (2019), Visuo-haptic grasping of unknown objects based on gaussian process implicit surfaces and deep learning, in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 402–409, IEEE.

Pan, S. J., I. W. Tsang, J. T. Kwok, and Q. Yang (2010), Domain adaptation via transfer component analysis, *IEEE Transactions on Neural Networks*, *22*(2), 199–210.

Pathak, D., P. Agrawal, A. A. Efros, and T. Darrell (2017), Curiosity-driven exploration by self-supervised prediction, in *CVPR Workshops*.

Peng, X. B., M. Andrychowicz, W. Zaremba, and P. Abbeel (2018), Sim-to-real transfer of robotic control with dynamics randomization, in *ICRA*, pp. 1–8, IEEE.

Piegl, L., and W. Tiller (1996), *The NURBS book*, Springer Science & Business Media.

Popović, M., T. Vidal-Calleja, J. J. Chung, J. Nieto, and R. Siegwart (2020), Informative path planning for active field mapping under localization uncertainty, in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10,751–10,757, IEEE.

Prattichizzo, D., and J. C. Trinkle (2016), Grasping, in *Springer handbook of robotics*, pp. 955–988, Springer.

Pugh, J. K., L. B. Soros, and K. O. Stanley (2016), Quality diversity: A new frontier for evolutionary computation, *Frontiers in Robotics and AI*, p. 40.

Romeo, R. A., and L. Zollo (2020), Methods and sensors for slip detection in robotics: A survey, *Ieee Access*, *8*, 73,027–73,050.

Rustler, L., J. Lundell, J. K. Behrens, V. Kyrki, and M. Hoffmann (2022), Active visuo-haptic object shape completion, *IEEE Robotics and Automation Letters*, *7*(2), 5254–5261.

Ryan, A., and J. K. Hedrick (2010), Particle filter based information-theoretic active sensing, *Robotics and Autonomous Systems*, *58*(5), 574–584.

Saund, B., and D. Berenson (2021), Diverse plausible shape completions from ambiguous depth images, in *Conference on Robot Learning*, pp. 1802–1813, PMLR.

Saund, B., and D. Berenson (2022), CLASP: Constrained Latent Shape Projection for Refining Object Shape from Robot Contact, in *Conference on Robot Learning*, pp. 1391–1400, PMLR.

Schubert, E., J. Sander, M. Ester, H. P. Kriegel, and X. Xu (2017), Dbscan revisited, revisited: why and how you should (still) use dbscan, *ACM Transactions on Database Systems (TODS)*, *42*(3), 1–21.

Schwarz, M., A. Milan, A. S. Periyasamy, and S. Behnke (2018), Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter, *The International Journal of Robotics Research*, *37*(4-5), 437–451.

Segal, A., D. Haehnel, and S. Thrun (2009), Generalized-icp., in *Robotics: science and systems*, vol. 2, p. 435, Seattle, WA.

Sim, R., and N. Roy (2005), Global a-optimal robot exploration in slam, in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 661–666, IEEE.

Sipos, A., and N. Fazeli (2022), Simultaneous contact location and object pose estimation using proprioception and tactile feedback, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3233–3240, IEEE.

Slavcheva, M., W. Kehl, N. Navab, and S. Ilic (2016), Sdf-2-sdf: Highly accurate 3d object reconstruction, in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pp. 680–696, Springer.

Smith, E., D. Meger, L. Pineda, R. Calandra, J. Malik, A. Romero Soriano, and M. Drozdzal (2021), Active 3d shape reconstruction from vision and touch, *Advances in Neural Information Processing Systems*, *34*, 16,064–16,078.

Snelson, E., and Z. Ghahramani (2005), Sparse gaussian processes using pseudo-inputs, *Advances in neural information processing systems*, *18*.

Stachniss, C., and W. Burgard (2003), Exploring unknown environments with mobile robots using coverage maps, in *IJCAI*, vol. 2003, pp. 1127–1134.

Stone, L. D., R. L. Streit, T. L. Corwin, and K. L. Bell (2013), *Bayesian multiple target tracking*, Artech House.

Suresh, S., M. Bauza, K.-T. Yu, J. G. Mangelson, A. Rodriguez, and M. Kaess (2020), Tactile slam: Real-time inference of shape and pose from planar pushing, *arXiv preprint arXiv:2011.07044*.

Suresh, S., M. Bauza, K.-T. Yu, J. G. Mangelson, A. Rodriguez, and M. Kaess (2021), Tactile slam: Real-time inference of shape and pose from planar pushing, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11,322–11,328, IEEE.

Suresh, S., Z. Si, S. Anderson, M. Kaess, and M. Mukadam (2022), Midastouch: Monte-carlo inference over distributions across sliding touch, *arXiv preprint arXiv:2210.14210*.

Tam, G. K., Z.-Q. Cheng, Y.-K. Lai, F. C. Langbein, Y. Liu, D. Marshall, R. R. Martin, X.-F. Sun, and P. L. Rosin (2012), Registration of 3d point clouds and meshes: A survey from rigid to nonrigid, *IEEE transactions on visualization and computer graphics*, *19*(7), 1199–1217.

Teh, Y. W., M. Welling, S. Osindero, and G. E. Hinton (2003), Energy-based models for sparse overcomplete representations, *Journal of Machine Learning Research*, *4*(Dec), 1235–1260.

Tobin, J., R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel (2017), Domain randomization for transferring deep neural networks from simulation to the real world, in *IROS*, pp. 23–30, IEEE.

Vespa, E., N. Nikolov, M. Grimm, L. Nardi, P. H. Kelly, and S. Leutenegger (2018), Efficient octree-based volumetric slam supporting signed-distance and occupancy mapping, *IEEE Robotics and Automation Letters*, *3*(2), 1144–1151.

Vo, B.-N., and W.-K. Ma (2006), The gaussian mixture probability hypothesis density filter, *IEEE Transactions on signal processing*, *54*(11), 4091–4104.

Wang, F., and Z. Zhao (2017), A survey of iterative closest point algorithm, in *2017 Chinese Automation Congress (CAC)*, pp. 4395–4399, IEEE.

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

Williams, G., N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou (2017b), Information theoretic mpc for model-based reinforcement learning, in *ICRA*.

Wong, J. M., et al. (2017), Segicp: Integrated deep semantic segmentation and pose estimation, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5784–5789, IEEE.

Wu, Y., J. Lim, and M.-H. Yang (2013), Online object tracking: A benchmark, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2411–2418.

Xu, J., H. Lin, S. Song, and M. Ciocarlie (2022), Tandem3d: Active tactile exploration for 3d object recognition, *arXiv preprint arXiv:2209.08772*.

Yi, Z., R. Calandra, F. Veiga, H. van Hoof, T. Hermans, Y. Zhang, and J. Peters (2016), Active tactile object exploration with gaussian processes, in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4925–4930, IEEE.

Yu, K.-T., and A. Rodriguez (2018), Realtime state estimation with tactile and visual sensing for inserting a suction-held object, in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1628–1635, IEEE.

Yuan, W., S. Dong, and E. H. Adelson (2017), Gelsight: High-resolution robot tactile sensors for estimating geometry and force, *Sensors*, *17*(12), 2762.

Zeng, A., et al. (2022), Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching, *The International Journal of Robotics Research*, *41*(7), 690–705.

Zhang, A., H. Satija, and J. Pineau (2018), Decoupling dynamics and reward for transfer learning, *arXiv preprint arXiv:1804.10689*.

Zhang, R., T.-Y. Lin, C. E. Lin, S. A. Parkison, W. Clark, J. W. Grizzle, R. M. Eustice, and M. Ghaffari (2021), A new framework for registration of semantic point clouds from stereo and rgb-d cameras, in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 12,214–12,221, IEEE.

Zhang, T., R. Ramakrishnan, and M. Livny (1996), Birch: an efficient data clustering method for very large databases, *ACM sigmod record*, *25*(2), 103–114.

Zhang, Y., J. Pan, L. K. Li, W. Liu, Z. Chen, X. Liu, and J. Wang (2024), On the properties of kullback-leibler divergence between multivariate gaussian distributions, *Advances in Neural Information Processing Systems*, *36*.

Zhong, S., and T. Power (2019), Pytorch model predictive path integral controller, `https://github.com/LemonPi/pytorch_mppi`.

Zhong, S., N. Fazeli, and D. Berenson (2022), Soft tracking using contacts for cluttered objects to perform blind object retrieval, *IEEE Robotics and Automation Letters*, *7*(2), 3507–3514.

Zhong, S., D. Berenson, and N. Fazeli (2023), Chsel: Producing diverse plausible pose estimates from contact and free space data, in *Robotics: Science and Systems*.

Zhou, Y., C. Barnes, J. Lu, J. Yang, and H. Li (2019), On the continuity of rotation representations in neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5745–5753.