Belief Representations for Planning with Contact Uncertainty

by

Brad Saund

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Robotics)
in The University of Michigan
2021

Brad Saund
bsaund@umich.edu
ORCID iD: 0000-0002-9765-0258

I dedicate this thesis to my newborn son, Kevin Saund. I've only known you for 6 months, but you have already brought me incredible joy. May your curiosity never wane and your happiness be plentiful.

ACKNOWLEDGEMENTS

The only way I know to develop ideas such as presented here is through discussion and iteration with smart and enthusiastic colleagues. Thanks first goes to the ARM lab. Dmitry provided me with incredible freedom to pursue a variety of topics, while nudging me towards the areas with the most potential. Dmitry's guidance was instrumental in both developing many of the ideas in this thesis, and endowing our initial nebulous thoughts with precise mathematical form.

I have grown from my many conversations with the members of the ARM lab. Dale McConachie and I have filled many whiteboards with robotics, software, networking, and more. With Andrew Price I have covered similar topics while also diving into philosophy. Peter, Tom, Johnson, Glen, Yu-Chi, and others have helped me through brainstorming, technical discussions, code reviews, and invaluable feedback.

Thanks must also go to Victor, the ARM lab robot with whom I am quite familiar. I love robotics for the combination of mechanical, electrical, mathematics, and software components, and Victor provided a wonderful playground for my ideas.

Finally, I thank my family. My parents raised me with such incredible love and have encouraged my curiosity my entire life. Every day I look at my wife Katie and cannot believe how much I love her, and every day I love her more than the last. I have never imagined a more perfect partner for me in life.

TABLE OF CONTENTS

DEDICATION ii

ACKNOWLEDGEMENTS iii

LIST OF FIGURES vii

LIST OF TABLES x

ABSTRACT xi

CHAPTER

I. Introduction 1

1.1 Problem Characterization 2
1.1.1 Reconstruction 2
1.1.2 Belief Space Planning 3
1.2 Summary of Contributions 4

II. Planning with Contact Sensing Uncertainty 5

2.1 Introduction 5
2.2 Related Work 6
2.3 Problem Statement 8
2.4 Representing Uncertain Contact Information 9
2.4.1 Baseline: Unified Cost Grid 10
2.4.2 Collision Hypothesis Sets 10
2.5 Interleaving Planning and Control 11
2.6 Experiments and Results 15
2.7 Conclusions and Future Work 18

III. The Blindfolded Traveler's Problem 20

3.1 Introduction 20
3.2 Related Work 22

3.3 Problem Statement 24
3.3.1 Blindfolded Traveler's Problem 24
3.3.2 Contact-based Planning Problem as an instance of BTP 25
3.4 Belief Representations for Contact-based Planning 27
3.4.1 Approach 1: Manifold Particle Filter (MPF) 27
3.4.2 Approach 2: Collision Hypothesis Sets (CHS) 28
3.5 Strategies for Solving the BTP 31
3.5.1 Collision Measure (CM) 31
3.5.2 Baselines 32
3.5.3 Pitfalls for Heuristic Strategies 33
3.6 Experiments 34
3.7 Conclusions 37

IV. Plausible Shape Sampling 38

4.1 Introduction 38
4.2 Related Work 39
4.3 Problem Formulation and Metrics 42
4.4 Method 43
4.5 Quantifying Plausibility 45
4.6 Experiments 46
4.7 Discussion 49
4.8 Conclusion 51

V. CLASP: Shape Completion with Contact 52

5.1 Introduction 52
5.2 Related Work 55
5.3 Problem Formulation 56
5.4 Method 57
5.4.1 Initial Belief 57
5.4.2 Projecting a single object 58
5.4.3 Multi-object completion 59
5.5 Experiments 59
5.5.1 Robot Contacts 60
5.5.2 Scenes 60
5.5.3 Baselines 62
5.5.4 Results 63
5.6 Discussion and Conclusion 64

VI. Conclusion 66

6.1 Summary of Contributions 66
6.1.1 Contact Information 66

6.1.2 Belief Representations from Contact Information . . 67
6.1.3 Planning 68
6.2 Future Work and Extensions 69
6.2.1 Movable Objects 69
6.2.2 Leaving the Graph 69
6.2.3 Context-dependent Goals 71
6.3 Final Remarks 74

APPENDICES 75
A.1 Overview of Robotic Hardware 76
A.2 Analysis of BTP 76
A.2.1 Mapping the BTP to a POMDP 76
A.2.2 Computational Complexity 77
A.2.3 Relation to the Bayesian Canadian Traveler's Problem 80
A.3 Strategies for Solving the BTP 80
A.3.1 Heuristic Estimates of Q-values 81
A.3.2 Simulation-based Policies 86
A.3.3 Planning to gather information 87
A.4 CLASP Experiment Details and Alternate Analysis 88
A.4.1 Shape Network Training 88
A.4.2 Robot Motion Generation 89
A.4.3 Likelihood Results 89

BIBLIOGRAPHY 92

LIST OF FIGURES

Figure

1.1 Overview of method in this Thesis 2
2.1 A plan for the green robot (left) results in a collision with the grey unknown obstacle. A red collision hypothesis set is added using links possibly in collision based on measured joint torques (center). The known free space is removed (right). 10
2.2 Plans for the green arm sweep through the blue region. Red: a CHS 12
2.3 Simulated Scenarios 15
2.4 Physical Robot environments 15
2.5 Experimental parameters 17
2.6 Total time and failure % for R4, averaged over 10 trials for each clearance with a 5 min. timeout 18
3.1 Overview of the BTP framework for planning with contact feedback. The robot is uncertain about location of the back wall. As it attempts to traverse edges, it partially localizes the wall and eventually finds its way to the goal. 21
3.2 Blindfolded Traveler's Problem 25
3.3 Manifold Particle Filter: The initial particles Φ_0 model configurations of the true obstacle before the robot moves (top). A collision during a motion causes particles to be resampled on the contact manifold (middle). Subsequent free space motions sweep through and eliminate some particles (bottom). 28
3.4 CHS: The robot initially plans a motion optimistic about unknown space (top). A motion sweeps out free space (blue) and a collision generates a CHS (middle). Future free space motion sweeps out more free space, potentially shrinking CHSs (bottom). 29
3.5 Pitfalls for various strategies for a 2D BTP problems. 33
3.6 `Refrigerator` - Victor moving to place an object inside a refrigerator. 34
3.7 `RealTable` - Victor moving from below to above a table. 35
3.8 Simulation scenarios. Here CM is used in all scenarios. Top left: `Easy` setting of `Box` using CHS. Top right: `Easy` setting of `Box` using MPF. Bottom left: `Hard` setting of `Box` using MoE. Bottom right: `Hard` setting of `Bookshelf` using CHS. 36

3.9 Results of applying various belief strategies and policies to the `Bookshelf` BTP. Our proposed MoE+CM is consistently fast and solves the BTP with low cost. 36
4.1 Our proposed PSSNet applied to a noisy segmented Kinect depth image of a mug produces multiple plausible reconstructions 40
4.2 Our network, PSSNet, has the structure of a VAE during inference. During training we separate the latent space into typical learned VAE features and "latent box" feature produced by a learned normalizing flow applied to the ground truth bounding box. These latent box features are used both as a loss on the encoder prediction and as input to the decoder during training. 43
4.3 Coverage (Eq. 4.1) of various methods for shapenet mugs at different rotation angles. Rotations where the mug handle is occluded are highlighted. 47
4.4 Completions (green) of a mug are sampled from the visible 2.5D view (grey). When the handle is visible (left) all methods produce similar mugs close to the ground truth (GT) (blue). When the handle is occluded (right) sampling from PSSNet yields mugs with different styles of handles in different orientations, with similar variation seen in the plausible set (4 shapes shown). 47
4.5 Completions of YCB objects as viewed through a 6 pixel narrow slit with the nearest plausible shape shown for each network sample. PSSNet generates diverse samples where other networks generate only small variations on the same sample. 49
5.1 A visual RGBD view of objects leave ambiguity final shape due to sensor noise and occlusion, which we store as a set of sampled scenes in a particle filter. Contact information (pink) reduces ambiguity, and using CLASP the particles converge to the true shape. 54
5.2 CLASP Architecture **Top:** Shapes sampled from RGBD. **Middle Right:** A robot motion detects free space (light blue) and a collision set (pink). **Middle Left:** Latent samples are projected to satisfy contacts (green). Ovals depict the latent prior. **Bottom:** Final samples satisfy the contact constraints. 54
5.3 Boxplots showing the Chamfer Distance from sampled particles to ground truth. The mean, middle quartiles (boxed colored region), and outer quartiles excluding outliers are shown. Rejection Sampling and VAE_GAN occasionally produced no valid shapes, in which case no box is displayed. 61
5.4 Boxplot results for the multiobject scene. PSSNet + CLASP: No contact disambiguation fails to project any samples for observations 4 and beyond, so there is not corresponding box. 62

5.5 The Deep Cheezit (left), Mug (Middle), Live Cheezit and Live Pitcher (Right) scenes. The occupied (black) and known free (not shown) voxels from vision with contact (transparent red) and robot free (not shown) voxels are all used by CLASP to generate completed shapes (green). 62
6.1 Leaving the Graph of the Blindfolded Traveler's Problem. Left: The initial BTP graph. Motion is limited to the graph edges. Middle: After collision a node could be created at the current configuration (red circle) with new edges (blue dashed lines) to traverse. Right: The agent could then closely follow the surface of obstacles. 70
6.2 A 2D search problem solved using Selective Densification with evaluated edges shown in black (valid) and red (invalid) and the final path shown in blue. Left: 2D view with red obstacles. Right: View of Layered Graph with unevaluated edges shown in light grey 71
6.3 Architecture for BTP with goals dependent on the belief over states 72
A.1 Victor shaking my hand. Victor was the main robot used in experiments. 77
A.2 Reduction from Optimal Decision Tree problem 80
A.3 Plots of likelihoods of CLASP and baselines under the particle filter belief and kernel function. 90
A.4 Likelihood of CLASP and baselines for the multiobject scene 91

LIST OF TABLES

Table

2.1 Simulated Scenarios: Successes within 15 min, total time (s), and number of planner calls averaged over 20 trials for each entry. Blue: Proposed methods. 19

2.2 Physical Robot Experiments: Successes within 5 min., total time (s), number of planner calls, planning time (s), and controller time (s) averaged over 10 trials. 19

4.1 Best sample accuracy, Coverage of the plausible set, Average sample plausibility and Plausible diversity in mm. PSSNet performs best relatively in “Shapenet: occluded handle” and “YCB: narrow slit”, as in these datasets there is ambiguity in the full shape given the partial view. 48

ABSTRACT

While reaching for your morning coffee you may accidentally bump into the table, yet you reroute your motion with ease and grab your cup. An effective autonomous robot will need to have a similarly seamless recovery from unexpected contact. As simple as this may seem, for decades manufacturing robots were not able to sense contact quickly and precisely enough to stop during a collision, so robots in factory environments lived in tightly controlled and expensively precise work zones. Recent collaborative robots can now stop after collision so successfully they have been deemed safe to work around people. However unexpected contact is still treated as an error that an operator is expected to resolved. Robots operating in our less-structured daily environments will need to reason about the information they have gained from contact and replan autonomously.

This thesis examines planning under uncertainty with contact sensitive robot arms. First addressed is the specific information gained from sensing contact. Most robots do not have skin and cannot precisely sense the location of contact. This leads to the proposed *Collision Hypothesis Set* model for representing a belief over the possible occupancy of the world sensed through contact. To capture the specifics of planning in an heavily occluded environment with this measurement model, we develop a POMDP approach called the *Blindfolded Traveler's Problem* and propose several strategies for practical approximate solutions. Finally, we examine belief representations for the occupancy of the world to more closely approximate a rich prior over possible objects. We propose a neural network for shape completion that combines both visual and contact information.

CHAPTER I

Introduction

Traditional robot motion planning algorithms were designed to prevent contact with the environment. With the introduction of collaborative robot arms such as the UR series, Kuka iiwa, and Franka, unintended collisions are no longer the detrimental event that breaks the robot or harms a human. These robots are able to sense external forces on the arms and come to a safe stop before damage occurs. However, in a typical industrial workflow an unintended contact will result in an emergency-stop behavior, and a human operator must remedy the discrepancy between the robot's understanding and real world obstacles. In academic labs the situation is similar. If a robot collides with a table because the perception system is not callibrated, we fix the perception system instead of using the information learned by the contact.

This dissertation is motivated by the research question "How can robots use sensed unexpected contact information to understand their environment and achieve tasks with an imperfect perception system?". By answering this question I hope to move towards enabling robots to operate in cluttered scenarios with imperfect sensing while not requiring frequent human corrections.

A challenge in all of my approaches will be the very limited information a contact measurement provides. Detecting no contact along the arms indicates that all points inside and on the surface of the robot are not part of the environment. By contract, detecting a contact indicates that *some* point on the robot surface is in contact, but there will be uncertainty as to which point caused this measurement. Consider the large hypothesis space of all possible ways of assigning occupancy, and the smaller version space of all hypothesis shapes consistent will all observations. Probing the occupancy of a single point divides the hypothesis space in half, providing one bit of information. A contact measurement is less informative, providing less than one bit of information. From this analysis using contact information may first appear hopeless.

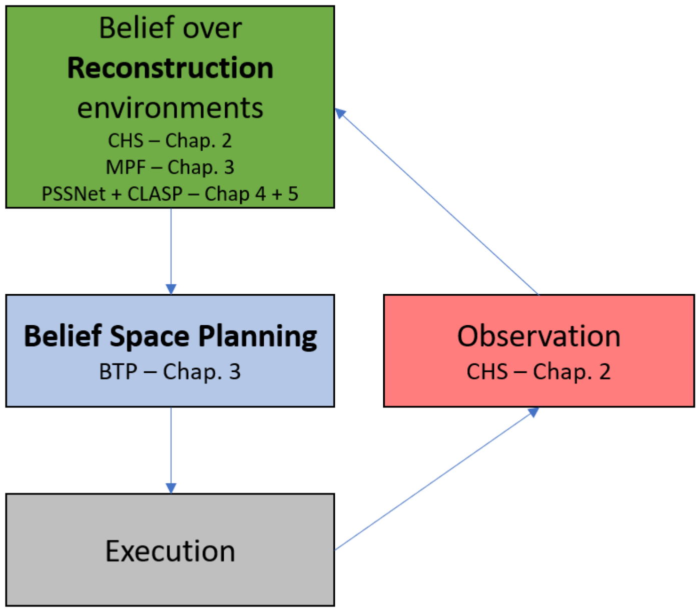


Figure 1.1: Overview of method in this Thesis

Fortunately there are two features we leverage. First, our primary objective is to achieve some task, such as reaching some final goal position, thus the robot does not need to determine the full occupancy of the world. Second, environments are highly structured, thus by knowing the occupancy of a few points the occupancy of much of the scene can be inferred. The space of reasonable shapes is far smaller and more orderly than the space of all possible occupancies. I explore both of these topics separately as well as combined them into a single planning framework.

1.1 Problem Characterization

This thesis examines two challenges of planning with contact information. The first problem of *Reconstruction* aims to estimate the geometry of the world using the sensed information. The second problem of *Belief Space Planning* instructs the robot how to move given the objectives and beliefs over reconstructed worlds. Figure 1.1 shows this interaction.

1.1.1 Reconstruction

Reconstruction is the task of estimating the geometry of the world from observations. In this work, the "world" refers to a binary voxelgrid of occupied and free space. Contact sensing and RGBD (depth camera) observations are the two forms of observation, with some chapters focusing on a single sensing modality.

Reconstruction Metrics: When considering reconstruction as an isolated task, approaches are evaluated on the accuracy of the reconstruction using either the Intersection over Union (IoU) or Chamfer Distance (*Barrow et al.*, 1977) metrics. Reconstruction accuracy is, however, a proxy for the true desired metric of reconstruction that improves the overall robot performance in reaching a goal. Informally, the desired metric would measure the reconstruction of only the important regions. This however cannot be measured without the full planning context.

Multi-hypothesis Reconstruction: We consider approaches capable of generating multiple distinct reconstructions. While single hypothesis approaches can score well on the reconstruction metrics, they do not integrate well into belief-planning frameworks. Many of our multi-hypothesis approaches are derived from Bayesian origins, producing samples conditioned on priors and observations. However the complexity of the observations requires approximations which break the formal Bayesian reasoning. Thus our belief approaches are often "Bayesian Inspired", rather than strictly Bayesian.

1.1.2 Belief Space Planning

The robot makes use of the reconstructions described using Belief Space Planning e.g. (*Somani et al.*, 2013). Since the true world is not known with certainty, the robot must evaluate potential actions for a belief over possible worlds to determine how to reach the goal.

Robot Motion Planning: This thesis considers the task of robot motion planning as traversing a sequence of collision-free joint configurations to reach a target configuration. Other forms of motion planning, such as force (impedance) control, feedback loops, and robot dynamics were implemented to make our robot function, but are not the focus of the algorithms described. We consider motion planning through the continuous free space which is done both using random sampling, and by constructing a roadmap graph.

Exploration vs. Exploitation: This thesis continues a theme common in many works on planning with uncertainty addressing two valuable components of actions. Exploration chooses actions that gain information yet do not make direct progress towards a goal. Exploitation leverages the information learned so far but may forgo informative actions in favor of progress towards the goal under the current estimate(s) of the world.

Static World: Throughout this work the world is assumed to be static. Objects must not move, even when bumped, for the guarantees and methods to apply exactly.

Of course, real robot collision will likely cause objects to move slightly, and these slight motions are often not problematic in practice. For extending this work to movable objects, see Chapter VI.

1.2 Summary of Contributions

This thesis makes the following contributions:

- I introduce the Collision Hypothesis Set observation model to capture the uncertainty of a contact measurement. I introduce a method of interleaving planning and control using the observed Collision Hypothesis Sets
- I introduce the Blindfolded Traveler's Problem to cast the motion planning under uncertainty with contact measurements into a graph search problem. I explore the effectiveness of many heuristic methods adapted to BTP.
- I create the Plausible Shape Sampling Network (PSSNet) for generating multiple diverse yet plausible completions of 3D objects from depth images.
- I create the Constrained Latent Shape Projection (CLASP) algorithm to predict plausible shape completions consistent with both the depth image and contact information from the robot. This can be integrated into the Blindfolded Traveler's Problem as the belief occupancy model.

CHAPTER II

Planning with Contact Sensing Uncertainty

This first chapter addresses the fundamental questions of contact sensing "What information is gained from a contact measurement?" and "How can we construct planning algorithms using a belief learned from contact measurements?". This chapter introduces the Collision Hypothesis Set belief distribution and proposes planning algorithms to minimize the probability of collision over this belief.

2.1 Introduction

Robots rely on sensors to construct models of the world for use in motion planning, but in many practical scenarios sensing limitations result in an incomplete or inaccurate model, resulting in plans that can collide with unobserved obstacles. Sensing limitations occur in manipulation tasks due to limited range and field of view, invalid measurements caused by glare, and insufficient accuracy for motion in tight areas. Furthermore, robots may reach into occluded areas during maintenance and assembly tasks (e.g. reaching into a car engine) and household tasks (e.g. reaching deep into a cabinet or behind a box). In the scenarios examined in this paper collisions between the arm and environment can be sensed without damage to the robot and used to inform future plans, but a lack of tactile sensing generates large non-Gaussian uncertainty over the belief of the occupancy of the environment.

The task we consider is to move from a given start configuration through free space to a goal configuration as quickly as possible, however the free space is not known *a priori* and the occupancy must be sensed through contact. We do not require tactile sensing to detect contact since most robots do not possess touch-sensitive skin. Even robots with such skin may pick up objects, effectively extending their kinematic chain with unsensorized geometry. We assume a robot is able to detect contact using joint torque feedback as is available on many robot arms. We further use that joint torque to determine which links may be in contact.

The task of moving to a goal in an unknown environment can be framed as a Partially Observable Markov Decision Process (POMDP), where the belief over occupancy is obtained through noisy collision measurements. In our implementation the workspace, observed collisions, and known obstacles are all stored in voxel grids of size N, thus the size of a belief state is 2^N. A measurement either indicates a configuration along a path did not collide and thus the voxels occupied by the robot do not contain obstacles, or that a collision occurred and thus at least one voxel blocked movement to the new configuration. Measurement uncertainty does not primarily come from sensor noise, but because a measurement only provides a set of voxels where at least one is occupied. The size of the belief space and the measurement uncertainty make this problem intractable for a standard POMDP solver, thus we propose an approach which is specialized to our domain.

The key contribution of this chapter is a representation for contact uncertainty that we call *Collision Hypothesis Sets (CHS)*, which enables planners to more accurately reason about potential collisions. This chapter first reviews prior work on planning in uncertain environments (Sec. 2.2). The task is then defined (Sec. 4.3), the CHS representation is motivated and presented (Sec. 2.4), and the planning and control architecture is then described (Sec. 2.5). Results are reported for experiments performed in simulation and on a physical robot, comparing the CHS representation to a baseline unified cost grid (Sec. 2.6). With enough open free space the two methods perform similarly. However, when the robot must enter narrow passages to reach a goal the CHS representation reduced the total execution time by approximately a factor of 1.5 to 3. Furthermore, in narrow passage scenarios where the baseline method only reached the goal in approximately 40-60% of trials, using CHSs produced a success rate of 100%.

2.2 Related Work

A core component of motion planning is the ability to check the validity of a path. Many motion planning algorithms assume deterministic collision checking but with uncertainty in the robot or environment it is only possible to calculate the probability of collision. With an arbitrary belief Monte Carlo simulations (MCS) can be used to estimate collision probabilities although this approach is computationally expensive and generally not practical for large state spaces, so many approaches apply only to specific belief distributions.

Assuming Gaussian uncertainty over the robot and specific objects enables computation of collision probability for a single configuration (*Blackmore*, 2006; *van den Berg et al.*, 2012; *Toit and Burdick*, 2011). To estimate the collision probability along a full path, rather than at an individual configuration, (*Patil et al.*, 2012) propagates only the portion of a Gaussian robot belief that are not in collision. Other forms of uncertainty, such as point cloud measurements from range sensors, may be better modeled with specifically-tailored distributions dependent on distance and incidence angle (*Bae et al.*, 2009). However, none of these distributions accurately model the information obtained when a robot contacts the environment.

Localization methods exist that are specifically tailored to model the unusual contact sensing uncertainty. Using a particle filter, a belief consistent with a contact measurement can be updated using rejection sampling (*Saund et al.*, 2017), or sampling from the contact manifold (*Klingensmith et al.*, 2016b). Both of these methods require models of objects in the world, which we do not assume are known. Other methods use joint position and torques to estimate the contact location on the robot surface (*Koonjul et al.*, 2011; *Bicchi et al.*, 1993), but require accurate torque measurements to produce accurate estimates. Since we wish to sense contact just above the noise threshold the torque measurements will contain significant noise, thus we use a comparatively simple method that more reliably estimates which links may be in contact.

By treating the probability of collision as a cost, the problem of planning under uncertainty can be framed as an optimal path planning problem. While there exist numerous optimal planners through continuous space, such as RRT* (*Karaman and Frazzoli*, 2011) and its many variants, these methods rely on a quick computation of path cost to run efficiently and since computing collision probability is far more expensive than a typical path length cost these methods ultimately explore too few nodes for our problem in a reasonable time.

Rather than attempting to minimize the probability of collision, many algorithms search for a path with some acceptably-low probability of collision. This can be done conservatively by inflating the robot (*Lee et al.*, 2013), iteratively considering simplified dynamics (*Bry and Roy*, 2011), or in a Probabilistic Roadmap where each edge collision cost is bounded (*Guibas et al.*, 2008). These methods approximate the full path probability of collision from the probability of collision of the individual states, thereby assuming that probabilities of collision are independent. In our work the occupancy uncertainty is heavily coupled across space, thus this independence assumption does not hold (see Sec. 2.5 for further discussion).

While executing a plan a robot may learn new information that causes the plan to become invalid or suboptimal, and the robot can replan a new path given this new information. Most prior work on replanning assumes collisions can be sensed at a distance and a primary goal is to avoid collisions (*Janson et al.*, 2018a). Some methods do allow for collisions and store contact locations sensed using tactile skin on a robotic arm (*Bhattacharjee et al.*, 2014; *Killpack et al.*, 2016). Using a simulated skin a fast planning architecture was demonstrated with a fast local controller that falls back to a slower sampling-based planner when stuck in a local minimum (*Park et al.*, 2014). We use a similar approach for mixing planning and control, but our method does not require tactile skin to sense precise contact points.

2.3 Problem Statement

Given a robot with configuration space $\mathcal{C}$, define a workspace voxel grid of size $N \times N \times N$ as W with static workspace occupied voxels $W_{\mathcal{O}} \subseteq W$ and free space $W_{\mathcal{F}} = W \setminus W_{\mathcal{O}}$. For any configuration $q \in \mathcal{C}$ the robot occupies a subspace of workspace, defined by the mapping $\mathcal{R}(q) : \mathcal{C} \to \mathbb{P}(W)$ where $\mathbb{P}$ denotes the powerset. A collision occurs when $\mathcal{R}(q) \cap W_{\mathcal{O}} \neq \emptyset$, and is detected by joint torque feedback (discussed in Sec. 2.4). While the planner does not have access to $W_{\mathcal{O}}$ directly, the swept volume of the previously visited configurations $q^i_{visited}$ is known to be free and is stored in $W_{SV} = \cup_i \mathcal{R}(q^i_{visited}) \subseteq W_{\mathcal{F}}$.

A discretized path ξ consists of configurations q_i such that $||q_{i+1} - q_i|| < \delta$ where δ is set based on the desired resolution. A robot at q_i attempting to follow the ith path ξ_i^{plan} takes execution time $T_{exec}(\xi_i^{plan}) > 0$ to arrive at

$$M(q_i, \xi_i^{plan}) = q_{reached} \tag{2.1}$$

If ξ_i^{plan} collides then $q_{reached} \in \xi_i^{plan}$ is the configuration directly before the first $q \in \xi_i^{plan}$ in collision. After a collision a new ξ_{i+1}^{plan} may be executed.

Define an algorithm $\mathcal{P}$ that takes time t_i^{plan} to produce ξ_i^{plan}, a path that must begin at q_i. $\mathcal{P}$ is aware of past planned paths and visited configurations $Q_i^{visited} = \{q_0, q_1, ..., q_i\}$. We refer to $\mathcal{P}$ as a "planner" if it plans ξ^{plan} reaching the goal, and a "controller" if it computes a short ξ^{plan} towards the goal. A single $\mathcal{P}$ may choose to act as either a planner or controller depending on context.

$$\mathcal{P}(Q_i^{visited}, \xi_1^{plan}, \ldots, \xi_{i-1}^{plan}) = (\xi_i^{plan}, t_i^{plan}) \tag{2.2}$$

Given a robot in configuration q_{init} and a set of goal configurations Q_{goal} our objective is to choose a $\mathcal{P}$ to reach a goal configuration in the least time.

$$\begin{aligned}
\underset{\mathcal{P},n}{\text{minimize}} \quad & \sum_{i=0}^{n-1} t_i^{plan} + T_{exec}(\xi_i^{plan}) & (2.3)\\
\text{subject to:} \quad & q_0 = q_{init}, q_n \in Q_{goal} & (2.4)\\
& (\xi_i^{plan}, t_i^{plan}) = \mathcal{P}(Q_i^{visited}, \xi_1^{plan}, \ldots, \xi_{i-1}^{plan}) & (2.5)\\
& q_{i+1} = M(q_i, \xi_i^{plan}) & (2.6)
\end{aligned}$$

Since planning time is part of the objective, $\mathcal{P}$ must choose a tradeoff between analyzing all information to choose the best ξ_i^{plan}, and minimizing planning time t_i^{plan}. A controller typically achieves a small t_i^{plan}, while a planner spends more time to produce better ξ_i^{plan}. As ξ^{plan} are executed, $\mathcal{P}$ has access to more knowledge about W, and effectively using this information is key to minimizing total time. We will not solve Eq. 2.3 computationally, but instead design a $\mathcal{P}$ with desirable qualities and justify our choices with experimental trials.

2.4 Representing Uncertain Contact Information

When a collision is detected during execution the swept volume of a set of configurations $q^i_{collision}$ along the robot path within some small distance $d_{\mathcal{K}}$ after collision is assumed to contain the point of contact. While in theory the set of points on the robot surface contains the contact point, in practice joint measurement uncertainties, robot geometry uncertainties and approximations, and material compliance require a larger set to guarantee encapsulation of the contact point. A set containing the contact point is constructed $\mathcal{K}_i = \cup_i \mathcal{R}(q^i_{collision}) \setminus W_{SV}$, the total volume of the robot in the possible collision configurations with the known free space removed (Fig. 2.1).

In our problem collisions are detected using measured joint torque $\tau_{meas} \in \mathbb{R}^J$, where J is the number of robot joints. Using a mass model of the robot the expected joint torque due to gravity and dynamics τ^{exp} is calculated and used to estimate the external joint torque $\tau^{ext} = \tau^{meas} - \tau^{exp}$. A noise threshold τ^{th} is set for each joint and τ^{ext} triggers a collision detection whenever any joint exceeds its threshold.

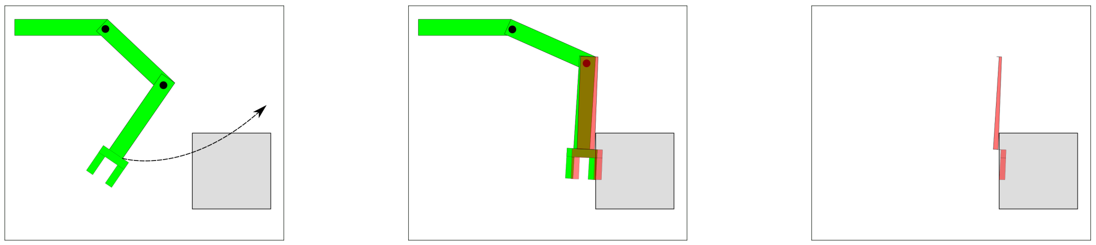


Figure 2.1: A plan for the green robot (left) results in a collision with the grey unknown obstacle. A red collision hypothesis set is added using links possibly in collision based on measured joint torques (center). The known free space is removed (right).

Joint i exceeding τ_i^{th} implies an external (contact) force on a link after joint i on the kinematic chain. A set of links that must contain a contact $\mathcal{L}_{contact}$ is constructed by first finding the highest i where $\tau_i^{ext} > \tau_i^{th}$, then adding all links downstream from joint i to $\mathcal{L}_{contact}$. Only the links in $\mathcal{L}_{contact}$ are used to create $\mathcal{K}_i$.

2.4.1 Baseline: Unified Cost Grid

When provided with noisy measurements a common approach to modelling uncertainty extends a binary occupancy map to a Unified Cost Grid (UCG), a voxel grid where each voxel stores its likelihood of occupancy. We compute this likelihood as the count of how many observed collisions could be explained if that voxel were occupied. The UCG representation computes a path cost by summing the likelihood of the voxels in the swept volume of the path.

Unfortunately, by combining all measurements into a single grid the UCG representation loses the information that each collision was caused by at least one occupied voxel. As we show in Section 2.6, this approach fails when entering narrow passages as collisions near the entrance create a high cost for feasible paths. Therefore, we construct a representation that maintains this information.

2.4.2 Collision Hypothesis Sets

Define a *Collision Hypothesis Set* (CHS) as a set of points in the robot workspace containing at least one point in collision. The $\mathcal{K}_i$ constructed after a collision are CHSs, since $\mathcal{K}_i \cap W_{\mathcal{O}} \neq \emptyset$. However, unlike in UCG, the CHS representation never combines $\mathcal{K}_i$s into a unified grid, and instead maintains a set $\mathcal{K}$ of all generated $\mathcal{K}_i$.

Planning will require evaluating the probability of collision for a path using $\mathcal{K}$. Let the swept volume of a path ξ on the robot be $W_\xi \subseteq W$. We define the probability of collision of ξ with a single $\mathcal{K}_i$ as

$$p_{collision}(W_\xi, \mathcal{K}_i) = \frac{|W_\xi \cap \mathcal{K}_i|}{|\mathcal{K}_i|} \tag{2.7}$$

Assuming there exists a single occupied voxel uniform randomly selected from $\mathcal{K}_i$, this is precisely the probability the path collides. While this will likely be an underestimate of the true probability, it encourages exploration and further collision measurements will help localize the contact. The probability of collision for a full path is computed assuming independence between $\mathcal{K}_i$s, thus

$$p_{collision}(W_\xi, \mathcal{K}) = 1 - \prod_i (1 - p_{collision}(W_\xi, \mathcal{K}_i)) \tag{2.8}$$

This definition for collision probability captures several key features that the UCG representation lacks. A $\mathcal{K}_i$ with fewer voxels represents a more precise knowledge of where the contact occurred and thus a more precise estimate of workspace occupancy. In addition, a path that moves the robot through an entire CHS is guaranteed to collide, since $\frac{|W_\xi \cap \mathcal{K}_i|}{|\mathcal{K}_i|} = 1$, representing the information that a CHS contains at least one point in collision. A path ξ that is attempted but blocked due to a detected collision creates a $\mathcal{K}_i$ that lies entirely within W_ξ, so the updated collision probability for ξ will now be 1.

2.5 Interleaving Planning and Control

To achieve our objective of reaching a goal configuration in minimal time (Eq. 2.3), we must choose an algorithm $\mathcal{P}$ that compute good motions ξ_i^{plan} in low planning times t_i^{plan}. Local controllers quickly compute locally good ξ_i^{plan} but may get stuck in local minima. Global planning can escape these minima by planning a full path, but requires significant computation time. Our environments include many undetected obstacles and a local controller may produce many collisions before getting stuck, thereby providing more information to the global planner without adding much total time. Thus we use a planner initially and when stuck in cul-de-sacs, and local control otherwise. Our full architecture is presented in Algorithm 1.

Planning: The objective of the planner is to find a path to the goal with the minimal probability of collision. Unfortunately, as discussed in Section 2.2, previous methods that plan over obstacle uncertainty are not applicable when using CHSs, as these planners typically rely on a path cost definition that is purely the sum(*Karaman and Frazzoli*, 2011) or maximum(*Toit and Burdick*, 2011) of costs of states/edges. Figure 2.2 illustrates two problems with inferring path collision probability using the

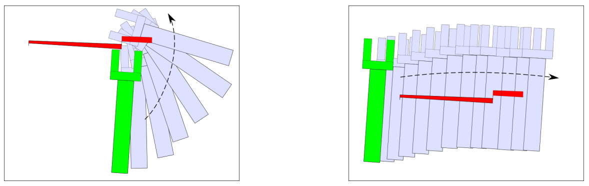

(a) (b)

Figure 2.2: Plans for the green arm sweep through the blue region. Red: a CHS

costs of only individual states along the path. In Fig. 2.2a the swept volume of adjacent states overlap significantly, thus summing costs of states could significantly overestimate the probability of collision. In Fig. 2.2b multiple states collectively intersect the entirety of a CHS thus guaranteeing a collision, but each state individually only intersects a fraction of that CHS, thus taking the maximum over all state costs would significantly underestimate the probability of collision.

Algorithm 1: MainLoop(q_{cur}, q_{goal})

```
1 $\mathcal{K} \leftarrow \emptyset$; $W_{SV} \leftarrow \emptyset$ **while** $q_{cur} \neq q_{goal}$ **do**
2     $\xi \leftarrow$ Planner($q_{cur}, q_{goal}, \mathcal{K}$) $q_{cur} \leftarrow$AttemptPath($\xi, \mathcal{K}, W_{SV}$) **while**
      $q_{cur} \neq q_{goal}$ ***and*** ($\xi \leftarrow$ *Controller*($q_{cur}, q_{goal}, \mathcal{K}$)) $\neq \emptyset$ **do**
3         $q_{cur} \leftarrow$ AttemptPath($\xi, \mathcal{K}, W_{SV}$)
```

Algorithm 2: AttemptPath($\xi, \mathcal{K}, W_{SV}$)

```
1 **for** $q_i$ *in* $\xi$ **do**
2     **if** $q_i$ *causes collision* **then**
3         $\mathcal{K}$.addNew($q_i, \xi, d_{\mathcal{K}}$) **break** $q_{cur} \leftarrow q_i$ $W_{SV} \leftarrow W_{SV} \cup \mathcal{R}(q_{cur})$
4 $\mathcal{K}$.subtract($W_{SV}$) **return** $q_{cur}$
```

To plan a path we create PathBiRRT (Alg. 3), a planner based on bi-directional RRT (*Kuffner and LaValle*, 2000) that ensures the cost of the path generated is below a specified threshold p_{thr}. For the CHS representation the `Cost` function is given by Eq. 2.8. For the baseline UCG representation, which we compare to in the results, the `Cost` sums the cost of all voxels in W_ξ, i.e. $\sum_i |W_\xi \cap \mathcal{K}_i|$.

Ideally we desire a `Connect` function that ensures the cost of the entire path to any new node is below p_{thr}. In practice, computing this cost for every explored node is prohibitively expensive, thus we approximate this cost as an accumulation of branch costs computed in the `Connect` function (Alg. 4). When extending towards q_{target} the full path cost of the branch from q_{near} is calculated (Line 5). The cost from the root to the q_{new} is approximated by accumulating the approximate cost to q_{near}

(computed previously) and the cost of the new branch (Line 7). When using CHS the `Accumulate` function is the combination of independent probabilities: $1-(1-p_1)(1-p_2)$. UCG accumulates by adding the costs: c_1+c_2. By computing the cost over full branches this approximation is significantly better than accumulating cost purely based on states, however, this approximation may still over or underestimate the true cost, as separate branches along a path may overlap in W.

PathBiRRT (Alg. 3) repeatedly calls `Connect` to build a tree from the start and a tree from the goal, generating a potential path ξ when the two trees meet. ξ may exceed p_{thr} due to the approximation error within `Connect` and because ξ is the combination of paths from two trees. The cost of ξ is checked (Line 11) and if it exceeds p_{thr} then the highest cost edge from ξ is pruned along with all child edges, and planning continues.

For planning within fixed time t_{plan}, we provide two methods for setting p_{thr}. The anytime APathBiRRT (Alg. 5) begins with $p_{thr}=\infty$ and continues searching for lower-cost paths until time runs out. In contrast, IPathBiRRT (Alg. 6) begins with an optimistic $p_{thr}=c_{init}$. In each iteration IPathBiRRT allocates a fraction ψ_f of the remaining time t_ψ for planning with the current p_{thr}. If a plan is not found within t_ψ, p_{thr} is increased for the next iteration, approaching p_{max} for CHS, or v_{max} for UCG. Once a path is found, IPathBiRRT then invokes APathBiRRT for the remainder of the planning time.

As a benchmark, we also implemented ABiRRT, which iteratively decreases a cost threshold (as in Alg. 5), but when checking to add a new node (as in Alg. 4, Line 4-8) only the configuration cost is considered ($\texttt{Cost}(\mathcal{R}(q_{new}),\mathcal{K})<p_{thr}$) and there is no full path check (Alg. 3, Line 11). To support the claim that asymptotically optimal planners are not practical for this problem we also compare against RRT* (*Karaman and Frazzoli*, 2011), which always computes the full path cost when considering new connections.

Local Control: The local controller samples a specified number n_c of straight-line motions of length d_c uniformly from the half-sphere in $\mathcal{C}$-space that reduce the robot's distance to the goal. From these samples, the controller greedily selects the motion with lowest probability of collision, using Eq. 2.8. If no motion is found with probability of collision $< p_c$ for CHS, or cost $<\mathrm{nvox}_c$ for UCG, the controller assumes it is stuck (Alg. 1 Line. 2) in a cul-de-sac and invokes the planner.

Algorithm 3: PathBiRRT($q_{init}, q_{goal}, \mathcal{K}, p_{thr}, t$)

```
1  $\mathcal{T}_A$.init($q_{init}$)
2  $\mathcal{T}_A[q_{init}]$.approxCost $\leftarrow 0$
3  $\mathcal{T}_B$.init($q_{goal}$)
4  $\mathcal{T}_B[q_{goal}]$.approxCost $\leftarrow 0$
5  while $timeElapsed() < t$ do
6      $q_r \leftarrow$ sampleConfig()
7      status, $q_{new}$ = Connect($\mathcal{T}_A, q_r, p_{thr}$)
8      if $status \neq Trapped$ then
9          if $Connect(\mathcal{T}_B, q_{new}, p_{thr}) = Reached$ then
10             $\xi \leftarrow$ path($\mathcal{T}_A, \mathcal{T}_B$)
11             if $Cost(W_\xi, \mathcal{K}) < p_{thr}$ then
12                 return $\xi$
13             else
14                 e $\leftarrow$ highestCostEdge($\xi$)
15                 if $e$ in $\mathcal{T}_A$ then
16                     $\mathcal{T}_A$.prune(e)
17                 else
18                     $\mathcal{T}_B$.prune(e)
19     swap($\mathcal{T}_A, \mathcal{T}_B$)
20 return $\emptyset$
```

Algorithm 4: Connect($\mathcal{T}, q_{target}, \mathcal{K}, p_{thr}$)

```
1  $q_{near} \leftarrow$ nearest($\mathcal{T}, q_{target}$) $W_{seg} \leftarrow \{\}$
2  $\xi \leftarrow$interpolate($q_{near}, q_{target}, \delta$)
3  for $q_{new}$ in $\xi$ do
4      $W_{seg} \leftarrow W_{seg} \cup \mathcal{R}(q_{new})$
5      $c_{seg} \leftarrow$ Cost($W_{seg}, \mathcal{K}$)
6      $c_{near} \leftarrow \mathcal{T}[q_{near}]$.approxCost
7      $c_{approx} \leftarrow$ Accum($c_{seg}, c_{near}$)
8      if $collides(q_{new})$ or $1.5emc_{approx} \geq p_{thr}$ then
9          if $q_{new} = q_{near}$ then
10             return {Trapped, $q_{new}$}
11         return {Advanced, $q_{new}$}
12     $\mathcal{T}$.add($q_{new}$)
13     $\mathcal{T}[q_{new}]$.approxCost $\leftarrow c_{approx}$
14 return {Reached, $q_{new}$}
```

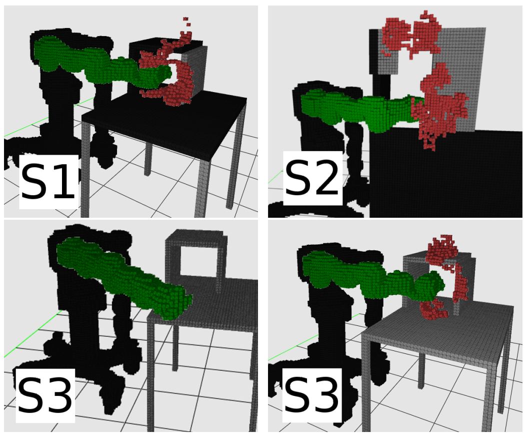


Figure 2.3: Simulated Scenarios

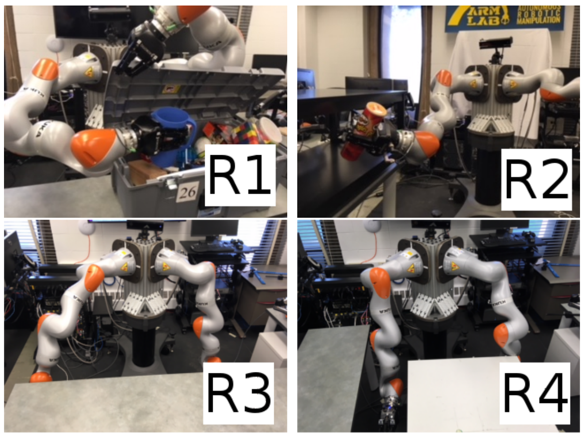


Figure 2.4: Physical Robot environments

Algorithm 5: APathBiRRT $(q_{init}, q_{goal}, \mathcal{K}, p_{thr} = \infty)$

```
1 $\xi_{best} \leftarrow \emptyset$
2 while $timeRemaining() > 0$ do
3     $\xi \leftarrow$ PathBiRRT($q_{init}, q_{goal}, p_{thr}$ timeRemaining())
4     if $\xi \neq \emptyset$ then
5         if $Cost(W_\xi, \mathcal{K}) = 0$ then
6             return $\xi$
7         $p_{thr} \leftarrow$ Cost($W_\xi, \mathcal{K}$) - $\epsilon$
8         $\xi_{best} \leftarrow \xi$
9 return $\xi_{best}$
```

Algorithm 6: IPathBiRRT $(q_{init}, q_{goal}, \mathcal{K}, c_{init}, c_{max}, \psi_f)$

```
1 $p_{thr} \leftarrow c_{init}$
2 while $timeRemaining() >$ 0 do
3     $t_\psi \leftarrow$ timeRemaining() $\cdot \psi_f$
4     $\xi \leftarrow$ PathBiRRT($q_s, q_{goal}, p_{thr}, t_\psi$)
5     if $\xi \neq \emptyset$ then
6         return APathBiRRT($q_s, q_{goal}, p_{thr}$ =Cost($W_\xi, \mathcal{K}$))
7     $\alpha \leftarrow$ timeElapsed()/totalTime()
8     $p_{thr} \leftarrow \alpha \cdot c_{max} + c_{init}$
9 return $\emptyset$
```

2.6 Experiments and Results

To demonstrate the advantages of our representation we compared Collision Hypothesis Sets (CHS) to the baseline Unified Cost Grid (UCG) in multiple environments in simulation and on a physical robot using multiple planning approaches. Parameters used in experiments are given in Fig. 2.5. Voxel grids were implemented on the GPU using GpuVoxels(*Hermann et al.*, 2014). Planners were implemented using OMPL(*Şucan et al.*, 2012) with modification. Code was run on a computer with i7-7700 processor and a NVidia 1080 Ti GPU. For all planners each path was smoothed using 100 iterations of shortcut smoothing.

Simulation Experiments simulated Victor's right arm (a Kuka iiwa) in the environments in Fig. 2.3, shown with red $\mathcal{K}_i$s, grey unobserved obstacles, and black observed obstacles. Rather than simulating joint torque, collisions were determined when the simulated robot moved into an obstacle in the workspace. All links downstream of the true link in collision were used to generate a CHS. In scenario S1, the simplest environment, the robot's goal was to reach inside a box located on a table with some occupancy known from a simulated depth sensor. Scenario S2 was harder as the robot needed to move the entire arm through a narrow slot occluded from the sensor. Scenario S3 was identical to S1 except the robot used no depth sensor information. Each simulation trial allowed 15 minutes for the robot to reach the goal.

Physical Robot Experiments were conducted on Victor's right arm (a Kuka iiwa capable of sensing joint torque). A Kinect depth sensor created known obstacle occupancy. Figure 2.4 shows the physical robot experimental setups. Each physical trial allowed 5 minutes for the robot to reach the goal. In physical robot scenario R1 the robot placed a pitcher inside a box with the lid occluding the side and back walls from the Kinect. R2 involved placing a cylindrical can on a short shelf. Glare and occlusions resulted in a sparse and noisy occupancy map from the Kinect, both blocking a feasible path to the goal and missing portions of the top and bottom of the shelf. To accommodate noise up 30 intersections were allowed between the robot and the Kinect occupancy map.

In scenario R3 the robot arm moved from below to above a table. The sensed table was artificially shifted 5 cm away from the robot, simulating localization or sensor error. R4 tested the behavior moving through a narrow passage between two tables. This gap was adjusted between 15.5cm to 28cm, corresponding to a clearance of 2.5cm to 15cm for the 13cm wide robot hand. In R4 Kinect data was not used, thus the robot only sensed obstacles through contact.

Results: Table 2.1 reports the results for each simulated scenario, comparing CHS and UCG using the proposed APathBiRRT and IPathBiRRT planners as well as RRT* and ABiRRT. RRT* did not reach the goal within the time limit in any

	Description	Value
p_c	controller max collision probability	0.9
nvox_c	controller max collision voxels	50
d_c	controller motion length	0.3 radians
n_c	controller number of samples	20
δ	path discretization size	0.14 radians
$d_{\mathcal{K}}$	dist. for $\mathcal{K}_i$ creation	0.05 radians
τ^{th}	torque threshold	[20, 20, 15, 5, 4, 3, 1] Nm
t_{plan}	allowed planning time per iteration	30s
N	voxel grid size	200x200x200
ψ_f	IPathBiRRT phase fraction	1/4
c_{init}	IPathBiRRT initial cost	0.3
p_{max}	IPathBiRRT CHS c_{max}	1
v_{max}	IPathBiRRT UCG c_{max}	400
ϵ	APathBiRRT improvement factor	0.0001

Figure 2.5: Experimental parameters

scenario. ABiRRT performed far worse than our proposed methods, indicating that considering full path collision probability is superior to only per-configuration collision probability. Table 2.2 reports the results for each physical scenario, comparing CHS and UCG using IPathBiRRT only, as this planner performed the best in simulation trials. In both simulation and physical trials we observed our CHS formulation outperforms the UCG approach in terms of computation time.

In simpler scenarios (S1, S2, R1) early collisions were navigated better by the local controller, causing fewer lengthy planning iterations. For example in R1 our method required invoking the global planner only once in all ten trials, leading to an average success time of 18.6s compared to 76.6s when using a unified cost grid. In harder scenarios (S3, R2) collisions occurred on multiple sides of narrow passages. Using a unified cost grid, the planner and controller avoided the center of the passage as this area has cost accumulated from multiple collisions, thus only 40% of the trials successfully reached the goal for R2. Paths through the center do not intersect with all voxels from any CHS, thus our approach correctly identified possible paths and succeeded in 100% of trials in R2. Even considering only trials where UCG succeeded our approach still reached the goal in approximately one third of the time on average.

In R3 the table obstacle created a cul-de-sac for the local controller, though there was significant free space for the planner to conservatively avoid the table. Since plans were able to avoid much of the CHSs, accurately modeling collision probability was

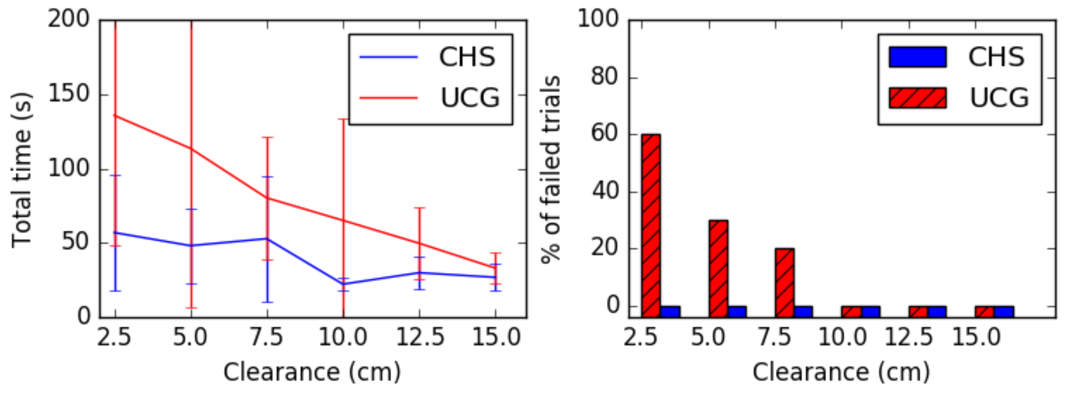


Figure 2.6: Total time and failure % for R4, averaged over 10 trials for each clearance with a 5 min. timeout

less important, thus the baseline method and proposed method performed similarly.

In R4 with a large gap between tables both methods found a path quickly. For both methods, successful trials primarily invoked the local controller and rarely needed the global planner. As the gap narrowed both methods required more time to find the opening, however UCG took longer on average and in 60% of trials did not find the opening within the allowed time of 5 minutes (Fig. 2.6).

2.7 Conclusions and Future Work

Most robots do not have touch-sensitive skin, and those that do may manipulate unsensorized objects. The proposed Collision Hypothesis Sets allow reasoning about the knowledge gained when these robots and objects come into contact with the environment. We showed how collision hypothesis sets can be used in controllers and planners to search for paths with minimum probability of collision. We performed simulated and physical robot experiments and found for simpler environments our approach takes less time to reach the goal while for more complex environments our methods succeeds where other approaches fail.

We explained why many existing methods for planning under uncertainty cannot be applied to contact observations and presented two planners that first approximate, then compute the full path cost. However, the planners implemented could be made more efficient and the resulting paths are often significantly suboptimal. Our future work seeks to improve these results.

	Planner	S1			S2			S3		
		Succ.	Time	P.Calls	Succ.	Time	P.Calls	Succ.	Time	P.Calls
CHS	APathBiRRT	100%	110	4.1	100%	100	3.4	100%	220	6.5
	IPathBiRRT	85%	69	3.0	100%	64	2.6	100%	180	5.3
	RRT*	0%	-	-	0%	-	-	0%	-	-
	ABiRRT	100%	78	3.5	30%	230	20	60%	200	15
UCG	APathBiRRT	100%	110	4.2	95%	120	3.9	40%	440	18
	IPathBiRRT	55%	71	2.9	90%	130	3.9	60%	280	15
	RRT*	0%	-	-	0%	-	-	0%	-	-
	ABiRRT	100%	113	4.4	80%	250	11	65%	500	22

Table 2.1: Simulated Scenarios: Successes within 15 min, total time (s), and number of planner calls averaged over 20 trials for each entry. Blue: Proposed methods.

	Planner	R1					R2				
		Succ.	Time	P.Calls	Plan	Ctrl	Succ.	Time	P.Calls	Plan	Ctrl
CHS	IPathBiRRT	100%	19	1.1	3.4	15	100%	62	2.1	38	24
UCG	IPathBiRRT	100%	77	2.3	57	19	40%	180	4.8	130	43

	Planner	R3					R4: clearance=2.5cm				
		Succ.	Time	P.Calls	Plan	Ctrl	Succ.	Time	P.Calls	Plan	Ctrl
CHS	IPathBiRRT	100%	100	1.0	32	65	100%	57	1.2	1.5	50
UCG	IPathBiRRT	100%	100	1.0	34	65	40%	140	2.5	50	82

Table 2.2: Physical Robot Experiments: Successes within 5 min., total time (s), number of planner calls, planning time (s), and controller time (s) averaged over 10 trials.

CHAPTER III

The Blindfolded Traveler's Problem

The previous chapter examines the uncertainty observed from a contact measurement and creates the Collision Hypothesis Set framework. The planning approaches of Chapter II modified the RRT algorithm and search in the continuous joint-space of the robot arm. Because the probability of collision along a path under the CHS model does not obey the Markov property, these searches were relatively slow. Here in Chapter III, we cast the planning problem into a Graph Search. In doing so we are able to introduce an algorithm that perform significantly faster and is more effective than those proposed in the previous chapter. This chapter further explores a more informed uncertainty model based on the Manifold Particle Filter, and proposes a method for combining the MPF and CHS beliefs.

3.1 Introduction

This chapter examines the same problem of robot motion planning in partially-known environments where obstacles are sensed only through contact. Again, this problem occurs quite frequently in manipulation tasks with sensing limitations such as a narrow field of view, occlusions in the environment, lack of ambient light, or insufficient sensor precision. For example, a robot may reach into dark confined areas during maintenance and assembly (e.g. inspecting the insides of aircraft (*Siegel et al.*, 1998)) or during everyday household tasks (e.g. reaching deep into a cabinet or behind a box (*Park et al.*, 2014)). Here, the goal is to minimize the total time it takes for the robot to move around obstacles sensed on-the-fly and reach a target configuration. As before, we consider both planning and execution time, though now examine each independently.

Consider the scenario where a robot arm is tasked with reaching into a box whose location is uncertain (Fig. 3.1). This could be framed as a POMDP, where the

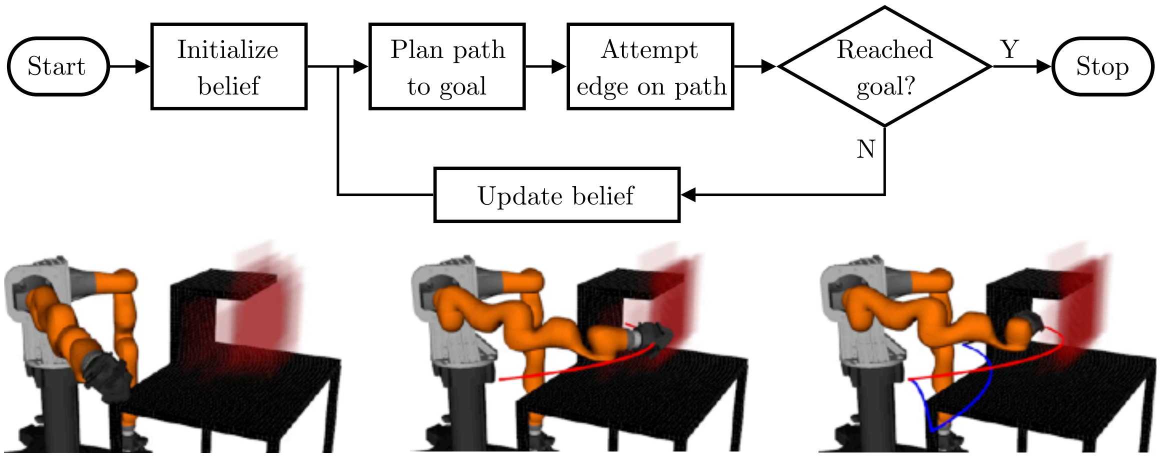


Figure 3.1: Overview of the BTP framework for planning with contact feedback. The robot is uncertain about location of the back wall. As it attempts to traverse edges, it partially localizes the wall and eventually finds its way to the goal.

belief over occupancy is obtained through noisy collision measurements. However the possible states of the POMDP include all possible arrangements of obstacles, and the action space includes all possible motions. The general POMDP is thus intractably large.

Instead, such planning problems may be solved by constructing a graph (*Kavraki et al.*, 1996), where vertices represent robot configurations and edges represent potentially valid movements of the robot between these configurations. Here, the validity of edges is unknown *a priori*. A natural strategy is *Optimism in the Face of Uncertainty* (OFU) (*Stentz*, 1997) — assume untraversed edges are valid, plan the shortest path and execute it. If the shortest path is indeed valid, the robot reaches the goal optimally. Otherwise, it removes the invalid edge from the graph and replans. OFU is effective in less-cluttered environments, where the robot finds a path to the goal after a few collisions. However, on problems with narrow passages such as Fig. 3.1, OFU can lead the robot down a "rabbit hole" trying paths that are not likely to be valid.

The key insight of this chapter is that *the validity of edges in the graph is correlated*. There are two main reasons for this correlation. First, edges overlap in swept workspace volume. Second, objects in the world occupy multiple workspace cells. Given a prior on edges, a robot can exploit such correlations to infer edge validities and reach the goal quickly (Fig. 3.1). We address the following research question:

> How should a robot navigate on a graph with unknown edge validites to minimize the expected traversal cost?

We refer to this broader problem as the *Blindfolded Traveler's Problem* (BTP). We show that this problem is NP-Complete and discuss a set of approximation-based policies. We also propose a new policy, Collision Measure, that is both efficient to compute and has theoretical guarantees.

We formulate robot arm planning with contact feedback as a BTP. We face an additional challenge for realistic scenarios – *the initial belief is approximate and can be misleading*. With a good initialization we show a particle filter that updates hypothesis worlds from contact observations suffices. Without a good initialization, we show an algorithm that starts with free-space and builds up a world model consistent with observations is effective. Since both scenarios occur in practice, we propose a Mixture of Experts framework for mixing these two belief update strategies.

In summary, this chapter makes the following contributions:

- Formulate the *Blindfolded Traveler's Problem*. (Section 3.3)
- Map the planning with contact feedback task to a BTP. Since the posterior is not specified, we propose a set of belief approximation strategies. (Section 3.4)
- Propose a set of approximation strategies to solve the BTP. (Section 3.5)
- Provide empirical evaluation of different strategies and belief approximations on simulated and real robot arm BTP instances. (Section 3.6)

We evaluate all strategies on Victor, our robot with two 7 DOF arms, in planning scenarios in simulation, each with three varying levels of difficulty (by adding error in prior). We also evaluate strategies with practical computation times on a live version of Victor. We find that the Collision Measure strategy using a Mixture of Experts belief tends to outperform all other baselines by planning consistently low cost paths with consistently low computation time. Furthermore, we find using the BTP framework significantly outperforms a baseline strategy used in planning with contact feedback.

3.2 Related Work

We examine planning under contact sensing uncertainty which leads to a number of challenges. While some approaches consider tactile skin (*Bhattacharjee et al.*, 2014), with only torque feedback contact observations cannot precisely localize collision points. One approach is to use non-parametric particle filters, however, they encounter problems with contact measurements (*Saund et al.*, 2017). The Manifold Particle Filter overcomes this by sampling from different proposal distributions depending if contact/no contact (*Klingensmith et al.*, 2016a), though this method requires an accurate prior over obstacles. Without a prior over obstacles we use the Collision Hypothesis Set belief, which we have previously employed in search using RRT (*Saund and Berenson*, 2018).

Our problem is closely related to that of real-time motion planning on roadmaps (*Kavraki et al.*, 1996). Roadmaps, which are graphs in configuration space, are efficient because they can be reused across planning iterations. In robot motion planning, edge evaluation dominates computational complexity (*Hauser*, 2015), therefore the key to minimizing search times is laziness (*Bohlin and Kavraki*, 2000; *Cohen et al.*, 2015). LazySP (*Dellin and Srinivasa*, 2016), shown to be optimally lazy (*Haghtalab et al.*, 2018), optimistically plans the shortest path and checks edges sequentially till an infeasible edge is encountered. Priors on edge validities can be further exploited to minimize edge evaluation (*Choudhury et al.*, 2016; *Mandalika et al.*, 2019; *Narayanan and Likhachev*, 2017). These problems can be further mapped to Bayesian active learning (*Tong and Koller*, 2001; *Golovin et al.*, 2010; *Chen et al.*, 2015) to compute policies that actively choose edges to evaluate to minimize uncertainty about which path is feasible (*Choudhury et al.*, 2018, 2017). An alternate formulation is online shortest path routing (*Awerbuch and Kleinberg*, 2004; *György et al.*, 2007; *Talebi et al.*, 2017) which is a particular instance of combinatorial bandits (*Cesa-Bianchi and Lugosi*, 2012). However, unlike our problem, these methods have full flexibility to telelport to and evaluate any edge.

Our work falls under the domain of planning under sensing uncertainty. D* (*Stentz*, 1997) and variants (*Koenig and Likhachev*, 2002; *Ferguson and Stentz*, 2007) typically replan optimistically and re-using the search graph. An alternative is to cast the problem in a Bayesian paradigm using an occupancy map (*Richter et al.*, 2018). However, such methods usually plan to short horizons. Since this problem arises from the mobile robot community, the focus is primarily robot safety (*Janson et al.*, 2018b). For our problem, the robot is able to collide safely and we seek to minimize the travel cost.

The BTP problem is closely related to the Canadian Traveler's Problem (CTP) (*Papadimitriou and Yannakakis*, 1991b) where neighboring edge costs are revealed when an agent visits a vertex. DAGs can be solved exactly via DP (*Nikolova and Karger*, 2008) but the general problem is PSPACE-complete (*Fried et al.*, 2013). Typically CTPs are solved using heuristics (*Eyerich et al.*, 2010) adopted from probabilistic planning (*Yoon et al.*, 2008) or using Monte-carlo Tree Search (*Gelly and Silver*, 2007; *Guez et al.*, 2012). CTP can also be cast in a Bayesian framework (*Lim et al.*, 2017) and solved near-optimally using informative path planning techniques (*Lim et al.*, 2016, 2015). While we evaluate some of these strategies for our robot arm planning, others are prohibitively expensive due to expensive collision checking and posterior update. We therefore adapt the Collision Measure (*Choudhury et al.*, 2016) as a computationally efficient strategy for the CTP/BTP.

3.3 Problem Statement

We propose the Blindfolded Traveler's Problem as a graph search problem to model the contact feedback planning problem. In a BTP the traveler traverses a graph attempting to reach a goal. While traversing an edge the traveler may encounter a blockage and be forced to retrace back to the previous node and plan an alternate route. While the traveler only directly senses the validity of the attempted edge, blockages may be correlated, thus providing implicit information about the validity of other edges in the graph.

3.3.1 Blindfolded Traveler's Problem

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ be an explicit directed graph where $\mathcal{V}$ denotes the set of vertices, $\mathcal{E}$ denotes the set of edges and $\mathcal{W} : \mathcal{E} \to \mathbb{R}_{\geq 0}$ denotes the weight of each edge. For each edge $e \in \mathcal{E}$, let $x(e) = \{0, 1\}$ denote if the edge is invalid (0) or valid (1). Note that $x(e)$ is *latent*. Additionally, let $\eta(e) \in [0, 1]$ be the *latent* blockage of an edge. The blockage is the fraction of an edge that can be traversed before encountering an obstruction.

A traveler located at vertex v_1 may attempt to traverse any edge $e_{1,2}$ connecting a neighboring vertex v_2. An attempt $(v_1, e_{1,2})$ is mapped to a resultant vertex and traversal cost specified by the following function:

$$\Gamma(v_1, e_{1,2}, x, \eta) = \begin{cases} (v_2, w(e_{1,2})) & x(e) = 1 \\ (v_1, 2\eta(e_{1,2})w(e_{1,2})) & x(e) = 0 \end{cases} \tag{3.1}$$

Traversing a valid edge moves the traveler to the new vertex v_2 with a traversal cost equal to the weight of the edge $w_{e_{1,2}}$. Traversing an invalid edge returns the traveler to the original vertex v_1 with a traversal cost equal to the distance travelled to the blocked point and back, $2\eta(e_{1,2})w(e_{1,2})$.

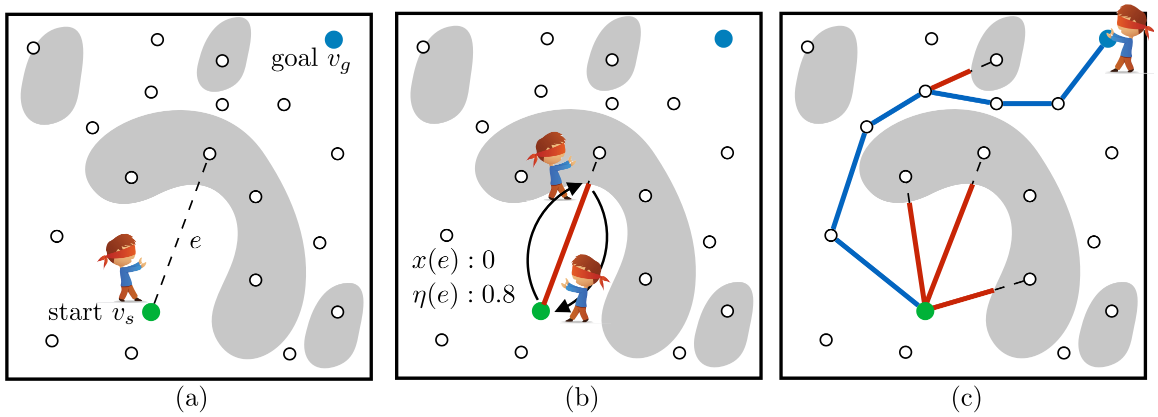


Figure 3.2: Blindfolded Traveler's Problem

The traveler has a prior $\mathcal{P}$ on the joint probability $P(x, \eta)$. When attempting to traverse edge e, the traveler receives the observation $o = (x(e), \eta(e))$. The traveler maintains a history of all observations, i.e. $\psi_T = \{o_t\}_{t=1}^{T}$. The Blindfolded Traveler's Problem can be fully specified by the tuple $\langle \mathcal{G}, \mathcal{P}, v_s, v_g \rangle$ where $v_s, v_g \in \mathcal{V}$ are the initial and goal vertices.

The solution to the BTP can be represented as a policy tree π, where the nodes specify an edge e that the traveler attempts to traverse. Each branch is labelled by an observation o_e. The root node of the tree is an edge emanating from start vertex v_s. To follow the policy, the traveler attempts to traverse the edge e and takes the branch matching the observation o_e to go to a next edge e'. The procedure repeats until the traveler reaches a terminal node which is always the edge e_{v_g,v_g}, i.e., a loop at the goal vertex.

The cost of a policy for a given (x, η), $c(\pi(x, \eta))$ is the sum of traversal costs. The goal of the traveler is to minimize the expected cost

$$\min_{\pi} \mathbb{E}_{(x,\eta)\sim\mathcal{P}} \left[c(\pi(x, \eta)) \right] \tag{3.2}$$

We show that BTP is NP-complete. We do so by constructing a mapping between any Optimal Decision Tree (ODT) problem, where the goal is to find a hypothesis with minimum tests, to an equivalent BTP. Since ODT is NP-Complete, so is BTP. For the proof and further details refer to the appendix A.2.

3.3.2 Contact-based Planning Problem as an instance of BTP

We now examine the problem of a robot arm planning with unknown workspace obstacles sensed only through contact and map this problem to an instance of BTP.

The robot's configuration space $\mathcal{C}$ is composed of free space $\mathcal{C}_{\mathcal{F}}$ and obstacles $\mathcal{C}_{obs} = \mathcal{C} \setminus \mathcal{C}_{\mathcal{F}}$. The robot operates in a workspace W containing workspace obstacles W_{obs}. A robot configuration $q \in \mathcal{C}$ occupies a workspace volume $\mathcal{R}(q) \subset W$. We say q is *in collision* if $\mathcal{R}(q) \cap W_{obs} \neq \emptyset$.

The graph $\mathcal{G}$ is a roadmap where vertices $\mathcal{V}$ are configurations and edges $\mathcal{E} : [0, 1] \rightarrow \mathcal{C}$ are paths through $\mathcal{C}$ connecting vertices, with $w(e) = ||e(0) - e(1)||$. An edge therefore represents the swept volume $W_e = \cup_{d \in [0,1]} \mathcal{R}(e(d))$. The prior $\mathcal{P}$ is a probability density over W_{obs}. This is mapped to $\mathcal{C}$ via $\mathcal{R}(\cdot)$ thus inducing a joint probability $P(x, \eta)$.

As in the previous chapter, we consider a robot that senses obstacles indirectly though collision using measured joint torque $\tau^{meas} \in \mathbb{R}^J$, where J is the number of robot joints. Using a mass model of the robot the expected joint torque due to gravity and dynamics τ^{exp} is calculated and used to estimate the external joint torque $\tau^{ext} = \tau^{meas} - \tau^{exp}$. A noise threshold τ^{th} is set for each joint and τ^{ext} triggers a collision observation at q_{col} whenever any joint exceeds its threshold. A successful edge traversal results in $o = (1, 1)$, while a collision yields $o = (0, \eta)$ where $e(\eta) = q_{col}$.

Furthermore, as a slight augmentation of BTP, a collision yields additional information. Joint i exceeding τ_i^{th} implies an external (contact) force on a link after joint i on the kinematic chain. A set of links $\mathcal{L}_{contact}$ that must contain a contact is constructed by first finding the largest i where $\tau_i^{ext} > \tau_i^{th}$, then adding all links downstream from joint i to $\mathcal{L}_{contact}$. Define $\mathcal{R}(q, \mathcal{L}) \subseteq \mathcal{R}(q)$ as the workspace occupancy for only links $\mathcal{L}$. A traveler may use the knowledge that an object must be in contact with $\mathcal{R}(q, \mathcal{L}_{contact})$, as opposed to anywhere on $\mathcal{R}(q)$.

The BTP for contact planning has a few defining characteristics that warrant attention. First, the edges of this BTP are highly correlated, because a single workspace obstacle can block multiple C-space edges. Hence even an independent prior over workspace occupancy translates to correlation amongst edges. The robot exploits this to gain information about untraversed edges. Second, it's unclear how one obtains priors. A uniform random distribution is certainly not realistic. A finite dataset of worlds has realizability issues on account of continuous observations. Designing parametric distributions that capture all likely worlds is difficult. Finally, a manually-specified prior might be inaccurate. How should the robot detect and compensate for this in a principled manner? We propose solutions that deal with these issues in the next section.

3.4 Belief Representations for Contact-based Planning

An agent maintains a belief over workspace occupancy W_{obs}, which we refer to as a world $\phi \in \Phi$ and represent it using a voxel grid. The belief at timestep t is represented as $b_t(\phi)$. Since each voxel can be either occupied or free, the set of worlds is $\Phi = \{0, 1\}^N$ where N is the number of voxels, thus explicitly enumerating all possible worlds is infeasible. We follow two approaches for maintaining the belief. The first is a non-parametric particle filter where a set of candidate hypotheses are maintained and possibly ruled out. The second is an approach that adds new hypotheses that are consistent with measurements. We also motivate and discuss mixing these methods.

3.4.1 Approach 1: Manifold Particle Filter (MPF)

A particle filter is a non-parametric Bayes filter that represents belief $b_t(\phi)$ as a finite set of possible candidate worlds $\Phi_t = \{\phi_t^1, \phi_t^2, \dots\}$ with associated weights $\{\mu_t^1, \mu_t^2, \dots\}$. In this paper, the particles model objects with known geometry but with varying positions. Since in the BTP objects are stationary, the process model is static, and particles are only updated due to the measurement model, thus we only update the particle weights and do not resample.

A known issue with particle filters is poor performance when the proposal distribution does not match the target distribution. A conventional particle filter performs measurement updates via importance sampling: sampling from $\phi_{t-1}^i \sim b_{t-1}$ and weighing by $\mu_t^i = P(o_t|\phi_t^i)$. In the case of a highly discriminative measurement such as a contact, the target distribution represents a thin manifold of possible object configurations which does not match the proposal b_{t-1}, causing particle starvation.

We therefore adopt the strategy used in the Manifold Particle Filter (MPF) (*Klingensmith et al.*, 2016a), depicted in Fig. 3.3 and detailed in Algorithm 7. For robot motions through free space where no collision is observed the MPF updates using importance sampling as in a conventional particle filter (Line 6). With our static process model this is equivalent to eliminating particles inconsistent with the new known free space.

When a collision is observed the MPF instead uses the contact manifold as the proposal distribution, sampling particles from obstacle configurations in contact with the robot arm (Line 10). The importance weights are then calculated using $P(\phi_t^i|b_{t-1}^i)$. b_{t-1}^i is approximated by applying a Gaussian kernel to Φ_{t-1}, called a Kernel Density Estimate. We implement the Implicit Manifold Particle Filter (*Klingensmith et al.*, 2016a) which approximates the proposal distribution by projecting the prior particles

Algorithm 7: Manifold Particle Filter

```
   input  : particles $\Phi_{t-1}$, $e, o_t = (x_t, \eta_t)$, $\mathcal{L}_{contact}$
   output: particles $\Phi_t$
1  $\Phi_t \leftarrow \emptyset$
2  for $\phi^i_{t-1} \in \Phi_{t-1}$ do
3      for $d \in [0, \eta_t)$ do
4          $q = e(d)$
5          $\phi^i_t \leftarrow \phi^i_{t-1}$
6          $\mu^i_t \leftarrow P(\mathcal{R}(q) \cap W_{obs} = \emptyset | \phi_t)\mu^i_{t-1}$
7      if $x_t = 0$ then
8          $W_{CM} \leftarrow \mathcal{R}(e(\eta_t), \mathcal{L}_{contact}))$
9          $\phi^i_t \leftarrow$ PROJECT $(\phi^i_{t-1}, W_{CM})$
10         $\mu^i_t \leftarrow$ KERNELDENSITYESTIMATE
             $(\Phi_{t-1}, \phi^i_t)$
```

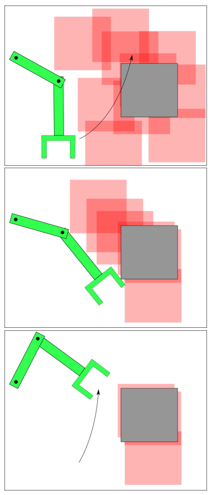


Figure 3.3: Manifold Particle Filter: The initial particles Φ_0 model configurations of the true obstacle before the robot moves (top). A collision during a motion causes particles to be resampled on the contact manifold (middle). Subsequent free space motions sweep through and eliminate some particles (bottom).

onto the contact manifold. Though computationally efficient, this projection does introduce significant bias, as the previous estimate appears both in the sampling and the re-weighting. In our implementation we translate each particle the minimum distance so that it overlaps with the robot in the collision configuration. This choice of projection can generate new particles that are inconsistent with past contact observations. While a more sophisticated projection operation is of interest, it is beyond the scope of this work.

MPF performs well when given an accurate initialization b_0, but for robots in the real world it is often unrealistic to assume the distribution over obstacles is known accurately. One such instance is when b_0 clusters the correct object far from the correct position. Another common and more difficult instance is when the particles model the incorrect object geometry, so no particle is capable of representing the true world.

3.4.2 Approach 2: Collision Hypothesis Sets (CHS)

To overcome the reliance on an accurate prior we can adopt the Collision Hypothesis Set (CHS) from Chapter II (*Saund and Berenson*, 2018) belief. To briefly review, a single CHS $\kappa_i \in W$ is the complete set of voxels that could explain observed collision

```
Algorithm 8: Collision Hypothesis Set
   input : CHSs $\mathcal{K}$, Known Freespace $W_F$,
           $e, o_t = (x_t, \eta_t), \mathcal{L}_{contact}$
   output: $\mathcal{K}, W_F$
1  for $d \in [0, \eta_t)$ do
2      $q = e(d)$
3      $W_F \leftarrow W_F \cup \mathcal{R}(q)$
4  if $x_t = 0$ then
5      $\mathcal{K}$.append($\mathcal{R}(e(\eta), \mathcal{L}_{contact})$)
6  for $\kappa_i \in \mathcal{K}$ do
7      $\kappa_i \leftarrow \kappa_i \setminus W_F$
```

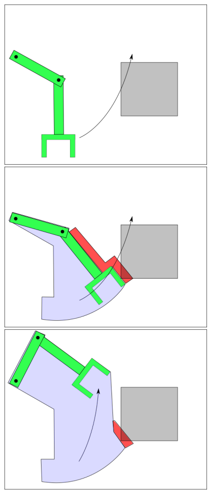


Figure 3.4: CHS: The robot initially plans a motion optimistic about unknown space (top). A motion sweeps out free space (blue) and a collision generates a CHS (middle). Future free space motion sweeps out more free space, potentially shrinking CHSs (bottom).

i. The CHS belief builds up a set $\mathcal{K} = \{\kappa_1, \kappa_2, \dots\}$ to explain all measurements.

Fig. 3.4 depicts the CHS update described in Algorithm 8. As the robot moves without collision, the swept volume of the motion is marked as known free space in the voxel grid (Line 3). When a collision is encountered during robot motion a CHS is added containing voxels of the links possibly in collision (Line 5). The known free space is then removed from all CHSs (Line 7).

$\mathcal{K}$ induce a belief $P(x)$ as follows:

$$P(x(e) = 0|\kappa_i) = \frac{|W_e \cap \kappa_i|}{|\kappa_i|} \qquad \text{effect of single CHS} \qquad (3.3)$$

$$P(x(e) = 1|\mathcal{K}) = \prod_i 1 - P(x(e) = 0|\kappa_i) \qquad \text{effect of all CHSs} \qquad (3.4)$$

where (3.3) captures the optimistic assumption that each κ generates exactly one occupied voxel, and (3.4) comes from the assumption that each κ is independent. Note that the CHS method never mark a valid edge as invalid. $P(x(e) = 1) = 0$ (i.e. e is marked invalid) only if W_e completely contains a κ. By construction a κ must contain an occupied voxel. Additionally note that when an invalid edge is attempted, the new κ created will cause $P(x(e) = 1) = 0$.

The CHS method is optimistic about free space. Sampling $\phi \sim \mathcal{K}$ yields worlds with only a few occupied voxels, not representative of realistic scenarios, though as a single voxel still blocks an edge the edge validities x may still match realistic scenarios.

However, while a particle filter with good initialization begins with a good estimate of $P(x)$, it may take many collisions to build up $\mathcal{K}$ sufficiently.

3.4.2.1 Approach 3: Mixture of Experts

We would like to benefit from an MPF prior, but also recover in the case of a bad initialization. In real world examples, it is unknown if an initial b_0 for the MPF is accurate *a priori*. Intuitively, online adaptation can be achieved by comparing particles Φ_t to Φ_0. If measurement updates cause particles to congregate in regions predicted by particles Φ_0 then the prior likely provides a reasonable model of the world. If instead particles update to unlikely regions or disappear entirely the prior was likely not accurate, and we would like to fall back to the CHS belief.

To achieve this behavior we mix the CHS belief b_t^{CHS} and MPF belief b_t^{MPF} using weights $\beta_t = (\beta_t^{MPF}, \beta_t^{CHS})$ to get the following:

$$b_t(\phi) = \frac{\beta_t^{MPF} b_t^{MPF}(\phi) + \beta_t^{CHS} b_t^{CHS}(\phi)}{\beta_t^{MPF} + \beta_t^{CHS}} \tag{3.5}$$

To set β_t^{MPF}, we consider three terms of interest: Φ_t is the current set of particles in the MPF, b_0^{MPF} is the initial MPF belief before any observations, and b^U is a uniform belief over a support set of volume V. The weights are set as:

$$\beta_t^{CHS} = 1 \tag{3.6}$$

$$\beta_t^{MPF} = \mathbb{E}_{\phi \sim b_t^{MPF}} \Big[\frac{P(\phi | b_0^{MPF})}{P(\phi | b^U)} \Big] \tag{3.7}$$

$$= \sum_{\phi_t^i \in \Phi_t} \mu_t^i \frac{P(\phi_t^i | b_0^{MPF})}{P(\phi_t^i | b^U)} = \sum_{\phi_t^i \in \Phi_t} \mu_t^i \frac{b_0^{MPF}(\phi_t^i)}{1/V} = V \sum_{\phi_t^i \in \Phi_t} \mu_t^i b_0^{MPF}(\phi_t^i) \tag{3.8}$$

where V is a tuning parameter. In other words, we set the weight of the MPF belief β_t^{MPF} by iterating over all particles and doing a weighted sum of the likelihood of the particle *under the original MPF belief* b_0^{MPF}. The weight β_t^{CHS} is set to be constant.

The rationale for setting β_t^{MPF} in this way is to measure how much the current MPF belief b_t^{MPF} has deviated from the original belief b_0^{MPF}. A large deviation indicates that the prior was not a good estimate and we should instead trust CHS. When the MPF prior b_0^{MPF} is accurate, there are at least some particles that have both a high weight μ_t^i and high likelihood under the original prior $b_0^{MPF}(\phi_t^i)$. Hence β_t^{MPF} is high. The deviation w.r.t b_0^{MPF} is measured relative to a uniform distribution with volume V.

When the prior is inaccurate, particles may still have a high weight μ_t^i. However $b_0^{MPF}(\phi_t^i)$ will be small since the particles have moved significantly, thus resulting in a small β_t^{MPF}.

3.5 Strategies for Solving the BTP

Since we established that BTP is NP-complete (Appendix A.2.2), we explore a number of efficient approximation strategies to solve the problem, by drawing from heuristics used in the related Canadian Traveler's Problem (CTP) (*Eyerich et al.*, 2010) (Section 3.5.2). We also propose a new heuristic (Section 3.5.1) that (to the best of our knowledge) has not been applied to a CTP.

3.5.1 Collision Measure (CM)

This heuristic balances exploration (assuming unexplored edges are free) with exploitation (penalizing edges with low validity likelihoods). The agent is at a vertex v_t and decides which edge e_t from the set of outgoing edges $\mathcal{N}(v_t)$ to traverse as follows:

$$\begin{aligned} \widehat{\mathcal{G}} &= (\mathcal{V}, \mathcal{E}, w(e) - \alpha \log P(x(e) = 1|\psi_t)) \\ e_t &= \left\{ e \in \mathcal{N}(v_t) \;\middle|\; e \in \textsc{ShortestPath}(\widehat{\mathcal{G}}, v_t, v_g)) \right\} \end{aligned} \tag{3.9}$$

Here $\widehat{\mathcal{G}}$ is an optimistic graph created by removing all edges that are invalid with probability 1 given observation history ψ_t. Further, the weights are penalized by log-probability. Log-probability is chosen because for a path ξ, the log-probability is additive over edges assuming independence, i.e., $\log P(x(\xi)) = \sum_{e \in \xi} \log P(x(e))$. A known blocked edge ($P(x(e) = 1|\psi) = 0$) yields a weight of ∞, and a known free edge ($P(x(e) = 1|\psi) = 1$) yields $w(e)$. At each iteration the CM strategy finds the shortest path over $\widehat{\mathcal{G}}$ and attempts the first edge.

We provide the outline of theoretical justification for using this heuristic, (see supplementary material (*Saund et al.*, 2019) for a detailed discussion). We first map BTP to a Bayesian search (*Ross*, 2014). In Bayesian search, an agent repeatedly inspects a series of n boxes until an item is found. The goal is to minimize the expected cost of searching the box. A greedy policy selects the box with the largest ratio of probability of containing an object over the cost of searching the box $\frac{p_i}{c_i}$. *Dor et al.* (1998)(Theorem 4.1) proved that a greedy policy has cost at most 4 times optimal cost.

We modify BTP as follows - the agent picks a path, travels along it till an obstacle is encountered, backtracks to the start and tries another path. This is a Bayesian search problem. A greedy policy is equivalent to a more general notion of the collision measure policy that can solve the following optimization

$$e_t = \left\{ e \in \mathcal{N}(v_t) \;\middle|\; e \in \arg\min_{\xi} \frac{w(\xi)}{P(x(\xi) = 1 | \psi_t)} \right\} \tag{3.10}$$

This has a bound of 4 w.r.t the optimal policy in the modified problem, and a bound of 8 w.r.t the original BTP problem.

The optimization in (3.10) is intractable as $P(x(\xi) = 1)$ is not additive. However we can instead solve $\arg\min_{\xi} w(\xi) - \alpha \log P(x(\xi) = 1 | \psi_t)$ where the cost function is additive (as log-probabilities are additive) and decomposes nicely. We show in supplementary material (*Saund et al.*, 2019) that this is a suitable approximation of the near-optimal policy. Furthermore, Collision Measure is complete on the modified BTP even when using the CHS approximation of the belief. Using CHS there are finite ξ, each attempt either reaches a goal or marks an edge as invalid, and no valid edge will ever be eliminated.

3.5.2 Baselines

To benchmark our proposed Collision Measure strategy we consider three categories of strategies commonly used in POMDPs – approaches that approximate the optimal expected cost-to-go of an action, also referred to as Q-value, with heuristics, approaches that use simulation to evaluate actions, and approaches that plan to gather information. For more details, refer to the Appendix A.3.

Optimism in the Face of Uncertainty (OFU) (*Brafman and Tennenholtz*, 2002): Find the shortest path on the optimistic graph and move along the edge on it.

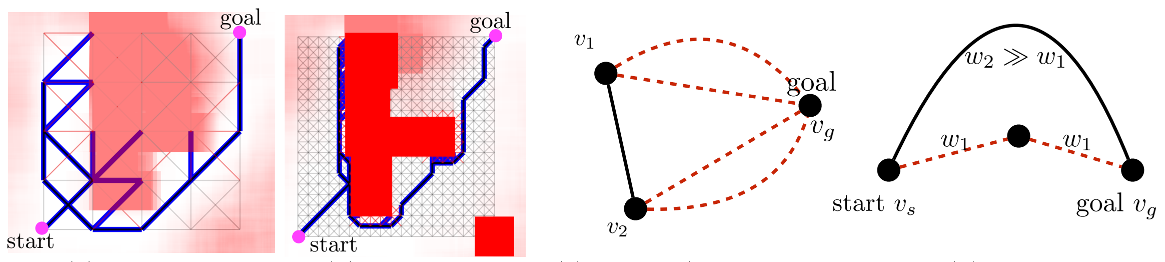
Thompson Sampling (TS) (*Littman et al.*, 1995): Sample a world from the current belief, find the shortest path in that world, and move along the edge on it.

Qmdp (*Littman et al.*, 1995): Given current belief, move along the edge with the least expected cost-to-go assuming the world is revealed at the next timestep.

Most Common Best Edge (MCBE): Given the current belief, move along the edge that has the highest probability of belonging to a shortest path.

Optimistic Rollout (ORO) (*Eyerich et al.*, 2010): Sample a world from the current belief, simulate moving along an edge and rollout with an optimistic policy. Move along the edge with best Q-value.

Upper Confidence Tree (UCT) (*Gelly and Silver*, 2007): Conduct a



(a) OFU trap (b) ORO trap (c) QMDP/MCBE trap (d) CM trap

Figure 3.5: Pitfalls for various strategies for a 2D BTP problems.

Monte-Carlo Tree Search (*Kocsis and Szepesvári*, 2006) where nodes are belief states and actions are edges to move along. The value of each belief is averaged over successors. To select actions for expansion during search, Upper Confidence Bound (UCB) is used.

Interleaving Planning and Control (Chapter II): Alternate between a global RRT planner and greedy local controller to plan a path to the goal through $\mathcal{C}$ with the least probability of collision. Note this is a strategy for the planning with contact feedback problem, but does not directly map to a BTP.

3.5.3 Pitfalls for Heuristic Strategies

Since all strategies considered are heuristics, it is important to recognize the pitfalls that they face. We illustrate these in Fig. 3.5. OFU is easily tricked into exploring cul-de-sacs that do not lead to the goal (Fig. 3.5(a)). A Bayes-aware heuristic would be able to predict the cul-de-saac and backtrack earlier. ORO offers significant improvement over OFU as it simulates executing OFU. However simply increasing the density of the grid yields a BTP where all neighbors of v_s fall into a cul-de-sac (Fig. 3.5(b)). ORO is not able to discover the non-myopic sequence of actions.

QMDP and MCBE avoid such optimistic pitfalls. However they rely on uncertainty disappearing after performing the first action. This can lead to infinite loops as shown in Fig. 3.5(c). The belief is such that the solid edge is known to be feasible while only one of dotted edges is feasible. When the agent is at v_1, it wishes to move to v_2 and vice-versa.

CM is also susceptible to pitfalls because it treats $P(x)$ independently. Fig. 3.5(d) shows an example where the solid edge is feasible while only one of the dotted edges is feasible. The only feasible path is the longer path with weight w_2. CM will choose the lower path as long as $2w_1 - \alpha \log 0.5 < w_2$.

However, of the four traps, the CM trap is the least concerning. In Fig. 3.5(d),

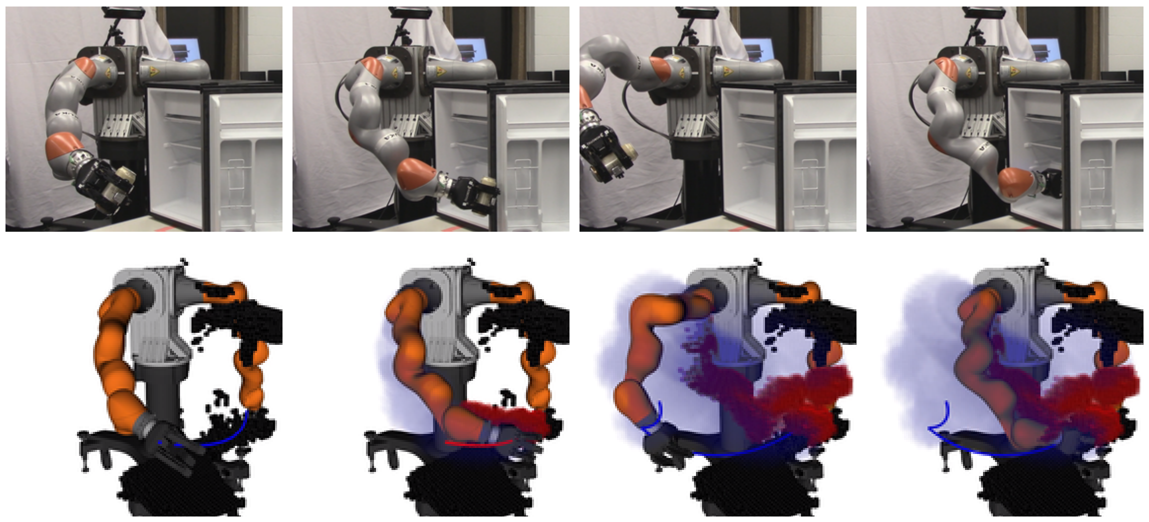


Figure 3.6: `Refrigerator` - Victor moving to place an object inside a refrigerator.

the suboptimality of CM is at most $\frac{4w_1+w_2}{w_2}$ which is small as $w_2 \gg w_1$. Moreover, an appropriate α would lead to the optimal answer. This suggest a sweep over α parameter in practice would help prevent such pitfalls.

3.6 Experiments

We performed experiments on simulated and real worlds for the "Victor" robot's right arm, a KUKA iiwa 7DOF arm that provides joint torque feedback.

Implementation Details: W is represented by a 200x200x200 voxel grid implemented on the GPU using GPUVoxels (*Hermann et al.*, 2014). Computing $P(x(e)|\psi)$ involves the expensive computation of swept volumes W_e, approximated by discretizing the configurations with a distance of 0.02 rad. For efficiency we lazily compute and cache W_e.

We constructed $\mathcal{G}$ in the $\mathbb{R}^7$ configuration space corresponding to the right arm of the Victor robot with 10000 vertices generated from the 7D Halton sequence and with edges connecting vertices within 1.8 rad, yielding $|\mathcal{E}| = 259146$. All strategies considered in Section 3.5 involve repeated shortest path queries over subgraphs of $\mathcal{G}$ with modified edge weights. Although any best-first search method is sufficient, we performed all shortest path queries using LazySP (*Dellin and Srinivasa*, 2016) to minimize the number of expensive edge-evaluation operations. All trials were conducted on an i7-7700K with a NVidia-1080Ti GPU.

Scenarios: We considered 2 real robot scenarios - `Refrigerator` and `RealTable`. In `Refrigerator`, Victor must reach into a refrigerator from behind (Fig. 3.6). In

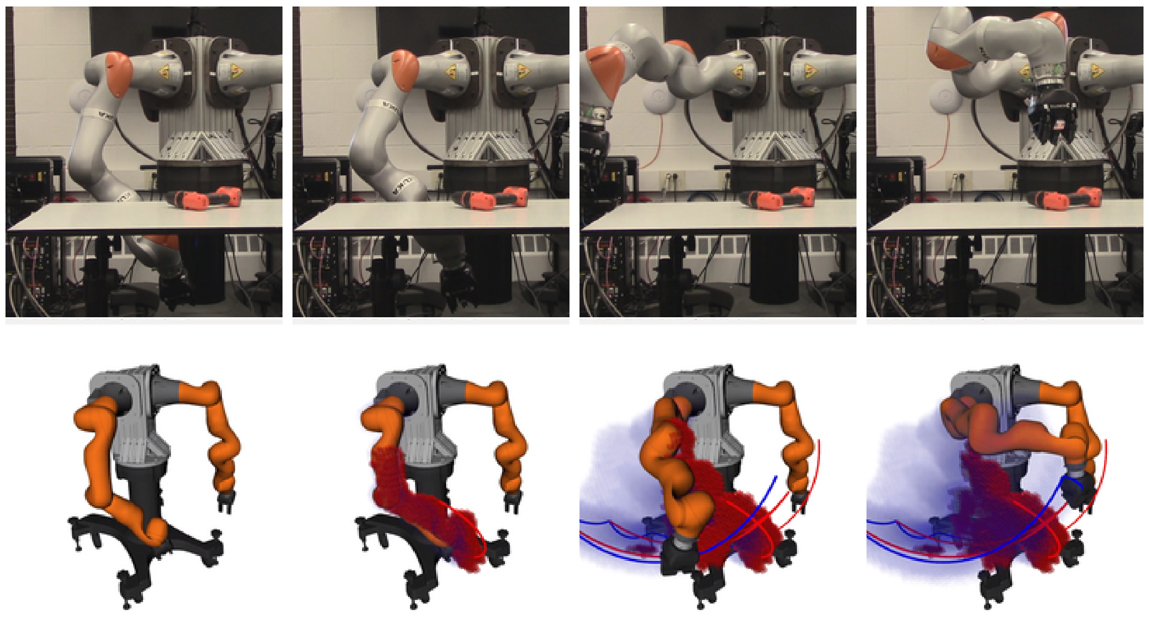


Figure 3.7: `RealTable` - Victor moving from below to above a table.

`RealTable`, Victor must move from below the table to above (Fig. 3.7). We also consider 2 simulated robot scenarios (Fig. 3.8) - `Bookshelf` and `Box`. In `Box`, Victor must reach into a box on a table where the back of the box unknown (which is a typical scenario due to sensor occlusion). In `Bookshelf`, Victor must reach into a bookshelf at a height above it.

We consider CHS, MPF with 100 particles, and MoE models of the belief. The MPF requires an initial belief b_0^{MPF}, which can have drastic effects on the behavior of strategies.

We consider three levels of difficulties based on how the prior b_0^{MPF} is chosen.

- `Easy:` true unknown obstacles with offset $\sim \mathcal{N}(0, 0.1)$
- `Medium:` true unknown obstacles with offset $\sim \mathcal{N}(0.1, 0.4)$
- `Hard:` a chair in the corner, with no knowledge of the relevant obstacles

In the real robot scenarios the `Easy` and `Medium` particle priors were manually generated, approximated the shape of the true obstacle. In the `Refrigerator` scenario W_{obs} is populated using a Kinect sensor mounted on Victor's head. In the `RealTable` scenario Victor is wearing a blindfold.

We compare across the three beliefs proposed in Section 3.4 and all strategies from Section 3.5, except UCT which was not tested due to excessive computational time.

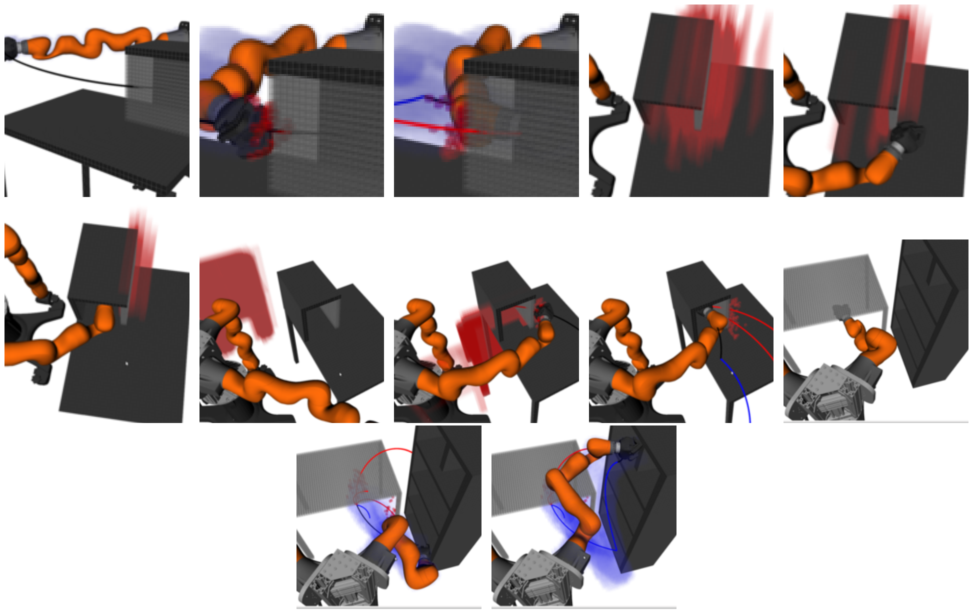


Figure 3.8: Simulation scenarios. Here CM is used in all scenarios. Top left: `Easy` setting of `Box` using CHS. Top right: `Easy` setting of `Box` using MPF. Bottom left: `Hard` setting of `Box` using MoE. Bottom right: `Hard` setting of `Bookshelf` using CHS.

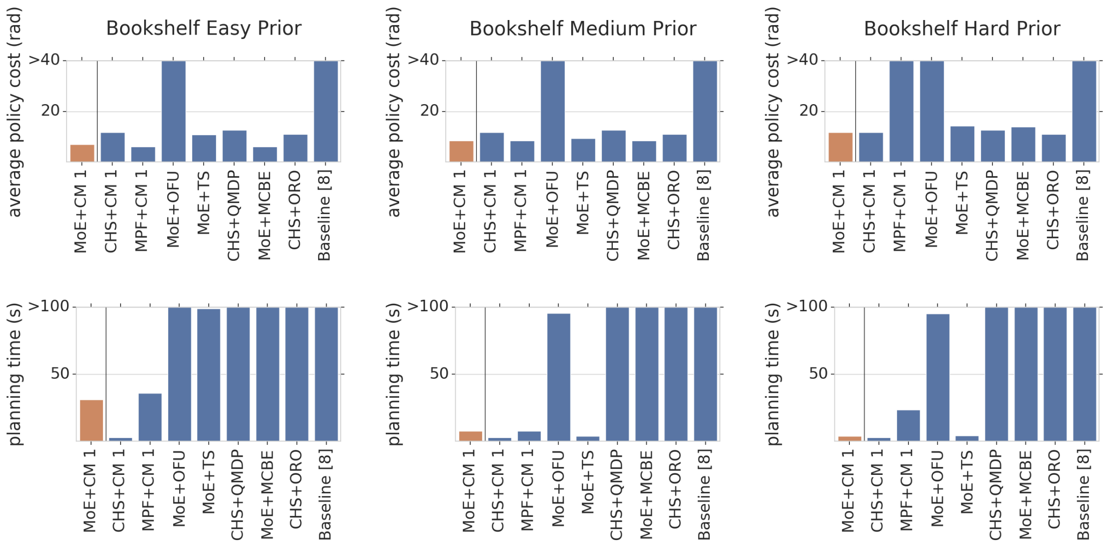


Figure 3.9: Results of applying various belief strategies and policies to the `Bookshelf` BTP. Our proposed MoE+CM is consistently fast and solves the BTP with low cost.

For the stochastic TS strategy we average across 10 trials. We test our proposed CM with $\alpha = 1$ and $\alpha = 10$. We also compare against the (non-BTP) baseline proposed in (*Saund and Berenson*, 2018) which interleaves an RRT with a local controller to find low cost paths through $\mathcal{C}$.

Results: Select results for the `Bookshelf` scenario are shown in Fig. 3.9 with full results for all scenarios shown in (*Saund et al.*, 2019). For the non-BTP baseline (*Saund and Berenson*, 2018) applied to the `Bookshelf` scenario we observe only 2 out of 20 trials succeeded within a 15 minute time limit.

Constraining motion to a roadmap yields a manageable action space and depth for the search for the strategies proposed. Furthermore, the roadmap allows reuse of the computationally expensive quantity $P(x(e)|\psi)$ within a single ShortestPath query, and reuse of the edge swept volume W_e between queries. Compared to the previous baseline (*Saund and Berenson*, 2018), we observe a significant improvement using the BTP framework.

Furthermore, we observe three key takeaways from the experiments.

1. CM performs well. CM consistently outperforms OFU, providing a lower cost policy in 19/24 experiments across scenarios, beliefs, and prior hardness. For our proposed MoE belief, CM outperforms OFU in 11/11 experiments, on average yielding 37% the cost. Compared to MCBE, CM yields a lower cost in 17/26 trials. In addition, averaged across all trials the planning time of CM is 15s, while MCBE is 217s.

2. MPF with a good prior performs well but breaks down when poorly initialized. MPF with the `Easy` prior outperforms CHS in 21/22 trials across all strategies and scenarios. MPF with the `Hard` prior only outperforms CHS in 1/22 trials, causing strategies to fail in half of trials.

3. MoE costs are approximately the minimum of MPF and CHS when using CM.

3.7 Conclusions

This chapter proposed the Blindfolded Traveler's Problem as a class of problems in planning under uncertainty. We showed that contact-based planning is an instance of BTP. We examined various strategies for approximating the belief over the workspace obstacles based on contact feedback and argue for a Mixture of Experts that work well with and without correct initialization. We also examined various policies for approximately solving the BTP and propose a new policy, Collision Measure, that is both efficient and has theoretical guarantees.

CHAPTER IV

Plausible Shape Sampling

The previous chapter introduced the Blindfolded Traveler's Problem and cast planning and replanning with contact uncertainty into this BTP framework. Many algorithms were examined for solving these BTP problems and a new heuristic was proposed. However a key limitation of Chapter III was the assumption a the shape of all obstacles was known and a prior was given specifying uncertainty of the object pose. Without such a prior, the previously proposed approaches for addressing BTP revert to the CHS uncertainty model. This chapter explores how to create a richer prior over obstacles observed in the world, capable of predicting diverse shapes from ambiguous depth images. Chapter V connects this richer prior to robot contact information.

4.1 Introduction

You look into a cabinet and see a coffee mug on the shelf. Though you only observe the front of the shell you have a rich prior of shapes and so can infer the occluded structure of the mug. Now suppose the handle is facing towards the back of the shelf, hidden from view. You may imagine scenarios where the handle is on the left, on the right, straight back, or perhaps there no handle at all. We propose a neural network architecture for generating these diverse samples over plausible completed shapes (Fig. 4.1).

More specifically, we generate a set of possible 3D shapes from a 2.5D depth image, such as that provided by a Kinect sensor. There is inherent ambiguity in this process as it is impossible to know the true occupancy of occluded space. We thus seek an algorithm which produces a set of plausible 3D shape estimates from the observed data.

Broadly, researchers have attempted two approaches when inferring 3D structure from a 2.5D depth image. Shape matching optimizes a model pose, potentially with uncertainty (*Desingh et al.*, 2016; *Peretroukhin et al.*, 2020), thus requiring meshes of any potential object, limiting their ability to generalize. Learned methods, such as Variational Auto Encoders (VAE) (*Wu et al.*, 2018), only require meshes during training and generate visibly pleasing shapes, but are optimized and evaluated on a single completion without consideration of other plausible completions.

Rather than operating on a single maximal-likelihood guess of the world, many robotics algorithms model and plan over a belief over worlds, thus we propose the Plausible Shape Sampling Network PSSNet, capable of generating diverse shape completions when multiple plausible shapes could fit a depth image. The *key insight* of this chapter is a restructuring of an Variational Auto Encoder to incorporate human-defined shape features during training. We use a normalizing flow to map the pose and size of a bounding box into a portion of the latent space of the VAE. During inference the network estimates a distribution over bounding boxes from which a specific box is sampled and used for reconstruction.

Evaluating the quality of our network presents a dilemma: if our network produces a shape different from the ground truth, how do we decide if that shape is plausible? We propose a classical non-learning method for generating plausible completions for a specific test dataset.

This chapter makes the following contributions:

1. A method to generate plausible completions for an evaluation dataset
2. Metrics to evaluate plausible diversity of a black-box shape completer
3. PSSNet: A network for sampling diverse and plausible shape completions

To validate our method, we perform experiments using mugs from shapenet (*Chang et al.*, 2015) and all YCB objects (*Calli et al.*, 2017) which show that for non-ambiguous completions our methods performs comparably to other recently-proposed shape completion methods. However, when there is ambiguity, baselines produce similar and poor quality completions while our method produces diverse yet plausible samples.

4.2 Related Work

Shape Matching: Robotics has studied the problem of inferring 3D structure from RGB and depth camera images for decades. In the *shape matching* variant the

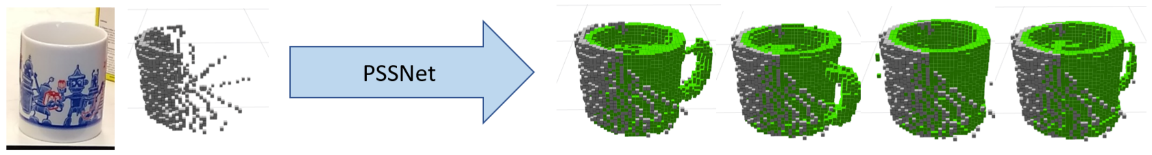


Figure 4.1: Our proposed PSSNet applied to a noisy segmented Kinect depth image of a mug produces multiple plausible reconstructions

pose or configuration of a target shape is estimated from observations. A classic but powerful non-learning approach uses the Iterative Closest Point (ICP) algorithm to align a mesh or pointcloud of a target object with the observed pointcloud (*Besl and McKay*, 1992; *Yang et al.*, 2020), with some newer methods accomplishing this using neural networks (*Narayanan and Likhachev*, 2016; *Deng et al.*, 2019; *Hodan et al.*, 2020).

Shape matching has inherent uncertainty due to both sensor noise and symmetries or repeated structures in the shapes, thus many robotics approaches predict a distribution of possible configurations for the target, often using particle filters (*Hausman et al.*, 2015; *Koval et al.*, 2015; *Desingh et al.*, 2019a, 2016, 2019b; *Liu et al.*, 2015; *Chen et al.*, 2017). Particle filters require an observation model that assigns a likelihood to the observed depth image given the proposed shape. Researchers have hand-crafted likelihood models using sum of squared pixel depth distances (*Desingh et al.*, 2016), outlier rejection (*Narayanan and Likhachev*, 2016), gaussian per-pixel error (*Wüthrich et al.*, 2013), and signed distance (*Schmidt et al.*, 2014).

Shape matching requires known meshes for objects, limiting the applicability in an unstructured novel world. Our work uses shape matching to construct an evaluation dataset of plausible shapes and configurations for each given depth image. Our construction uses ICP followed by an outlier rejection observation model to generate plausible particles. We use the plausible particles to evaluate how well PSSNet captures the uncertainty inherent in constructing a 3D model from a depth image. PSSNet does not perform shape matching, nor require models outside of the training process.

Shape Completion: In *shape completion* or *shape reconstruction* the 3D structure is directly predicted from the camera observation without fitting specific shapes, resolving the inherent ambiguity of unobserved space using a prior learned from a dataset. Shape datasets such as shapenet (*Chang et al.*, 2015) and YCB (*Calli et al.*, 2017) enable learning on sufficient examples to generate visually compelling results.

Recently, dozens of papers have proposed neural networks to perform shape completion. The most common architecture for shape completion learns an encoder to a feature space followed by a decoder to the shape output (*Zhirong Wu et al.*, 2015; *Choy et al.*, 2016; *Girdhar et al.*, 2016; *Wu et al.*, 2018, 2017; *Michalkiewicz et al.*, 2020; *Wen et al.*, 2019; *Xie et al.*, 2019; *Fan et al.*, 2017; *Yu et al.*, 2020; *Yang et al.*, 2018; *Wu et al.*, 2016). Though mostly addressed as a complete problem by itself, shape completion has been used as a component in robotics tasks (*Price et al.*, 2019). In different variants the encoder may accept voxelgrids (*Zhirong Wu et al.*, 2015; *Wu et al.*, 2017; *Choy et al.*, 2016; *Yang et al.*, 2018; *Dai et al.*, 2017), images (*Girdhar et al.*, 2016; *Xie et al.*, 2019), or point clouds (*Yuan et al.*, 2018a; *Fan et al.*, 2017). Similarly the decoder may produce voxelgrids (*Zhirong Wu et al.*, 2015; *Wu et al.*, 2017; *Choy et al.*, 2016; *Yang et al.*, 2018; *Dai et al.*, 2017; *Xie et al.*, 2019), point clouds (*Yuan et al.*, 2018a; *Fan et al.*, 2017), meshes (*Wen et al.*, 2019) or octrees (*Riegler et al.*, 2017). Our proposed network encodes to and from voxelgrids, however we expect out contributions to be applicable to other approaches either directly or by converting between representations (e.g. learning RGB to depth (*Wu et al.*, 2018)).

In these networks a reconstruction loss such as voxel-independent binary crossentropy guides the optimizer (*Zhirong Wu et al.*, 2015; *Choy et al.*, 2016; *Dai et al.*, 2017; *Wu et al.*, 2016), which leads to averaging over possible shapes when there is ambiguity, producing "blurry" completions. Generative Adversarial Networks (GANs) (*Goodfellow et al.*, 2014) penalize this averaging and are used to produce natural-looking 3D reconstructions (*Wu et al.*, 2016, 2018; *Yang et al.*, 2018). Even with recent improvements to GANs (*Gulrajani et al.*, 2017) some (*Wu et al.*, 2018) (including us) still find training GANs for shape completions unreliable, with some methods intentionally weakening the discriminator (*Yang et al.*, 2018) to improve stability. We might hope that by employing VAEs with GANs we could sample substantially different yet plausible completions for a single input, yet past work using this structure only evaluate a single sampled completion relative to ground truth (*Wu et al.*, 2016, 2018; *Yang et al.*, 2018). In our experience VAE-GANs have resulted in visually pleasing samples with low diversity.

Representing Bounding Box Uncertainty: Our proposal for encouraging diversity involves explicitly training the feature space of a VAE to represent means and variances in properties such as position, orientation, and size. Similar to a TL-network (*Girdhar et al.*, 2016) we provide ground truth information for these features during training, forcing the encoder to learn these features and providing the decoder with noiseless values. The vector representation of these chosen features and their uncertainties must be representable and learnable by a neural network, which is a notorious challenge when representing rotations in SO(3). Recently *Peretroukhin* *et al.* (2020) demonstrated a rotation belief representation amenable to deep learning, which would be interesting to add to our framework. We follow the approach of *Tremblay et al.* (2018) and represent pose as a bounding box using 8 3-dimensional points. However, the standard independent Gaussian prior of a VAE is a poor prior for boxes where we expect corner locations to be highly coupled.

Normalizing flows have become popular in image generation as a method to invertably and losslessly map the tightly coupled distribution of pixel values onto an independent Gaussian distribution (*Dinh et al.*, 2014, 2017; *Kingma and Dhariwal*, 2018). However, normalizing flows have also been proposed to model posterior distributions of VAEs (*Rezende and Mohamed*, 2015; *Vahdat and Kautz*, 2020). We take a similar, but inverted, approach and learn a normalizing flow as a map from the distribution of bounding boxes to the same independent Gaussian distribution used in our VAE.

4.3 Problem Formulation and Metrics

We assume a dataset of pairs (x, y) where x is the two voxelgrids (known occupied, known free) for voxelized shape y. In this work we refer to an *object* as a mesh at an unspecified pose and a *shape* as a voxelgrid produced by an object at a specific pose. We assume that for each x there is given a set of plausible completions $\mathcal{P}(x)$. We desire a non-deterministic function $\tilde{y}_i \sim f(x)$ where $\tilde{y}_i$ is a voxelgrid called a *completion* of x. Drawing n samples from $f(x)$ gives a set of completions $\tilde{Y}_x = \{\tilde{y}_1, ..., \tilde{y}_n\}$. Let $d(y_1, y_2)$ be a distance function between two voxelgrids. We define the Best Accuracy as $M_A(x) = \min_{\tilde{y}_i \in \tilde{Y}_x} d(\tilde{y}_i, y)$. For a given (x, y) pair in our test dataset we additionally evaluate the quality of f using 3 criteria:

1. **The coverage of plausible completions:**

$$M_C(x) = \frac{1}{|\mathcal{P}(x)|} \sum_{\hat{y} \in \mathcal{P}(x)} \min_{\tilde{y}_i \in \tilde{Y}_x} d(\tilde{y}_i, \hat{y}) \tag{4.1}$$

2. **The average plausibility of completions generated by f:**

$$M_P(x) = \frac{1}{|\tilde{Y}_x|} \sum_{\tilde{y}_i \in \tilde{Y}_x} \min_{\hat{y} \in \mathcal{P}(x)} d(\tilde{y}_i, \hat{y}) \tag{4.2}$$

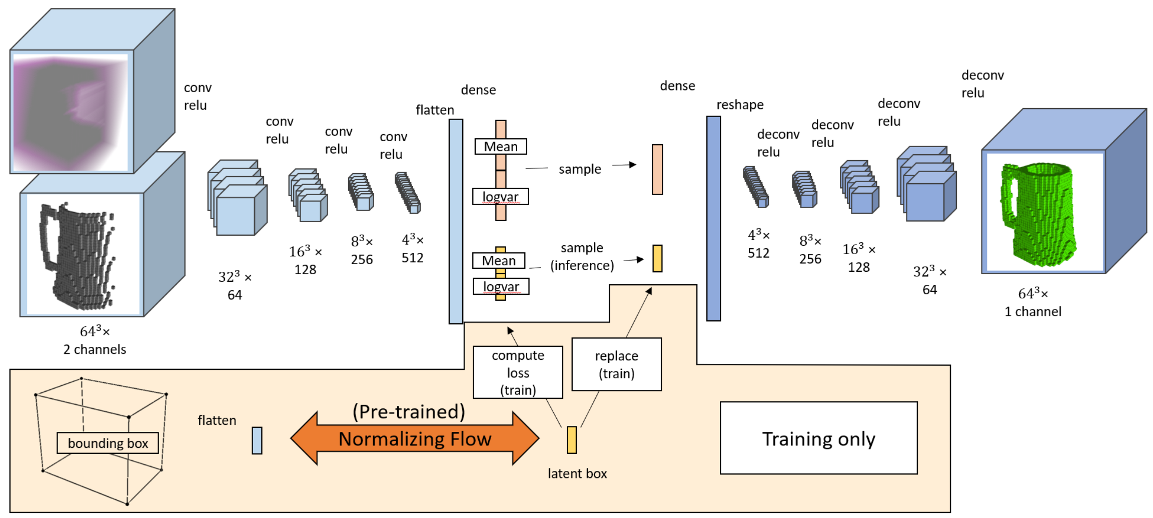


Figure 4.2: Our network, PSSNet, has the structure of a VAE during inference. During training we separate the latent space into typical learned VAE features and "latent box" feature produced by a learned normalizing flow applied to the ground truth bounding box. These latent box features are used both as a loss on the encoder prediction and as input to the decoder during training.

3. **The Plausible Diversity:**

$$M_{PD} = M_C + M_P \tag{4.3}$$

M_A is most similar to metrics used in previous work and is also not dependent on construction of $\mathcal{P}$. M_C penalizes plausible shapes that are not generated by f, whereas M_P penalizes network samples that are far from $\mathcal{P}$. We want to generate diverse samples that are plausible, thus we seek an f that achieves lowest M_{PD}, which is the chamfer distance between the sets $\mathcal{P}$ and $\tilde{Y}$.

4.4 Method

Our Plausible Shape Sampling Network, PSSNet, is an adaptation of a variational auto encoder (VAE). During inference PSSNet exactly follows a VAE, with an encoder that predicts a latent mean and variance from which a latent vector is sampled, and a decoder that produces a 3D voxelgrid from this latent vector. During training PSSNet differs from a VAE by replacing a portion of the latent space with a learned representation of an additional input.

Our training data starts with a set of mesh objects at a single pose. For each object we compute an axis-aligned bounding box. We then augment the dataset by applying rotations and translations to each object and bounding box. Finally, we compute the voxelized shape y and the known-free and known-occupied voxels x from a fixed view.

We train a normalizing flow on the bounding boxes of the training dataset with a Gaussian prior $\mathcal{N}(0,1)$. Each bounding box consists of 8 points, thus this flow maps from a 24 dimensional "box" space to a 24 dimensional "latent box" space ψ. The flow consists of 8 RealNVP networks (*Dinh et al.*, 2017), each with 2 hidden layers of size 512. During training batch normalization is performed between every other RealNVP network.

We then use this flow in training PSSNet (Fig. 4.2). The encoder takes as input x the known-occupied and the known-free 64^3 voxelgrids. $2 \times 2 \times 2$ convolutions with a stride of 2 and relu activation are applied 4 times sequentially using [64, 128, 256, 512] channels. The output is densely connected to a 200D latent-mean and 200D latent log-variance. During inference the network is identical to a VAE, and thus a latent vector is sampled from this mean and log-variance. The decoder inverts the structure of the encoder, with a dense layer reshaped into a 4x4x4x512 tensor followed by "deconvolution", or convolution-transpose layers again with a stride of 2. The output of the decoder, $\tilde{y}$, is a 64^3 voxelgrid that represents the probability of occupancy for each voxel, independently. We threshold this voxelgrid at 0.5 to produce a binary occupancy.

During training, PSSNet differs from a VAE during the latent space sampling. The latent space is partitioned into two vectors: z^f and the 24 dimensional latent box space z^b. During training z^b is replaced by ψ, the latent box produced by the normalizing flow applied to the bounding box, thus z^b has no effect on the final voxelgrid produced. A loss term L^{flow} rewards the log-likelihood of ψ given the latent mean z^b_μ and variance z^b_{logvar} produced by the encoder. Additional loss terms for binary cross-entropy reconstruction loss L^{rec} and L^{VAE} form the Monte Carlo estimate of the Evidence Lower Bound (ELBO) (*Kingma and Welling*, 2014) as applied to shape completion (*Yang et al.*, 2018; *Wu et al.*, 2018). With N as the total number of voxels (64^3), $y[i]$ as the target value $\{0,1\}$ of the ith voxel, and $\varphi(\mu, \sigma_{logvar})$ is the probability density at μ of a Gaussian with log-variance σ_{logvar}.

$$L^{\text{rec}} = p(y|z) \qquad = \frac{1}{N}\sum_{i=1}^{N} -y[i]\log(\tilde{y}[i]) - (1-y[i])\log(\tilde{y}[i]) \tag{4.4}$$

$$L^{\text{VAE}} = \log(p(z^f)) - \log(p(z^f|x)) \qquad = \log(\varphi(z^f, 0)) - \log(\varphi(z^f - z^f_\mu, z^f_{\text{logvar}})) \tag{4.5}$$

$$L^{\text{flow}} = \log(p(\psi|z^b_\mu, z^b_{\text{logvar}})) \qquad = \log(\varphi(\psi - z^b_\mu, z^b_{\text{logvar}})) \tag{4.6}$$

4.5 Quantifying Plausibility

Many shape completion methods evaluate results using the metric $d(f(x), y)$, which may be appropriate if the ground truth shape is unambiguous given the view from the depth camera. However, given two different shapes y_1, y_2 in the dataset with similar corresponding depth camera image $x_1 \approx x_2$ it is unreasonable to expect f to always generate the correct output. Furthermore, for our application we desire f to output diverse yet plausible shapes.

We propose two criteria to define some y_j as a plausible completion of x_i:

- Observing x_i given y_j must be sufficiently likely given a camera observation model
- The object represented by y_j is in the test database, possibly with a different pose

To address the first criterion we define an observation model $obs(x, y)$ as the likelihood of observing the depth image of the 2.5D view $Im(x)$, given that the true occupied voxels are y. Similar work uses the sum-of-squared depth differences of $Im(x) - Im(y)$ (*Desingh et al.*, 2016), yet we find this model is not sufficiently discriminative. On the other hand, applying a Gaussian belief to each pixel independently (*Wüthrich et al.*, 2013) is far too discrimative, as a single pixel can alter the likelihood by orders of magnitude. We have had the most success with an outlier rejection model (*Narayanan and Likhachev*, 2016).

We define our $obs(x, y)$ as a binary likelihood in Algorithm 9, indicating if x is or is not plausible. We first compute a mask of unreliable depth pixels as any pixel in $Im(y)$ with gradient greater than some threshold δ, and inflate this mask by one pixel (Line 3). We accept x as a plausible depth image of y if every reliable pixel of $||Im(x) - Im(y)||$ is below p_{max} = 4cm. Depending on sensor noise it may be appropriate to allow some outliers. We deem certain pixels in the depth image $Im(y)$

"unreliable" if they are at the boundary of shape, as discretization approximations due to pixelization may assign a vastly different depth value due to a slight translation orthogonal to the camera. We see this effect on physical hardware such as a Kinect as depth values near the boundary of shapes are sometimes far too large, causing points to trail off into the background.

With *obs* now defined, we generate candidate shapes using objects from the test dataset D_{TEST}. Uniformly sampling poses and objects is infeasibly inefficient, as the vast majority of samples are not plausible. As in previous particle filter approaches (*Klingensmith et al.*, 2016a), we sample candidate states and project these onto a manifold of states more likely to be plausible. Algorithm 10 describes our approach. For each $(x_i, y_i) \in D_{TEST}$ we attempt to create a plausible completion using every element $(x_j, y_j) \in D_{TEST}$. We find a transformation T to align the 2.5D voxelgrids x_j to x_i using ICP (*Rusu and Cousins*, 2011) (Line 3). We then check if the observation is plausible given this aligned shape.

Algorithm 9: Observation Plausible: $obs(x, y)$

```
1 obs_image = $Im(x)$
2 exp_image = $Im(y)$
3 mask = ComputeUnreliable(expected_image)
4 for each pixel index i not in mask do
5     if $||obs\_image[i]$ - $exp\_image[i]|| > p_{max}$ then
6         return False
7 return True
```

Algorithm 10: Compute Plausibles(x_i)

```
1 $\mathcal{P}(x_i) = \emptyset$
2 for $(x_j, y_j) \in D_{TEST}$ do
3     T = $ICP(x_j, x_i)$
4     if $obs(x_i|Ty_j)$ then
5         $\mathcal{P}(x_i) = \mathcal{P}(x_i) \cup Ty_j$
6 return $\mathcal{P}(x_i)$
```

4.6 Experiments

We present quantitative and qualitative results demonstrating that for non-ambiguous completions PSSNet performs on par with existing methods, and that when there is ambiguity PSSNet performs better. We created datasets from shapenet (*Chang et al.*, 2015) and YCB (*Calli et al.*, 2017) such that 2.5D views could have multiple consistent

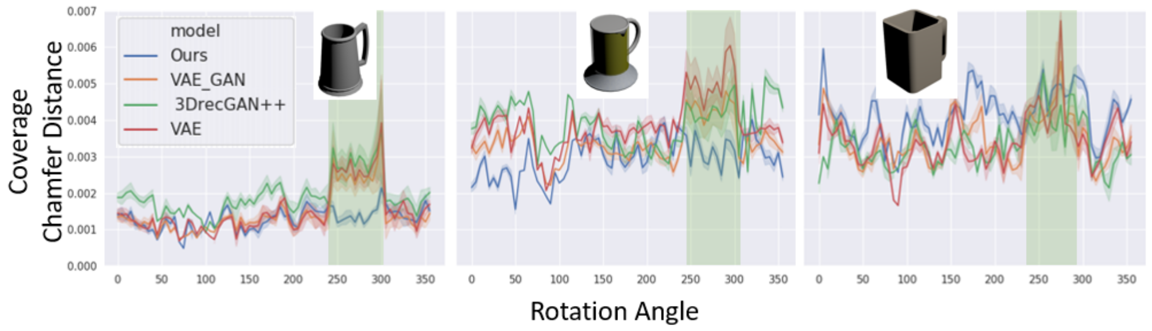


Figure 4.3: Coverage (Eq. 4.1) of various methods for shapenet mugs at different rotation angles. Rotations where the mug handle is occluded are highlighted.

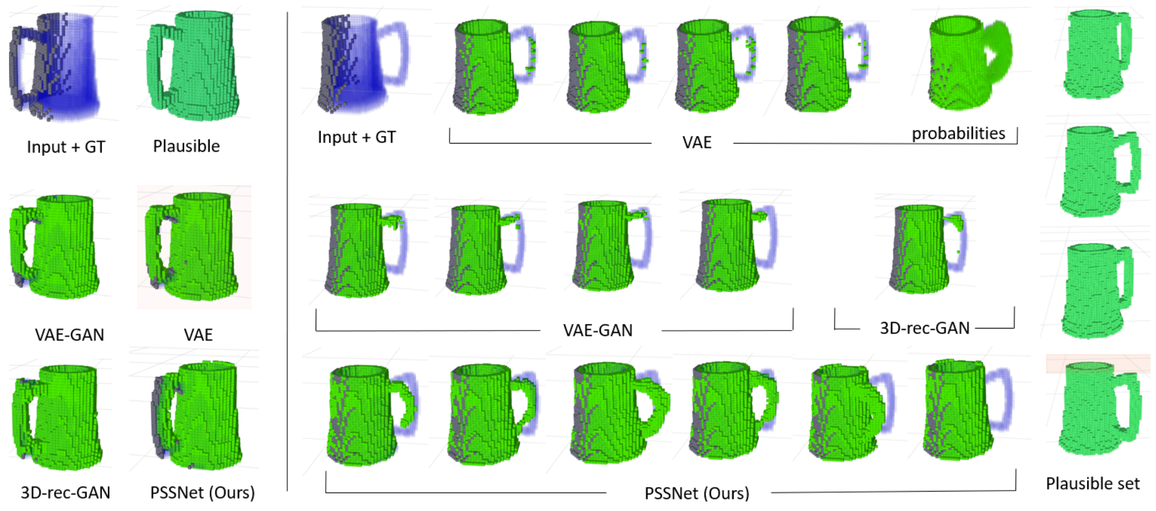


Figure 4.4: Completions (green) of a mug are sampled from the visible 2.5D view (grey). When the handle is visible (left) all methods produce similar mugs close to the ground truth (GT) (blue). When the handle is occluded (right) sampling from PSSNet yields mugs with different styles of handles in different orientations, with similar variation seen in the plausible set (4 shapes shown).

completions. We trained PSSNet as described above as well as a VAE, a VAE with GAN loss similar to (*Wu et al.*, 2016), and 3D-rec-GAN++ (without super-resolution layers) (*Yang et al.*, 2018), with networks accepting and producing voxelgrids of size 64^3. We constructed plausible completions $\mathcal{P}$ for each x in our test dataset and evaluated our metrics (Section 4.3) using $d(y_1, y_2)$, as chamfer distance between voxelgrids converted to pointclouds, as it is a common metric of shape completion quality (*Wu et al.*, 2018).

Shapenet Mugs: Using the *mugs* category from shapenet we constructed a dataset of 209 train and 5 test meshes. We rotated each mesh and associated bounded box in 5 degree increments about the vertical axis and voxelized using binvox (*Min*,

	Shapenet: all mugs				Shapenet: occluded handle			
	best acc	coverage of $\mathcal{P}$	avg. plaus	plausible diversity	best acc	coverage of $\mathcal{P}$	avg. plaus	plausible diversity
PSSNet (ours)	2.3	2.3	2.9	5.3	2.3	2.0	3.1	5.1
VAE	3.1	2.8	2.5	5.3	3.9	3.4	3.0	6.4
3D-rec-GAN	2.6	3.2	2.5	5.7	2.9	4.1	2.9	7.0
VAE-GAN	3.0	2.6	2.4	5.1	3.8	3.2	2.8	6.0
	YCB: 30 pixel wide slit				YCB: 6 pixel narrow slit			
PSSNet (ours)	1.3	1.7	3.2	4.8	2.3	4.5	4.4	8.9
VAE	1.5	3.1	1.8	4.9	3.0	7.8	2.8	10.6
3D-rec-GAN	1.2	3.6	1.2	4.8	4.6	9.6	2.9	12.4
VAE-GAN	1.3	3.3	1.6	5.0	3.1	7.9	2.7	10.6

Table 4.1: Best sample accuracy, Coverage of the plausible set, Average sample plausibility and Plausible diversity in mm. PSSNet performs best relatively in "Shapenet: occluded handle" and "YCB: narrow slit", as in these datasets there is ambiguity in the full shape given the partial view.

2004 - 2020), creating 15048 train and 360 test shapes. For approximately 1/5 of rotations, the handle is completely occluded from the 2.5D view.

We display the coverage metric for three of these shapes in Fig. 4.3. The left and middle mugs have a typical handle and when the handle is visible all methods obtain similar coverage. When the handle is occluded other methods perform far worse on M_C, meaning there are plausible completions that significantly differ from any samples produced by the network. PSSNet retains similar coverage even in these occluded regions. The right mug is square and unlike mugs in the training dataset, and the chamfer distance reconstruction error is dominated by the mug body reconstruction.

We visualize samples in Fig. 4.4 and qualitatively observe the same trends. When visible, all methods accurately reconstruct the mug handle, but when occluded other methods tend to average over plausible mugs and produce poor and non-diverse samples. For the 7 mugs from PSSNet the handles vary in orientation and style while remaining in the occluded region. We find PSSNet generates these diverse plausible handles for many but not all mugs. Qualitatively, we observe similar behavior for PSSNet with live Kinect depth images using a hard-coded segmentation of a mug (Fig. 4.1).

YCB with slit occlusion: We constructed a training dataset by applying a total of 24 rotations about the vertical and a horizontal axis for each YCB object. During training we occlude left and right portions of the depth image to simulate viewing the object through a vertical slit. We randomly translate the YCB shape and then

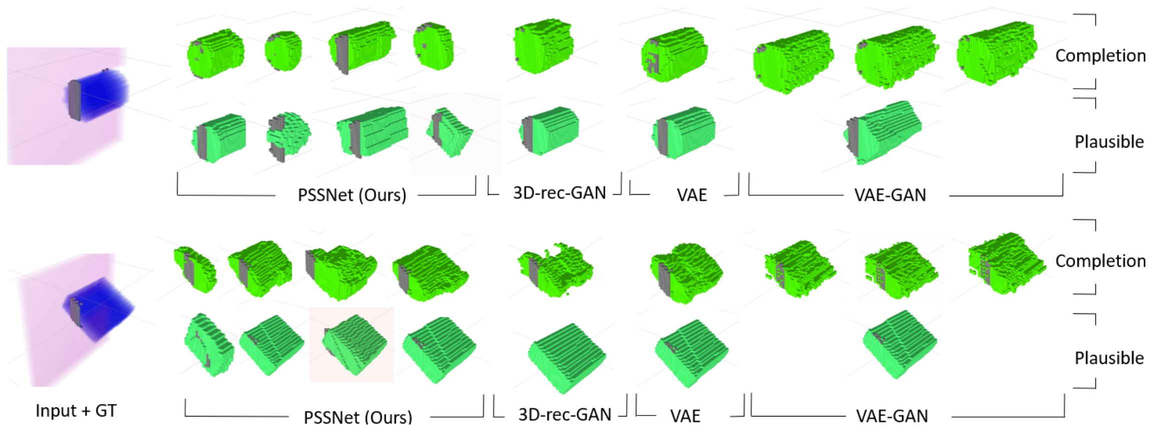


Figure 4.5: Completions of YCB objects as viewed through a 6 pixel narrow slit with the nearest plausible shape shown for each network sample. PSSNet generates diverse samples where other networks generate only small variations on the same sample.

randomly select a slit of width 5 to 30 pixels (1 pixel $\approx$ 0.6cm) and randomly place this slit so that the target object is visible in at least 5 columns of the image. A full 2.5D view of any YCB object leaves little ambiguity, this slit simulates viewing occluded objects in a cluttered scene.

We construct two test datasets for a subset of the YCB objects by using the same set of rotations but fix the translations and fix slit widths to 6 and 30 pixels. For each fixed slit width we construct a separate $\mathcal{P}$ by fitting (Alg. 10) each test shape at each orientation and each translation along the slit in 2 pixel increments. 6 pixels is a small portion of each object, thus in this dataset different objects with many different translations tend to match each x. The 30 pixel slit captures most of the object, so there is little ambiguity as to the 3D shape. We visualize completions in Fig. 4.5.

Metrics averaged over all test datasets are shown in Table 4.1. PSSNet consistently provides the best coverage. PSSNet performs comparably in plausible diversity for the datasets with lower ambiguity and outperforms baselines for datasets with greater ambiguity.

4.7 Discussion

We achieve our goal of creating a network that generates diverse samples, while other networks generate only small variations on a single completion. PSSNet, however, performs worse on M_P, indicating that either PSSNet sometimes produces poor quality samples, or that $\mathcal{P}$ lacks some plausible completions. Subjectively, we see both cases. Given a larger set of test shapes, $\mathcal{P}$ would contain more shapes, and likely M_P would improve.

Below we discuss advantages and limitations of our design choices for PSSNet and the plausible set:

Feature replacement: The main advantage we see in partial feature replacement in the latent space of the VAE is proper credit assignment between the encoder and decoder during training of ambiguous samples. For inputs where the reconstruction is inherently ambiguous we desire the encoder to predict variance in the latent space. Given this ambiguity the reconstruction loss is minimized when the decoder averages over the ambiguity. Replacing these latent box features during training removes some ambiguity so that the minimum of the reconstruction loss is a specific object.

The main limitation we perceive is the addition of a human-defined prior on feature structure, requiring domain-specific preprocessing of a dataset. However, as our network still retains many learned latent features we do not lose the ability to generalize beyond the human-defined features.

Normalizing flow: The normalizing flow transforms the human-defined features into a range appropriate for our $\mathcal{N}(0,1)$ VAE prior. In addition, features sampled with independent variance map back to (approximately) boxes, which is not true of independent sampling of bounding box coordinates. Furthermore, training this flow does not require defining a distance function in latent box space, and with a well-trained flow any latent box from the dataset will be relatively likely under the VAE prior.

Our choice to train the flow separately from the VAE simplifies the training but limits the distributions representable by our VAE latent space, as the latent distribution z^b assumes independent variance on each dimension. To illustrate this limitation, suppose a correlated change of $\psi[1]$ and $\psi[2]$ caused "box z-axis rotation" whereas the anticorrelated change caused "box x-axis translation". In this hypothetical it is impossible to represent uncertainty only in "box z-axis rotation" using independent uncertainty over z^b. We have not quantified this limitation, but in practice suspect it is mitigated from the over-parameterization of our features by using a 24-dimensional box.

Computing the Plausible Set: ICP finds local, not global, minima and typically ICP is run many times with different initializations. Our dataset D_{TEST} contains many copies of each object at different rotations, and these copies serve the function of different initializations.

However, there are some limitations of our plausible set computation. Our algorithm to compute $\mathcal{P}$ is quadratic in the size of the dataset, and although $\mathcal{P}$ is not needed for inference and only needs to be computed once for each D_{TEST}, this practically limits the size of our dataset. In addition, our observation model explicitly ignores small depth errors without considering correlation of errors between pixels, yet small but correlated depth differences could be used to identify larger shapes. For example, consider a dataset where some shapes have a small embossed logo, and thus detecting this logo should inform the completion of the full shape. Similarly, we explicitly discard depth values on the borders of shapes as independently these pixels tend to be noisy, yet again correlated depth values may provide useful information that is observable even with the independent noise. Our network f may use such features, but $\mathcal{P}$ will not, thus our evaluation may be overly harsh on our network, penalizing it for not generating shapes in $\mathcal{P}$ even when they are not plausible.

4.8 Conclusion

In this chapter, we proposed PSSNet, a method for generating diverse yet plausible 3D completions of a 2.5D depth image. A normalizing flow transforms the side information of the true shape bounding box into a feature space, which is used during training to encourage an encoder to generate diverse latent space samples, and to aid the decoder in producing plausible samples. To evaluate this method on a specific dataset we proposed a shape matching method to generate a set of plausible completions, as well as metrics for plausible diversity. In experiment PSSNet generated diverse samples and outperformed existing approaches for depth images with ambiguous reconstructions.

CHAPTER V

CLASP: Shape Completion with Contact

Chapter IV introduced PSSNet as a method for generating diverse yet plausible completed shapes from depth images. This chapter extends this shape reconstruction when robot contact information is available. An initial set of shapes is generated from depth camera information alone. As a robot moves around and bumps objects in the scene, the generated reconstructions update to reflect the learned contact information.

5.1 Introduction

You look into a cabinet and see a box of crackers. You reach in and attempt to grab the box from the side, but your fingers hit something. Perhaps this box is larger than you thought? Your mental model of the box updates, you try a wider grasp, and you successfully retrieve your snack. Robots are currently not so adept. While they can estimate the pose of known shapes (*Klingensmith et al.*, 2016a) or estimate parameters of objects (*Desingh et al.*, 2019b), they cannot yet fuse this visual and contact information to draw from the wide range of shape priors in the world. A robot could try to learn its next action directly from vision and force feedback instead (*Lee et al.*, 2020), but this approach lacks the logic to generalize to scenarios not seen in training.

This chapter proposes a method that allows robots to mimic the process of updating object shape from contact information. A shape completion neural network first generates beliefs over possible object shapes based on visual RGBD data. The belief updates the object shape to be consistent with contact information gathered by a robot moving in the scene. We make the realistic assumption that the RGBD camera perceiving the scene suffers from sensor noise and occlusion. As in the rest of this thesis, we assume the robot can sense *if* it collides with an object, but not *where* the contact was made (i.e., no sensorized skin). Many of the "cobot" platforms available today utilize this contact model to detect collision and stop before harming a person.

Formally, this type of contact creates a contact manifold, a thin space of shapes with a boundary bordering the robot. Past work has projected shapes onto the contact manifold in object pose space (*Koval et al.*, 2015) and robot configuration space (*Klingensmith et al.*, 2016a), but both require known shape geometry. Our objective is to update the unknown shape geometry to satisfy contact constraints. Returning to the cracker box example, the robot will be unsure if the contact occurred at the top finger, the bottom finger, or perhaps the back of the hand or the elbow. Filling in all possible contact points would lead to absurd scenes with robot shells protruding from the cracker box. However, ignoring this contact information leaves the robot with the original belief of the thinner cracker box and no explanation of why the attempted grasp failed. Our shape completion network generates a prior in latent shape space which can be decoded into shapes in workspace; however, shapes generated directly from this latent prior are unlikely to satisfy contact constraints.

The *key insight* of this chapter is that latent samples from our neural network can be projected onto the contact manifold in the latent shape space using iterative gradient descent, creating shapes both likely under the visual prior and consistent with the contact information. We further expect these projected shapes to be closer to ground truth than direct samples not considering contact.

We accomplish this with our proposed Constrained LAtent Shape Projection (CLASP), which stores a belief over shapes in a particle filter. Each particle represents a collection of latent object shapes which can be decoded into a scene. Every new robot measurement of contact and freespace triggers an update on all particles. During each particle update, gradient steps are taken to increase the occupancy likelihood of the most likely point(s) explaining the contact(s), decrease the occupancy likelihood of the free points, and increase the latent likelihood under the shape prior.

We test this method both in simulation and on a live robot by constructing scenes of objects on a tabletop, generating robot motions that generate freespace and contact observations (Fig. 5.1), updating the belief using CLASP as well as baselines, and comparing the set of sampled shapes to the ground truth scene. We find CLASP outperforms both ablations of CLASP, as well as the approaches of Rejection Sampling and directly updating the input to the shape completion network. We also find that CLASP produces more accurate scenes than a VAE_GAN shape completion network.

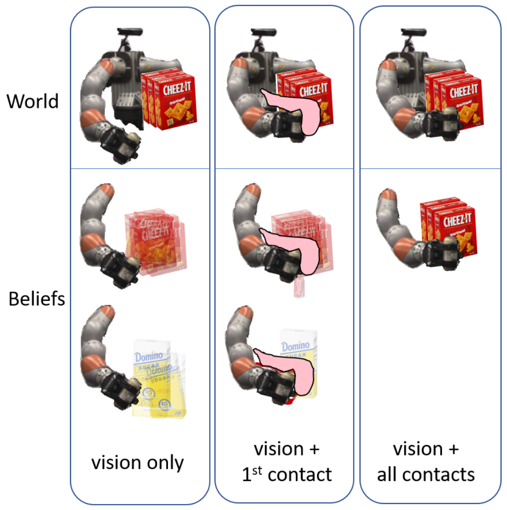


Figure 5.1: A visual RGBD view of objects leave ambiguity final shape due to sensor noise and occlusion, which we store as a set of sampled scenes in a particle filter. Contact information (pink) reduces ambiguity, and using CLASP the particles converge to the true shape.

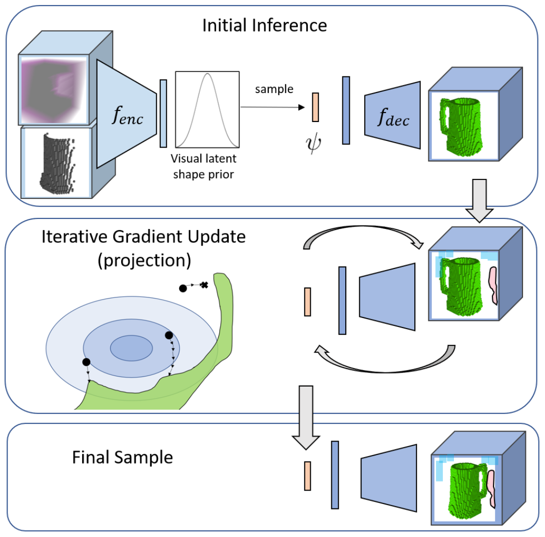

Initial Inference
f_{enc}
Visual latent shape prior
sample
ψ
f_{dec}
Iterative Gradient Update (projection)
Final Sample

Figure 5.2: CLASP Architecture
Top: Shapes sampled from RGBD.
Middle Right: A robot motion detects free space (light blue) and a collision set (pink).
Middle Left: Latent samples are projected to satisfy contacts (green). Ovals depict the latent prior.
Bottom: Final samples satisfy the contact constraints.

5.2 Related Work

Shape Completion from Vision: The goal of Shape Completion is to predict a full shape from a single partial input. Recently, neural networks have become a popular method of shape completion. A common network architecture learns an encoder to a feature space followed by a decoder to the shape output (*Zhirong Wu et al.*, 2015; *Choy et al.*, 2016; *Girdhar et al.*, 2016; *Wu et al.*, 2018, 2017; *Michalkiewicz et al.*, 2020; *Wen et al.*, 2019; *Xie et al.*, 2019; *Fan et al.*, 2017; *Yu et al.*, 2020; *Yang et al.*, 2018; *Wu et al.*, 2016). Scene Completion networks are trained on larger spatial volumes of occupied points and use similar architectures with adaptations to join information from multiple scales (*Roldão et al.*, 2021). While most methods predict the single best estimated shape, we build off work that uses a variational autoencoder architecture to produce plausible and diverse shapes (*Saund and Berenson*, 2020b). We draw from the vast work on shape and scene completion and contribute a method that improves scene estimates using contact information from a robot. For a more thorough overview of shape completion from vision, refer back to chapter IV 4.2.

Shape Completion from Touch: In different works, "touch" can refer to a single known contact point, a contact configuration, force-torque measurements, or a rich tactile sensor. Work using the definition of contact point or contact configuration typically uses touch to reduce the version space of shape possibilities (*Jung*, 2019), but such approaches cannot tractably capture the diversity of all shapes. Alternatively, some situations model known shapes with unknown poses (*Desingh et al.*, 2019b). Both classical Iterative Closest Point (*Besl and McKay*, 1992; *Yang et al.*, 2020) and neural networks (*Narayanan and Likhachev*, 2016; *Deng et al.*, 2019; *Hodan et al.*, 2020) have been used to predict valid poses. To generate samples consistent with contact information, the Implicit Manifold Particle Filter projects sampled poses onto the contact manifold using an iterative approach (*Koval et al.*, 2015). Analogously, we project sampled particles onto the contact manifold in the latent shape space of our neural network.

Touch can also refer to the rich tactile sensors such as GelSight (*Yuan et al.*, 2017) or soft-bubble grippers (*Alspach et al.*, 2019), with input more analogous to images. Neural networks have used these sensors for material classification (*Yuan et al.*, 2018b) and grasped pose estimation (*Kuppuswamy et al.*, 2020). We do not assume our contact sensing has such rich information.

Combining Vision and Touch: A neural network can combine vision and touch (force + torque (*Lee et al.*, 2020), or GelSight (*Li et al.*, 2019)) using separate encoders to a latent space for each sensing modality alongside a decoder to a variety of spaces. We considered a similar encoder structure with a decoder to produce completed shapes, but this would require a large dataset of (Shapes $\times$ Contact + Freespace) measurements and the resulting network would be only applicable to the robot used for training. Our method is most similar to the work of *Wang et al.* (2018), which uses gradient descent on the latent space of a shape completion network to enforce touch constraints. Where that work uses a high-resolution GelSight tactile sensor to refine shape details previously reconstructed from vision, our work focuses on ambiguous shapes (e.g. a box with unknown depth, or novel shapes not in the training data) and the lower information measurement of contact detection. We accomplish much larger shape updates by using a diverse set of predictions and a novel projection loss function.

5.3 Problem Formulation

Consider a robot $\mathcal{R}$ observing a static scene composed of specific objects o_j sampled from some distribution of objects $\mathcal{O}$. The objects divide the workspace into occupied space W_{occ} and free space $W_{\mathcal{F}} = W \setminus W_{occ}$ The robot has access to a training subset of $\mathcal{O}$ beforehand, but does not know the specific objects o_j in the current scene.

The robot observes the scene with two distinct sensing modalities. In the visual modality, the robot views the scene from a stationary RGBD camera receiving color depth images Im. Due to sensor noise and occlusion these depth images offer an incomplete and noisy measurement on the full region of W_{occ} occupied by the obstacles. From the camera image we assume the scene can be segmented into distinct objects from $\mathcal{O}$.

For the tactile modality, consider a robot that is able to sense *if* it has made contact with any object, but not *where* along the robot surface the contact was made. We assume the contact does not move the objects. For a configuration in configuration space $q \in \mathcal{C}$, let $\mathcal{R}(q) \subset W$ denote the region of workspace occupied by the robot. A robot that has visited configurations $\{q_1, q_2, ...\} = Q_{free} \subset \mathcal{C}$ without observing contact can carve out regions of known free space:

$$\cup_{q \in Q_{free}} \mathcal{R}(q) = W_{known\_free} \subset W_{\mathcal{F}} \tag{5.1}$$

For each configuration $q_{contact} \in Q_{contact}$ where contact is observed, there must be at least one object point in collision with the robot (and not in known freespace).

$$\forall q_{contact} \in Q_{contact} \; \exists \; p_{contact} \in \big(\mathcal{R}(q_{contact}) \setminus W_{known\_free}\big) : p_{contact} \in W_{occ} \qquad (5.2)$$

Using existing nomenclature, each such region is called a Collision Hypothesis Set (CHS) (*Saund and Berenson*, 2018).

Our objective is to model the conditional occupancy $p(W_{occ}|\mathcal{O}, Im, Q_{free}, Q_{contact})$. Specifically, we desire a stochastic function $g(\mathcal{O}, Im, Q_{free}, Q_{contact})$ which generates sample W_{occ} as similar as possible to the true conditional distribution. Since the true conditional distribution is unknown, in practice we seek to minimize the distance of drawn samples to the ground truth scene.

5.4 Method

Our approach is to use a particle filter storing a collection of latent shapes. We first segment the scene into distinct objects, then use an existing shape completion neural network to draw latent shape samples z_j for each object from $p(z_j|\mathcal{O}, Im)$, initializing the particle filter. Each particle can be decoded into the objects in a scene, thus the collection of particles represents the belief $p(W_{occ}|\mathcal{O}, Im)$. We propose Constrained LAtent Shape Projection (CLASP) as the measurement update, projecting these samples onto the constraints imposed by Q_{free} and $Q_{contact}$.

5.4.1 Initial Belief

The RGBD camera images are passed to a segmentation algorithm, which yields distinct pixel regions in the image corresponding to different objects o_j. For each object o_j the corresponding portion of the depth image is converted first to a point cloud, then voxelgrids of known-occupied and known-free space centered around the visible object points with a transform T_j mapping the voxelgrid to the workspace coordinates.

For each object o_j we use the Plausible Shape Sampling Network (PSSNet) (*Saund and Berenson*, 2020b) f to generate possible shape completions. PSSNet is structured as a variational autoencoder. An encoder f_{enc} maps the known-free and known-occupied voxelgrids to a mean and variance in latent space. A latent vector z can be sampled and passed to the decoder f_{dec}, which outputs a probability of occupancy for each voxel. Thresholding (e.g. $p > 0.5$ for each voxel) yields a completed shape.

An object o_j that is representable by f can be stored compactly as z_j such that

$f_{dec}(z_j) = o_j$. The transform T_j maps the completed shape into the workspace frame. A world is composed of static objects $\{o_1, o_2, ...\} \in \mathcal{O}$. A particle ϕ stores a specific world as a sequence of latent-space vectors $\{z_1, z_2, ...\}$. We sample worlds conditioned on only the depth-image observation by independently sampling latent vectors of objects. The initial belief is a set of particles $\{\phi_1, \phi_2, ...\} \in \Phi$ generated from the information from the depth camera before any robot motion.

5.4.2 Projecting a single object

Sampling particles using only camera information may yield worlds that are inconsistent with the robot contact information. For example, PSSNet may predict objects that extend far into occluded space that intersect regions the robot has moved through. Alternatively, PSSNet may predict objects that do not extend into occluded space, and so the robot may observe contact with no object to explain the collision. Predicting shapes from vision and robot contact in a single pass would require a dataset specific to each robot and a specific set of motions.

To resolve these inconsistencies, sampled particles are projected onto the constraints in the latent space of the shape completion network, shown in Fig. 5.2. For sample i of object j, z_j^i induces a workspace occupancy. Our constraints lie in the workspace, but we wish to project the latent space vector. Therefore, the projection is accomplished by optimizing a loss via gradient updates on z_j^i while holding f_{dec} fixed, mirroring the process of training a neural network but optimizing the input instead of the network weights. Consider the unthresholded voxelgrid with values between 0 and 1 produced by the decoder: $f_{dec}(z_j^i) = \phi_j^i$.

We optimize the loss: $L_{all} = L_{free} + L_{occ} + L_{prior}$

The first term L_{free} penalizes all voxels predicted above a threshold δ that are known to be free.

$$L_{free} = \sum_{x,y,z} \begin{cases} \max(\phi_j^i(x,y,z) - \delta, 0) & W_{known\_free}(x,y,z) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

The second term L_{occ} penalizes unexplained contact. Each contact $q_{contact}$ must be caused by some object s.t. $\mathcal{R}(q_{contact}) \cap W_{occ} \neq \emptyset$, however it is not obvious which object is responsible for the contact, or which voxel of the object contacted the robot. During optimization we consider a specific assignment of $Q_{contact}^j$ to object o_j. We define the assignment process in Section 5.4.3. Because a single occupied voxel is enough to explain a contact, at each iteration the loss is optimized based on the maximum prediction of occupancy overlapping with the collision hypothesis set.

$$L_{occ} = \sum_{q \in Q^j_{contact}} 1 - \max\left(\mathcal{R}(q) \cdot \phi^i_j\right) \tag{5.4}$$

The final term L_{prior} penalizes deviation of z from the original distribution predicted by the encoder. Without this constraint, z can deviate arbitrarily, losing all dependence on the depth image and even leaving the training domain of f_{dec}. This would produce completions that no longer look like objects. L_{prior} is weighted by α to maintain a similar magnitude of gradients to L_{free} and L_{occ}.

$$L_{prior} = -\alpha \log\left(P(z^i_j | f_{enc}(Im))\right) \tag{5.5}$$

Sampling the occupancy for a specific object o_j given Im, Q_{free} and $Q^j_{contact}$ is thus accomplished by sampling a z_j and optimizing until the constraints are satisfied. Projection can fail if an iteration limit, set to 100 steps, is reached without satisfying the constraints. For practical efficiency this failure can sometimes be detected early when gradient updates no longer change the loss and the constraints are not satisfied. We use Adam (*Kingma and Ba*, 2014) for optimization with a learning rate of 0.01.

5.4.3 Multi-object completion

CLASP stores an assignment of each $q_{contact}$ to a particular object o_j for each full-scene particle i. When a measurement contains a new contact $q_{contact}$, it is assigned to a specific object for each sampled particle i as follows. First, the output of our shape completer is a finite-sized voxelgrid, typically smaller than the full scene. The new $q_{contact}$ cannot be assigned to any object j where $\mathcal{R}(q_{contact})$ lies entirely outside the output region of the decoder $f_{dec}(z_j)$. Next, for each remaining j, a projection is attempted for each z^i_j to satisfy the new $q_{contact}$. If all attempts fail, we assume this new $q_{contact}$ was not caused by object j. For each full-scene particle i, a specific assignment of $q_{contact}$ is randomly and uniformly selected from the remaining objects j that could possibly explain the contact.

5.5 Experiments

We evaluated scenarios of different objects to determine if CLASP improves the estimate of the scene using robot contact information. We tested ablations of CLASP to evaluate the importance of the latent prior and constraint satisfaction. We also tested alternative approaches to CLASP that did not rely on projection. Finally, we compared CLASP on two different network architectures and trained on multiple datasets. We trained separate instances of PSSNet (*Saund and Berenson*, 2020b) on Axis-Aligned Boxes (AAB), YCB (*Calli et al.*, 2017), and ShapeNet mugs (*Chang et al.*, 2015) (training details in Section A.4.1).

5.5.1 Robot Contacts

Simulation: To generate contact measurements we moved the right arm of a robot composed of two Kuka iiwa arms with Robotiq 3-finger grippers. We generated scenes by manually placing simulated objects from AAB, YCB, or ShapeNet on a virtual table at about camera height. The known voxels were passed to our trained PSSNet to generate a set of possible completions.

We generated robot motions to gather information by moving near and sometimes contacting the objects using the procedure described in the appendix A.4.2. The first motion typically sweeps known free space rather than making contact. The second or third motion intentionally makes contact with the object.

Live Robot: The physical kinematics of our robot matched the simulated robot. A Kinect depth camera mounted at the "head" position generated the RGBD images. A calibrated motion capture system provided transforms between the Kinect and robot frames. We segmented the RGB image using the CSAIL semantic-segmentation-pytorch library (*Zhou et al.*, 2018) which we retrained on YCB objects. Each segmentation was converted into a voxelgrid and fed to PSSNet as in simulation to generate sample worlds. The same procedure was used to generate robot motions as in simulation.

On the live robot, contact was determined at each configuration by checking if the measured external torque exceeded a threshold of $2Nm$ per joint. This threshold was large enough to avoid generating false positives while remaining sensitive enough to detect contact with secured objects.

5.5.2 Scenes

We tested four scenes in simulation and two on the live robot. Each scene consisted of a single object secured to the table in front of the robot. We also tested a scene of multiple YCB objects. Contacts occurred with occluded sections of the objects, with examples shown in Fig. 5.5. In both simulation and the live robot, table occupancy

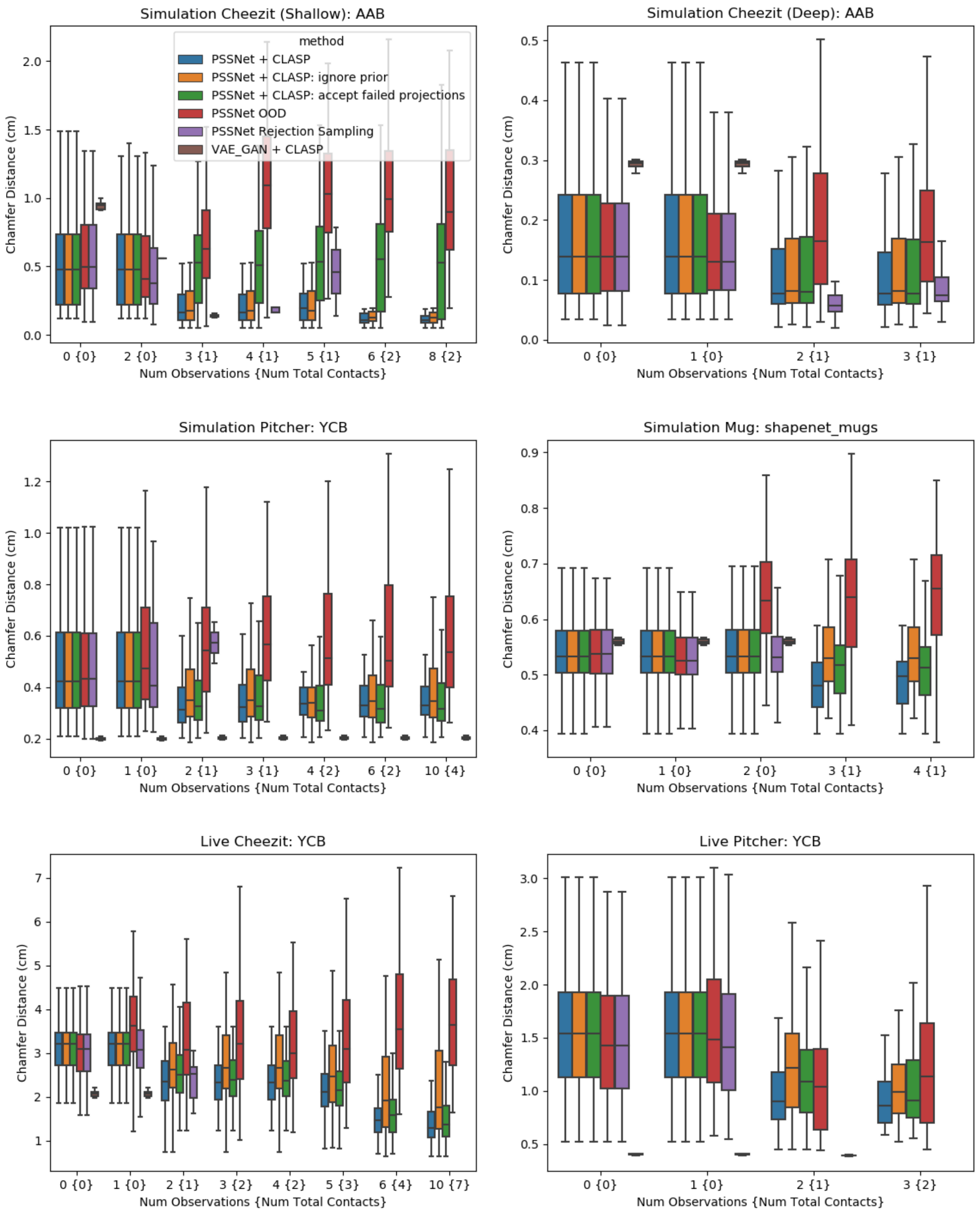


Figure 5.3: Boxplots showing the Chamfer Distance from sampled particles to ground truth. The mean, middle quartiles (boxed colored region), and outer quartiles excluding outliers are shown. Rejection Sampling and VAE_GAN occasionally produced no valid shapes, in which case no box is displayed.

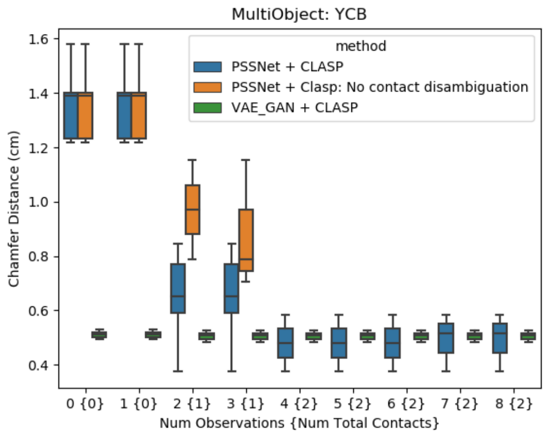


Figure 5.4: Boxplot results for the multiobject scene. PSSNet + CLASP: No contact disambiguation fails to project any samples for observations 4 and beyond, so there is not corresponding box.

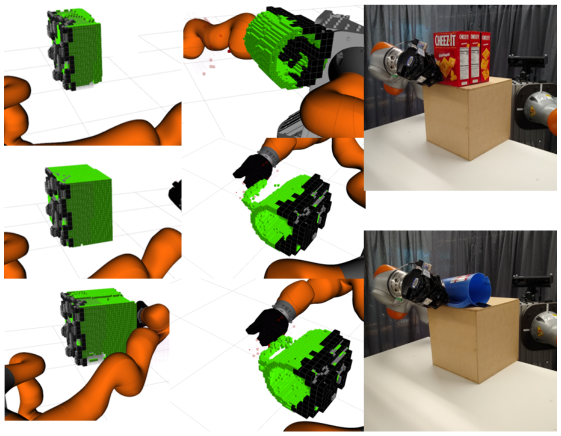


Figure 5.5: The Deep Cheezit (left), Mug (Middle), Live Cheezit and Live Pitcher (Right) scenes. The occupied (black) and known free (not shown) voxels from vision with contact (transparent red) and robot free (not shown) voxels are all used by CLASP to generate completed shapes (green).

was not considered when evaluating the quality of the completions.

Simulated Scenes: The first pair of scenarios used a single YCB Cheezit box (Shallow) and a stack of three Cheezit boxes (Deep). These setups generated similar depth images but different ground truth shapes. Both used networks trained on the AAB dataset. The Simulated Pitcher from YCB was positioned with the handle occluded from view and used networks trained on the full YCB dataset. The Simulated Mug from ShapeNet also had the handle occluded from view and used networks trained on all mugs in ShapeNet. The handles on these objects were localized through contact.

Live Scenes: The Live Cheezit also consisted of a stack of three boxes, and again the Live YCB Pitcher had the handle occluded. Both scenes used networks trained on the full YCB dataset. The Cheezit boxes were attached together and the pitcher was taped to prevent motion during contact. Simulated objects were manually aligned to the live scene to approximate the ground truth of live objects and were used for evaluation.

5.5.3 Baselines

We compared our proposed method to the following alternatives. In Rejection Sampling we sampled latent space vectors from the distribution predicted by the encoder, then decoded these into 3D objects and rejected any samples not satisfying the contact or free space constraints. For OOD (Direct Out-Of-Distribution Prediction), we augmented the RGBD known free and occupied voxels with the contact information. While other methods did not have access to the true contact point, we allowed this method this advantage and added the true contact point directly to the known occupied voxels from the depth image. Finally, while other approaches used the PSSNet shape completion network, we tested using a VAE_GAN (*Wu et al.*, 2016) network. This network tends to produce better average but less diverse samples.

We also tested ablations of our method. CLASP: IGNORE PRIOR tested removing the loss term L_{prior}. CLASP: ACCEPT FAILED PROJECTIONS tested accepting all projections, even those that do not satisfy the contact constraints. To test our contact assignment in the multi-object case (Section 5.4.3), CLASP: NO CONTACT DISAMBIGUATION determined if a projection of latent z_j could satisfy each new $q_{contact}$ as in CLASP, then assigned each $q_{contact}$ to all feasible objects j. This resulted in scenes explaining a single $q_{contact}$ with multiple objects.

100 particles were sampled in each method, with the threshold of L_{free} set at $\delta = 0.4$ and the weighting of L_{prior} set at $\alpha = 0.01$.

5.5.4 Results

Results for the various single-object scenes were tested on all baselines using PSSNet trained with the appropriate dataset. Fig. 5.3 compares the Chamfer Distance (CD) (*Barrow et al.*, 1977) of each accepted sample to the ground truth. We consider a different analysis in Section A.4.3. We find that REJECTION SAMPLING often performs well during the first few observations with zero or one contact. However, REJECTION SAMPLING soon fails to return any valid samples with two or more contacts. We see that OOD produces completions that are typically much worse than the original completions from only vision. The initial estimate (observation 0) from VAE_GAN are hit-or-miss. All networks saw the YCB Pitcher during training, and VAE_GAN recalls the pitcher more accurately than PSSNet during testing to the point where contacts are unnecessary. However, the recall of VAE_GAN in the other ambiguous scenarios is worse than PSSNet and projection to the contact constraints often fails, leaving no sampled shapes.

Considering ablations of CLASP, ACCEPT FAILED PROJECTIONS performed as well initially (when no projections fail) and significantly worse as the number of observations increases. Ignoring the latent prior during projection also performs worse and occasionally produces shapes that qualitatively look less like objects compared to completions from other methods.

Across all scenes, CLASP performed similarly to the best of all other methods with 0 or 1 contacts and the best with multiple contacts. CLASP successfully used the robot contact information in all scenarios to reduce the CD between the predicted and ground truth shapes in all scenarios. Robot measurements with a contact typically caused a larger reduction in CD than measurements with only freespace information. Numerically, the CD reduced most in the Cheezit scenarios, with a reduction of the mean from $0.5cm$ to $0.1cm$ for the Shallow and from $0.14cm$ to $0.08cm$ for the Deep. The CD reduction in the pitcher and mug scenarios was significantly smaller, as the general shape of the pitcher and mug could be predicted from the image. The prediction of the occluded handle was improved with contact. The trend of improvement in the live scenes matched the simulation. However, the numeric error of the live scenes was much larger, perhaps caused by imperfect transfer of the learned shape completer from training in simulation to prediction on live Kinect data as well as imperfect alignment of the robot frame to the Kinect frame.

In the multi-object scene (Fig. 5.4) the VAE_GAN method achieves a better completion from the RGBD data, but our proposed method produces better samples after 2 contacts. Our proposed contact assignment (Section 5.4.3) outperforms naively satisfying the contacts whenever possible.

5.6 Discussion and Conclusion

While we model CLASP using a particle filter and would like to have the Bayesian estimate of the scene given all observations, we acknowledge many non-Bayesian approximations. Particle filters approximate Bayes filters, but the 100 particles we sample may not be a sufficient coverage of the latent shape space. CLASP projects samples, which does not preserve Bayesian estimates.

While our shape network uses voxelgrids, implicit representations have recently become popular and offer advantages worth considering. Currently shape completion networks produce the most visually pleasing results when trained on a single object, visually decent results when trained on a single class of objects, and poor results when trained on large diverse shape datasets. In order to be practically applicable to robots, shape completion must handle a wide variety of objects. Shape completion is rarely the end goal, but rather a tool robots can use to aid in tasks. Choices of correct metrics and refinements to CLASP ultimately depend on the specific downstream application.

We demonstrated a method for estimating shape completions initialized with purely RGBD visual data, then updated from observations of a robot arm moving through unknown regions and sensing contact. We stored the belief of the scene as a particle filter of latent vectors from a shape completion network and used CLASP to enforce shape consistency with the robot observations. Most importantly, we showed that CLASP improves the estimate of object shape using these contact observations. Our results further showed that CLASP performs better than ablations of CLASP and alternative methods. We hope CLASP will be used within a larger robotics framework where reasoning over environment uncertainty based on shape priors aids in accomplishing larger goals.

CHAPTER VI

Conclusion

This dissertation presented an approach for planning with contact feedback. The key challenges addressed were modeling the belief over environment given the contact measurements, and planing under the uncertainty of these beliefs. Exploring the belief representations led to the Collision Hypothesis Set (CHS) model (Chapter II) and CLASP (Chapter V). Planning using these beliefs required the new formulation of the Blindfolded Traveler's Problem (Chapter III).

This conclusion summarizes and connects the contributions in Section 6.1. Section 6.2 discusses promising directions for future work to remove some assumptions and expand the applicability of this work.

6.1 Summary of Contributions

6.1.1 Contact Information

This thesis explored a very specific form of contact information which we have named Collision Hypothesis Sets. We assumed a contact generates a set of points on the surface of the robot containing at least one point that is an obstacle. This set of points could be as large as the entire robot surface, but frequently is much smaller because (A) the robot has already swept out known freespace, and (B) the torque of a joint (e.g. joint 5) has exceeded the non-contact threshold, so the contact occurs downstream (e.g. not links 1-4).

This contact model is seldom used in other academics works, and this thesis is a rare work that delves into the implications. Other works tend to use more informative contact models. Some assume the robot is endowed with contact sensitive skin. Others assume the contact location has be recovered or approximately localized from joint torques or force-torque sensors. Finally, some works use the dynamics of the robot after collision to infer contact location.

With all of these competing models, it is worth reiterating why I have spent the effort to explore Collision Hypothesis Sets. Conceptually, it appears other models can provide more information. However, I am motivated by the practicality of the approaches on real robots. Contact sensitive skin barely exists in laboratory robots, and is custom, rare, and expensive on commercial robots. The CHS model is applicable on a much wider set of robots for much less cost. Precise torque sensing can work quite well in limited applications, however our Victor robot uses two Kuka iiwa arms which have some of the most precise torque sensing available, yet still calibration is frequently required and not accurate for the entire robot workspace. The CHS model is much more robust to slight calibration errors. Finally, in order to use dynamic motion to infer contact location, the robot must be moving quickly and continue moving significantly after contact. This work explores sensing contact and stopping motion at the limits of reliable sensing. There is too much noise in the motion and joint torques to reliable use a method other than CHS.

6.1.2 Belief Representations from Contact Information

We have explored three belief representations of environment occupancy. The first constructs an occupancy belief using only the CHSs. The second uses the Manifold Particle Filter (MPF) assuming the shape but not pose of the object is known. The final, CLASP, uses a neural network to provide a prior over shapes.

The CHS belief is optimistic, assuming almost the minimal occupancy required to explain all contacts. The benefits of the CHS belief are the lack of further assumptions on the shape of objects in the world and ease of implementation. The CHS model can accommodate any shape and does not require any dataset. Depending on the application, this optimism may be acceptable. However, often knowledge of shapes in the world can be useful, which led us to develop other belief representations.

The Manifold Particle Filter was developed prior to this thesis, and the contribution here is the adaptation of the MPF to our contact model and planning framework. The MPF requires knowledge of the shapes of objects in the world. If provided with the true shapes, MPF can generate beliefs which closely match the true world. However, if provided with incorrect object shapes the MPF can become hopelessly stuck, attempting to match incorrect geometry to the world.

The final belief explored is the Constrained Latent Shape Projection (CLASP), using the Plausible Shape Sampling Network (PSSNet). This approaches uses a neural network to learn a prior over shapes, and a projection method to enforce that sampled shapes are consistent with the observed contact information. This belief model requires more machinery, but is capable of generating a much larger space of shapes to match the environment.

These representations can be combined into a single belief. As shown in Chapter III, we can construct a Mixture-of-Experts (MoE) belief that is a combination of other belief representations. As observations look more or less likely under each "expert", the MoE model adjusts the weight of that representation.

6.1.3 Planning

Planning motivates this model of beliefs over world occupancy. In our problem setup the robot's primary objective is to reach a goal. Since the world occupancy is uncertain, the robot may bump into objects. The previous belief representations help the robot understand the true world and navigate to the goal. In the work presented here, the goal was always to reach a specific configuration or any configuration from a set of goal configurations.

The first approaches attempted in Chapter II used interleaving planning and control. A greedy local controller attempted to move the robot towards the goal. If insufficient progress was being made, the robot invoked a global planner. This planner used approaches inspired by RRT to find the lowest probability collision path to the goal within a time limit. Ultimately these planning approaches of Chapter II were successful but slow. The planner was not able to reuse work between queries. Furthermore, each stage of growing the search tree in RRT requires a binary decision of collision, however since the true occupancy was uncertain only collision probabilities are available. Finally, the collision probability of a path is not Markovian in the edges. To calculate the probability of collision for a path, it is not sufficient to simply know the probability of collision for each edge.

Stepping back to the fundamentals of the planning problem led to the Blindfolded Traveler's Problem of Chapter III. By restricting the robot motion to a roadmap, a new problem class emerged. The Blindfolded Traveler's Problem is a problem of reaching a goal node when edge validity is only discoverable by attempted traversal. This mimics the task of our robot, which only learns which motions are allowed by attempting the motion. The robot will either collide or move freely. However, the validity of edges is correlated, so a collision (or successful motion) provides implicit information about many other edges. This coupling is accomplished through the belief representations above.

6.2 Future Work and Extensions

The development of the belief representations and planning approaches required formal assumptions, but there is an variety of promising future work to remove some assumptions and extend the capabilities of this contact sensing model and planning.

6.2.1 Movable Objects

This work assumed the environment was static, and thus objects did not move even when contacted. This assumption was made primarily to focus on other aspects of the planning with contact sensing problem, specifically the object priors and BTP formulation. However, if many real scenarios bumping an object can cause it to move.

Fortunately, object motion can be easily incorporated into our belief models. Both the MPF and CLASP belief models store a set of sampled objects as particles. A standard particle filter has both a motion model and an observation model. This thesis focused on the observation model, updating the belief over objects from contact measurements. The BTP of Chapter III assumed a static motion model. However, a static motion model is not required. Using priors over how objects move, the belief could estimate the updated object at each time. Presumably in many cases the object would remain stationary. Yet when the object is bumped, possible updates could be sampled and applied. Furthermore, object could be tracked with the camera, using visual feature to detect motion and estimate SE(3) transforms from frame to frame. The primary challenge of allowing movable object will be developing a motion model that is sufficiently accurate for the wide variety of shapes generated by the beliefs.

6.2.2 Leaving the Graph

BTP from Chapter III restricts all robot motion to the initial roadmap, limiting the action space. For freespace motion, this restriction is minor, as reasonably dense roadmaps contain paths only slightly worse than the optimal unconstrained path. The graph restriction is much more apparent after a contact is made.

Consider attempting to find a light switch in the dark. Likely you will wave your arm until you contact the wall, and then *stay in contact* with the wall while searching for the switch. You use the prior knowledge that the switch will be on the surface of the wall, and that exploring the contact manifold between you and the wall is a good strategy. In BTP, the robot's only option after contact is to follow motions prescribed by the roadmap. This roadmap was generated without knowledge of the

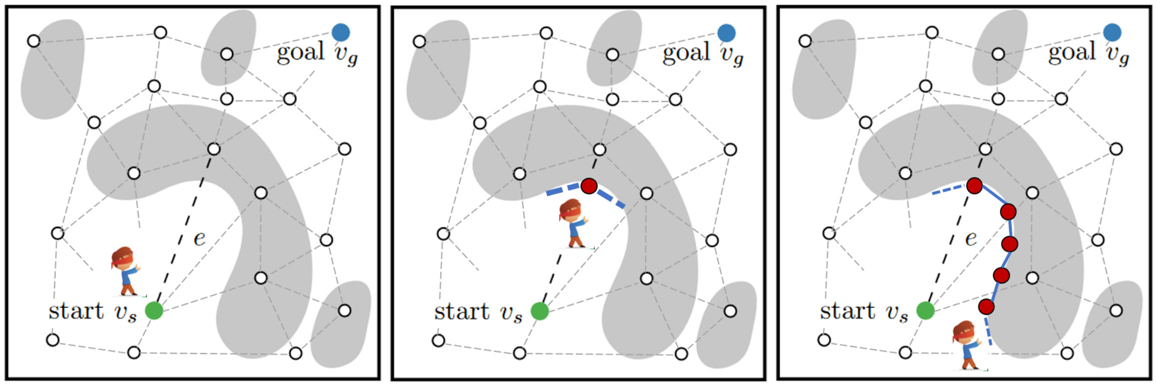


Figure 6.1: Leaving the Graph of the Blindfolded Traveler's Problem.
Left: The initial BTP graph. Motion is limited to the graph edges.
Middle: After collision a node could be created at the current configuration (red circle) with new edges (blue dashed lines) to traverse.
Right: The agent could then closely follow the surface of obstacles.

wall, and therefore will not (i.e. with probability 0) have any edges traversing the contact manifold.

A solution is to traverse a dynamic implicit graph, rather than a static precomputed roadmap. After contact, the robot can consider new motions along edges constructed conditioned on gathered information. For example, new edges can explicitly traverse tangential motions to a wall, and following an edge may require invoking a controller to maintain contact with the wall, visualized in Fig. 6.1. Such a solution may look like the "interleaving planning and control" approach of Chapter II 2.5.

A opportunity for research could define rules for constructing this implicit graph. Adding nodes arbitrarily could create endless loops, where the planner generates additional nodes ad infinitum, continuing motion forever without reaching the goal. Unlike the original BTP, a solution could exist but never be found. Further work could identify constraints on the node and edge generation to maintain some of the guarantees of BTP.

A related strategy I have explored is Selective Densification (*Saund and Berenson*, 2020a). In Selective Densification search is performed on a layered graph. The top layer is a sparse roadmap, with long edges traversing large distances leading to fast search in freespace but no solution traversing narrow passages. Each lower layer contains all nodes of the previous layer as well as additional nodes and edges. 0-cost vertical edges traverse layers and do not correspond to any robot motion. A heuristic biases search towards sparser layers to attempt to achieve faster planning. Figure 6.2 shows an example planning problem and visualization of the layered graph.

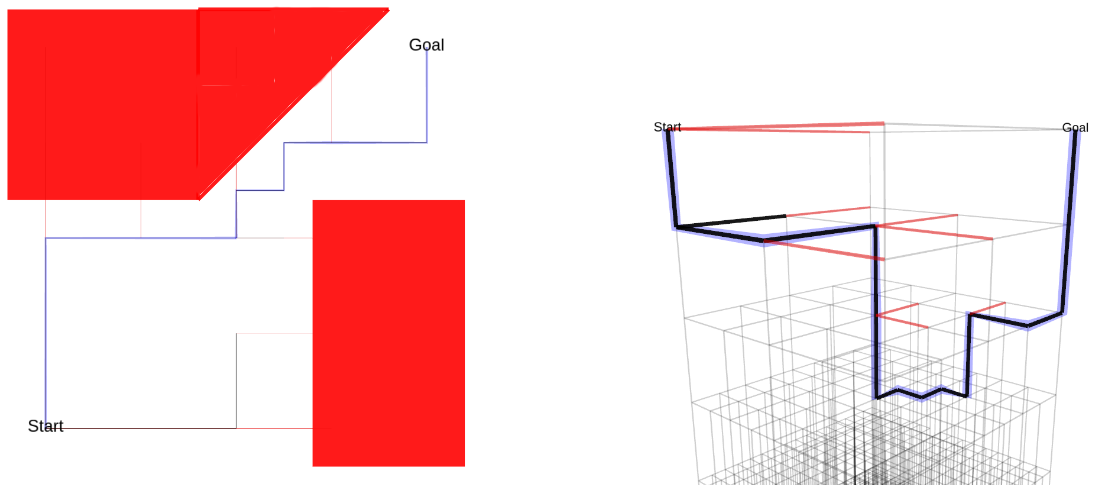


Figure 6.2: A 2D search problem solved using Selective Densification with evaluated edges shown in black (valid) and red (invalid) and the final path shown in blue. Left: 2D view with red obstacles. Right: View of Layered Graph with unevaluated edges shown in light grey

In Selective Densification, the layered graph is explicitly constructed using a sampling procedure such as a densifying grid (as in Fig. 6.2), halton sampling, or random sampling. Integrating Selective Densification into the contact planning problems discussed in this thesis will require a different sampling procedure that is dependent on the observations. One could design a sampling procedure to densify as shown in Fig. 6.1. The search procedure could use a similar Selective Densification Heuristic, biasing planning away from dense layers representing contact with objects, but not preventing such motion in cases where contact signficantly improves the plan.

6.2.3 Context-dependent Goals

The Blindfolded Traveler's Problem from Chapter III assumes the task is solved once the robot reach any goal from a set of goal configurations. The assumption that the goal configurations are known is odd, since the BTP explicitly handles unknown environments. For many tasks the goal will be dependent on the specific unknown environment, requiring the robot to localize some portions of the environment to even know how to complete a task.

I have implemented a cursory general approach. A goal generator function g is provided instead of an explicit goal set. g applied to a specific world occupancy ϕ provides a goal region G in configuration space $\mathcal{C}$.

$$g(\phi) = G \subset \mathcal{C} \tag{6.1}$$

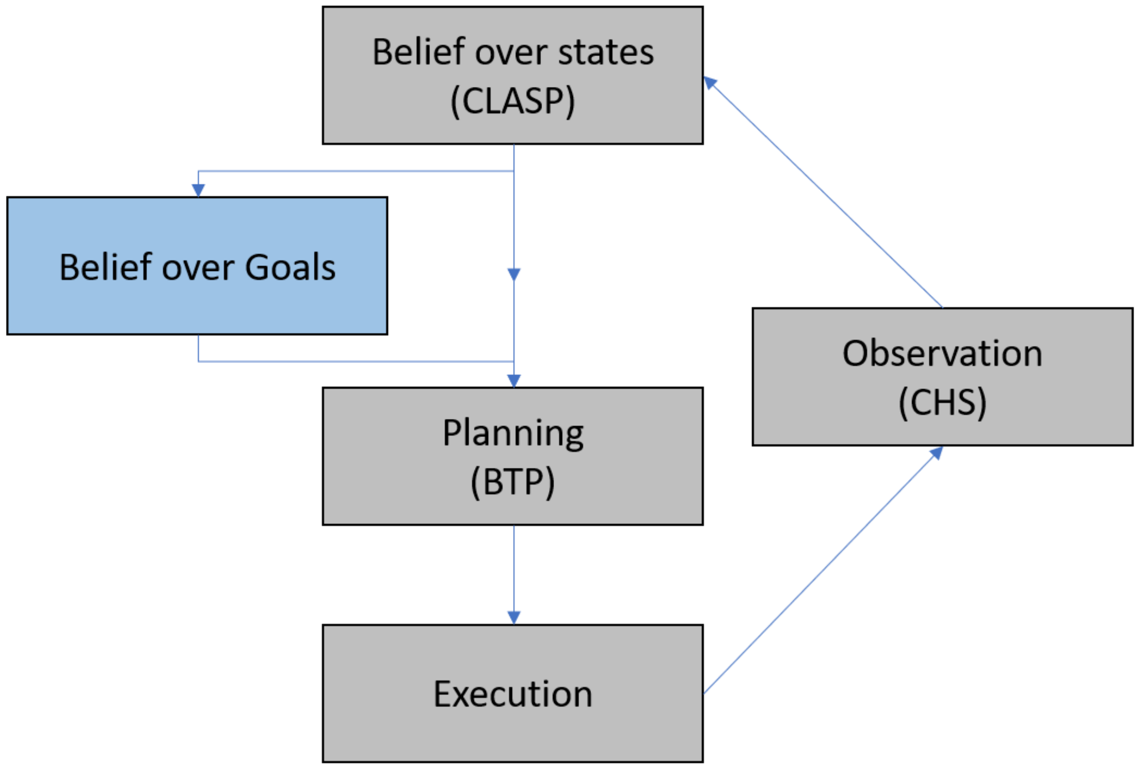


Figure 6.3: Architecture for BTP with goals dependent on the belief over states

The updated pipeline in Fig. 6.3 computes a belief over world occupancy as a set of sampled worlds as before, but then computes the goal set for each sampled world. The updated planner must reason over unknown worlds as they connect to unknown goal sets. Below I discuss areas for further refinement in this method.

Goal Generation: The generic function g leaves the problem both general but open-ended. For a picking task, g might generate the set of configurations for which closing the gripper will grasp the object. For a placing task, g might generate the set of configurations such that the held objects rests stably on the drop off surface. These specific examples have ample literature to guide a user on how to generate the goal set for a specific world instance. However, a more complete analysis would consider a wider variety of tasks and associated goal generators.

When to declare success: Without a known goal set, an algorithm must decide when to terminate. A naive approach could declare success whenever $P(q \in g(\phi)|\Phi)$ is sufficiently large, for example by requiring that q is in the goal set for every world ϕ sampled from the belief. The true success of an algorithm must be judged based on the goal set of the true but unknown underlying world ϕ_{true}.

One strict approach is to allow an algorithm to determine when to terminate, then assign an additional high cost (or mark as failure) if $q \notin g(\phi_{true})$. An alternative is to further augment BTP and allow the robot to query (for some cost) if $q \in g(\phi_{true})$. A physical analogy could be executing a grasp and observing if the target object becomes held by the gripper. If such a check fails, then the robot has accumulated some cost, but learns the task is not complete, can update the belief over worlds (i.e. removing worlds where the task is complete), and continue planning.

Planning: Without a specific goal set, the planning portion of the BTP becomes yet more coupled with the belief over worlds. The optimal solution would consider the full POMDP, taking into account the current uncertainty, the potential observations of each action, how each of these observations would update the belief over worlds, and finally how updating these worlds would update the belief over goals. All of these considerations would be unrolled, considering all possibilities for all actions until an optimal solution is found. This brute-force process is clearly intractable.

Instead, I have examined a simple heuristic. Consider the set of configurations that could possibly be goals under the current belief: $\mathcal{G} = \cup_{\phi \in \Phi} g(\phi)$. While the robot configuration satisfies $q \notin \mathcal{G}$, clearly the task is not solved. Here I use an exploitation heuristic and use the same planning procedure in BTP (Chapter III 3.5) to plan to any configuration in $\mathcal{G}$. This heuristic quickly reaches a possible goal configuration, yet ignores actions which could generate value information to refine $\mathcal{G}$. As the robot attempts to reach $\mathcal{G}$, observations will update the belief over worlds which may in turn alter $\mathcal{G}$.

Once the robot has entered some $q \in \mathcal{G}$ but is not sufficiently confident it has reach a goal of the true underlying world $g(\phi_{true})$, I use an exploration heuristic. The robot considers a set of possible next actions. For each action the robot calculated the information gain following the methods of (*Saund et al.*, 2017). In summary, the robot considers the different observations it might receive for each world $\phi \in \Phi$. The most informative action is the one which will likely lead to an observation that distinguishes between worlds in the belief. Note that there is an internal heuristic embedded in this method as well, as the information gain only considers distinguishing between worlds and not goals. It could be that two different worlds have the same goal set. This IG heuristic would favor action that distinguish these worlds, even though the distinction is unimportant for the overall planning problem.

In practice these two heuristics appear to produce desirable behavior on initial problems. Further work is needed to consider alternative approaches, and explore the benefits and limitations of these heuristics.

6.3 Final Remarks

This thesis has explored planning with contact feedback, examining the key issues of belief representation and belief-space planning. My development of this thesis brought about the Collision Hypothesis Set (CHS) observation model, the Blindfolded Traveler's Problem (BTP), the Plausible Shape Sampling Network (PSSNet), and the Constrained Latent Shape Projection (CLASP). These components provide a framework for robot planning in unknown environments.

As with many research projects, much of my time was spent on work that ultimately did not make it in to this document. Some projects manifested as software packages or other research papers, but most of my effort went into exploring ideas or tangents that ultimately proved to be unsuccessful. Do not despair over such result, for they make the successful discoveries more sweet.

I hope my framework is not the final word on these topics, but instead that it opens a door to greater ideas. Through the explanations of the successes and limitations of my work, I hope this thesis helps guide your path to the fruitful areas of undiscovered knowledge.

APPENDICES

APPENDIX A

A.1 Overview of Robotic Hardware

The main robot used for experiments in this thesis is Victor (Figure A.1). Victor consists of two Kuka iiwa robotic arms (KR-800) mounted horizontally. For all experiments these arms were set to "Impedance Mode", and used their joint torque sensing within an impedance controller internal to the Kuka software. On top of the Kuka stack we implemented controllers and planners to direct Victor. Joint torque measurements were published at 100Hz and used to stop the robot when collision was detected.

Each arm has a Robotiq three-finger end effector for manipulating objects. While this work uses the grippers to motivate the applications of this thesis, grasping itself is not core to the algorithms.

Victor sees the world through a Kinect RGBD sensor at it's head. Additional software components segment the images into distinct objects for further reasoning about the world. All components communicate using the Robotic Operating System (ROS).

A.2 Analysis of BTP

A.2.1 Mapping the BTP to a POMDP

We map BTP problem to a Partially Observable Markov Decision Process (POMDP) specified by the following tuple $\langle \mathcal{S}, \mathcal{A}, T, C, \mathcal{O}, Z \rangle$ which we define as follows.

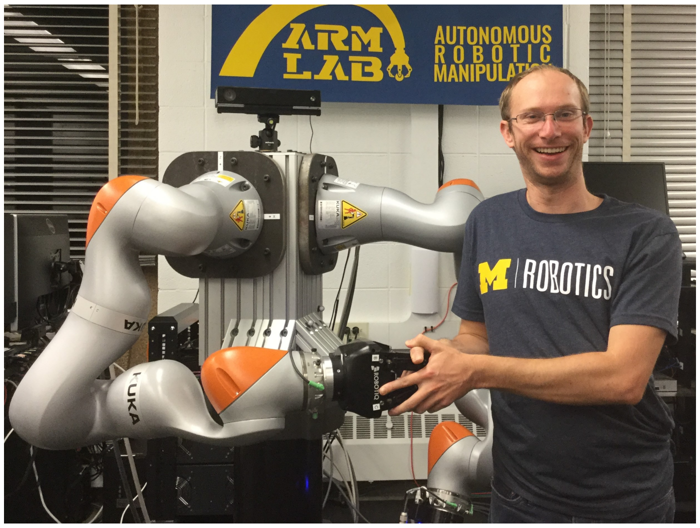


Figure A.1: Victor shaking my hand. Victor was the main robot used in experiments.

The state $s \in \mathcal{S}$ is the tuple $s = (v, x, \eta)$ where $v \in \mathcal{V}$ is the current location of the traveler on the graph $\mathcal{G}$, x is the binary vector of edge validities and η is a vector of edge blockages. The state is partially observable, i.e. v is observable but the rest is latent.

Given state $s \in \mathcal{S}$, the action $a \in \mathcal{A}(s)$ is any edge $e \in \mathcal{E}$ that can be traversed, i.e., whose parent is v. Let the result of the attempt be $(v', c) = \Gamma(v, e, x, \eta)$. The transition function $T(s, a, s')$ is deterministic, i.e. $s' = (v', x, \eta)$. Similarly, the one step cost is $C(s, a) = c$. The observation $o \in \mathcal{O}$ is the tuple $o = (x(e), \eta(e))$. Hence the observation model $Z(s', a, o)$ is deterministic.

Since the state is partially observable, the POMDP is viewed as a MDP over belief b. A POMDP policy $\pi(b)$ maps b to actions. The optimal policy π^* accumulates the minimum cost in expectation. The Q-value of action a in a belief state is the expected total cost of taking a and subsequently following π^*, i.e.

$$Q(b, a) = \mathbb{E}_{s \sim b}\left[C(s, a)\right] + \mathbb{E}_{b' \sim P(.|b,a)}\left[V^{\pi^*}(b')\right]. \tag{A.1}$$

A.2.2 Computational Complexity

In BTP, the belief b is over a continuous space $\mathcal{S}$ due to blockages η, i.e. the exact belief is infinite dimensional. This necessitates approximation based approaches that rely on non-parametric sample-based belief representations. For the proofs in this section we consider a discrete/simplified BTP with discrete b by fixing $\eta(e) = 1$. Furthermore, we examine the BTP decision problem instead of the optimally problem.

We follow an analysis parallel to *Lim et al.* (2017) to show that the BTP decision problem is NP-complete by showing it is both in NP and NP-Hard.

We first prove that the BTP decision problem is in NP. For this result we consider an explicit description of the input $\mathcal{P}$, that is $\mathcal{P}$ specifies probability of each possible world. Note that $\mathcal{P}$ could be exponentially larger than $|\mathcal{E}|$. In this case BTP would still be in NP, though $\mathcal{P}$ (part of the input) would be so large as to make this claim of limited use.

We also prove that BTP is NP-hard by reduction from the Optimal Decision Tree (ODT) problem. The ODT problem is as follows. We have a finite set of hypotheses $\mathcal{H} = (h_1, h_2, \dots, h_n)$ and a finite set of tests $\mathcal{T} = (t_1, t_2, \dots, t_m)$. A test t_i leads to an outcome $o_i \in \{0, 1\}$ depending on the latent hypothesis $h^* \in \mathcal{H}$. The objective is to find a policy that identifies h^* with the least number of tests when h^* is uniformly distributed. The policy is a binary decision tree where nodes are tests, edges branch on outcomes and the terminal nodes stores the latent object $h^* \in \mathcal{H}$. The decision version of the problem, which asks if a policy with expected cost of less than or equal to w is NP-complete (*Laurent and Rivest*, 1976).

Theorem A.1. *We define the decision version of Blindfolded Traveler Problem as the question of whether there is a policy with expected cost less than or equal w. The decision version of discrete/simplified BTP is NP-complete.*

Proof. The solution of BTP can be represented as a policy tree. Note that nodes and edges in this policy tree are distinct from nodes and edges in the graph $\mathcal{G}$ of the BTP. Nodes of this policy tree represent testing an unevaluated edge in $\mathcal{G}$. A node in the policy tree may even represent traversing several known edges in $\mathcal{G}$ to reach the unknown edge in $\mathcal{G}$. Each edge of the policy tree is an observation o received upon performing an edge. A BTP is solved by traversing the policy tree till the leaf node is reached, i.e. evaluating unknown edges, receiveing observations until the goal is reached.

The optimal policy tree is polynomial size in the input of BTP. Consider that each edge in the policy tree corresponds to an action (or actions) in the BTP that will determine the validity of one edge in $\mathcal{G}$, thus the policy tree can be at most $|\mathcal{E}|$ deep. Furthermore, there can be at most one unique path through the policy tree for each hypothesis world in $\mathcal{P}$. Since we assume each hypothesis world is explicitly represented in $\mathcal{P}$, the optimal policy tree is polynomial in $|\mathcal{G}|$ and $|\mathcal{P}|$.

Finally, computing the expected cost of a policy is simply a weighted sum for all paths through the policy tree. Hence the BTP decision problem is in NP.

We now show that ODT is polynomial time reducible to BTP and thus BTP is NP-hard. Given an instance of ODT$(\mathcal{H}, \mathcal{T})$, we consider an instance of BTP $\langle \mathcal{G}, \mathcal{P}, v_s, v_g \rangle$

as follows. Consider the BTP problem shown in Fig. A.2. The cluster of edges $\{e_1, \ldots, e_m\}$ correspond to the tests $\{t_1, \ldots, t_m\}$. Note again that the blockages for all tests is fixed at $\eta(e) = 1$. An agent attempting to traverse the edge e_j will either be successful and reach the vertex v_j, or unsuccessful and the agent will return back to v_s. The cluster of edges $\{e_{m+1}, \ldots, e_{m+n}\}$ has only one valid edge correspond to identifying the correct hypothesis from $(h_1, h_2, \ldots, h_n)$. The weights of the left cluster of edges $\{e_1, \ldots, e_m\}$ is 1 and the right cluster of edges is $2m$.

We set up the prior $\mathcal{P}$ to be uniform over a set of candidate vectors x_i, each of which corresponds to a h_i. For the latent hypothesis h_i, we set the edge validities $x(e_j) = o_j$ for $j = \{1, \ldots, m\}$, i.e. the outcome of the tests for h_i. For the other cluster, we set $x(e_{m+i}) = 1$ and all other edges to 0, i.e., $x(e_j) = 0$ for $j = \{m, \ldots, m+n\}, j \neq i$. We now argue that expected cost of ODT instance is less than or equal to some value w iff cost of BTP instance is less than or equal to $2w + 2m$.

First, if the cost of the ODT is $\leq w$ then the agent can traverse the left cluster using the policy tree of ODT and identify the correct hypothesis h^* with cost $\leq 2w$. The agent then goes to v_g using the valid edge incurring $2m$. Hence the total cost of the BTP is $\leq 2w + 2m$.

Next, we prove the converse that if the cost of the BTP is $\leq 2w + 2m$, then the cost of the ODT is $\leq w$. Note that $w > m$ is vacuous because ODT is clearly solved by at worst evaluating all tests, which would incur cost m. Thus we consider $w \leq m$ which implies the cost of the BTP is $\leq 4m$. First consider that if an edge to v_g is attempted before identifying the correct hypothesis, there will be at least two equally likely paths with cost $2m$ and so the expected cost of any policy that tries to go directly to the goal is $\geq 4m$. Hence the agent will try to identify the true hypothesis before going to the target. If the agent solves the BTP by identifying the correct hypothesis with cost $\leq 2w + 2m$ then it also has a policy to solve the ODT with cost w.

Thus ODT is reducible to BTP in polynomial time, and since ODT is known to be NP-hard then BTP is also NP-hard. Since we also showed BTP is in NP, BTP is therefore NP-complete. □

Note that if $\mathcal{P}$ is not represented explicitly (e.g. not by a matrix of size $|\mathcal{E}|$ by the number of hypothesis worlds), but with factored representations, then the problem may no longer be in NP. Also if we further consider the location of the contact (η), the size of the hypothesis space is now continuous and this analysis no longer holds.

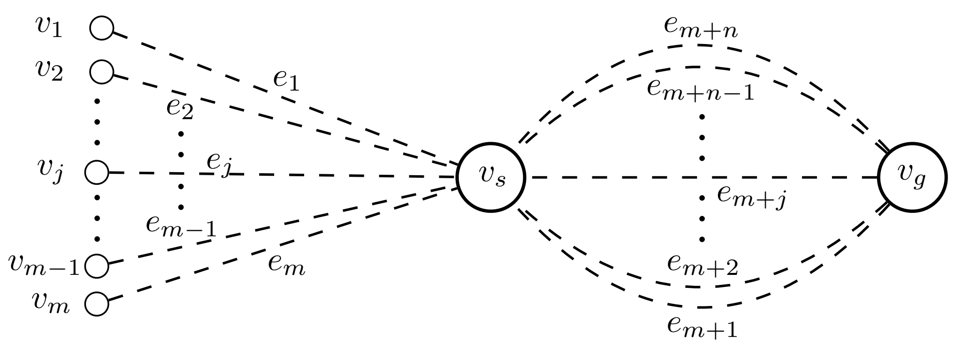


Figure A.2: Reduction from Optimal Decision Tree problem

A.2.3 Relation to the Bayesian Canadian Traveler's Problem

The BTP is closely related to the Canadian Traveler's Problem (CTP) (*Papadimitriou and Yannakakis*, 1991a). In graph search an agent executes a polity to reach a goal with the minimum expected cost. Consider the k-lookahead graph search problem, where an agent only observes the true validity of edges within k steps of its location. The Shortest Path Problem over known graphs in an instance of ∞-lookahead. The CTP is a 1-lookahead instance. For $k \geq 1$ an agent knows the state of adjacent edges and therefore will never attempt an invalid edge. In BTP, with $k = 0$, an agent might attempt invalid edges, which is the reason for the more complicated cost formulation.

In the original CTP $x(e)$ are independent. In the more general Bayesian CTP (BTCP) (*Lim et al.*, 2017) $x(e)$ are correlated through beliefs of underlying worlds ϕ rather than beliefs directly over x. As defined, the BTP is analogous to the Bayesian CTP.

A.3 Strategies for Solving the BTP

Since we established that BTP is a hard problem (Section A.2.2), we explore a number of efficient approximation strategies to solve it. We organize these approaches into three categories – approaches that approximate the Q-value with heuristics, approaches that use simulation to evaluate actions and approaches that plan to gather information. Note that while the latter approaches have theoretical guarantees, they come at the cost of computational complexity.

For all of these strategies, we assume that the agent is current at a vertex v_t and must decide which edge e_t from the set of outgoing edges $\mathcal{N}(v_t)$ to traverse. The history of observations is encoded in ψ_t.

A.3.1 Heuristic Estimates of Q-values

One class of approaches try to approximate optimal Q-value $Q^*(b, a)$ with an estimate $\hat{Q}(b, a)$. These approximations are motivated by different relaxations of the original problem. Since these approximations are myopic, i.e., only consider the instantaneous belief, they do not offer any performance guarantees in general. However, they are efficient to compute and perform quite well in practice.

A.3.1.1 Optimism in the Face of Uncertainty (OFU)

A common approach for planning under uncertainty is to be optimistic (*Brafman and Tennenholtz*, 2002), i.e., pick a world from the plausible set of worlds that leads to the lowest action value. The rationale is that either the assumption is correct and the agent does the best it can do, or the possibility is eliminated and the search space is reduced. This heuristic is commonly used in navigation (*Stentz*, 1997; *Koenig and Likhachev*, 2002) as well as for solving CTP (*Bnaya et al.*, 2009).

Formally, the approximation is $\hat{Q}(b, a) \approx \min_{s, b(s)>0} Q(s, a)$. An optimistic policy selects the best action $\pi^{\text{OFU}} = \underset{a}{\arg\min}\ \hat{Q}(b, a)$. Mapping this back to the BTP, the agent chooses edge e_t as follows:

$$\begin{aligned} \widehat{\mathcal{G}} &= (\mathcal{V}, \mathcal{E} \setminus \{e \mid P(x(e) = 0|\psi) = 1\}, \mathcal{W}) \\ e_t &= \left\{ e \in \mathcal{N}(v_t) \,\middle|\, e \in \textsc{ShortestPath}(\widehat{\mathcal{G}}, v_t, v_g)) \right\} \end{aligned} \tag{A.2}$$

Here $\widehat{\mathcal{G}}$ is the optimistic graph created by removing all edges that are invalid with probability 1. The agent invokes a search subroutine $\textsc{ShortestPath}(\widehat{\mathcal{G}}, v_t, v_g)$ to compute the shortest path from current vertex v_t to goal v_g. It then looks at which of the outgoing edges $\mathcal{N}(v_t)$ belongs to the shortest path and takes that.

We can bound the sub-optimality of the optimistic policy if we alter it to backtrack whenever the shortest path is in collision. Let this policy be π^{OFU2}. This results in the following iterative policy

1. At iteration i, the agent computes shortest path from start to goal on the optimistic graph, i.e. $\xi_i = \textsc{ShortestPath}(\widehat{\mathcal{G}}_i, v_s, v_g)$
2. It moves along ξ_t till it either reaches the goal or hits a blocked edge $x(e) = 0$.
3. If it hits a blocked edge, it back tracks to start v_s and repeats.

Then the following theorem is true

Theorem A.2. *Given a configuration (x, η), let w^* be the length of the shortest feasible path between v_s and v_g, and K be the number of shorter paths that are infeasible. For all such configurations, the cost of the optimistic backtracking policy π^{OFU2} is upper bounded by*

$$c(\pi^{\text{OFU2}}(x, \eta)) \leq 2Kw^* \tag{A.3}$$

Proof. The optimistic backtracking policy will attempt the shortest path from v_s on $\widehat{\mathcal{G}}$, which must be no longer than the shortest path on $\mathcal{G}$. Each attempted path therefore incurs at most a cost of $2w^*$. Since each attempt either reaches the goal or invalidates a path shorter than w^*, there will be at most K attempts. □

Interestingly, if we changed the problem to the following:

1. The agent has to reach the goal via the shortest path from start
2. The agent is allowed to backtrack for free

this problem becomes equivalent to the shortest path planning problem on expensive graphs (*Dellin and Srinivasa*, 2016). π^{OFU2} is then equivalent to LazySP (*Dellin and Srinivasa*, 2016) which has been shown to be optimal (*Mandalika et al.*, 2018).

A.3.1.2 Thompson Sampling (TS)

This is a commonly used heuristic for Bayesian Multi-armed Bandit (MAB) problem based on the idea of randomized probability matching (*Thompson*, 1933). At every decision step, it samples a model from a posterior and selects the optimal action for that model. Hence action selection probability is matched to the posterior of actions being optimal. In recent literature, Thompson Sampling has shown to be empirically successful (*Chapelle and Li*, 2011), theoretically sound (*Agrawal and Goyal*, 2013) and applicable beyond MAB to RL (*Osband et al.*, 2013).

Formally, the TS policy is $\pi^{\text{TS}} = \arg\min_a Q^*(s, a)$ where $s \sim b$. Mapping this back to BTP, the agent chooses edge e_t as follows:

$$\begin{aligned} \hat{x} \sim P(x|\psi_t),\ \widehat{\mathcal{G}} = (\mathcal{V}, \mathcal{E} \setminus \{e \mid \hat{x}(e) = 0\}, \mathcal{W}) \\ e_t = \left\{ e \in \mathcal{N}(v_t) \,\middle|\, e \in \textsc{ShortestPath}(\widehat{\mathcal{G}}, v_t, v_g)) \right\} \end{aligned} \tag{A.4}$$

Here $\widehat{\mathcal{G}}$ is the sampled valid graph from the posterior on which the agent plans the shortest path and takes a step along it. Thompson sampling usually provides a bound for MAB w.r.t Bayesian regret, i.e., the expected regret under the prior (*Russo and*

Roy, 2018). These bounds are meaningful for repeated trials on the same world, which is not the case for BTP.

A.3.1.3 Qmdp

This is one of the most commonly used heuristics for POMDPs (*Littman et al.*, 1995). It assumes that all uncertainty will disappear at the next timestep. Hence the optimal action is the one with the least expected value based on the current uncertainty.

Formally, the approximation is $\hat{Q}(b, a) \approx \mathbb{E}_{s\sim b}\left[Q^*(s, a)\right]$ and the policy is $\pi^{\text{QMDP}} = \arg\min_a \hat{Q}(b, a)$. Mapping this back to BTP, the agent chooses edge e_t as follows:

$$\begin{aligned} e_t = \underset{e\in\mathcal{N}(v_t)}{\arg\min}\ \mathbb{E}_{(x,\eta)\sim P(\cdot|\psi_t)}\left[c + w(\textsc{ShortestPath}(\mathcal{G}(x), v', v_g))\right] \\ \text{where } (v', c) = \Gamma(v_t, e, x, \eta) \text{ and } \mathcal{G}(x) = \left(\mathcal{V}, \mathcal{E} \setminus \{e \mid x(e) = 0\}, \mathcal{W}\right) \end{aligned} \tag{A.5}$$

Here we sample a set of worlds $(x, \eta) \sim P(\cdot|\psi_t)$. For each candidate edge $e \in \mathcal{N}(v_t)$, we simulate moving along the edge (which may or may not result in a success) and subsequently plan the shortest path on the revealed world.

It's straightforward to see QMDP lowerbounds the optimal value $\hat{Q}(b, a) \leq Q^*(b, a)$. There are two known drawbacks. Firstly, the policy never acts to gain information because it ignores potential observations. Secondly, and perhaps more relevant to BTP, it's susceptible to a clairvoyance trap.

A.3.1.4 Most Common Best Edge (MCBE)

This is a further relaxation of the QMDP heuristic. Note that QMDP calls $\textsc{ShortestPath}(\cdot)$ a total of kN times, where k is the degree of the graph and N is the number of samples. We can reduce this to N if the agent chooses action based on the current belief, without first simulating an action.

Formally, the policy is $\pi^{\text{MCBE}} = \arg\max_a \mathbb{E}_{s\sim b}\left[\mathbb{I}(a \in \arg\min_{a'} Q^*(s, a'))\right]$. Mapping this back to BTP, the agent chooses edge e_t as follows:

$$\begin{aligned} \mathcal{G}(x) = \left(\mathcal{V}, \mathcal{E} \setminus \{e \mid x(e) = 0\}, \mathcal{W}\right) \\ e_t = \underset{e\in\mathcal{N}(v_t)}{\arg\max}\ \mathbb{E}_{(x,\eta)\sim P(\cdot|\psi_t)}\left[\mathbb{I}(e \in \textsc{ShortestPath}(\mathcal{G}(x), v_t, v_g))\right] \end{aligned} \tag{A.6}$$

Here we sample a set of worlds $(x, \eta) \sim P(\cdot|\psi_t)$, find the shortest path for each world and store the first edge along the path. The agent moves along the most common edge.

MCBE and QMDP do not necessarily agree on the same actions. One can construct examples where MCBE has a very high QMDP value because the action maybe quite suboptimal for worlds for which it is not on the shortest path. MCBE too is susceptible to the clairvoyance trap.

A.3.1.5 Collision Measure (CM)

A drawback of the OFU policy is that it does not reason about the likelihood of a path to be valid. This can lead to excessive exploration of implausible paths. Augmenting the original $\mathcal{W}$ with a term penalizing small $P(x)$ retains the graph substructure needed for efficient search while hedging against likely blocked edges. We examine weight augmentation using the collision measure proposed in (*Choudhury et al.*, 2016) for fast motion planning with C-space beliefs.

This heuristic balances exploration (assuming unexplored edges are free) with exploitation (penalizing edges with low validity likelihoods). The agent is at a vertex v_t and decides which edge e_t from the set of outgoing edges $\mathcal{N}(v_t)$ to traverse as follows:

$$\begin{aligned} \widehat{\mathcal{G}} &= (\mathcal{V}, \mathcal{E}, w(e) - \alpha \log P(x(e) = 1 | \psi_t)) \\ e_t &= \left\{ e \in \mathcal{N}(v_t) \;\middle|\; e \in \textsc{ShortestPath}(\widehat{\mathcal{G}}, v_t, v_g)) \right\} \end{aligned} \tag{A.7}$$

Here $\widehat{\mathcal{G}}$ is an optimistic graph created by removing all edges that are invalid with probability 1 given observation history ψ_t. Further, the weights are penalized by log-probability. Log-probability is chosen because for a path ξ, the log-probability is additive over edges assuming independence, i.e., $\log P(x(\xi)) = \sum_{e \in \xi} \log P(x(e))$. A known blocked edge $(P(x(e) = 1|\psi) = 0)$ yields a weight of ∞, and a known free edge $(P(x(e) = 1|\psi) = 1)$ yields $w(e)$.

We provide theoertical justification behind such a heuristic. We begin by mapping BTP to a Bayesian Search (*Ross*, 2014) problem. Let $\Xi = (\xi_1, \xi_2, \ldots, \xi_n)$ be the set of simple paths from v_s to v_g. The probability of edge validity $P(x)$ maps to a joint probability $P((\xi_1, \xi_2, \ldots, \xi_n))$ of paths being valid. For each path ξ_k, we assign a cost twice the length of the path $c_i = 2w(\xi_i)$. We now describe a sequential game of at most n rounds. In each round the agent attempts to traverse a path ξ_k. If the path is valid, it reaches the goal and receives a cost of c_k and the game terminates. Else, it receives a cost of c_k, remains at the start and the game continues.

Let σ be a sequence of attempting paths, i.e. a particular permutation of $\{1, \cdots, n\}$.

Let $\mathbb{E}\left[c(\sigma)\right]$ be the expected cost of a sequence. The optimal sequence σ^* has minimal expected cost, i.e. $\mathbb{E}\left[c(\sigma^*)\right] \leq \mathbb{E}\left[c(\sigma)\right]$ for all sequences σ^*.

Let σ^g be a sequence corresponding to a greedy policy that selects the path with the maximum posterior to cost ratio. Formally, this rule is defined as follows.

$$\sigma^g(i+1) = \arg\max_j \frac{P(\xi_j = 1 | \xi_{\sigma^g(1)} = 0, \xi_{\sigma^g(2)} = 0, \cdots, \xi_{\sigma^g(i)} = 0)}{c(\xi_j)} \tag{A.8}$$

where the numerator is the posterior probability of a path given the observations seen thus far and the denominator is cost of the path.

(*Dor et al.*, 1998)(Theorem 4.1) proved that greedy has an optimality bound of 4

Theorem A.3. *Given the following conditions on the game:*

1. *There exists at least one valid path*
2. *Ratio of costs are bounded* $\sup_{i,j} \frac{c_i}{c_j} < \infty$

The performance of the greedy sequence σ^g *is bounded*

$$\mathbb{E}\left[c(\sigma^g)\right] \leq 4\mathbb{E}\left[c(\sigma^*)\right] \tag{A.9}$$

Proof. We refer the reader to Theorem 4.1 in (*Dor et al.*, 1998). $\square$

We now map this result back to BTP. Note that BTP has an *asymmetric cost* of attempting a path. If traversal is successful, the agent pays half price of $0.5c_i$, else in the worst case pays the full price of c_i for going all the way to goal and returning. Let $\bar{c}(\sigma)$ be the cost of a sequence under these new rules. Note that the greedy policy σ^g remains the same with these new rules. We can transfer the bound from Theorem A.3

Corollary A.4. *The performance of the greedy sequence* σ^g *is bounded*

$$\mathbb{E}\left[\bar{c}(\sigma^g)\right] \leq 8\mathbb{E}\left[\bar{c}(\bar{\sigma}^*)\right] \tag{A.10}$$

Proof. Let $\bar{\sigma}*$ be the optimal policy for the new game. Then $\bar{c}(\bar{\sigma}^*) \geq 0.5c(\bar{\sigma}^*)$ where the bound is tight if the optimal policy never encounters a blocked path. It's straightforward to see that

$$\bar{c}(\sigma^g) \leq c(\sigma^g) \leq 4c(\sigma^*) \leq 4c(\bar{\sigma}^*) \leq 8\bar{c}(\bar{\sigma}^*) \tag{A.11}$$

$\square$

The greedy sequence is equivalent to a more general notion of the collision measure policy that can solve the following optimization

$$\pi^{\mathrm{CM2}} \equiv \left\{ e \in \mathcal{N}(v_t) \,\middle|\, e \in \arg\min_{\xi} \frac{w(\xi)}{P(x(\xi) = 1 | \psi_t)} \right\} \tag{A.12}$$

The optimization in (A.12) is intractable as $\frac{1}{P(x(\xi)=1)}$ is not additive. We choose to approximate this with log-probability. We utilize the following inequality for $p \in (p_{\min}, 1]$ and $\alpha \geq \frac{\frac{1}{p_{\min}} - 1}{\log \frac{1}{p_{\min}}}$

$$(1 - \log p) \leq \frac{1}{p} \leq (1 - \alpha \log p) \tag{A.13}$$

Hence $(1 - \alpha \log p)$ is a good family of approximators to $\frac{1}{p}$ which justifies (3.9) is an approximation.

A.3.2 Simulation-based Policies

This class of approaches employ *simulation* to estimate action values. We refer to the policy being simulated as the *rollout* policy $\pi(b)$. Let $V^{\pi(b)}(s)$ be the cumulative cost of the rollout policy initialized with belief b and simulated on the underlying MDP from state s. Note that unlike Section A.3.1, the simulator only has access to s and not the policy π. The simulator is thus able to provide observations o to the policy which updates the belief used in the rollout. We can then approximate action value as $\hat{Q}(b, a) \approx \mathbb{E}_{s \sim b}\left[c(s, a) + V^{\pi(b')}(s')\right]$, where s', b' is the next state and belief.

The attractive aspect of these approaches is that any policy from Section A.3.1 can be used as a rollout policy. For any such policy, we have the following upper bound

$$\hat{Q}(b, a) \geq \mathbb{E}_{s \sim b}\left[c(s, a) + V^{\pi^*}(s')\right] \geq Q^*(b, a) \tag{A.14}$$

If this is close to matching lower bounds from Section A.3.1, the value can be known exactly. However, the simulator invokes these policies $O(NTk)$ where N is the number of samples and T is the maximum horizon length, and k is the degree of the graph. Each invocation requires at least one belief update and perhaps several calls to ShortestPath. Even with parallelization this is memory and computation heavy.

A.3.2.1 Optimistic Rollout (ORO)

One of the simplest rollout policies is the OFU policy because it involves only one invocation of ShortestPath. Let π^{OFU} be the OFU policy. Let $V^{\pi^{\mathrm{OFU}}(v,\psi)}(x,\eta)$ be the evaluation of the policy starting from vertex v with history ψ on an underlying graph (x,η). The agent chooses edge e_t as follows:

$$\begin{aligned} e_t &= \operatorname*{arg\,min}_{e \in \mathcal{N}(v_t)} \mathbb{E}_{(x,\eta)\sim P(\cdot|\psi_t)} \left[c + V^{\pi^{\mathrm{OFU}}(v',\psi')}(x,\eta) \right] \\ &\text{where } (v',c) = \Gamma(v_t, e, x, \eta) \text{ and } \psi' = \psi_t \cup (x(e), \eta(e)) \end{aligned} \tag{A.15}$$

A.3.2.2 Upper Confidence Tree (UCT)

This is a state of the art algorithm from planning under uncertainty (*Kocsis and Szepesvári*, 2006) which combines the framework Monte-Carlo Tree Search with Upper Confidence Bound (UCB) for action selection. It has successfully been used for solving games (*Gelly and Silver*, 2007; *Silver et al.*, 2016), POMDPs (*Silver and Veness*, 2010) and Bayesian RL (*Guez et al.*, 2012). The idea builds on top of simulation based evaluation but differs on how actions are selected and how estimates are backed up.

Each UCT rollout begins with a belief sate b_0 and grows a tree where each node is a successor b. The value of each action $\hat{Q}(b,a)$ is an average over successors. To expand a given node, the search has to select one of k actions that according to the following rule:

$$\operatorname*{arg\,max}_{a_i} B\sqrt{\frac{\log N(b,a_i)}{N(b,a_i)}} - \hat{Q}(b,a) \tag{A.16}$$

Once the search goes off the tree, it uses a roll out policy (such as π^{OFU}) to finish the episode. UCT has been proved to converge to the exact Q-values (*Eyerich et al.*, 2010) asymptotically, i.e. $\hat{Q}(b,a) \to Q(b,a)$. However there is no such guarantee on the rate of convergence. Hence, in practice, UCT might have to do a large number of simulations.

A.3.3 Planning to gather information

The final class of approach we consider is where an agent plans to explicitly gather information. One such approach is the Hedged Shortest Path under Determinization (HSPD) (*Lim et al.*, 2017) algorithm which was original defined for the Bayesian Canadian Traveler Problem. HSPD determinizes the graph according to the most likely edge (MLE) assumption - each edge is set to valid if the marginal posterior probability is 0.5. The agent at every timestep plans two paths - exploitation and exploration. The exploitation is simply the shortest path to goal. The exploration path is the shortest path that reduces the version space to less than 0.5 fraction. The agent then takes the shorter of these paths and travels till it encounters a blocked edge, following which it returns to the start. This happens only a logarithmic number of times till it finds a path to goal.

This method for the BCTP has a near-optimality guarantee of $4(\log \delta + 1)$ where δ is the minimum prior probability of an underlying world. However, there are two concerns with the approach. Planning in belief space requires several invocations to the Bayes filter which can be expensive. Secondly, for the case of BTP the value of δ can be quite small as the observations are continuous. For these reasons, we chose not to proceed with this method although an efficient implementation for BTP would be of great interest.

A.4 CLASP Experiment Details and Alternate Analysis

A.4.1 Shape Network Training

We generated 3 distinct datasets of voxelized objects with size 64^3 from random axis-aligned boxes (AAB), YCB objects (*Calli et al.*, 2017), and ShapeNet mugs (*Chang et al.*, 2015). Boxes for AAB had width, depth, and height uniformly sampled with 2 to 41 voxels. For YCB and ShapeNet we generated ground truth voxelgrids centered on the object with different rotations using binvox (*Nooruddin and Turk*, 2003; *Min*, 2004 - 2020). For YCB we applied all 15 degree increment rotations about both the vertical and a horizontal axis. For Shapenet we applied all 5 degree increment rotations about the vertical axis.

During each epoch of training, each voxelgrid was augmented with translations sampled uniformly from -10 to 10 voxels in each direction. The 2.5D "known occupied" and "known free" voxelgrids were generated assuming a sensor looking down the x-direction. Sensor noise was simulated by sampling IDD 0-mean 2cm-std. deviation gaussian random noise in a depth image of 16x16, scaling that depth image to 64x64 using bilinear interpolation, then applying that noise to the x-direction of the known occupied and free voxelgrids.

We trained separate instances of PSSNet (*Saund and Berenson*, 2020b) on AAB, YCB, and Shapenet mugs, training for at least 100 epochs ($\sim$1 day), and used the iteration with minimal loss for experiments.

A.4.2 Robot Motion Generation

The following procedure was used to generate the robot motion which in turn generated the contact and freespace observations. The robot began each trial with a roadmap: a graph of nodes corresponding to configurations, and edges of robot motions connecting the nodes. Each scene contained a Goal Generator function, which mapped the completed objects to a goal Task-Space Region (TSR) (*Berenson et al.*, 2011). Using 10 sampled worlds, there were a corresponding 10 separate TSRs. In the scene above, the Goal Generator took the mean of the completed object points, and generated a TSR centered 10cm back in the occluded region.

At each iteration if the robot did not currently satisfy any TSR, approximately 80 configurations were sampled from each TSR and added to the roadmap, and the robot would attempt to traverse the roadmap to the closest configuration in a TSR. If the robot satisfied all TSRs the task was considered complete.

If instead the robot satisfied at least one but not all TSRs, the robot took an information gathering action. For each outgoing edge of the robot's node on the roadmap, the Information Gain (IG) was calculated from the existing particles using the method in (*Saund et al.*, 2017). The robot took the action with highest information gain, which often (intentionally) contacted an object. The belief was updated and the next iteration began.

Detecting contact in simulation: The voxelized robot was computed using GpuVoxels (*Hermann et al.*, 2014) with a much larger 256^3 voxelgrid with 1cm voxel side lengths. This robot voxelgrid was converted to an occupied point cloud, then transformed to the object frame, and converted into a voxelgrid matching the size and position of the depth image voxelgrid. Contact was determined by checking for overlap between the robot and object voxelgrid. For each configuration visited not in contact, the voxelized robot was added to the known freespace. Each configuration in contact generated a Collision Hypothesis Set, added to $Q_{contact}$.

A.4.3 Likelihood Results

We consider an alternative analysis of the experiment data presented in Section 5.5.4. Given that we model scenes by sampling shapes in a particle filter, we consider the likelihood of the ground truth scene given the particles. Since a particle filter models discrete samples, none of which will exactly match the ground truth, we apply a kernel to our particles in workspace. Specifically, we apply a non-normalized

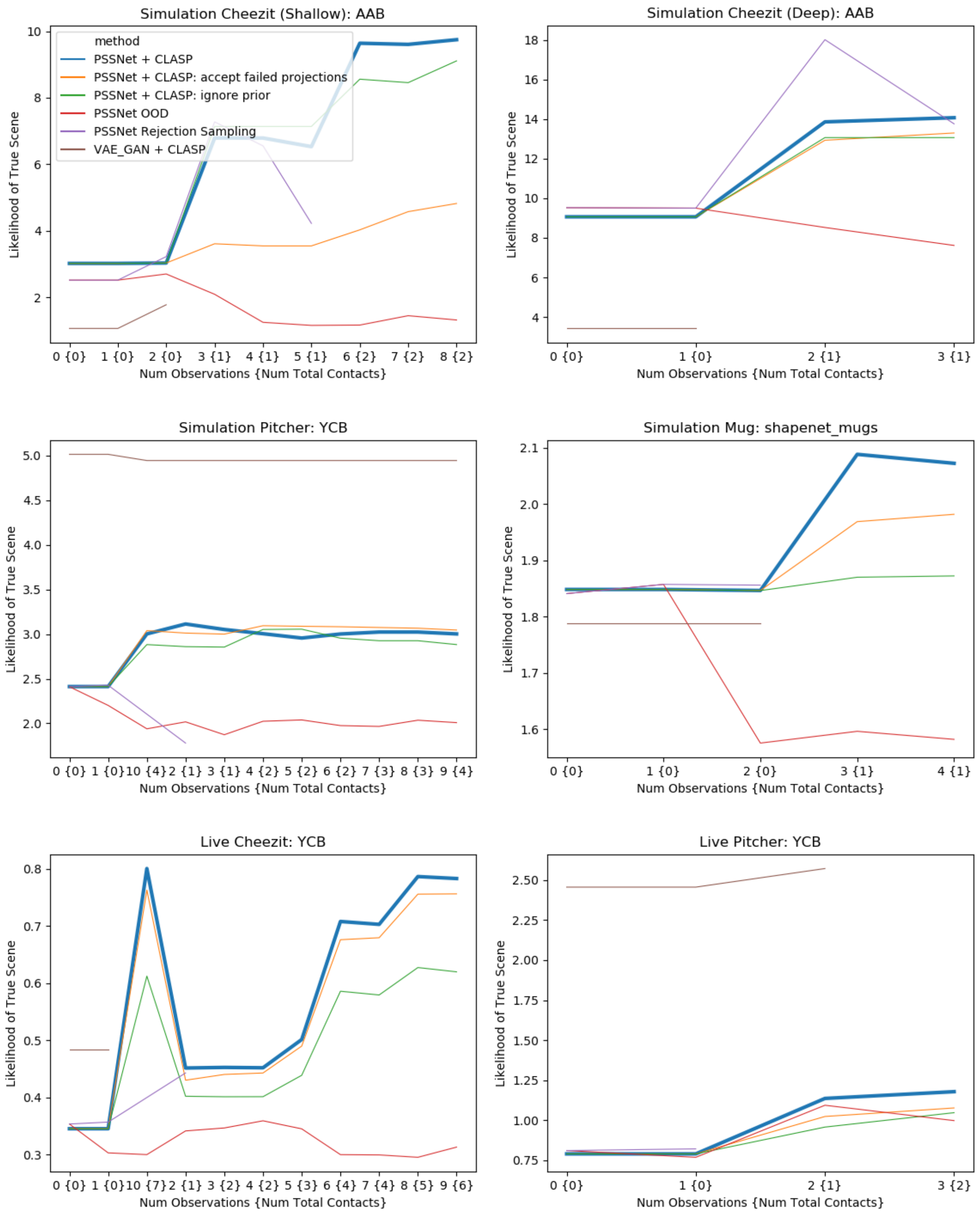


Figure A.3: Plots of likelihoods of CLASP and baselines under the particle filter belief and kernel function.

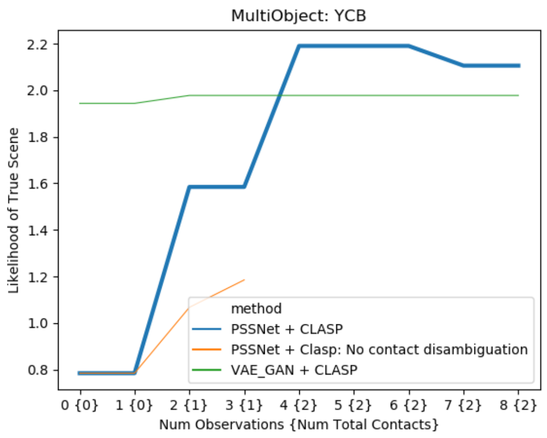


Figure A.4: Likelihood of CLASP and baselines for the multiobject scene

kernel based on the Chamfer Distance between two shapes s_1, s_2:

$$k(s_1, s_2) = \frac{1}{CD(s_1, s_2)} \tag{A.17}$$

The (non-normalized) likelihood of a particular scene occupancy s under the belief of n particles Φ is then

$$p(s|\Phi) = \sum_{\phi \in \Phi} \frac{1}{n} k(f_{dec}(\phi), s) \tag{A.18}$$

where $f_{dec}(\phi)$ decodes all latent shape vectors $z \in \phi$ into a scene.

We plot the likelihood of the true scene in Fig. A.3 and Fig. A.4, and find similar trends as in Section 5.5.4. The magnitude of the likelihood is not meaningful, however the relatively likelihoods between the methods are. Initially methods perform similarly, except VAE_GAN which is either better or worse than other methods. With contact and freespace observations, our proposed CLASP with PSSNet tends to increase the likelihood of the ground truth scene, while VAE_GAN tends to decrease the likelihood.

BIBLIOGRAPHY

BIBLIOGRAPHY

Agrawal, S., and N. Goyal (2013), Further optimal regret bounds for thompson sampling, in *Artificial intelligence and statistics*.

Alspach, A., K. Hashimoto, N. Kuppuswamy, and R. Tedrake (2019), Soft-bubble: A highly compliant dense geometry tactile sensor for robot manipulation, in *RoboSoft*.

Awerbuch, B., and R. Kleinberg (2004), Adaptive routing with end-to-end feedback: Distributed learning and geometric approaches, in *ACM symposium on Theory of computing*.

Bae, K., D. Belton, and D. D. Lichti (2009), A closed-form expression of the positional uncertainty for 3D point clouds, *TPAMI*.

Barrow, H., J. Tenenbaum, R. Bolles, and H. Wolf (1977), Parametric correspondence and chamfer matching: Two new techniquesfor image matching, *IJCAI*.

Berenson, D., S. Srinivasa, and J. Kuffner (2011), Task space regions: A framework for pose-constrained manipulation planning, *International Journal of Robotics Research*, *30*(12), 1435 – 1460.

Besl, P. J., and N. D. McKay (1992), A method for registration of 3-D shapes, *TPAMI*.

Bhattacharjee, T., P. Grice, A. Kapusta, M. Killpack, D. Park, and C. Kemp (2014), A robotic system for reaching in dense clutter that integrates model predictive control, learning, haptic mapping, and planning, *IROS*.

Bicchi, A., J. K. Salisbury, and D. Brock (1993), Contact sensing from force measurements, *IJRR*, *12*(3), 249–262.

Blackmore, L. (2006), A probabilistic particle control approach to optimal , robust predictive control, in *AIAA*.

Bnaya, Z., A. Felner, and S. Shimony (2009), Canadian traveler problem with remote sensing, in *IJCAI*.

Bohlin, R., and L. E. Kavraki (2000), Path planning using lazy PRM, in *ICRA*.

Brafman, R. I., and M. Tennenholtz (2002), R-max-a general polynomial time algorithm for near-optimal reinforcement learning, *Journal of Machine Learning Research*.

Bry, A., and N. Roy (2011), Rapidly-exploring random belief trees for motion planning under uncertainty, in *ICRA*, doi:10.1109/ICRA.2011.5980508.

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, *IJRR*.

Cesa-Bianchi, N., and G. Lugosi (2012), Combinatorial bandits, *Journal of Computer and System Sciences*.

Chang, A. X., et al. (2015), ShapeNet: An Information-Rich 3D Model Repository, *Tech. Rep. arXiv*.

Chapelle, O., and L. Li (2011), An empirical evaluation of thompson sampling, in *NIPS*.

Chen, S., B. Saund, and R. Simmons (2017), The datum particle filter: Localization for objects with coupled geometric datums, in *IROS*.

Chen, Y., S. Javdani, A. Karbasi, D. Bagnell, S. Srinivasa, and A. Krause (2015), Submodular surrogates for value of information., in *AAAI*.

Choudhury, S., C. Dellin, and S. Srinivasa (2016), Pareto-optimal search over configuration space beliefs for anytime motion planning, in *IROS*.

Choudhury, S., S. Javdani, S. Srinivasa, and S. Scherer (2017), Near-optimal edge evaluation in explicit generalized binomial graphs, in *NIPS*.

Choudhury, S., S. Srinivasa, and S. Scherer (2018), Bayesian active edge evaluation on expensive graphs, in *IJCAI*.

Choy, C. B., D. Xu, J. Gwak, K. Chen, and S. Savarese (2016), 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction, in *ECCV*.

Cohen, B., M. Phillips, and M. Likhachev (2015), Planning single-arm manipulations with n-arm robots, in *Eigth Annual Symposium on Combinatorial Search*.

Dai, A., C. R. Qi, and M. Nießner (2017), Shape completion using 3D-encoder-predictor CNNs and shape synthesis, *CVPR*.

Dellin, C. M., and S. Srinivasa (2016), A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors, in *ICAPS*.

Deng, X., A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox (2019), PoseRBPF: A rao-blackwellized particle filter for 6D object pose estimation, in *RSS*.

Desingh, K., O. C. Jenkins, L. Reveret, and Z. Sui (2016), Physically plausible scene estimation for manipulation in clutter, *Humanoids*.

Desingh, K., S. Lu, A. Opipari, and O. C. Jenkins (2019a), Efficient nonparametric belief propagation for pose estimation and manipulation of articulated objects, *Science Robotics*.

Desingh, K., S. Lu, A. Opipari, and O. C. Jenkins (2019b), Factored pose estimation of articulated objects using efficient nonparametric belief propagation, in *ICRA*.

Dinh, L., D. Krueger, and Y. Bengio (2014), Nice: Non-linear independent components estimation, *CoRR*.

Dinh, L., J. Sohl-Dickstein, and S. Bengio (2017), Density estimation using Real NVP, *ArXiv*.

Dor, A., E. Greenshtein, and E. Korach (1998), Optimal and myopic search in a binary random vector, *Journal of applied probability*.

Eyerich, P., T. Keller, and M. Helmert (2010), High-quality policies for the canadian traveler's problem., in *AAAI*.

Fan, H., H. Su, and L. J. Guibas (2017), A point set generation network for 3D object reconstruction from a single image, *CVPR*.

Ferguson, D., and A. Stentz (2007), Field d*: An interpolation-based path planner and replanner, in *Robotics research*.

Fried, D., S. E. Shimony, A. Benbassat, and C. Wenner (2013), Complexity of canadian traveler problem variants, *Theoretical Computer Science*.

Gelly, S., and D. Silver (2007), Combining online and offline knowledge in uct, in *ICML*.

Girdhar, R., D. Fouhey, M. Rodriguez, and A. Gupta (2016), Learning a predictable and generative vector representation for objects, in *ECCV*.

Golovin, D., A. Krause, and D. Ray (2010), Near-optimal bayesian active learning with noisy observations, in *NIPS*.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014), Generative adversarial nets, in *NIPS*.

Guez, A., D. Silver, and P. Dayan (2012), Efficient bayes-adaptive reinforcement learning using sample-based search, in *Advances in neural information processing systems*.

Guibas, L., D. Hsu, H. Kurniawati, and E. Rehman (2008), Bounded uncertainty roadmaps for path planning, in *WAFR*.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville (2017), Improved training of wasserstein gans, in *NIPS*.

György, A., T. Linder, G. Lugosi, and G. Ottucsák (2007), The on-line shortest path problem under partial monitoring, *Journal of Machine Learning Research*.

Haghtalab, N., S. Mackenzie, A. Procaccia, O. Salzman, and S. Srinivasa (2018), The provable virtue of laziness in motion planning, in *ICAPS*.

Hauser, K. (2015), Lazy collision checking in asymptotically-optimal motion planning, in *ICRA*.

Hausman, K., S. Niekum, S. Osentoski, and G. S. Sukhatme (2015), Active articulation model estimation through interactive perception, in *ICRA*.

Hermann, A., F. Drews, J. Bauer, S. Klemm, A. Roennau, and R. Dillmann (2014), Unified GPU voxel collision detection for mobile manipulation planning, in *IROS*.

Hodan, T., D. Barath, and J. Matas (2020), EPOS: Estimating 6d pose of objects with symmetries, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Janson, L., T. Hu, and M. Pavone (2018a), Safe motion planning in unknown environments: Optimality benchmarks and tractable policies, *CoRR*.

Janson, L., T. Hu, and M. Pavone (2018b), Safe motion planning in unknown environments: Optimality benchmarks and tractable policies, *arXiv preprint arXiv:1804.05804*.

Jung, J. (2019), Active shape completion using tactile glances, in *Master's Thesis: Lulea University of Technology*.

Karaman, S., and E. Frazzoli (2011), Sampling-based algorithms for optimal motion planning, *The International Journal of Robotics Research*, *30*(7), 846–894.

Kavraki, L., P. Svestka, J. Latombe, and M. Overmars (1996), Probabilistic roadmaps for path planning in high-dimensional configuration spaces, *T-RO*.

Killpack, M., A. Kapusta, and C. Kemp (2016), Model predictive control for fast reaching in clutter, *Autonomous Robots*, *40*(3), 537–560.

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

Kingma, D. P., and P. Dhariwal (2018), Glow: Generative flow with invertible 1x1 convolutions, *NeurIPS*.

Kingma, D. P., and M. Welling (2014), Auto-encoding variational bayes., in *ICLR*, edited by Y. Bengio and Y. LeCun.

Klingensmith, M., M. Koval, S. Srinivasa, N. Pollard, and M. Kaess (2016a), The manifold particle filter for state estimation on high-dimensional implicit manifolds.

Klingensmith, M., M. Koval, S. Srinivasa, N. Pollard, and M. Kaess (2016b), The manifold particle filter for state estimation on high-dimensional manifolds, *CoRR*.

Kocsis, L., and C. Szepesvári (2006), Bandit based monte-carlo planning, in *European conference on machine learning*.

Koenig, S., and M. Likhachev (2002), D* lite, in *AAAI*.

Koonjul, G., G. Zeglin, and N. Pollard (2011), Measuring contact points from displacements with a compliant, articulated robot hand, in *ICRA*.

Koval, M. C., N. S. Pollard, and S. S. Srinivasa (2015), Pose estimation for planar contact manipulation with manifold particle filters, *IJRR*.

Kuffner, J. J., and S. M. LaValle (2000), RRT-Connect: An efficient approach to single-query path planning, in *IEEE International Conference on Robotics and Automation, ICRA*.

Kuppuswamy, N., A. Alspach, A. Uttamchandani, S. Creasey, T. Ikeda, and R. Tedrake (2020), Soft-bubble grippers for robust and perceptive manipulation, in *IROS*.

Laurent, H., and R. Rivest (1976), Constructing optimal binary decision trees is np-complete, *Information processing letters*.

Lee, A., Y. Duan, S. Patil, J. Schulman, Z. McCarthy, J. van den Berg, K. Goldberg, and P. Abbeel (2013), Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles, *IROS*.

Lee, M. A., Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg (2020), Making sense of vision and touch: Learning multimodal representations for contact-rich tasks, *IEEE Transactions on Robotics*, doi: 10.1109/TRO.2019.2959445.

Li, Y., J.-Y. Zhu, R. Tedrake, and A. Torralba (2019), Connecting touch and vision via cross-modal prediction, in *CVPR*.

Lim, Z. W., D. Hsu, and W. S. Lee (2015), Adaptive stochastic optimization: From sets to paths, in *Advances in Neural Information Processing Systems*.

Lim, Z. W., D. Hsu, and W. S. Lee (2016), Adaptive informative path planning in metric spaces, *IJRR*.

Lim, Z. W., D. Hsu, and W. S. Lee (2017), Shortest path under uncertainty: Exploration versus exploitation, in *UAI*.

Littman, M. L., A. R. Cassandra, and L. P. Kaelbling (1995), Learning policies for partially observable environments: Scaling up, in *Machine Learning Proceedings 1995*.

Liu, Z., D. Chen, K. M. Wurm, and G. von Wichert (2015), Table-top scene analysis using knowledge-supervised mcmc, *Robotics and Computer-Integrated Manufacturing*.

Mandalika, A., O. Salzman, and S. Srinivasa (2018), Lazy Receding Horizon A* for Efficient Path Planning in Graphs with Expensive-to-Evaluate Edges, in *icaps*.

Mandalika, A., S. Choudhury, O. Salzman, and S. Srinivasa (2019), Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles, *ICAPS*.

Michalkiewicz, M., E. Belilovsky, M. Baktashmotlagh, and A. Eriksson (2020), A simple and scalable shape representation for 3D reconstruction, *arXiv*.

Min, P. (2004 - 2020), binvox, `http://www.patrickmin.com/binvox`.

Narayanan, V., and M. Likhachev (2016), Discriminatively-guided deliberative perception for pose estimation of multiple 3D object instances, in *RSS*.

Narayanan, V., and M. Likhachev (2017), Heuristic search on graphs with existence priors for expensive-to-evaluate edges, in *ICAPS*.

Nikolova, E., and D. R. Karger (2008), Route planning under uncertainty: The canadian traveller problem., in *AAAI*.

Nooruddin, F. S., and G. Turk (2003), Simplification and repair of polygonal models using volumetric techniques, *IEEE Transactions on Visualization and Computer Graphics*.

Osband, I., D. Russo, and B. Van Roy (2013), (more) efficient reinforcement learning via posterior sampling, in *NIPS*.

Papadimitriou, C., and M. Yannakakis (1991a), Shortest paths without a map, *Theoretical Computer Science*.

Papadimitriou, C. H., and M. Yannakakis (1991b), Shortest paths without a map, *Theoretical Computer Science*.

Park, D., A. Kapusta, J. Hawke, and C. C. Kemp (2014), Interleaving planning and control for efficient haptically-guided reaching in unknown environments, in *Humanoids*.

Patil, S., J. van den Berg, and R. Alterovitz (2012), Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty, in *ICRA*.

Peretroukhin, V., M. Giamou, D. Rosen, and W. N. Greene (2020), A smooth representation of belief of SO(3) for deep rotation learning with uncertainty, *RSS*.

Price, A., L. Jin, and D. Berenson (2019), Inferring occluded geometry improves performance when retrieving an object from dense clutter, *International Symposium on Robotics Research*.

Rezende, D. J., and S. Mohamed (2015), Variational inference with normalizing flows, *arXiv*.

Richter, C., W. Vega-Brown, and N. Roy (2018), Bayesian learning for safe high-speed navigation in unknown environments, in *International Symposium on Robotics Research*.

Riegler, G., A. O. Ulusoy, and A. Geiger (2017), Octnet: Learning deep 3D representations at high resolutions, in *CVPR*.

Roldão, L., R. de Charette, and A. Verroust-Blondet (2021), 3d semantic scene completion: a survey, *ArXiv*, *abs/2103.07466*.

Ross, S. (2014), *Introduction to stochastic dynamic programming*, Academic press.

Russo, D., and B. V. Roy (2018), A tutorial on thompson sampling, *Foundations and Trends in Machine Learning*.

Rusu, R. B., and S. Cousins (2011), 3D is here: Point Cloud Library (PCL), in *ICRA*.

Saund, B., and D. Berenson (2018), Motion planning for manipulators in unknown environments with contact sensing uncertainty, in *ISER*.

Saund, B., and D. Berenson (2020a), Fast planning over roadmaps via selective densification, in *ICRA*.

Saund, B., and D. Berenson (2020b), Diverse plausible shape completions from ambiguous depth images, *CoRL*.

Saund, B., S. Chen, and R. Simmons (2017), Touch based localization of parts for high precision manufacturing, in *ICRA*.

Saund, B., S. Choudhury, S. Srinivasa, and D. Berenson (2019), The blindfolded robot : A bayesian approach to planning with contact feedback.

Schmidt, T., R. A. Newcombe, and D. Fox (2014), Dart: Dense articulated real-time tracking., in *Robotics: Science and Systems*.

Siegel, M., P. Gunatilake, and G. Podnar (1998), Robotic assistants for aircraft inspectors, *IEEE Instrumentation & Measurement*.

Silver, D., and J. Veness (2010), Monte-carlo planning in large POMDPs, in *NIPS*.

Silver, D., A. Huang, C. Maddison, et al. (2016), Mastering the game of go with deep neural networks and tree search, *nature*.

Somani, A., N. Ye, D. Hsu, and W. S. Lee (2013), Despot: Online pomdp planning with regularization, in *NIPS*.

Stentz, A. (1997), Optimal and efficient path planning for partially known environments, in *Intelligent Unmanned Ground Vehicles*.

Şucan, I., M. Moll, and L. Kavraki (2012), The Open Motion Planning Library, *IEEE Robotics & Automation Magazine*, *19*.

Talebi, M. S., Z. Zou, R. Combes, A. Proutiere, and M. Johansson (2017), Stochastic online shortest path routing: The value of feedback, *IEEE Transactions on Automatic Control*.

Thompson, W. R. (1933), On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, *Biometrika*.

Toit, N. E. D., and J. W. Burdick (2011), Probabilistic collision checking with chance constraints, *IEEE T-RO*, *27*(4), 809–815.

Tong, S., and D. Koller (2001), Support vector machine active learning with applications to text classification, *Journal of machine learning research*.

Tremblay, J., T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield (2018), Deep object pose estimation for semantic robotic grasping of household objects, *CoRL*.

Vahdat, A., and J. Kautz (2020), NVAE: A deep hierarchical variational autoencoder, in *arxiv*.

van den Berg, J., D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha (2012), LQG-Obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty, *ICRA*.

Wang, S., J. Wu, X. Sun, W. Yuan, W. Freeman, J. Tenenbaum, and E. Adelson (2018), 3d shape perception from monocular vision, touch, and shape priors, in *IROS*.

Wen, C., Y. Zhang, Z. Li, and Y. Fu (2019), Pixel2mesh++: Multi-view 3D mesh generation via deformation, *ICCV*.

Wu, J., C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum (2016), Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling, in *NIPS*.

Wu, J., Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum (2017), MarrNet: 3D Shape Reconstruction via 2.5D Sketches, in *Advances In Neural Information Processing Systems*.

Wu, J., C. Zhang, X. Zhang, Z. Zhang, W. T. Freeman, and J. B. Tenenbaum (2018), Learning shape priors for single-view 3D completion and reconstruction, in *ECCV*.

Wüthrich, M., P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal (2013), Probabilistic object tracking using a range camera, in *IROS*.

Xie, H., H. Yao, X. Sun, S. Zhou, and S. Zhang (2019), Pix2vox: Context-aware 3d reconstruction from single and multi-view images, in *ICCV*.

Yang, B., S. Rosa, A. Markham, N. Trigoni, and H. Wen (2018), Dense 3D object reconstruction from a single depth view, in *TPAMI*.

Yang, H., J. Shi, and L. Carlone (2020), TEASER: Fast and Certifiable Point Cloud Registration, *arXiv*.

Yoon, S. W., A. Fern, R. Givan, and S. Kambhampati (2008), Probabilistic planning via determinization in hindsight., in *AAAI*.

Yu, Y., Z. Huang, F. Li, H. Zhang, and X. Le (2020), Point encoder gan: A deep learning model for 3D point cloud inpainting, *Neurocomputing*.

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

Yuan, W., T. Khot, D. Held, C. Mertz, and M. Hebert (2018a), PCN: Point completion network, in *3DV*.

Yuan, W., Y. Mo, S. Wang, and E. Adelson (2018b), Active clothing material perception using tactile sensing and deep learning, *ICRA*.

Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao (2015), 3D shapenets: A deep representation for volumetric shapes, in *CVPR*.

Zhou, B., H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, and A. Torralba (2018), Semantic understanding of scenes through the ade20k dataset, *International Journal on Computer Vision*.