Distributed Inference for Robotic Perception and Planning Under Uncertainty

by

Jana Pavlasek

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

Jana Pavlasek
pavlasek@umich.edu
ORCID iD: 0000-0001-6332-2646

DEDICATION

To Granddad, for inspiring me to become Dr. Pavlasek, too.

ACKNOWLEDGEMENTS

I am grateful to many people for their support that made the work in this dissertation possible. First, I would like to thank my advisor, Chad Jenkins, who provided me with a collaborative environment with many opportunities to grown and learn. I have been fortunate to benefit from the mentorship of someone with a strong commitment to making our community better and more inclusive. I am indebted to Chad for his support and patience, through good times and tough ones, and for sharing his wisdom (even when it came in the form of 80's pop culture or Naruto references).

I am tremendously grateful to Professors David Fouhey, Jessy Grizzle, Mark Guzdial, and Fabio Ramos for serving on my committee and providing invaluable support and feedback. I would particularly like to thank Fabio, for insightful discussions about probability and kernels, and for his close collaboration on many of the ideas contained in this document. I would also like to thank Tucker Hermans, for guiding me as I delved into the world of planning as inference, and whose mentorship and support have been invaluable to me in my career.

It is somewhat atypical for a robotics dissertation to include a chapter on education. I have been lucky to have the opportunity to explore my passion for teaching during my PhD, with the help of many passionate educators and leaders in engineering education. I am grateful to Mark, for sharing his in-depth knowledge of computing education research and for his collaboration in turning my work on teaching into a dissertation chapter. Thanks also to Drs. Jasmine Jones, Jan Pearce, Wei Wu, Dwayne Joseph and Todd Shurn for joining us in the experiment of distributed teaching. Thank you to Peter Gaskell for trusting me with the MBot and helping me fix it when I broke it. Thank you to the students and instructional aides who supported the class for their efforts and enthusiasm.

I am proud to be a member of the Laboratory for Progress, where I have been able to collaborate with a tremendous group of people. Thank you to Karthik Desingh, Zhen Zeng, Zhiqiang Sui, Zheming Zhou, and Kevin French, for getting me through the first years of my graduate studies and passing on the culture of the lab. Particular thanks to Karthik whose support and mentorship has remained crucial to me throughout the years. Thanks to Xiaotong Chen, for always knowing in what order transforms should be multiplied, to

Stanley Lewis, for always knowing where the clamp is, to Anthony Opipari, for obsessing over probability theory with me, and to Elizabeth Olson, for many long phone calls and lunches. Life in the lab would not have been the same without Emily Sheetz, Alphonsus Adu-Bredu, Jasmine Berry, Jorge Vilchis, Tommy Cohn, Cameron Kisailus, Jace Aldrich, Amber Green, Joseph Taylor, Joshua Mah, Multy Xu, and Thiru Ashokkumar.

I am deeply indebted to the staff in Robotics, who I am convinced are collectively in possession of all of human knowledge: Denise, Kimberly, Damen, Dan, Abhishek, Christina, Samantha, and David. Thank you for all you have done for me and for making Robotics a supportive and welcoming environment.

To the friends I made along the way, especially Christina, Leanne, Tom, Thomas, Julia, Julie, Austin, Josh, Jenn, Nik, Peter, Grant, and Andrea. Thank you for keeping me sane and making my time in Ann Arbor worthwhile.

To my loving parents, who have given me everything and been endlessly supportive of whatever I do. You're the best parents in the world. Thank you to my sister, the next Dr. Pavlasek and my lifelong partner-in-crime.

Finally, to my husband, for his endless patience and constant support. Thanks for moving to a small midwest town we had never heard of with me and making it a home. I count myself very lucky to have a partner to share both my work and my life with. Finally, thank you to Darcy, for reminding me that there are more important things than work.

The work in this dissertation was supported in part by the Robotics Department at the University of Michigan, Amazon, the Ford Motor Company, NVIDIA, the Alfred P. Sloan Foundation, and a Natural Sciences and Engineering Research Council of Canada (NSERC) Postgraduate Scholarship.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

AI Artificial Intelligence

BP Belief Propagation

CER Computing Education Research

DTC Distributed Teaching Collaborative

GPU Graphical Processing Unit

MOOC Massively Open Online Courseware

MPC Model Predictive Control

MRF Markov Random Field

NBP Nonparametric Belief Propagation

PBP Particle Belief Propagation

SVBP Stein Variational Belief Propagation

SVGD Stein Variational Gradient Descent

ABSTRACT

Autonomous robots promise to improve human productivity and quality of life by assisting in our homes, factories, and labs to automate the dull, dirty, and dangerous. To achieve these long-promised versatile robot assistants, robots must be able to operate robustly in unstructured real-world environments. Environments made for humans remain challenging for autonomous robots due to their highly unstructured nature which arises from environmental occlusions, dynamic environments, and the diversity of possible objects a robot might encounter. These challenges result in a plurality of competing hypotheses present at all levels of the system. Modelling the uncertainty inherent to the robotic system is a crucial capability to enable robust operation in unstructured environments.

This dissertation considers the problem of scalable, robust operation under uncertainty through distributed probabilistic inference. To handle the intractable nature of these problems, *distributed* probabilistic inference decomposes high-dimensional problems into simpler, parallelizable subproblems. These are represented as probabilistic graphical models and solved via message passing. Further, the resulting distributions must encompass arbitrary, multi-modal uncertainty which results from competing hypotheses and noisy estimates. We employ nonparametric distributions for their flexible representational capabilities. We present novel approaches leveraging these insights and demonstrate their application to robotic perception and planning problems.

First, we consider the problem of articulated object localization in cluttered scenes towards robot manipulation of hand-tools. We take a parts-based approach, modelling object geometries by decomposing them into their component parts. We employ Nonparametric Belief Propagation to perform distributed inference over the resulting graphical model. A learned observation likelihood is leveraged alongside object geometry in order to infer the belief over the part poses. We demonstrate that the proposed method is robust to challenging observations with heavy occlusion on a custom dataset.

Second, we turn to the task of robotic planning for object manipulation. Many tasks in robotic manipulation are described by goals that are intractable to model explicitly (e.g. stable grasping or user preferences). We present a planning framework which considers uncertainty in the goal specification. To accomplish this, we consider robotic planning through the lens of probabilistic inference, modelling both the trajectory and goal as distributions. We propose a novel differentiable loss over arbitrary nonparametric goals which is demonstrated on high-dimensional robotic manipulation tasks of grasping and placement.

Third, we extend the problem of planning as inference to the high-dimensional problem of multi-robot coordination. We propose Stein Variational Belief Propagation, an algorithm for performing inference over graphical models. We show that the proposed algorithm is more effective at representing the underlying distribution than sampling-based baselines. We demonstrate the capability of this method to solve challenging, dynamic problems in robotics through multi-robot coordination experiments.

The promise of robotics coupled with the open challenges that remain in the field as described in this dissertation highlight the immediate need to train the next generation of diverse talent with expertise in robotics. Towards this objective, this dissertation formalizes recent trends in robotics undergraduate education. We present a modular introductory robotics curriculum which involves programming a custom robotic platform appropriate for undergraduate instructional use. Finally, we suggest best practices for teaching robotics as a discipline at the undergraduate level based on lessons learned teaching the described course at the University of Michigan and three partner institutions to over 100 students over the past three years.

CHAPTER 1

Introduction

The field of robotics has seen impressive growth over the past decades, but the promise of ubiquitous, versatile robot assistants has yet to be fulfilled. These autonomous robots have the potential to assist humans in home environments, factories, and laboratory environments. Autonomous robotic execution has seen tremendous growth in structured, static environments such as in manufacturing facilities or in highly constrained tasks (e.g. assembly line automation, vacuuming a room). However, versatile operation in *unstructured* environments, like those humans inhabit in their daily lives, remains an open challenge for autonomous mobile manipulators. Reliable operation in these environments poses a plurality of challenges for modern robots. Unlike structured lab environments, our unstructured human spaces are highly challenging for robotic systems due to the variability of possible surroundings and objects present. Challenging conditions such as heavy clutter lead to partial observability in the robot's perception. Additionally, these spaces can be highly dynamic due to other humans and agents moving in the environment. The challenge of modern robotics is to achieve robust operation in these natural environments.

Robotic manipulation tasks can be decomposed at a high-level into two key components: perception and action. Robotic *perception* is challenging due to the high-dimensional nature of RGB and depth camera data, and the partial observability of objects in the environment (e.g. in Figure 1.1, the objects of interest to the robot are heavily occluded). To interact with the environment, robots must then *plan actions* to manipulate objects, taking advantage of the scene perception (e.g. in Figure 1.1, the robot manipulates the hand tools to accomplish a task). Versatile robotic assistants require robust perception of the objects of interest and flexible planning objectives which scale across tasks.

Uncertainty estimation is a crucial capability in robotic systems for both perception and planning. Considering uncertainty enables robots to maintain multiple completing hypothesis over time, lending robustness to uncertainty and dynamic environments. Robots exist in a time-varying system, in which the state and observations continuously evolve in response to changing environmental conditions and the robot's own actions. As such, uncertainty forms

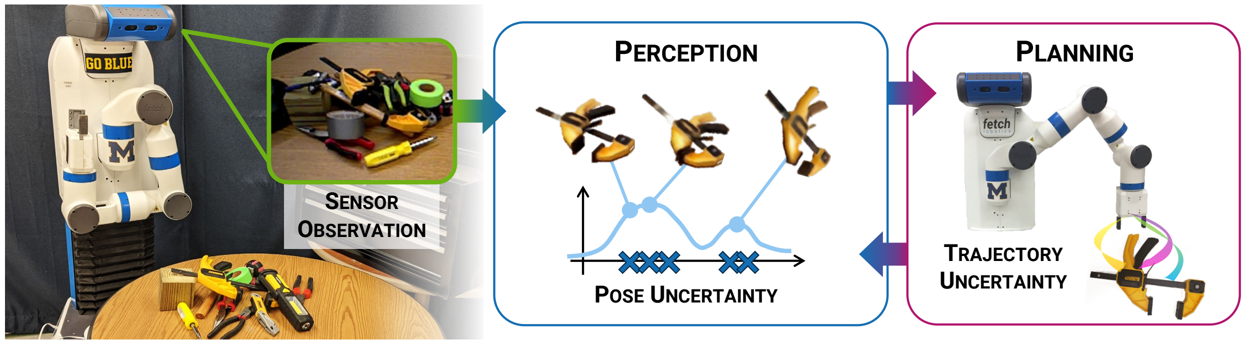


Figure 1.1: Versatile robotic assistants must be capable of performing various assembly and maintenance tasks. A robot faced with a highly cluttered scene (left) must first perceive the objects of interest (middle). The robot must then manipulates objects to complete the task (right), which involves performing complex interactions with the objects.

a key signal to the robot to reason over. To model such uncertainty, probabilistic inference techniques can be employed in order to maintain belief distributions over the robot state and action. Probabilistic approaches have a rich history of success in robotics [161, 150, 8, 61]. However, these methods often must contend with a tradeoff between the representational capability and the computational tractability of the algorithm. For example, in the manipulation task depicted in Figure 1.1, both the perception and planning problems are high-dimensional and multi-modal, due to the plurality of possible object locations and trajectory modes. While common approximations (e.g. Gaussian distributions) enable convenient mathematical properties to be exploited, yielding efficient algorithms [179, 161, 37], these representations are limiting. Fully specifying distributions of possible states and actions in cases such as that in Figure 1.1 requires more flexible representations.

Probabilistic inference techniques applied to robotics pose a number of open challenges. First, the high-dimensional nature of robotic vision and planning make it these techniques computationally expensive. Second, maintaining multi-modal distributions with the ability to represent the multiple competing hypotheses present in the robotic tasks described is challenging in probabilistic inference. As an alternative to limiting Gaussian or discrete approximations, some probabilistic inference techniques opt to represent these complex distributions *nonparametrically*, e.g. as mixtures of Gaussians or as implicit distributions. These approaches lend rich representational power, but often rely on expensive computational operations [26, 152, 144, 42]. Problems which suffer from both model misspecification and computational complexity are known as *doubly intractable*.

This dissertation is concerned with the latter of the approaches, arguing that the challenging nature of modelling multiple, diverse competing hypotheses prevalent in robotics applications necessitates the representational power of nonparametric distributions. This body of work focuses on distributed solutions to robotics problems, taking a divide-and-conquer approach to solving these complex problems by considering multiple smaller subproblems which can be efficiently solved in parallel. Problems of particular interest are those which can be represented as graphs and solved via graphical probabilistic inference. This is demonstrated on both perception and planning problems in robotics. We model object localization as graphical inference using Belief Propagation and apply the resulting approach to articulated object pose estimation in cluttered scenes [126]. We explore the use of learned models within the probabilistic framework to achieve methods that are both generalizable and efficient, and robust and explainable. We then consider manipulation planning, tackling the challenging case in which the goal region is uncertain and challenging to model [127]. We propose an approach based on Stein Variational Inference [92] which enables generalizable objectives under nonparametric uncertainty. This same technique is then applied within a graphical inference framework to solve the challenging problem of multi-robot planning. We propose an algorithm called Stein Variational Belief Propagation [128] and demonstrate its capabilities as an inference algorithm for robotic perception.

1.1 Uncertainty in Robotic Perception

Robotic perception for manipulation is primarily concerned with the detection and localization of objects towards accomplishing a task. A common representation of object localization is the object *pose*, consisting of the 3-dimensional (3D) position and orientation, resulting in a 6-dimensional (6D) quantity. This problem is high-dimensional and requires processing unstructured image (RGB) and sometimes depth (RGB-D) data. Object pose estimation gained a significant boost with the advancement of data-driven methods [184, 167]. However, robustness in highly cluttered scenes remains a challenge. Consider the scene in Figure 1.2. The robot observation is heavily cluttered, leading to multiple possible competing hypotheses for the object pose. Axes of symmetry and articulations in the object further increase the possible hypotheses for the pose. The resulting distribution representing the belief of the pose is therefore *multi-modal*.

In Chapter 3, we describe an approach to tractably solving the problem of articulated pose estimation suitable for cluttered conditions through distributed inference. We take a parts-based approach, representing articulated objects as a Markov Random Field (MRF) describing geometric constraints between component parts of the object. We maintain belief through Particle Belief Propagation (PBP) [69], a message passing-based inference algorithm which represents the belief over each node in the graph using a nonparametric distribution

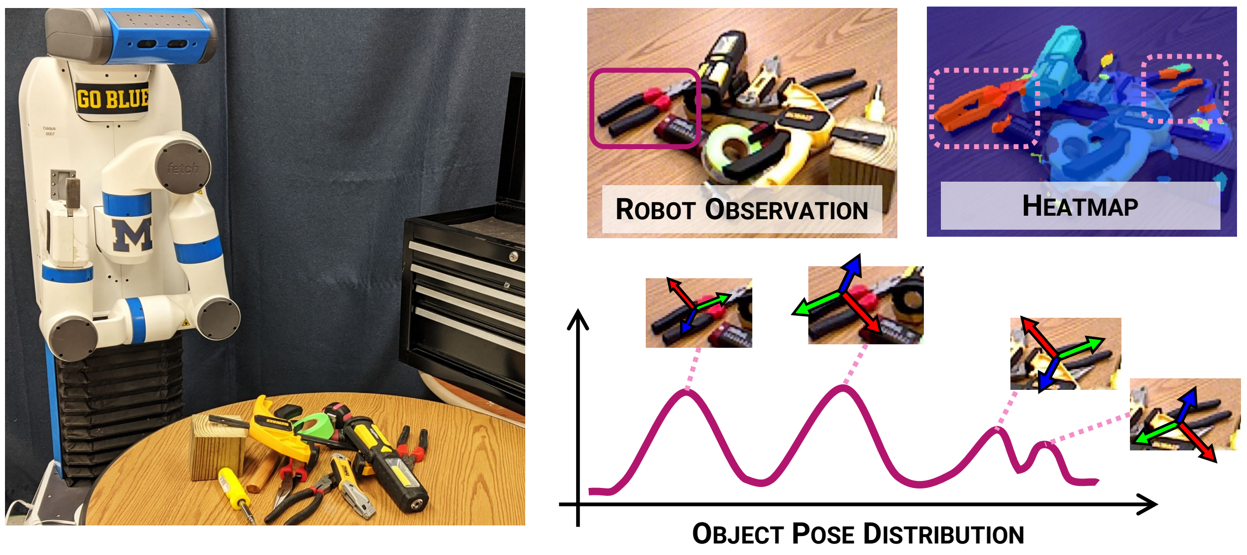


Figure 1.2: An illustration of the challenge of robotic perception for manipulation under uncertainty. In a cluttered scene (left), the robot observation may be occluded, leading to partial observations of the object of interest (top right). Occlusions and properties of the object like axes of symmetry result in a complex, multi-modal distribution over the object pose estimate (bottom right).

in the form of a particle set. This enables the representation of complex distributions, as illustrated in Figure 1.2. We formulate a likelihood within the framework of the MRF, using geometric priors over the object in combination with a data-driven likelihood over the observation based on a Convolutional Neural Network (CNN). By leveraging data-driven factors within a probabilistic framework, we obtain the representational capabilities of learning-based methods alongside the robustness of probabilistic inference. We demonstrate that our method yields improved pose estimation accuracy compared to baselines.

1.2 Uncertainty in Robotic Planning

Once an object is localized within the robot observation, a robotic manipulator must then generate action trajectories in order to manipulate the object towards accomplishing a given task. For a manipulation platform such as the Franka Emika Panda, shown in simulation in Figure 1.3, this is a high-dimensional planning problem involving planning in 7 degree-of-freedom (DoF) space. In the presence of obstacles, the number of possible trajectories contains multiple competing hypotheses. In addition, the goal of such a manipulation task is often intractible to model. In the case of stable grasping, there exist multiple valid grasp points, as shown in Figure 1.3. This poses several challenges. First, modelling the distribu-

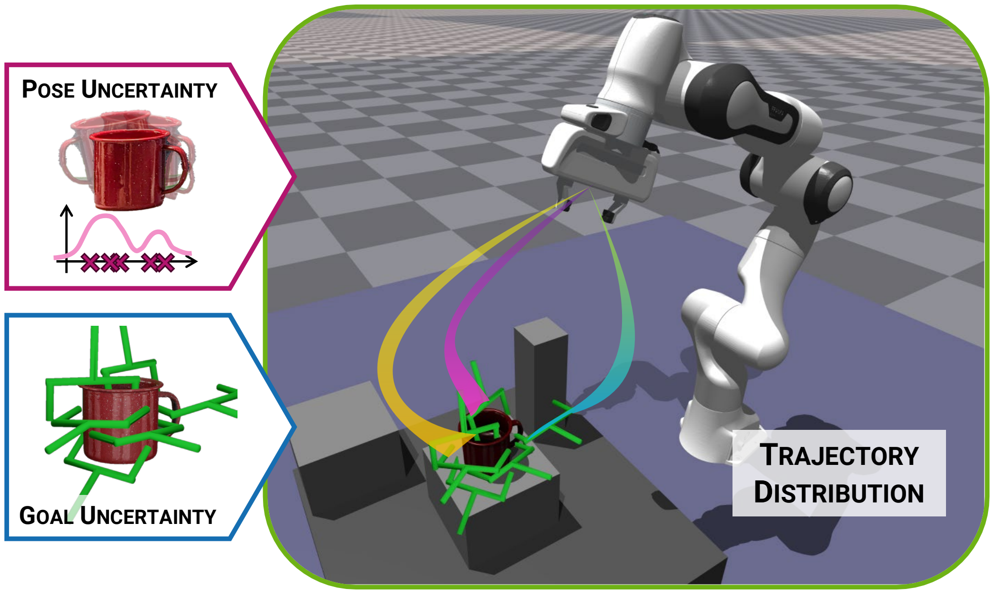


Figure 1.3: Planning for robotic manipulation must consider multiple competing hypotheses in both the trajectory and in the final goal position. Given an estimate of the pose of an object, which may contain uncertainty (top left), and possible grasp points (bottom left), the robot must plan a trajectory to grasp the object (right). Considering these sources of uncertainty and obstacles in the environment results in multiple possible trajectories.

tion of stable grasps explicitly is typically intractable. Second, selecting a grasp point as a planning objective requires considering environmental occlusions and reachability, resulting in a computationally expensive process.

One way to model the uncertainty inherent in robotic planning is through *planning as inference* [8, 166, 132, 181], in which the trajectory is modelled as a distribution. In Chapter 4, we introduce a planning as inference technique which considers uncertainty in the goal. The goal is modelled as an implicit distribution, represented by a set of valid goal samples (e.g. stable grasps, as in Figure 1.3). In practice, these samples might come from high-fidelity simulation or user demonstration. We leverage Stein Variational Gradient Descent (SVGD) [92], an emerging inference technique which is capable of maintaining diverse modes using a non-parametric distribution, which has shown promise as applied to planning as inference [81]. We extend this technique to generic goal representations through the introduction of a differentiable goal loss, based on the principles of Generalized Bayesian Inferences [18, 72, 104]. We demonstrate that this method is more effective at planning under cluttered conditions without the need for domain-dependent, computationally expensive pre-processing steps.

Having considered the high-dimensional challenge of planning on a robotic manipulator, we turn our attention to multi-robot coordination in order to further explore the scalability of SVGD for planning as inference. In multi-robot planning, uncertainty over a neighboring agent's chosen trajectory often leads to *deadlock*, a failure mode in which robots select incompatible trajectories, resulting in both agents getting stuck. We hypothesize that explicitly modelling uncertainty over the neighbors' potential actions can mitigate these cases, enabling the robots to find feasible paths more effectively.

In Chapter 5, we describe a solution to multi-robot planning as inference, relying on the principles of distributed inference. By modelling the multi-robot system as an MRF, we can apply message-passing algorithms to propagate information through the whole robot swarm. We extend SVGD to the graphical inference setting, and introduce Stein Variational Belief Propagation (SVBP), an inference algorithm for inferring marginal distributions on graphs. The proposed algorithm represents the node distributions nonparametrically, and is capable of maintaining diverse modes through the use of SVGD. We demonstrate that this algorithm is more effective at escaping deadlock scenarios than baselines, and can better represent true underlying multi-modal distributions compared to sampling-based baselines.

1.3 Robotics Education

The growth of the field in robotics has resulted in an increased demand for robotics education in undergraduate and graduate education. This is fueled by the growth in the industry demand for skilled talent in areas of robotics and artificial intelligence. Robotics has a rich history as a tool for computing and engineering education [49, 9, 164, 105]. Following the growth of robotics research, robotics has emerged in undergraduate and graduate curricula through specialized courses, primarily as technical electives or advanced specializations. The past decade has seen a growth in popularity in graduate specializations in robotics, followed by the more recent appearance of trailblazing undergraduate robotics programs [57, 71].

These trends exemplify the expansion of robotics from an area of specialization into the present-day emergence of *robotics as a discipline*. This new era in undergraduate education creates specific demands on the development of new curricula. The majority of robotics educational content available is intended for a graduate audience, begging the question: *what should an introductory robotics curriculum for the undergraduate level look like?* We argue that an early undergraduate course in robotics should introduce key concepts fundamental to the discipline (e.g. programming and math) while introducing key areas of robotics. Additionally, robotics poses unique challenges in terms of the uncertainty inherent in embodied autonomous systems which must be introduced in the course of an undergraduate robotics specialization.

This dissertation proposes a curriculum for a first-year undergraduate course in robotics

Figure 1.4: Instructors collaborate on development of an introductory robotics curriculum, offered across multiple institutions. Students at each institution complete modules which focus on writing autonomous programs for operating a real robot platform. *Photo: Brenda Ahearn/University of Michigan, College of Engineering, Communications and Marketing.*

which introduces programming and computational thinking through the lens of robotics and artificial intelligence by programming a real robot platform. We develop an ecosystem of tools comprised of robot hardware and educational modules which is flexible and configurable to the needs of a specific course. The course was developed and taught through a *Distributed Teaching Collaborative (DTC)*, and has served over 100 undergraduate students across four institutions, including the University of Michigan and collaborating Historically Black Colleges and Universities (HBCUs) and Minority Serving Institutions (MSIs). This collaborative, distributed approach to robotics education draws on lessons from work presented in this dissertation, by taking a divide-and-conquer approach in which effort across institutions benefits the collective. Lessons learned from the instruction of this course offer key insights and opportunities for future developments in robotics education.

1.4 Dissertation Contributions

This dissertation examines probabilistic techniques for robust and flexible representations of uncertainty for robotics applications. Specifically, the question this dissertation seeks to address is: **How can we apply distributed algorithms for scalable, robust robotic operation?** This question is examined through the lens of robotic perception, planning, and education. The contributions of each chapter are as follows:

1. **Articulated object localization in cluttered scenes** *(Chapter 3).* We consider perception for robotic tasks in the presence of heavy clutter, specifically pose estimation of articulated hand tools towards mobile manipulation. We propose a parts-based approach and apply belief propagation with a data-driven likelihood to obtain robust estimates of object location [126]. Results demonstrate improved performance under challenging, cluttered conditions.

2. **Planning to uncertain goal sets** *(Chapter 4).* We consider the problem of planning for a robotic manipulator in the case where the goal is a region which is intractable to model. We propose a novel goal loss which enables flexible goal region representation under uncertainty and apply Stein Variational Inference to plan trajectories [127]. This technique outperforms approaches which rely on domain-specific heuristics, and generalizes to any goal that can be represented by a set of example configurations.

3. **Stein Variational Belief Propagation for multi-robot coordination** (*Chapter 5*). We consider the challenging problem of multi-robot coordination through the lens of planning as inference, building on the methods explored in Chapter 4. We introduce a new algorithm, Stein Variational Belief Propagation (SVBP) [128], for probabilistic graphical inference, and show its ability to maintain diverse distributions. We demonstrate the robustness and efficiency of SVBP for multi-robot planning on both simulation and real-robot scenes.

4. **An introductory course for robotics as a discipline** *(Chapter 6).* We formalize the problem of robotics education as it applies to the study of *robotics as a discipline* and discuss the unique challenges in teaching computational robotics at the undergraduate level. We describe an introductory course developed as part of the Robotics undergraduate major at the University of Michigan. The course introduces fundamental programming concepts through the lens of robotics and AI and employs real robot platforms as educational tools, specifically developed for undergraduate robotics education [56]. Drawing on insights from the distributed methods discussed in previous chapters, we propose a *distributed* collaborative approach to teaching robotics in higher education which has been applied to offer the proposed course at three other institutions. We conclude with a summary of best practices and future opportunities based on multiple successful offerings of the proposed course.

CHAPTER 2

Background and Related Work

This chapter summarizes the related work in probabilistic inference applied to robotics for pose estimation and planning that motivate the contributions of this dissertation. Additionally, we introduce the fundamental concepts in nonparametric inference and graphical inference necessary for the discussions in Chapters 3 to 5.

2.1 Probabilistic Inference for Robotics

Probabilistic inference has a long history of success in robotics [161]. This dates back to seminal work on the Kalman Filter [74], and the Particle Filter [59], as dominant techniques for robot localization at the beginning of the 21st century [134, 111, 54, 85]. The success of these methods has been widely attributed to their ability to explicitly model the uncertainty due to noise in robotic sensors (e.g. Lidar range finders, odometry). Models for the noise distributions can be identified offline and integrated into the estimate online.

Over the past decade, advancements in data-driven deep learning [79, 4] has led to the rise in popularity of these methods being applied to robotics [184, 80, 29]. Data-driven methods take advantage of the growing availability of large datasets [24, 66, 27, 173]. The representational ability of learned methods eliminates the need to hand-design and tune likelihood functions manually. The ability to train and execute such models has benefited greatly from advancements in Graphical Processing Unit (GPU) technology, making them faster than traditional inference methods.

While deep learning has dominated many fields in Artificial Intelligence (AI) in the past decade [145, 77, 1], adoption by the robotics community has encountered some key challenges. Notably, robotics domains have stronger accuracy constraints since the robot must execute actions in the real world in 3D, leading to the need for error recovery. This has motivated the resurgence of inference in robotics, particularly hybrid learning and probabilistic techniques [75, 40, 142]. These methods take a data-driven approach to inference, leveraging the representational ability and speed of deep learning alongside the robustness of inference.

2.2 Object Pose Estimation

Given a scene containing a set of objects, the problem of pose estimation is concerned with localizing one or more objects of interest. The objective of object localization in the context of robotic manipulation is to enable robots to interact with the objects in question, for example to grasp them and use them towards a task. While there are many ways to express the object localization (e.g. a set of keypoints [100], 3D bounding boxes [184], category-based models [177]), we focus our review on *pose estimation.* The pose of an object is comprised of a 3D position and 3D orientation relative to a reference frame, resulting in a 6D quantity, X. In the context of pose estimation via probabilistic inference, the goal is to infer the posterior distribution over possible poses, $p(X \mid Z)$, where Z is the observation (e.g. camera image) of the environment. The posterior distribution is sometimes referred to as the *belief* over the pose, $bel(X)$.

Pose estimation techniques for rigid-body objects include geometry-based registration approaches [15], generative approaches [155, 41], and end-to-end learning approaches [184, 167, 175]. Generative approaches leverage domain-specific prior information, such as inter-object relationships [155] and environment physics [41]. These provide robustness through grounding, but are often challenging to define and expensive to compute. Deep learning methods, on the other hand, eliminate the need for hand-defined models, relying instead on data, but can be noisy, especially in challenging cluttered scenarios. Of particular interest are hybrid approaches combining data-driven and generative methods. These have been successful under challenging conditions such as background and foreground clutter [118, 109, 40], adversarial environment conditions [28], and uncertainty due to robot actions [156, 188]. This dissertation employs a hybrid approach, in which a learned likelihood is formulated as a factor in a generative inference framework (see Chapter 3).

Articulated Object Pose Estimation. The works mentioned until now are primarily concerned with objects which consist of a single rigid body. *Articulated* object pose estimation is concerned with the problem of localizating an object with one or more movable joints. In this case, the pose estimation can be formulated in terms of joint pose and configuration estimation, (X, θ). Alternatively, the pose of an articulated object with N rigid parts can be represented as a set of poses for each part s, $\mathcal{X} = \{X_s\}_{s=1}^{N}$.

Articulated pose estimation has been approached via probabilistic inference in the context of tracking articulated objects [137, 136, 31]. Due to the added challenge resulting from the higher-dimensionality of the problem, these frameworks have historically been informed by ground-truth initialization or data from joint encoders. Parts-based approaches have been applied to human pose estimation [144, 50, 51] and hand-pose estimation [153]. Within robotics, single-frame estimation has been explored using learning [88] and inference [107, 42], but these have largely relied on large, primarily planar objects and depth data. In Chapter 3, we draw on the success of parts-based approaches for articulated pose estimation, and propose a general framework for estimating pose of articulated objects using their geometric models as priors that relies on both RGB and depth information.

2.3 Planning as Inference

Trajectory planning describes the problem of finding a trajectory, τ, defined as a sequence of actions, $\tau = \{(u_t, x_t)\}_{t=1}^{T}$, where x is the robot state, u is an action, and t is the timestep,over a finite horizon, T. The robot state is governed by dynamics function, $x_{t+1} = f(x_t, u_t)$. The optimal trajectory, τ^*, satisfies the following:

$$\tau^* = \underset{\tau \in \mathcal{T}}{\arg\min} \; c_{\text{term}}(x_T) + \sum_{t=1}^{T-1} c_t(x_t, u_t), \tag{2.1}$$

where $\mathcal{T}$ is the space of possible trajectories, $c_t(x_t, u_t)$ is the *running cost* and $c_{\text{term}}(x_T)$ is the *terminal cost.*[1] The costs are selected by the user based on the problem domain. Trajectory planning has largely been considered as an optimization problem in the style of the formulation in Equation (2.1), alongside optional constraints [131, 139].

Planning as inference [8, 166, 132, 181] instead formulates the planning problem in Equation (2.1) as an inference problem, where the objective is to infer a *distribution* of trajectories. More formally, given an observation of the environment, z, the trajectory optimization problem becomes:

$$\tau^* = \underset{\tau \in \mathcal{T}}{\arg\max} \, p(\tau \mid z), \tag{2.2}$$

where $p(\tau \mid z)$ is the posterior trajectory distribution. Planning as inference is concerned with estimating the posterior distribution. We can further reformulate the problem in terms of the prior, $p(\tau)$, and the likelihood, $p(z \mid \tau)$, by applying Bayes' rule:

$$p(\tau \mid z) = \frac{p(z \mid \tau)p(\tau)}{p(z)} \propto p(z \mid \tau)p(\tau). \tag{2.3}$$

[1]This problem can also be formulated in the continuous domain, in which case the sum in Equation (2.1) becomes an integral. This dissertation employs the discrete trajectory optimization formulation.

Defining the likelihood in terms of the cost,

$$p(z \mid \tau) \propto \exp(-\alpha C(\tau)), \tag{2.4}$$

where $C(\tau) = c_{\text{term}}(x_T) + \sum_{t=1}^{T-1} c_t(x_t, u_t)$, and α is a hyperparameter, yields an equivalence between the optimization problems in Equations (2.1) and (2.2).

The planning as probabilistic inference formulation from Equation (2.2) gained popularity due to its ability to represent uncertainty in planning [132, 81, 78, 181, 19]. Different approaches exist for inferring the posterior trajectory distribution. One approach is to employ variational inference, solving an optimization problem to minimize the Kullback–Leibler (KL) divergence between a candidate posterior and the true posterior [132, 81, 129]. Other approaches employ a nonparametric representation, in which the posterior distribution is approximated by a set of trajectory samples, iteratively updated through importance sampling [78, 181]. Chapter 4 introduces a novel formulation of the posterior for planning as inference, in which the goal is represented as a nonparametric distribution and solved via differentiable nonparametric inference. Chapter 5 demonstrates an approach to multi-robot coordination, formulated as planning as inference applied in a graphical inference context.

2.4 Background: Probabilistic Inference Techniques

In this section, we describe relevant methods for probabilistic inference in the context of inferring a posterior distribution,[2] $p(x \mid z)$, where x is a random vector consisting of unknown variables, and z is a random vector consisting of observed data. Before inference can be performed, the form of the distribution must first be selected. This choice depends heavily on the characteristics of the problem domain.

In the case where x is a discrete variable, then $p(x \mid z) = \Pr(x = X \mid z)$ is the probability that the random variable x is equal to one possible value X, where $\Pr : \mathbb{R} \rightarrow [0, 1]$, which can be explicitly evaluated for each possible discrete value of x. In the case of a continuous random variable, x, the density of x given observed data z is described by a nonnegative function, $p(x \mid z)$. Performing probabilistic inference starts with selecting a form for the density function, which requires introducing assumptions about the underlying data. The problem of inference is then to infer the *parameters* of the density in question. For example, a common choice is the Gaussian distribution, $p(x \mid z) \approx \mathcal{N}(\mu, \Sigma)$. The Gaussian distribution is fully described by its mean, μ, and covariance, Σ. This distribution introduces convenient

[2]Note that we use the term *distribution* to refer to the *probability density function* describing the distribution of the random variable.

mathematical properties, but the assumption of normally distributed data can be too limiting for certain problems.

This dissertation is primarily focused on solving inference problems related to robotic perception and planning, which are typically multi-modal and cannot be fully captured by single-modal Gaussian distributions. In Section 2.2, the posterior in question is the belief over the pose given the robot camera observation, $p(x \mid z)$. Clutter and object symmetries can result in multiple competing modes (see Figure 1.2). In Section 2.3, the posterior represents the distribution of possible trajectories given the environment representation, $p(\tau \mid z)$. These distributions can include multiple valid solution modes, for example, taking different paths around an obstacle or grasping an object from different points (see Figure 1.3). It follows that we require a representation capable of representing multi-modal distributions.

2.4.1 Problem Statement: Nonparametric Inference

Nonparametric inference is concerned with minimizing the assumptions on the distribution of the data. This is accomplished by using models with *infinite* parameters. Of course, it is intractable to perform exact inference in these cases, so we instead consider *approximate inference* techniques. Instead of estimating parameters for a closed-form distribution, we can draw a set of *samples* whose density approximates the true posterior. More formally, the posterior is approximated by a set of N independent samples, or *particles*:

$$\mathbb{X} = \{x^{(i)}\}_{i=1}^{N}, \quad \text{where} \quad x^{(i)} \sim p(x \mid z). \tag{2.5}$$

Depending on the objective, these particles might be used to compute a quantity (e.g. the expectation), to reconstruct a parametric form of the posterior, or to estimate x by drawing directly from the set of particles, $\mathbb{X}$.

In general, this approximation can represent a broad class of distributions given a sufficiently large N compared to finite parametric representations, like the normal distribution. Practically, increasing the size of the sample set will increase the computation time and memory requirements of the algorithm. Below, we review techniques for performing nonparametric inference via a particle set $\mathbb{X}$ that are relevant to the works proposed in this dissertation in Chapters 3 to 5.

2.4.2 Importance Sampling

Importance sampling is a Monte Carlo method [30] for generating samples from a distribution. The objective is to generate samples from a *target distribution*, $f(x)$, which we

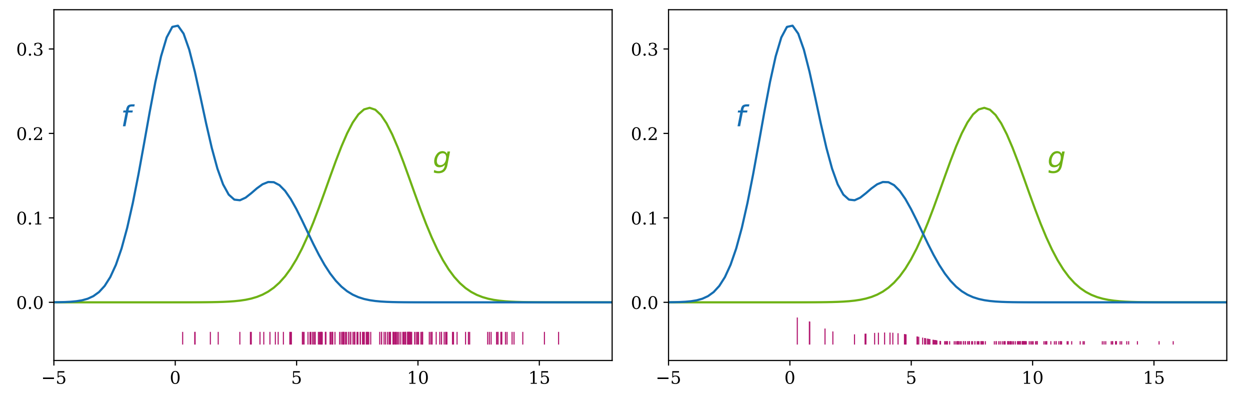


Figure 2.1: Illustration of the importance sampling procedure. The goal is to generate samples from f. Samples are drawn from a proposal distribution, g (left). The samples are then weighed using function f (right). Example inspired by [161].

cannot directly sample from, but which we can evaluate for a given sample, $x^{(i)}$. It works by defining a *proposal distribution*, $g(x)$, which can be sampled from easily. We must have that $f(x) > 0$ implies $g(x) > 0$, meaning there is a non-zero chance of sampling from the proposal distribution everywhere there might be a sample from $f(x)$.

To illustrate, consider the example in Figure 2.1. We cannot directly draw samples from the target distribution $f(x)$, but $g(x)$ is a Gaussian distribution, from which we can generate samples. The left panel shows particles drawn from the proposal distribution $g(x)$. We weigh the particles from $g(x)$ by the ratio:

$$w^{(i)} = \frac{f(x^{(i)})}{g(x^{(i)})} \tag{2.6}$$

The right panel shows the result, where the height of the sample represents its weight.

It is common to apply importance sampling iteratively, by randomly drawing particles from the sample set $\mathbb{X}$ proportional to the weights from Equation (2.6), with replacement. This step is called *resampling*. Since this process results in repeated particles, a common practice is to slightly perturb, or *jitter*, particles before the next iteration. This results in a *reweigh-resample* paradigm.

Derivation of the importance sampling weight. Given a random variable, x, with density $f(x)$, its expectation is defined as follows:

$$\mathbb{E}[x] = \int x f(x)\, dx \tag{2.7}$$

Monte Carlo methods are a class of numerical methods which take advantage of the following approximation:

$$\mathbb{E}\left[x\right] \approx \frac{1}{N}\sum_{i=1}^{N}\phi(x^{(i)}), \quad x^{(i)} \sim f \tag{2.8}$$

where $x^{(i)}$ are independent and identically distributed (i.i.d.) samples drawn with density $f(x)$. Note that we assume that $\mathbb{E}\left[x^2\right] < \infty$ [30].

Since we cannot draw from f, we can instead introduce the proposal distribution, and express the expectation from Equation (2.8) as follows:

$$\mathbb{E}\left[x\right] = \int x\frac{f(x)}{g(x)}g(x)\,dx \tag{2.9}$$

This allows the Monte Carlo approximation from Equation (2.8) to be applied to approximate the expectation from Equation (2.7) as follows:

$$\mathbb{E}\left[x\right] \approx \frac{1}{N}\sum_{i=1}^{N}w^{(i)}\phi(x^{(i)}), \quad w^{(i)} = \frac{f(x^{(i)})}{g(x^{(i)})}, \quad x^{(i)} \sim g \tag{2.10}$$

This trick allows the expectation to be approximated by a weighted average of particles drawn from the proposal distribution, $g(x)$.

Importance Sampling in Robotics. The *reweigh-resample* paradigm based on importance sampling is a has been applied to a breadth of robotics problems based on nonparametric distributions, such as the particle filter for localization [54], pose estimation [41], and planning as inference [181]. The same paradigm has also appeared in more recent work fusing deep learning with inference techniques [89, 40]. In Chapter 3, we use importance sampling in a reweigh-resample paradigm to approximate belief over object pose.

2.4.3 Stein Variational Inference

The objective of variational inference is to infer the parameters of a distribution $q(x)$ such that it is a good approximation of the true posterior of random variable x, $p(x \mid z)$, where z is observed data. Specifically, we want to minimize the KL-divergence:

$$q^*(x) = \arg\min_{q\in\mathcal{Q}} D_{KL}\left(q(x) \mid\mid p(x \mid z)\right) \tag{2.11}$$

where $\mathcal{Q}$ is a family of distributions. Recently, *Stein Variational Inference* has been proposed [92] as a method for solving for $q^*(x)$, represented using a nonparametric distribution.

The Stein identity says that for a density $p(x)$ and some smooth function $\phi(x)$,

$$\mathbb{E}_{x\sim p}\left[\mathcal{A}_p \cdot \phi(x)\right] = 0, \qquad \text{where} \quad \mathcal{A}_p \cdot \phi(x) = \phi(x)\nabla_x \log p(x)^\top + \nabla_x \phi(x) \tag{2.12}$$

$\mathcal{A}_p$ is called the *Stein operator*. Given another density, $q(x)$, $\mathbb{E}_{x\sim q}\left[\mathcal{A}_p\phi(x)\right] \neq 0$.[3] The maximum violation of the Stein identity over all $\phi \in \mathcal{F}$ can be used to define a discrepancy between distributions p and q, called the *Stein discrepancy*:

$$D_S(q,p) = \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}\left[\operatorname{tr}(\mathcal{A}_p\phi(x))\right]^2 \tag{2.13}$$

where $\mathcal{F}$ is a function space. In practice, it is intractable to search over an arbitrary function space $\mathcal{F}$. A key insight is to select $\mathcal{F}$ to be a reproducing kernel Hilbert space (RKHS), $\mathcal{H}^d$, where d is dimension:

$$D_S(q,p) = \max_{\phi\in\mathcal{H}^d} \mathbb{E}_{x\sim q}\left[\operatorname{tr}(\mathcal{A}_p\phi(x))\right]^2 \quad \text{s.t.} \quad \|\phi\|_{\mathcal{H}^d} \leq 1, \tag{2.14}$$

then there is a closed form solution to the Stein discrepancy in Equation (2.13). Given kernel $k(x,x')$ associated with RKHS $\mathcal{H}^d$, the solution is:

$$\hat{\phi}(x) = \frac{\phi^*_{q,p}(x)}{\|\phi^*_{q,p}(x)\|_{\mathcal{H}^d}}, \quad \text{where} \quad \phi^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}\left[\mathcal{A}_p k(x,\cdot)\right], \quad \text{and} \quad D_S(q,p) = \|\phi^*_{q,p}(x)\|_{\mathcal{H}^d} \tag{2.15}$$

It can be shown that $D_S(q,p) = 0 \iff p = q$ when $k(x,x')$ is a valid positive definite kernel. We refer the reader to Lui and Wang [92] for details.

Stein Variational Gradient Descent. Lui and Wang [92] proposed Stein Variational Gradient Descent (SVGD), an iterative approach to variational inference using the results of Stein above. First, we define an incremental transform on x, $T(x) = x + \epsilon\phi(x)$, and $q_{[T]}$ is the new density on $T(x)$. Then,

$$\nabla_\epsilon D_{KL}\left(q_{[T]} \| \, p\right)\Big|_{\epsilon=0} = -\mathbb{E}_{x\sim q}\left[\operatorname{tr}(\mathcal{A}_p\phi(x))\right]. \tag{2.16}$$

This means that $\phi^*_{q,p}$ from Equation (2.15) gives the direction of steepest descent on the KL divergence. The direction of steepest descent is given by:

$$\phi^*_{q,p}(\cdot) = \mathbb{E}_{x\sim q}\left[k(x,\cdot)\nabla_x \log p(x)^\top + \nabla_x k(x,\cdot)\right]. \tag{2.17}$$

[3]Intuitively, the gradients across all $x \sim p$ will average to zero, but if we imagine sampling $x \sim q$, a different distribution, and evaluating $\mathcal{A}_p\phi(x)$, the gradients will not average to zero.

Each step decreases the KL divergence by $\epsilon D_S^2(q, p)$. It follows that the KL divergence between $q_{[T]}$ and p approaches zero as the incremental transform is iteratively applied.

We can approximate Equation (2.17) using Monte Carlo integration as in Equation (2.8), using a set of particles $\{x^{(i)}\}_{i=1}^N$, where $x^{(i)} \sim q$. At an iteration k, the following update rule is applied to each particle $x^{(i)}$:

$$x_k^{(i)} \leftarrow x_{k-1}^{(i)} + \gamma \phi(x_{k-1}^{(i)}) \tag{2.18}$$

$$\phi(x) = \frac{1}{N} \sum_{j=1}^{N} k(x^{(j)}, x) \nabla_{x^{(j)}} \log p(x^{(j)}) + \nabla_{x^{(j)}} k(x^{(j)}, x) \tag{2.19}$$

where $k(x, \cdot)$ is a kernel function. Intuitively, the first part of the equation drives the particles to areas of high probability, and the second part acts as a repulsive force between particles. Compared to sampling-based methods, SVGD is more particle-efficient and can more effectively represent diverse modes in distributions due to the repulsive force component. Additionally, the update steps in Equations (2.18) and (2.19) are deterministic and can be computed in parallel, enabling efficient implementation on modern GPUs.

Stein Variational Inference in Robotics. Stein Variational Inference has been applied across a number of robotics applications, including in control, planning, and point cloud matching [81, 11, 98, 82]. In Chapter 4, we use Stein Variational Gradient Descent (SVGD) in order to formulate a planning as inference problem in the case of goal uncertainty. In Chapter 5, we extend the above formulations to infer marginal distributions on probabilistic graphical models.

2.5 Belief Propagation

Belief Propagation (BP) is an algorithm for inferring the *marginal distributions* of nodes in a graph [174]. Specifically, let $\mathcal{X}$ be a set of N hidden random vectors, $\mathcal{X} = \{x_s\}_{s=1}^N$. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graphical model with nodes $\mathcal{V}$ and edges, $\mathcal{E}$, describing random variables $\mathcal{X}$, known as a Markov Random Field (MRF). The power of the Markov Random Field (MRF) model is that it allows the joint probability of the graph $\mathcal{G}$ can be written as a product of its *clique factors*, $p(\mathcal{X}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(x_c)$, where Z is a normalizing constant. Cliques $\mathcal{C}$ are defined as all fully-connected subgraphs in $\mathcal{G}$, and factors are non-negative functions defined over the random variables.

The objective is to infer the marginal distribution over a node given observed data, $\mathcal{Z}$. For ease of expression, which will prove useful later, we assume that there exists a unique, independent observation corresponding to *each hidden node*, such that $\mathcal{Z} = \{z_s\}_{s=1}^N$. At

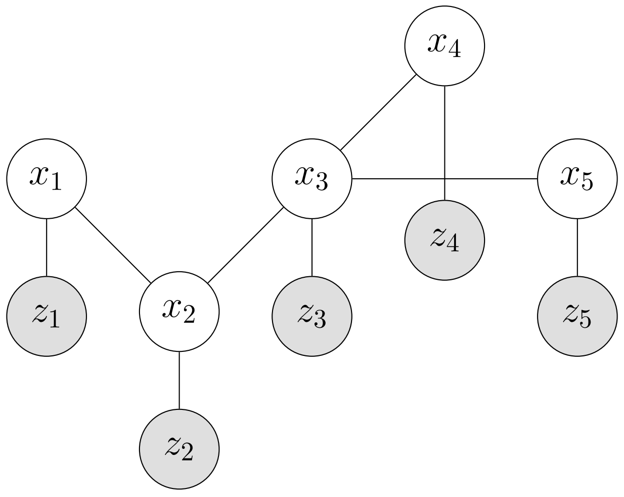


Figure 2.2: An example Markov Random Field (MRF), where the variables $\{x_s\}_{s=1}^5$ are hidden, and the variables $\{z_s\}_{s=1}^5$ are observed.

present, we restrict our discussion to tree- or chain-structured graphs, i.e. those without loops. An example MRF conforming to these properties with $N = 5$ is illustrated in Figure 2.2.

Given an MRF with the properties described, the joint probability can be written as a product of factors as follows:

$$p(\mathcal{X}, \mathcal{Z}) = \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \psi_{s,t}(x_s, x_t) \prod_{s\in\mathcal{V}} \phi_s(x_s, z_s). \tag{2.20}$$

The function $\psi_{s,t}$ is the *pairwise factor*, and ϕ_s is the *unary factor*. Intuitively, the pairwise factor describes the correspondence between neighbouring nodes and the unary factor describes the correspondence of a hidden variable x_s with the observed variable z_s. Given that $\mathcal{Z}$ are observed variables and we are only interested in inferring hidden variables $\mathcal{X}$, it will be useful to use the fact that $p(\mathcal{X}, \mathcal{Z}) = p(\mathcal{X} \mid \mathcal{Z})p(\mathcal{Z})$. In practice, we can assume $p(\mathcal{Z})$ is a constant and can be absorbed into normalization constant Z, enabling Equation (2.20) to be written interchangeably as either the joint density $p(\mathcal{X}, \mathcal{Z})$ or the joint posterior density $p(\mathcal{X} \mid \mathcal{Z})$.

2.5.1 Sum Product Belief Propagation

Sum Product BP estimates the marginal posterior distribution of each hidden variable in graph $\mathcal{G}$, $p(x_s \mid \mathcal{Z})$. The belief for node s can be obtained via a marginalization of the expression in Equation (2.20):

$$p(x_s \mid \mathcal{Z}) \propto \phi_s(x_s, z_s) \prod_{t \in \rho(s)} m_{t \to s}(x_s), \tag{2.21}$$

where $\rho(s)$ denotes the neighbours of s. Given the independence relationship between x_s and z_t, $t \neq s$, $p(x_s \mid \mathcal{Z}) = p(x_s \mid z_s)$. The message from node t to node s, $m_{t \to s}$, is defined as:

$$m_{t \to s}(x_s) = \int_{x_t} \phi_t(x_t, z_t) \psi_{t,s}(x_t, x_s) \prod_{u \in \rho(t) \backslash s} m_{u \to t}(x_t)\, dx_t \tag{2.22}$$

The messages are derived via a straight-forward factorization of the marginalization of Equation (2.20). Sum Product BP gets its name from the integral and product in the message in Equation (2.22). The power of this factorization is that messages can be pre-computed and passed through the graph, using a message passing algorithm that can tractably compute the marginal beliefs. Sum Product BP yields exact marginal posteriors on tree- and chain-structured graphs.

2.5.2 Loopy Belief Propagation

Our discussion up until now has only considered graphs without loops. For loopy graphs, an approximate algorithm known as Loopy BP has been applied [117].This algorithm does not provide exact marginals but has proven effective in practice.[4]

In Loopy BP, messages are initialized to some value, $m^0_{t \to s}(x_s)$, and iteratively updated at iteration k according to the following rule:

$$m^k_{t \to s}(x_s) = \int_{x_t} \phi_t(x_t, z_t) \psi_{t,s}(x_t, x_s) \prod_{u \in \rho(t) \backslash s} m^{k-1}_{u \to t}(x_t)\, dx_t. \tag{2.23}$$

The insight is that instead of recursing through the whole graph as in the tree case, we instead update messages using the most recent messages until convergence. As a consequence, Loopy BP is sensitive to the order in which messages are updated, known as the message passing schedule.

2.5.3 Algorithms for Belief Propagation

In order to apply BP in practice, we must select a form for the distribution $p(x_s \mid z_s)$, and the factors in Equation (2.20). In the case that x_s is discrete, the integral in Equation (2.22)

[4]Loopy BP has convergence guarantees under some conditions.

becomes a sum and can be evaluated over each possible value of the variable. This dissertation is primarily concerned with the case in which x_s is continuous, as is the case in many robotic applications. In the continuous case, the integral in Equation (2.22) becomes intractable to evaluate in practice.

A number of BP algorithms have been proposed in the literature for continuous variables. Gaussian Belief Propagation (GaBP) is an efficient algorithm when the node distributions and their corresponding factors can be represented as Gaussian [179, 37]. This method leverages convenient properties of Gaussians in order to perform the products and marginals required for the evaluation of Equations (2.21) and (2.22) which are efficient to compute. However, many applications in robotics are complex and multi-modal, and cannot be fully represented by unimodal Gaussian uncertainty.

Nonparametric Belief Propagation. Nonparametric Belief Propagation (NBP) [152, 70] represents distributions nonparametrically as mixtures of Gaussians, yielding a flexible representation. However, these algorithms involve expensive product operations between mixture distributions. Particle Belief Propagation (PBP) [69] is another nonparametric BP algorithm which relies on sampling, enabling the representation of arbitrarily complex distributions and offering an efficient message, eliminating the need for expensive message products.

PBP defines a sampling-based algorithm for computing the messages for cases where $\mathcal{X}$ is large and the sums cannot be evaluated in practice. Instead, a set of M particles at each node, $\{x_s^{(i)}\}_{i=1}^N$,is obtained using importance sampling. Given samples $\{x_t^{(j)}\}_{j=1}^M$ drawn from a sample distribution W_t, the PBP message is defined as follows:

$$\hat{m}_{t\to s}(x_s) = \frac{1}{M}\sum_{j=1}^{M}\frac{\phi_t(x_t^{(j)}, z_t)}{W_t(x_t^{(j)})}\psi_{t,s}(x_t^{(j)}, x_s)\prod_{u\in\rho(t)\setminus s} m_{u\to t}(x_t^{(j)}). \tag{2.24}$$

PBP relies on the definition of a sampling distribution, W_t, typically selected by the user. Later work proposed to estimate the sampling distribution via expectation maximization [90]. A more efficient particle-based pull message has also been proposed [42]. Importance sampling is prone to mode collapse, an effect which has been mitigated by using multiple sampling distributions [121].

2.5.4 Belief Propagation in Robotics

Gaussian belief propagation has been shown to be effective for multi-robot collision avoidance and localization [124, 115]. Nonparametric belief propagation has been applied to robotic perception of articulated objects, using an efficient sampling-based message product technique [42], learned unary factors [126], and end-to-end learned factors [120]. While these methods enable complex representations of belief distributions, they rely on expensive sequential sampling operations. In Chapter 3, we apply belief propagation to the challenge of parts-based articulated object pose estimation, in which the unary factors are learned. In Chapter 5, we present a novel nonparametric belief propagation technique that uses SVGD to infer the marginal distributions within the BP framework.

CHAPTER 3

Parts-Based Articulated Pose Estimation with Belief Propagation

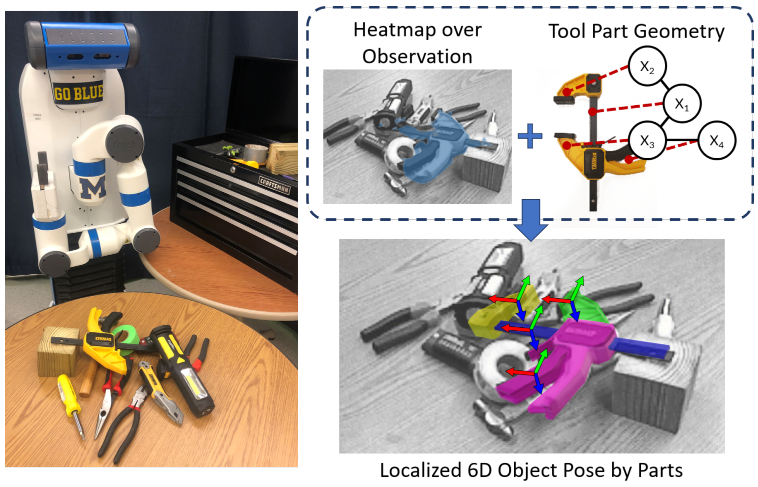


Figure 3.1: The ProgressLab Fetch robot perceiving a cluttered scene with tools (articulated objects): The top right image shows the pixelwise heatmap over the RGB observation for the clamp object, which misses the top part of the clamp as it is occluded by another object. Our parts-based localization method is able to use this partially informed heatmap along with the clamp geometry to localize the 6D pose of the entire object (bottom right image).

3.1 Introduction

Robot assistants operating in real-world environments should be capable of performing maintenance and repair tasks. Going beyond pick-and-place actions, we aim to enable robots to use the diversity of objects it might encounter. The ability to use commercial, off-the-shelf hand tools is critical for robots to perform tasks in unstructured, everyday environments. In order to accomplish this, robots must be able to identify and localize tools in an arbitrary cluttered scene to plan appropriate actions toward performing a task.

Recognizing hand tools and localizing their pose remains challenging in common human environments. These challenges arise from uncertainty caused by physical clutter and the high-dimensionality of the space of poses multiple objects in contact may occupy. Many hand tools are articulated, adding complexity to the localization problem by introducing additional degrees-of-freedom. Figure 3.1 shows one example of hand tools in a cluttered scene that could be typical in a work area.

State-of-the-art object and pose recognition methods have been proposed that estimate the six degree-of-freedom (6D) pose of objects using convolutional neural networks (CNNs) [184, 167]. Other methods have accomplished pose estimation using probabilistic inference [155, 40]. However, localizing *articulated* objects remains a challenge for these methods due to both the added degrees of freedom that arise from articulations and occlusions due to clutter. Parts-based representations [50] have the potential to achieve higher levels of robustness under these conditions than whole-object based approaches. For such methods to be suitable for robot manipulation tasks, they must be able to localize 6D pose with reasonable computational efficiency. We suggest that generative inference methods, if made more computationally efficient, offer compelling and complementary benefits to modern deep learning. Additionally, a parts-based representation can provide information about the affordances of an object, because robot actions are typically applied to the object parts.

In this chapter, we present a method for recognition and localization of articulated objects in clutter suited to robotic manipulation of object affordances. We formulate the problem of articulated object pose estimation as a Markov Random Field (MRF), representing the 6D poses of each rigid object part and the articulation constraints between them. We propose a method to perform inference over the MRF based on message passing. We are inspired by work by Desingh et al. [42], in which parts-based articulated object localization is facilitated by combining information from both the observation as well as the compatibility with neighbouring parts within the inference process. Our method is informed jointly by a learned likelihood modelled by a CNN, as well as by the known articulation constraints between each component part. We assume known object mesh models and kinematic constraints in the form of a Unified Robot Description Format (URDF) file, a standard geometrical object representation in the field of robotics.

By employing generative inference to integrate both data-driven techniques and domain knowledge about the object models, we leverage the speed and representational abilities of deep CNNs, while retaining the ability to reconcile noisy results and provide structure and context to the estimate. Methods we present emphasize novel synthesis of (1) efficient discriminative-generative inference via nonparametric belief propagation for pose estimation of articulated objects, and (2) a learned part-based likelihood to evaluate hypotheses of articulated object pose against RGB observations. We present results using a custom dataset made up of commercial, off-the-shelf hand tools with robot observations containing varying levels of clutter.

3.2 Related Work

Pose estimation has received considerable attention in robotics. Here, we discuss related work that focuses on rigid body, parts-based, and articulated object pose estimation.

3.2.1 Rigid Body Pose Estimation

Methods that tackle the problem of rigid body pose estimation include geometry-based registration approaches [15], generative approaches [155, 41], approaches combining discriminative and generative methods [118, 156, 28, 188, 109, 40], and end-to-end learning approaches [184, 167]. Here, we focus on the discriminative-generative methods and end-to-end learning methods that are most relevant to this work.

Combining the discriminative power of feature-based methods with generative inference has been successful under challenging conditions such as background and foreground clutter [118, 109, 40], adversarial environment conditions [28], and uncertainty due to robot actions [156, 188]. We are inspired by the success of the above approaches in taking advantage of the speed of discriminative methods to perform analysis and synthesis based generative inference.

Xiang et al. propose an end-to-end network for estimating 6D pose from RGB images [184]. This work was further extended to use synthetic data generation and augmentation techniques to improve performance [167]. Wang et al. [175] propose an end-to-end network that uses depth information along with RGB information. These methods rely significantly on the textured appearance of objects. More importantly, the state representation used in these methods assume rigidity. Our attempts to adapt these methods for articulated objects required considerably more training data and computation time. In addition, estimates from the end-to-end methods can be noisy, especially in challenging cluttered scenarios. We believe estimates from these methods will be a good prior to help generative methods recover under challenging scenarios. Hence, in this work, we learn a likelihood function over the observation to inform the generative inference.

3.2.2 Parts-Based Pose Estimation

Understanding objects in terms of their parts paves the way to meaningful and purposeful action execution, such as tool-use. Parts-based representations have been proposed to aid scene understanding and action execution [50, 51, 183], and have recently garnered attention within the robotics and perception communities [110, 93]. Parts-based localization has led to research in recognizing objects and their articulated parts [186]. Parts-based perception for objects in human environments is often limited to recognition and classification tasks. Parts-based pose estimation is often considered for human body pose [144] and hand pose [153] estimation with fixed graphical models. Here, we propose a general framework for estimating pose of articulated objects, such as hand tools, that includes parts with fixed transforms as constraints.

3.2.3 Articulated Pose Estimation and Tracking

Probabilistic inference is a popular technique in robot perception for articulated body tracking [31, 137, 136], where filtering-based approaches alongside novel observation models have been proposed. These tracking frameworks are either initialized to the ground truth poses of objects, or applied to robot manipulators, where the inference is informed by joint encoder readings. In this work, we aim to perform pose estimation of multiple articulated objects using a single RGB-D frame with weak initialization from pixel-wise segmentations.

Interactive perception [21] for articulated object estimation [62] has been a problem of interest in the robotics community. Various works [103, 151, 150], propose methods for estimating kinematic models from demonstration of manipulation or articulation examples. We instead focus on using known kinematic models to estimate the objects in challenging cluttered environments.

Li et al. [88] explore category-level localization of articulated bodies in a point cloud, however their method does not consider clutter and occlusions from the environment. Michel et al. [107] perform one-shot pose estimation of articulated bodies using 3D correspondences with optimization over hypotheses. Desingh et al. consider pose estimation of articulated objects in cluttered scenarios using efficient belief propagation [42], but do not consider RGB information. All of these approaches consider large, primarily planar objects that cover significant portion of the observation as opposed to the small objects in clutter in this work.

Li et al. [87] developed techniques to handle the challenges of hand tools and small objects with no articulation, however the techniques proposed require multi-viewpoint information, as opposed to the single image approach that we propose.

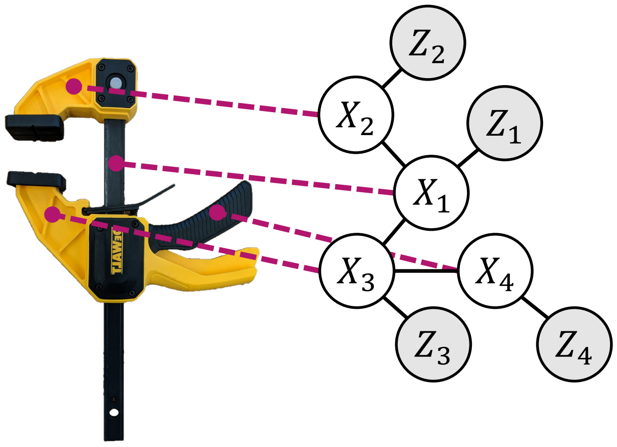


Figure 3.2: The Markov Random Field representation of the clamp object. The clamp is broken up into four parts to fully represent both its affordances and articulations. The hidden nodes (white) represent the pose of each part, X_s. The observed nodes (grey) correspond to the sensor observation.

3.3 Problem Statement

This work considers the problem of object pose estimation, as described in Section 2.2. Given a scene containing objects O, such that $\{O_k\}_{k=1}^{K}$ is the set of K relevant objects, we wish to localize each object O_k. The state of an object O_k is represented by the set of part poses $\mathcal{X} = \{X_s\}_{s=1}^{P_k}$, where X_s is the 6D pose of an articulating rigid part s of O_k, with P_k parts. Each object O_k in the scene is estimated independently.

This estimation problem is formulated as a Markov Random Field (MRF), $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, an undirected graph with nodes $\mathcal{V}$ and edges $\mathcal{E}$. An example MRF for an object considered in this work is illustrated in Figure 3.2. The problem of pose estimation of an articulated model O_k is interpreted as the problem of inferring the collection of part poses $\{X_s\}_{s=1}^{P_k}$, that maximizes the joint probability of the graph $\mathcal{G}$. Using the results from Section 2.5, the joint probability of the graph $\mathcal{G}$ for parts-based pose estimation is expressed as:

$$p(\mathcal{X}, \mathcal{Z}) \propto \prod_{(s,t)\in\mathcal{E}} \psi_{s,t}(X_s, X_t) \prod_{s\in\mathcal{V}} \phi_s(X_s, Z_s), \tag{3.1}$$

where $\mathcal{X}$ denotes the hidden pose variables to be inferred and $\mathcal{Z} = \{Z_s\}_{s=1}^{P_k}$ denotes the observed sensor information in the form of an RGB-D image for each part. The function $\psi_{s,t}$ is the *pairwise potential*, describing the correspondence between part poses based on the articulation constraints, and ϕ_s is the *unary potential*, describing the correspondence of a part pose X_s with its observation Z_s. The problem of pose estimation of an articulated object O_k is interpreted as the problem of estimating the marginal distribution of each part pose, called the belief, $bel(X_s)$, as a marginalization of the joint probability as expressed in Equation (3.1).

In addition to the sensor data, the articulation constraints and 3D geometry of the object, in the form of a Unified Robot Description Format (URDF), and the 3D mesh models of the objects are provided as inputs. We assume that the object articulations are produced by either fixed, prismatic or revolute joints. We consider scenes which contain only one instance of an object. In Section 3.4, our proposed inference mechanism is detailed, along with a description of our modelling of the potentials in Equation (3.1).

3.4 Methodology

Belief propagation via iterative message passing is a common approach to infer hidden variables while maximizing the joint probability of a graphical model [42, 153]. We adopt the sum-product iterative message passing approach to perform inference described in Sections 2.5.1 and 2.5.2, where messages are passed between hidden variables until their beliefs converge. A message, denoted by $m^n_{t\to s}(X_s)$, can be considered as the belief of the receiving node s as informed by its neighbor t at iteration n. An approximation of the message, denoted by $\hat{m}^n_{t\to s}(X_s)$, is computed using the incoming messages to t:

$$\hat{m}^n_{t\to s}(X_s) = \sum_{X_t \in \mathbb{X}_t} \phi_t(X_t, Z_t)\psi_{s,t}(X_s, X_t) \prod_{u\in\rho(t)\setminus s} \hat{m}^{n-1}_{u\to t}(X_t) \tag{3.2}$$

where $\rho(t)$ denotes neighboring nodes of t, and $\mathbb{X}_t$ denotes the particle set of node t.

The marginal belief of a hidden node is a product of all the incoming messages weighted by the node's unary potential:

$$bel^n_s(X_s) \propto \phi_s(X_s, Z_s) \prod_{t\in\rho(s)} \hat{m}^n_{t\to s}(X_s) \tag{3.3}$$

Our particle optimization algorithm aims to approximate the joint probability of the MRF, as in Equation (3.1), by maintaining the marginal belief, as in Equation (3.3) for each object part. The belief of a rigid part pose, $bel_s(X_s)$, is represented nonparametrically as a set of N weighted particles $\mathbb{X}_s = \{X^{(i)}_s, w^{(i)}_s\}^N_{i=1}$.

Section 3.4.1 describes the message passing algorithm. Section 3.4.2 describes how the function $\phi_s(X_s, Z_s)$ is represented, and Section 3.4.3 describes the function $\psi_{s,t}(X_s, X_t)$.

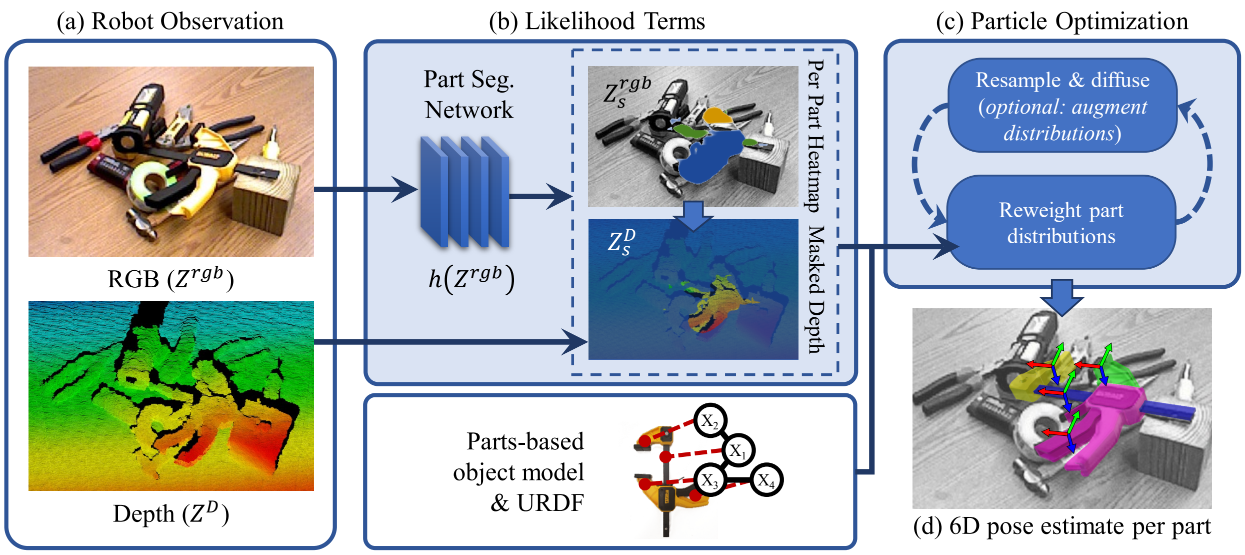

(a) Robot Observation
RGB (Z^{rgb})
Depth (Z^D)
(b) Likelihood Terms
Part Seg. Network
$h(Z^{rgb})$
Z_s^{rgb}
Z_s^D
Per Part Heatmap
Masked Depth
Parts-based object model & URDF
X_1
X_2
X_3
X_4
(c) Particle Optimization
Resample & diffuse (optional: augment distributions)
Reweight part distributions
(d) 6D pose estimate per part

Figure 3.3: The inference pipeline. (a) The robot observes a scene as an RGB-D image, $\mathcal{Z} = (Z^{rgb}, Z^D)$. (b) The RGB image is passed through a trained part segmentation network, $h(Z^{rgb})$, that generates a pixel-wise heatmap for the P_k parts of an object class of interest, $\{Z_s^{rgb}\}_{s=1}^{P_k}$ (in this example, the clamp, which has one fully occluded part). The heatmaps are used to generate masked depth images, $\{Z_s^D\}_{s=1}^{P_k}$ (c) The inference is initialized with part poses using these heatmaps and the depth image. Hypotheses are iteratively reweighed using Equation (3.4), and resampled with importance sampling. (d) The inference process generates an estimate of the 6D pose of each part. (Best viewed in color).

3.4.1 Belief Propagation via Message Passing

Our method adopts the traditional reweigh and resample paradigm for particle refinement methods, as described in Section 2.4.2. The particles are first *reweighed* using an approximated sum-product message. The particles are then *resampled* using importance sampling based on the calculated weights.

The high-dimensional nature of the estimation problem and the cluttered settings with similar parts and partial observations make the inference prone to convergence to local minima. To mitigate this problem while computing messages, we can optionally add an augmentation step before the reweight step to accommodate different proposals. The augmentation technique is adapted from Pacheco et al. [122] and is discussed in Section 3.4.4.

The overall system is summarized in Figure 3.3.

Reweighing and Resampling Steps. Each particle $X_s^{(i)} \in \mathbb{X}_s$ is reweighed as follows:

$$w_s^{(i)} = \phi_s(X_s^{(i)}, Z_s) \prod_{t \in \rho(s)} \hat{m}_{t \to s}^{n}(X_s^{(i)}) \tag{3.4}$$

where $\hat{m}^n_{t\to s}(X_s^{(i)})$ is the sum-product message:

$$\hat{m}^n_{t\to s}(X_s^{(i)}) = \sum_{X_t^{(j)}\in\mathbb{X}_t} \psi_{s,t}(X_s^{(i)}, X_t^{(j)})\phi_t(X_t^{(j)}, Z_t) \tag{3.5}$$

which only takes into account the immediate neighbors of the node. Since the number of parts in each object is small, this approximation has negligible effect in practice and saves computation time. For numerical stability, the log-likelihoods are used in practice. The weights are normalized and then the particles are resampled using importance sampling. The object pose estimate is made by selecting the maximum likelihood estimates (MLE) from each of the marginal beliefs.

3.4.2 Unary Likelihood

The unary potential represents the compatibility of each pose hypothesis with the RGB-D observation, $\mathcal{Z}$. The RGB and depth portions of the observation, Z^{rgb} and Z^D, are treated as independent such that the unary likelihood is:

$$\phi_s(X_s, Z_s) = \phi_s^{\text{rgb}}(X_s, Z_s^{rgb})\phi_s^D(X_s, Z_s^D) \tag{3.6}$$

where ϕ_s^D and ϕ_s^{rgb} are the likelihoods with respect to depth and RGB parts of the observations.

RGB Unary Likelihood. The RGB portion of the unary likelihood makes use of the Dilated ResNets architecture [187]. This architecture maintains a high dimensional feature space which is beneficial for semantic segmentation tasks.

The CNN outputs a pixelwise score for each object part class s. We apply a sigmoid function so the final scores lie between zero and one. This constitutes a learned heatmap $Z_s^{rgb} = h_s(Z^{rgb})$ over an RGB observation Z^{rgb}, where h is the Dilated ResNets model trained on parts and $h_s(\cdot)$ is the output indexed at class s. For each particle hypothesis X_s, we generate a mask $\mathcal{M}_s$ over the image for the object part at the hypothesis pose. We transform the mesh model of the part to pose X_s and use the camera parameters to obtain a corresponding binary mask in image space. We represent the likelihood of a particle X_s over the heatmap Z_s^{rgb} using the Jaccard index [113], commonly called the Intersection over Union (IoU), between the heatmap and the rendered mask:

$$\phi_s^{\text{rgb}}(X_s, Z_s^{rgb}) = \frac{|\mathcal{M}_s \cap Z_s^{rgb}|}{|\mathcal{M}_s| + |Z_s^{rgb}| - |\mathcal{M}_s \cap Z_s^{rgb}|} \tag{3.7}$$

The CNN is trained using the analagous intersection over union (IoU) loss, and as such, $\phi_s^{\text{rgb}}(X_s)$ represents a learned likelihood function over the image.

Depth Unary Likelihood. For a given part s, depth observation Z_s^D is generated using a threshold over the heatmap Z_s^{rgb} to mask the depth image Z^D. For a particle X_s, $\phi^D(X_s, Z_s^D)$ is the exponential of the negative average pixelwise error between Z_s^D and the mesh model of part s, rendered at pose X_s. The error is only evaluated over areas in which the two depth images overlap. If there is no overlap between the masked observation and the hypothesis, we assign a maximum error instead, which is a chosen constant.

3.4.3 Pairwise Likelihood

The pairwise likelihood between neighbouring particles $\psi_{t,s}(X_t, X_s)$ measures how compatible X_s is with respect to X_t. If X_s falls within the joint limits of s with respect to t at pose X_t, then $\psi_{t,s}(X_t, X_s) = 1$. Otherwise, the likelihood is the exponential of the negative error between X_s and the nearest joint limit. We refer to [42] for further details.

3.4.4 Particle Augmentation

At each node s, the particle set $\mathbb{X}_s$ can be augmented by drawing particles from various proposal distributions. Given N particles in $\mathbb{X}_s$, Gaussian noise is first added to the current particles, then the distribution is augmented to $\mathbb{X}_s^{prop} = \mathbb{X}_s \cup \mathbb{X}_s^{aug}$, where $\mathbb{X}_s^{aug}$ represents the particles generated from the augmentation procedure q. The set $\mathbb{X}_s^{prop}$ contains αN particles, where $\alpha > 1$. Various proposals q_s^{pair}, q_s^{unary}, and q_s^{rand}, as described below can be used to augment the particle set. This optional variant is evaluated and discussed in the results section.

Pairwise: The pairwise proposal distribution $q_s^{pair}(X_s) \propto \psi_{s,t}(X_s, \tilde{X}_t)$ is conditioned on a sample $\tilde{X}_t$, drawn from neighboring node t. Using the known geometric relationship between nodes t and s, a compatible proposal for node s, $\tilde{X}_s$, is generated from $\tilde{X}_t$.

Unary: The unary proposal distribution $q_s^{unary}(X_s) \propto \phi_s(X_s, Z_s)$ draws samples based on the unary potential ϕ_s.

Random: The random proposal distribution $q_s^{rand}(X_s) \propto \mathcal{N}(X_s, \Sigma)$ draws additional noisy samples. This can be used to avoid the belief falling into a local minima due to the high dimensionality of the orientation space, and to account for mirror symmetry in some objects.

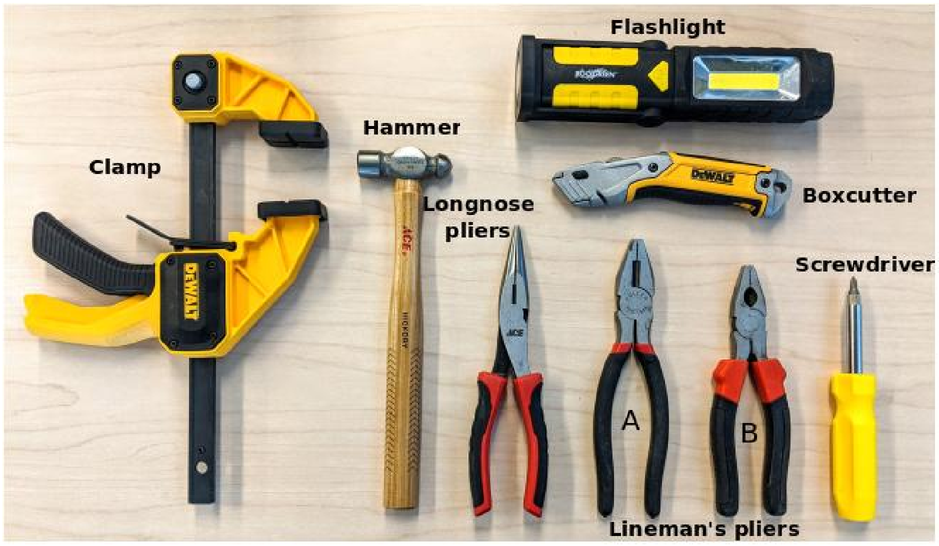


Figure 3.4: Objects in our custom dataset. The clamp has one prismatic joint. The three sets of pliers each have one revolute joint. The other four objects are treated as rigid. The objects are separated into parts based on the presence of both affordances and articulations.

3.5 Experiments

We evaluate our methods for articulated object localization in uncluttered and cluttered scenes. We run experiments on each component of our method and provide an analysis of their effects. These results provide quantitative and qualitative evidence of the accuracy and practicality of our methods.

We test on 20 uncluttered and 17 cluttered test scenes, unseen in the training data. We localize 196 total object instances in these scenes. We do not include results on objects which are severely or fully occluded such that there is no clear observation of any part. We remove 19 objects which fall into this category. An example of such a case is shown in the highly cluttered scene in Figure 3.7a, where the flashlight (behind the hammer) and lineman's pliers (behind the clamp) are almost entirely occluded.

3.5.1 Dataset & Training

Our custom dataset consists of hand tools with eight distinct tool instances: hammer, clamp, boxcutter, flashlight, screwdriver, longnose pliers, and two instances of lineman's pliers (see Figure 3.4). We collect videos of both cluttered and uncluttered scenes using the Fetch Mobile Manipulator's onboard Primesense Carmine 1.09 sensor. The articulated hand tools span the full range of possible articulations in the data. Semantic masks and 6D poses for the objects are labelled using Label Fusion [101], which generates annotations for each video once the first scene is manually labelled. Semantic part masks and part poses are calculated using the object URDFs. The pixels in the images which do not correspond to a tool part are given class label "background." After downsampling to remove adjacent frames in the videos, the dataset contains $\sim 6k$ RGB-D images of 640×480 pixel resolution.

We train the Dilated ResNets *DRN-D-22* architecture [187] to perform semantic segmentation on 90% of the dataset, and reserve 10% for validation. We further augment the training images with random crops, flips, and rotations. We increase the training set size by applying two transforms per image. The backbone is pre-trained on ImageNet, and the last layers are finetuned on our dataset. We employ the Intersection over Union (IoU) loss with an Adam optimizer. We train for 10 epochs on a RTX 2080 Max-Q GPU.

3.5.2 Implementation Details

Our implementation performs efficient unary potential computation on the GPU, to evaluate the heatmap from DRN and to generate binary masks and depth images for pose hypotheses. The current implementation is vectorized and processes all object parts in $\sim 0.5 - 2s$ for one iteration with 300 particles. The computation time could be further reduced with more efficient implementation.

The x and y locations of particle poses are initialized randomly in areas corresponding to high heat pixels of the heatmap over the RGB observation. The z-axis is initialized to the corresponding depth in the observed depth image. The initial orientations are uniformly distributed. For completely occluded parts which do not appear on the segmentation mask, we generate compatible poses from the neighbour initializations.

3.5.3 Evaluation Metric

For evaluation, we use the average point matching error proposed by Hinterstoisser et al. [65], which measures the average point pairwise distance between the rigid object model's point cloud in the ground truth and estimated poses:

$$m(P_{gt}, \hat{P}) = \frac{1}{\mathbb{N}} \sum_{(p_{gt}, \hat{p}) \in (P_{gt}, \hat{P})} ||\hat{p} - p_{gt}|| \tag{3.8}$$

where $(p_{gt}, \hat{p}) \in (P_{gt}, \hat{P})$ are corresponding points in the ground truth and estimated point clouds respectively, each with $\mathbb{N}$ points in the rigid object model. We also report the symmetric point matching error, which measures the average pairwise distance between points in the estimated point cloud and the *nearest* point in the ground truth point cloud:

$$m_{\text{sym}}(P_{gt}, \hat{P}) = \frac{1}{\mathbb{N}} \sum_{\hat{p} \in \hat{P}} \min_{p_{gt} \in P_{gt}} ||\hat{p} - p_{gt}|| \tag{3.9}$$

The symmetric matching error represents the error in symmetric objects, such as the screwdriver, better by not penalizing estimates rotated around a degree of symmetry in the object. However, it tends to provide artificially low errors for incorrect estimates.

3.5.4 Baselines

We implement two baselines, described below.

Segmentation with ICP (DRN+ICP): We initialize the 3D position of the particle hypotheses using the depth image and segmentation mask generated by Dilated ResNets, with random orientations. We use Iterative Closest Point (ICP) [15] to find the transform from the initialized point cloud to the observed point cloud. ICP works best on local refinements, and is prone to failure when the initial orientation is incorrect. To accommodate for this failure, we generate N proposal poses per part, perform ICP, and select the one with the best final fitness score as the estimate. In our experiments, $N = 20$. A similar method is used by Wong et al. [182].

Part-based Particle Filter (Parts-PF): This baseline consists of independent particle filters at each tool part. We use the unary potential from Equation (3.6) to calculate the weights for each hypothesis, and use importance sampling to select particles at each iteration. We use 300 particles per part, and run for 85 iterations.

For both *DRN+ICP* and *Parts-PF* baselines, if a part is completely occluded in the image, a pose estimate cannot be generated from the segmentation. In such cases, we randomly select a neighboring part for which an estimate was made and use the object model to generate a corresponding pose for the occluded part. If the edge between the parts is articulated, a joint value is uniformly sampled within the joint limits.

3.5.5 Parts-Based Pose Estimation

To fully understand the performance of our method, we perform an ablation study over the components of the proposed method. We focus on three factors: the message passing (*MP*), the use of *RGB* only vs. the inclusion of depth (*RGB-D*) in the unary potential, and the augmentation step (*Aug*). We use 300 particles (before augmentation) for all experiments. The choice of particles was observed qualitatively to achieve sufficient results in most cases. While representation of the underlying belief improves with more particles, computation

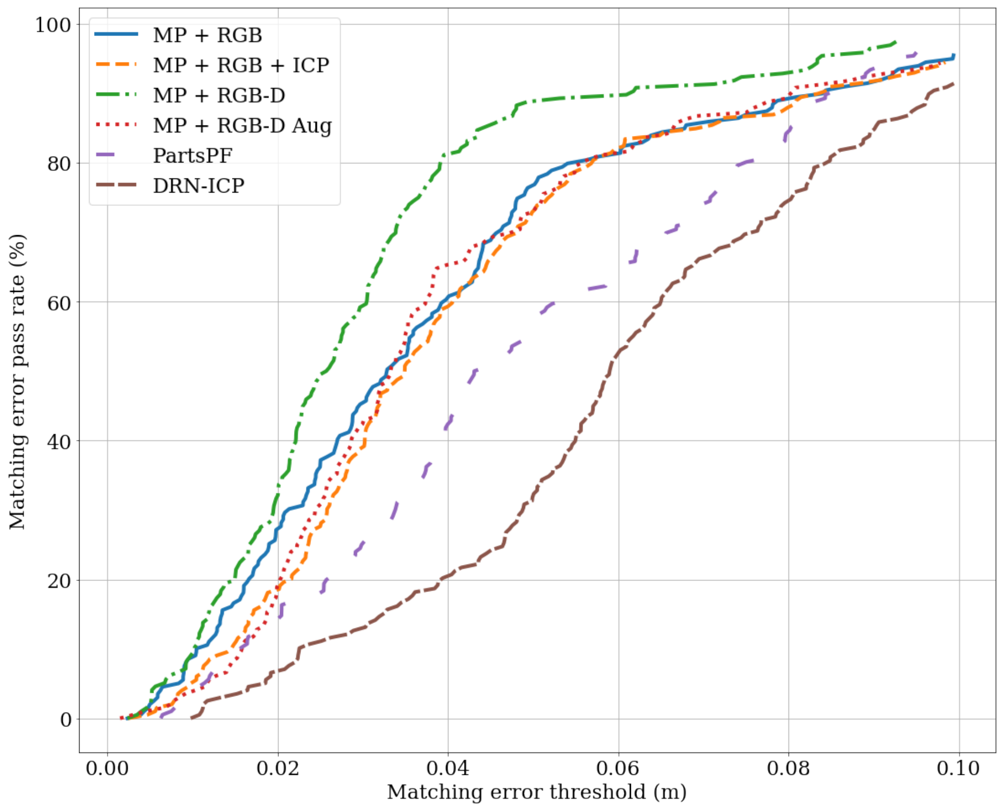


Figure 3.5: Average pairwise distance pass rate for each described method. All tools use the matching error described in Equation (3.8), except the screwdriver, which due to its symmetrical nature, uses the symmetrical form (m_{sym}).

Table 3.1: Average matching errors m (cm) and symmetric matching errors m_{sym} (cm)

Method	m	m_{sym}
DRN+ICP	6.47 ± 6.32	3.65 ± 8.22
Parts-PF	5.23 ± 3.74	2.03 ± 2.98
MP+RGB	4.06 ± 3.40	2.41 ± 2.50
MP+RGB+ICP	4.28 ± 3.32	2.64 ± 2.44
MP+RGB-D+Aug	4.21 ± 3.23	2.51 ± 2.27
MP+RGB-D	**3.08** ± 2.32	**1.57** ± 1.56

Table 3.2: Average matching errors m (cm) and symmetric matching errors m_{sym} (cm) for cluttered and uncluttered scenes

Method	**Cluttered**		**Uncluttered**	
	m	m_{sym}	m	m_{sym}
DRN+ICP	6.59 ± 3.36	3.31 ± 2.69	6.38 ± 8.00	3.93 ± 10.84
Parts-PF	6.03 ± 4.64	2.73 ± 4.04	4.55 ± 2.57	1.43 ± 1.33
MP+RGB	4.57 ± 3.96	2.83 ± 2.95	3.60 ± 2.75	2.04 ± 1.95
MP+RGB+ICP	4.77 ± 3.89	3.03 ± 2.88	3.85 ± 2.65	2.29 ± 1.90
MP+RGB-D+Aug	4.80 ± 3.80	2.90 ± 2.60	3.71 ± 2.55	2.18 ± 1.88
MP+RGB-D	$\mathbf{3.58} \pm 2.56$	$\mathbf{1.89} \pm 1.84$	$\mathbf{2.65} \pm 2.01$	$\mathbf{1.30} \pm 1.21$

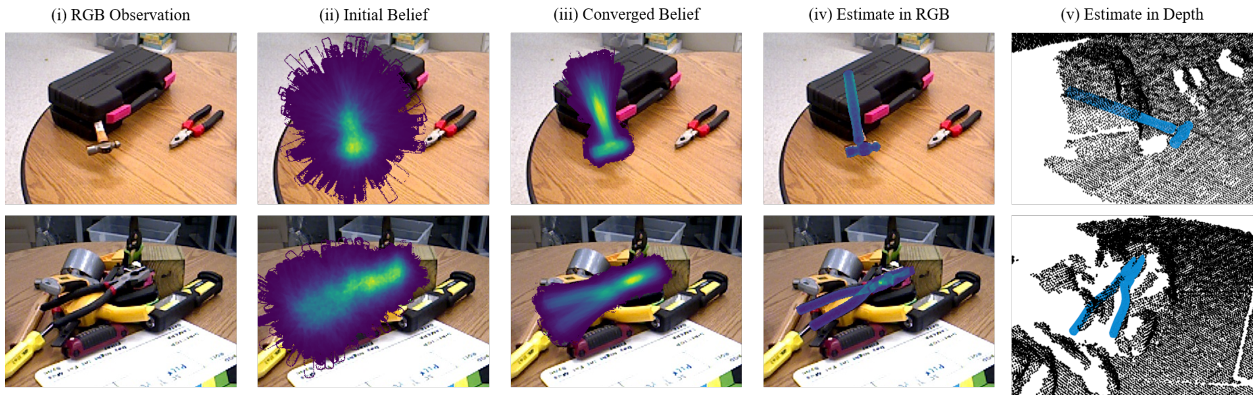
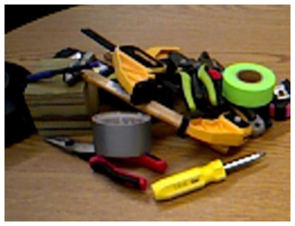


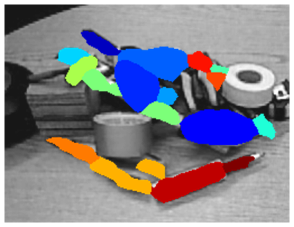
Figure 3.6: Qualitative results for each stage of the *MP+RGB-D* method on cluttered scenes. The method results in accurate pose estimates despite significant occlusions (top) and articulations (bottom).

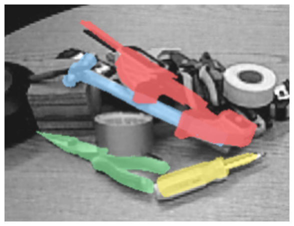
becomes intractable for very large particle sets. We run each method for 100 iterations, after which we observe little change in the estimate.

The results of each method are shown in Table 3.1. We further examine the results for cluttered and uncluttered scenes in Table 3.2. Figure 3.5 shows the results for all scenes. Message passing leads to superior results compared to the baselines, *DRN-ICP* and *Parts-PF*, which do not use message passing. Best performance is achieved by using the full RGB-D observation (*MP-RGB-D*). Further description and analysis of each method is provided below.

Message Passing: RGB Unary (MP+RGB): To test the effect of the depth component of the unary potential, we evaluate using only the RGB component, informed by the heatmap, such that Equation (3.6) becomes $\phi_s(X_s, Z_s) = \phi_s^{\text{rgb}}(X_s, Z_s^{rgb})$. We use message passing to calculate the final likelihood for each particle using the pairwise potential, as


(a) RGB input image


(b) Segmentation mask from CNN


(c) Localized objects in RGB image

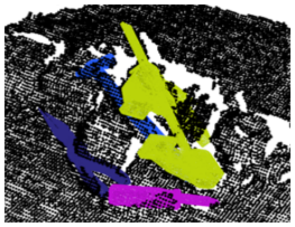

(d) Localized objects in depth point cloud

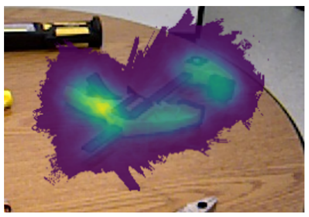
Figure 3.7: Qualitative results for localization of each object in a cluttered scene using the *MP + RGB-D* method. Although the segmentation map (b) is missing information for occluded parts, our iterative method is able to recover the 6D pose of the parts (c), (d).

described in Equation (3.4). Using only the RGB image, we obtain lower accuracy on the pose estimates. The RGB image captures the position and orientation of the objects well *in image space* (see Figure 3.8(c)), but is prone to falling into local minima in the axes which are not well represented by the image, namely z, pitch, and roll (see Figure 3.8(d)).

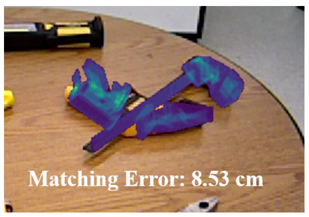
Message Passing: RGB Unary and ICP (MP+RGB+ICP): To attempt to recover from the errors in z, pitch, and roll, we add an ICP step on the final estimate from *MP+RGB* to align it to the masked depth image. We estimate the offset in the z-axis based on the depth image. Since ICP is a local refinement method which relies on an accurate estimation of the initial transform, the ICP step does not always reconcile the orientation error in pitch and roll, for which the initial transform is unknown.

Message Passing: RGB-D Unary (MP+RGB-D): We hypothesize that by including depth information in the unary potential, we can more reliably estimate the full 6D pose of the parts and make up for missing information in the 2D image. The depth term improves the estimation accuracy by discouraging all unoccluded particles from deviating from the depth image at each iteration. This performs better than *MP+RGB+ICP* because the latter only attempts to align to the depth image in the final iteration, where it often has converged to a local minimum. This is the best performing method. Selected qualitative results are shown in Figures 3.6 and 3.7.

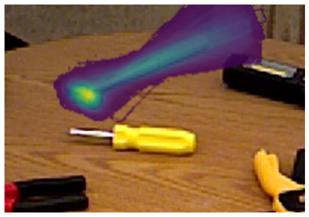
Message Passing with Augmentation (MP+RGB-D+Aug): Using the unary informed by both RGB and depth, as well as message passing, we augment the particle set at the beginning of each iteration as described in Section 3.4.1. We use $\alpha = 1.5$, with 5% of the additional particles drawn from the unary distribution. At the first iteration, the remaining 95% of the particles are drawn from q^{rand}. The percentage of particles drawn from q^{pair} is increased by 10% every 5 iterations, and the percentage from q^{rand} is decreased, up to a maximum of 90% of particles from q^{pair}.



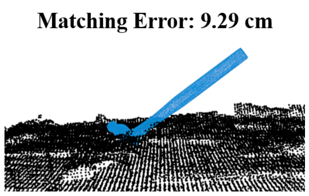
(a) Belief for all parts from *Parts-PF*.



(b) Estimate for each part from *Parts-PF*.



(c) Belief for all parts from *MP+RGB*



(d) Estimate for each part from *MP+RGB*

Figure 3.8: Common failure cases. The first row is a common failure of the baseline method, *Parts-PF*, which does not use message passing. In (a) the parts are clustered in the correct positions in the image, but (b) shows that the part estimates are oriented incorrectly in an incompatible configuration. The second row is a common failure case in *MP+RGB*. In (c), the particles have converged in the 2D image, but in (d), the hammer 6D pose is rotated about an axis which is not well represented in the 2D image plane.

Qualitatively, we observe that the augmentation step leads to quicker convergence in some highly cluttered scenarios. On average, this method performs worse than *MP+RGB-D* in some cases because it is susceptible to propagating incorrect estimates with artificially inflated pairwise scores, due to the addition of perfectly compatible pose estimates. However, these results depend on careful selection of parameters, and might be improved by further tuning. Further analysis of the effect of the parameters on the final estimate is left to future work.

3.5.6 Analysis on Tool Classes

The results for the *MP+RGB-D* and *MP+RGB-D+Aug* methods for each object in the dataset are shown in Table 3.3. We present the percentage of class indices which have error under 4 cm. We observed high error in the flashlight which is likely due to its symmetrical nature. The unary potential does not explicitly encode texture information, so geometrically symmetric parts can tend to flip. The clamp is among the most difficult objects to localize due to its high-dimensionality and significant self-occlusions.

3.5.7 Qualitative Analysis

Selected examples of the *MP+RGB-D* method are shown in Figure 3.6. We show a selected scene which demonstrate the effectiveness of the *MP+RGB-D* method at localizing hand tools, even under partial or full occlusion of some of their parts, in Figure 3.7. While the heatmap may provide little to no information for some parts, by leveraging geometric

Table 3.3: Fraction of tools with matching error less than 4 cm. All objects are evaluated with the average pairwise matching error m, except the screwdriver which uses the symmetric matching error, m_{sym}.

Class	*MP+RGB-D*	*MP+RGB-D+Aug*
clamp	0.677	0.387
hammer	0.920	0.400
longnose pliers	1.000	0.938
lineman's pliers A	0.909	0.818
lineman's pliers B	0.917	0.833
boxcutter	0.833	0.750
flashlight	0.640	0.640
screwdriver (m_{sym})	*0.759*	*0.689*

information through message passing, we are able to resolve the pose of all the visible tools in the scene.

3.6 Conclusion

In this chapter, we present an inference technique for estimating articulated parts-based object pose in clutter. We model the part poses of each articulated object as a Markov Random Field (MRF) and perform efficient particle-based belief propagation. We use articulation constraints between parts and a novel learned likelihood function to perform message passing in the MRF. We perform a thorough analysis of our method and show that it performs well on both uncluttered and cluttered scenes. We demonstrate that the message passing step is highly beneficial in terms of enforcing geometric consistency to inform pose estimation in the high dimensional space of 6D articulated object pose.

CHAPTER 4

Stein Variational Inference for Planning to Goal Sets

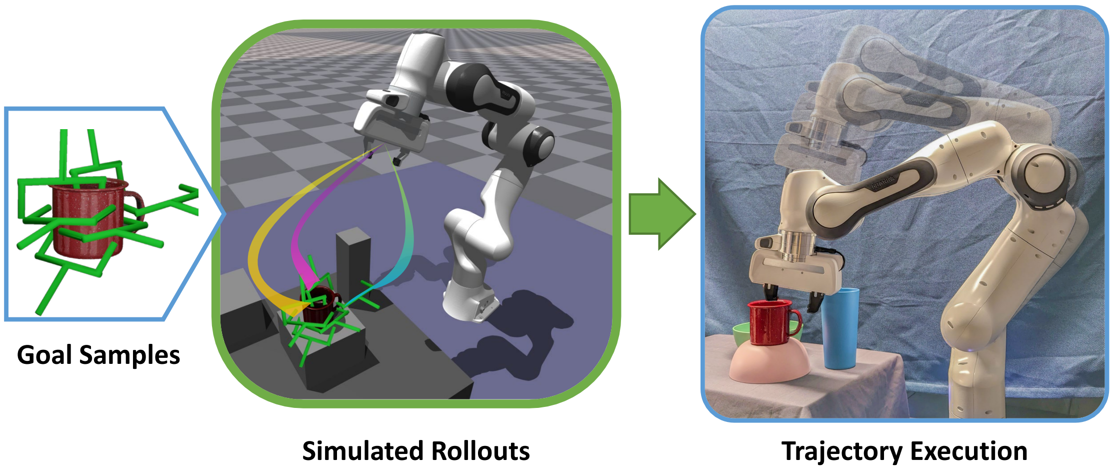


Figure 4.1: Our *goal set planner* considers a discrete set of goal samples as the planning objective. By considering multiple possible goals within the trajectory optimization process, we can handle challenging environments with clutter and dynamic obstacles.

4.1 Introduction

In order to accomplish diverse tasks in unstructured environments, robots need the ability to generate motion plans to different user-specified goals. Such goals are often naturally expressed as *goal regions* in the workspace, rather than single point goals (e.g. navigate to a certain room, place an object on a shelf). Traditional trajectory planning techniques require the goal regions to be explicitly specified as planning objectives. This specification is challenging in cases where the goal region is *uncertain*. Examples of uncertain goal regions include those which are intractable to model explicitly, such as stable grasp poses across diverse objects [97, 114] or user preferences [135, 189].

We seek to solve planning problems under goal region uncertainty given only a fixed set of goal demonstrations. The key insight of our approach is to model these demonstrations as samples from an underlying goal distribution. Our *goal set planner* plans directly to these goal samples, resulting in a generalizable, data-driven planning objective which can be employed across goal regions, eliminating the need to explicitly model the goal. Planning to goal distributions has been formulated as a flexible representation for a number of objectives [34], but requires a fully specified parametric goal distribution. Our approach instead treats the goal as an *implicit distribution*, from which we can obtain samples (e.g. successful grasp poses [47]) but which cannot be evaluated explicitly. We formulate this problem as inference-based planning [8, 166, 181], and seek a distribution of trajectories which terminate in high-density regions of the goal distribution (see Figure 4.1). Due to the complex, multi-modal nature of the planning problem, we consider particle-based trajectory distributions and perform inference using Stein Variational Gradient Descent [92, 81]. In contrast to previous data-driven approaches to intractable planning objectives [97, 114, 135, 189, 47, 33, 94], we require no upfront learning before planning begins.

We formulate the problem of goal set planning as *generalized Bayesian inference* (GBI), an emerging statistical technique which enables inference for intractable likelihoods. GBI refers to a class of problems in which the likelihood function is specified through a generic loss, typically a divergence between two distributions evaluated against observations [18, 72, 104]. We demonstrate that divergences can be computed over implicit goal distributions, which allows us to leverage a loss function that quantifies the divergence between the terminal state samples and the goal samples to infer the posterior trajectory distribution [39]. We then propose a Stein variational inference methodology to solve the problem as a sequence of gradient updates of a set of trajectories.

We present a number of potential loss functions which can be applied within this framework. We additionally compare to several previously proposed heuristic approaches [34, 185]. We demonstrate the applicability of our approach in planar navigation and in a high-dimensional grasping task. By considering the full set of goal samples throughout the planning process, our planner reliably finds reachable grasps in scenarios where heuristic approaches fail to reach their selected pose from the goal set.

4.2 Related Work

Trajectory planning to a goal can be stated as an optimization problem with the objective of finding a minimum cost path to a point in the robot workspace and can be efficiently solved using differentiable costs and dynamics [131, 73, 138]. Casting planning as probabilistic inference has become a popular technique for achieving robust control under uncertainty [132, 81, 78, 181, 19]. One popular approach is to employ a nonparametric, particle-based representation of the trajectory distribution, iteratively updating the posterior through importance sampling [78, 181]. Other methods use variational inference to efficiently infer the trajectory distribution [132, 81, 129].

Most related to our work is that of Conkey and Hermans, who formulate trajectory planning to goals defined as a probability distribution [34]. Our proposed approach also considers the goal to be a distribution, but we represent the goal as an implicit (sampled) distribution [39] eliminating the need for the distribution to be explicitly specified by the user. The problem of planning to a set of goals has been considered by the manipulation community for robotic grasping. A popular approach for grasp planning to a goal set has been to formulate trajectory optimization and grasp selection as independent steps [13, 158, 185]. These methods first select a single goal from the set prior to planning based on a scoring metric, then treat the selected sample as a point goal. Similar approaches have been used to heuristically guide sampling-[64] and graph-based [3] motion planners to goal samples from sets in other contexts. Another approach is to jointly perform trajectory optimization and grasp selection. Dragan et al. [45] add a manually defined goal set constraint to the problem of trajectory optimization. Wang et al. [178] perform grasp selection and refinement online by estimating the grasp distribution. These methods require domain-specific information about the grasping problem and do not generalize across planning problems. Data-driven approaches to represent intractable planning objectives either employ high-fidelity simulation to measure success (e.g of grasps) [97, 114, 47, 94], or demonstrations to represent user preferences [135, 189, 33]. In these techniques, an upfront computational cost is required before planning begins, and must be performed again whenever environmental conditions, such as observability or dynamic environments, necessitate replanning.

4.3 Goal Sample Set Planning as Inference

Given a robot with state $x_t \in \mathcal{X}$ at time t, we consider the problem of planning to a *goal set* within the state space, $\mathcal{X}_g \subset \mathcal{X}$. We consider the case of arbitrarily complex goal regions $\mathcal{X}_g$ that do not admit explicit analytic specification. Instead we can obtain a set of valid goal samples, $\tilde{G} = \{x_g^{(i)}\}_{i=1}^N$, $x_g^{(i)} \in \mathcal{X}_g$. Examples of this type of goal include data labelled by simulation or human demonstration.

We seek a policy $u_t = \pi(x_t)$ such that the terminal state after a time horizon T lies within the goal region, $x_T \in \mathcal{X}_g$, given only goal samples $\tilde{G}$. The robot trajectory is defined as a sequence of states and actions, $\tau = \{(x_t, u_t)\}_{t=0}^{T}$. We seek the minimum cost trajectory, τ^*, which solves:

$$\tau^* = \underset{\tau \in \mathcal{T}}{\arg\min}\ C\left(x_T; \tilde{G}\right) + \sum_{t=0}^{T-1} c_t(x_t, u_t; z), \tag{4.1}$$

where z denotes the environmental observation (e.g. the map of the environment). The running cost, $c_t(x_t, u_t; z)$, is chosen based on the domain (e.g. quadratic cost, obstacle cost, etc.). Our primary contribution is the formulation of the terminal cost, $C(x_T; \tilde{G})$, when the goal region is represented only by a set of samples, $\tilde{G}$.

The planning objective in Equation (4.1) can be restated in terms of probabilistic inference using the formalism of planning as inference [8, 166, 132, 181], as described in Section 2.3. In planning as variational inference, we seek to find a distribution over trajectories $q(\tau) \in \mathcal{Q}$ that minimizes the divergence with the posterior distribution of trajectories, $p_O(\tau \mid z)$, defined from the cost in Equation (4.1):

$$q^*(\tau) = \underset{q \in \mathcal{Q}}{\arg\min}\, D_{KL}\Big(q(\tau) \,||\, p_O(\tau \mid z)\Big). \tag{4.2}$$

We can factorize the posterior over trajectories, τ, given an environment observation, z, and a set of goal samples, $\tilde{G} = \{x_g^{(i)}\}_{i=1}^{N}$, as:

$$\begin{aligned} p_O(\tau \mid z, \tilde{G}) &\propto p_{\text{goal}}(\tilde{G} \mid x_T) \prod_{t=1}^{T-1} p(z \mid x_t) \prod_{t=1}^{T-1} p(x_{t+1} \mid x_t, u_t) && (4.3) \\ &= p_{\text{goal}}(\tilde{G} \mid x_T) p_O(z \mid \tau') p(\tau'), && (4.4) \end{aligned}$$

where $\tau' = \{(x_t, u_t)\}_{t=0}^{T-1}$ denotes the portion of the trajectory up to time $T-1$, x_T denotes the terminal state of the trajectory, and $p_{\text{goal}}(\tilde{G} \mid x_T)$ is the goal likelihood given the implicit goal distribution $\tilde{G}$. We model the likelihood $p_O(z \mid \tau')$ in terms of the trajectory cost yielding:

$$p_O(z \mid \tau') \approx \frac{1}{Z} \exp\left(-\alpha \sum_{t=0}^{T-1} c_t(x_t, u_t; z)\right), \tag{4.5}$$

where Z is a normalization term. The likelihood $p_{\text{goal}}(\tilde{G} \mid x_T)$ in Equation (4.4) is intractable due to the implicit goal distribution. The following section addresses the key technical question of this work: how can we perform inference on the intractable posterior arising from an implicit goal distribution?

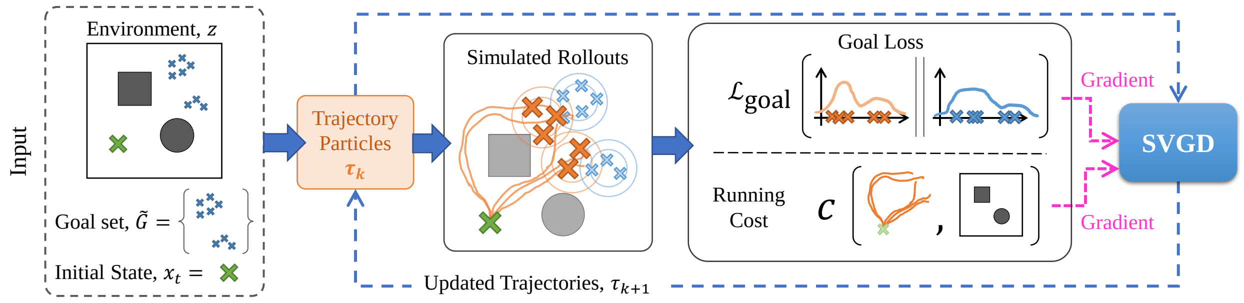


Figure 4.2: Our SVGD-based planner takes as input a set of goal samples, $\tilde{G}$, an environmental observation, z, and an initial state, x_0 and iteratively updates a set of trajectory particles, τ. At each iteration, the particles are rolled out using a simulator. Each rollout is then evaluated in terms of its running cost, c. A goal loss , $\mathcal{L}_{\text{goal}}$, computes the discrepancy between the terminal points of the trajectory distribution and the goal set. SVGD updates the trajectory particles using the gradients of the combined running and goal losses.

4.4 Planning to Implicit Goal Distributions as Generalized Bayesian Inference

We propose applying generalized Bayesian inference to directly approximate the solution of Equation (4.4). While one could fit a parametric model of $\hat{p}_{\text{goal}}(\mathcal{X}_g \mid x_T)$ to the sampled data $\tilde{G}$, we seek a generic objective for planning directly to the implicit distribution that avoids task-specific pre-processing. We now give a brief overview of generalized Bayesian inference and Stein variational inference before explaining how we combine them to solve the goal set planning problem. Figure 4.2 visualizes our approach to planning as generalized Bayesian inference using SVGD.

Generalized Bayesian Inference. Generalized Bayesian inference (GBI) accounts for intractable inference problems arising from evaluating a loss function instead of a likelihood that factorizes over each observation [104]. Given a posterior $p(x \mid \mathbb{Y})$, where x are arbitrary random variables and $\mathbb{Y} = \{y^{(i)}\}_{i=1}^{N}$ is a set of N observed data points, GBI approximates the posterior as follows:

$$p_{\mathcal{L}}(x \mid \mathbb{Y}) \propto p(x) \exp\left(-\beta \mathcal{L}(x, \mathbb{Y})\right) \tag{4.6}$$

where $\mathcal{L}(x, \mathbb{Y})$ defines a loss between the random variables and observed data. Typically, the loss measures divergence (e.g. KL) between the data and query variables. The posterior $p_{\mathcal{L}}(x \mid \mathbb{Y})$ recovers the conventional Bayesian posterior when $\beta = 1$ and $\mathcal{L}(x, \mathbb{Y}) = -\sum_{i=1}^{N} \log p(y^{(i)} \mid x)$. We can compute the approximate posterior by solving the following variational optimization problem:

$$q_{\mathcal{L}}(x \mid \mathbb{Y}) = \underset{q \in \mathcal{Q}}{\arg\min}\, \beta\, \mathbb{E}_{x \sim q}\left[\mathcal{L}(x, \mathbb{Y})\right] + D_{KL}\Big(q(x) \,||\, p(x)\Big) \tag{4.7}$$

Stein Variational Inference. We build on work that approximately solves Equation (4.2) using Stein variational inference [81, 11]. For a complete description of this technique, see Section 2.4.3. Stein variational inference represents the candidate distribution $q(\tau)$ nonparametrically by a set of particles, each representing a trajectory, $\{\tau^{(i)}\}_{i=1}^{M}$. Stein variational gradient descent (SVGD) [92] employs gradient-based optimization over the particle set to minimize the kernelized Stein discrepancy [91] between the true and the approximate posterior. SVGD can solve high-dimensional planning problems involving complex, multi-modal posteriors [81, 11]. SVGD is an iterative algorithm which applies the following update to each particle i at iteration k:

$$\tau_k^{(i)} \leftarrow \tau_{k-1}^{(i)} + \gamma \phi(\tau_{k-1}^{(i)}) \tag{4.8}$$

$$\phi(\tau) = \frac{1}{M} \sum_{j=1}^{M} k(\tau^{(j)}, \tau) \nabla_{\tau^{(j)}} \log p_O(\tau^{(j)} \mid z) + \nabla_{\tau^{(j)}} k(\tau^{(j)}, \tau) \tag{4.9}$$

where $k(\tau^{(j)}, \tau)$ is a kernel function between trajectories. We can interpret the first term inside the summation of Equation (4.9) as an attractive force that moves particles according to the gradient of the log-posterior, while the second term acts as a repulsive term keeping particles from collapsing to a single point estimate. Thus SVGD leverages parallel gradient-based optimization to generate a diverse set of samples more efficiently than Markov chain Monte Carlo (MCMC) samplers [181].

Terminal Losses for Goal Sets. We leverage results from the GBI literature which approximate the optimization problem in Equation (4.7) using a divergence metric between two implicit distributions [39]. We define the implicit terminal distribution as $\tilde{X}_T = \{x_T^{(i)}\}_{i=1}^{M}$ and solve:

$$q_{\mathcal{L}}(x_T \mid \tilde{G}) := \underset{q \in \mathcal{Q}}{\arg\min}\, \mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G}) \tag{4.10}$$

The loss function $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G})$ measures the discrepancy between the terminal state distribution and the distribution of goal states, represented implicitly by samples $\tilde{X}_T$ and $\tilde{G}$ respectively.

We note that the loss defined in Equation (4.10) provides a measure over the set of terminal states, rather than an individual likelihood per trajectory particle required by Equation (4.4). A key insight of our approach is to solve the resulting planning as inference task with SVGD, which employs the gradient of the particle likelihood, without the need to explicitly evaluate the goal loss per particle. We define the gradient in Equation (4.9) as follows, combining the results from Equation (4.4) and Equation (4.10):

$$\nabla_{\tau^{(j)}} \log p_O(\tau^{(j)} \mid z) = \nabla_{\tau^{(j)}} \log p_O(z \mid \tau^{(j)}) + \nabla_{\tau^{(j)}} \log p(\tau^{(j)}) - \beta \, \nabla_{\tau^{(j)}} \mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G}) \tag{4.11}$$

The terminal loss must be differentiable with respect to the trajectory samples and efficient to compute so it can be used in a control loop. We consider the following *two sample* tests which meet this criteria: The kernel Maximum Mean Discrepancy (MMD) [60], a KL divergence approximation based on classification [154, 106], a smooth k-Nearest Neighbor test [143], and the energy statistic [159]. In the following section, we describe each of these losses.

4.4.1 Terminal Losses

This section provides detailed derivations of the different terminal loss functions, $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G})$, compared in our generalized Bayesian inference planner. Recall, the loss function $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G})$ measures the discrepancy between the terminal state distribution and the distribution of goal states, represented implicitly by samples $\tilde{X}_T$ and $\tilde{G}$ respectively.

Kernel Maximum Mean Discrepancy (MMD). The squared MMD [60] measures discrepancy between two distributions $p(x)$ and $q(y)$, and is defined as:

$$\text{MMD}^2(p, q) = \mathbb{E}\left[k(x, x')\right] - 2\mathbb{E}\left[k(x, y)\right] + \mathbb{E}\left[k(y, y')\right] \tag{4.12}$$

where $k(\cdot, \cdot)$ is a kernel function. In the case where distributions p and q are implicit, an unbiased two-sample approximation given sample sets $\mathbb{X} = \{x^{(i)} \sim p\}_{i=1}^{M}$, and $\mathbb{Y} = \{y^{(i)} \sim q\}_{i=1}^{N}$ of the squared MMD is given by:

$$\begin{aligned} \text{MMD}_u^2(\mathbb{X}, \mathbb{Y}) = \frac{1}{M(M-1)} \sum_{i=1}^{M} \sum_{j \neq i}^{M} k(x^{(i)}, x^{(j)}) + \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j \neq i}^{N} k(y^{(i)}, y^{(j)}) \\ - \frac{2}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} k(x^{(i)}, y^{(j)}). \end{aligned} \tag{4.13}$$

When applied to our setting we simply define $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G}) = \text{MMD}^2(\tilde{X}_T, \tilde{G})$ where $\tilde{X}_T = \{x_T^{(i)}\}_{i=1}^{M}$ correspond to samples from the terminal state associated with the corresponding trajectory particle $\tau^{(i)}$.

KL Divergence. A differentiable two-sample KL divergence approximation can be obtained through density ratio estimation via classification [154, 106]. The KL divergence between distributions p and q is defined as:

$$D^{\text{KL}}(p, q) = \mathbb{E}_{x \sim p(x)} \left[\log \frac{p(x)}{q(x)} \right]. \tag{4.14}$$

This quantity can be approximated over samples $\{x^{(i)} \sim p\}_{i=1}^{M}$ if the ratio $r(x) = p(x)/q(x)$ can be computed analytically:

$$\mathbb{E}_{x \sim p(x)} [\log r(x)] \approx \frac{1}{M} \sum_{i=1}^{M} \log r(x^{(i)}) \tag{4.15}$$

In the case where the goal distribution and posterior distribution are implicit, the ratio $r(x)$ cannot be evaluated. Instead, the ratio can be approximated through classification, where binary label $y = 1$ indicates that a sample x was drawn from p and $y = 0$ indicates a sample was drawn from q. The ratio can then be approximated as:

$$r^*(x) = \frac{P(x \mid y = 1)}{P(x \mid y = 0)}, \tag{4.16}$$

which can be further simplified to:

$$r^*(x) = \exp(\sigma^{-1}(P(y = 1 \mid x))). \tag{4.17}$$

This approximation reduces the density ratio estimation to a classification problem. Thus, the ratio is computed by evaluating a binary classifier for each sample which can be performed efficiently.

Intuitively, this approach assumes that the KL divergence should be highest when the classifier cannot distinguish between the samples from each distribution. The classifier must be trained individually for each estimated trajectory distribution in order to use this divergence. To improve computational efficiency in practice, the classifier can be initialized with the result from the previous iteration. Finally, we can define $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G}) = D^{\text{KL}}(\tilde{X}_T, \tilde{G})$ where the KL divergence is approximated following Equation (4.15).

Smooth K-Nearest Neighbor. It is possible to define a differentiable two-sample test based on the well known k-NN algorithm as demonstrated in [143]. The Smooth K-Nearest Neighbor test possesses important statistical properties such as consistency and convergence of its statistics to f-divergence. This is despite the complexity of having to solve a combinatorial optimization problem (nearest neighbor match) required by the k-NN method. Let M

be the number of samples of p, and N be the number of samples of q. The k-NN divergence can be defined as

$$D_{\alpha}^{\mathrm{NN}}(p,q) = \int \frac{\alpha^2 p^2(x) + (1-\alpha)^2 q^2(x)}{\alpha p(x) + (1-\alpha) q(x)} dx, \tag{4.18}$$

for $M/(M+N) \to \alpha \in (0,1)$.

As proved in [63], the statistic

$$1 - \frac{T(\mathbb{X},\mathbb{Y})}{(M+N)^k}, \tag{4.19}$$

where $T(\mathbb{X},\mathbb{Y})$ refers to the number of edges connecting samples in a set $\mathbb{X} = \{x^{(i)} \sim p\}_{i=1}^{M}$ to a set $\mathbb{Y} = \{y^{(i)} \sim q\}_{i=1}^{N}$ from a k-neighborhood graph created with points in $\mathbb{X}$ and $\mathbb{Y}$, converges in probability to the D_{α}^{NN} divergence, and can be used as an efficient approximation. To make the computation differentiable, the authors of [44] define

$$T(\mathbb{X},\mathbb{Y}) = \sum_{i=1}^{M}\sum_{j=1}^{N} s_i(\{x^{(l)}\}_{l=1}^{M+N})_j, \tag{4.20}$$

where $s_i(\{x^{(l)}\}_{l=1}^{N+M})$ denotes the `softmax` function computed on the Euclidean distances between all points, except point i. As the `softmax` function is differentiable, the statistic in Equation (4.19) becomes differentiable and can be used directly as our loss function $\mathcal{L}_{\mathrm{goal}}(\tilde{X}_T, \tilde{G})$. To avoid specifying a particular value for α, our implementation computes the statistic for several values and averages them as the final result.

Energy Statistic. This two-sample test is based on Newton's gravitational potential energy which relates two entities by the Euclidean distance between them [159]. Given two distributions $p(x)$ and $q(y)$, the energy distance is defined as:

$$D^{\mathrm{E}}(p,q) = 2\mathbb{E}\left[||x-y||^2\right] - \mathbb{E}\left[||x-x'||^2\right] - \mathbb{E}\left[||y-y'||^2\right], \tag{4.21}$$

where x and y are independent random variables. The corresponding two-sample statistic given two sets of samples $\mathbb{X} = \{x^{(i)} \sim p\}_{i=1}^{M}$ and $\mathbb{Y} = \{y^{(i)} \sim q\}_{i=1}^{N}$ can then be written as:

$$D^{\mathrm{E}}(\mathbb{X},\mathbb{Y}) = \frac{2}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\|x^{(i)} - y^{(j)}\|^2 - \frac{1}{M^2}\sum_{i=1}^{M}\sum_{j=1}^{M}\|x^{(i)} - x^{(j)}\|^2 - \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\|y^{(i)} - y^{(j)}\|^2. \tag{4.22}$$

This provides a computationally efficient statistic which can be directly used as our loss $\mathcal{L}_{\mathrm{goal}}(\tilde{X}_T, \tilde{G}) = D^{\mathrm{E}}(\tilde{X}_T, \tilde{G})$.

4.4.2 Practical Considerations

The statistics considered in this work are good local approximations of distribution divergences. In the case of trajectory optimization, when the terminal states in early planning iterations are far from the goal set, the goal loss gradients can be uninformative. We therefore include a prior in our set planning method consisting of a smooth uniform distribution constructed by placing a bounding box around the goal samples. This can be included in our framework by multiplying a prior over the terminal state $p(x_T)$ with the goal likelihood in Equation (4.10) This mitigates the poor divergence approximation in early iterations. Note that the uniform prior is insufficiently informative on its own, particularly in cases where the goal set is multi-modal. Furthermore, the prior needs to be differentiable, which is not the case for a standard uniform distribution. Therefore we define a smooth uniform prior in the region $R = x_T \colon a \leq x_T \leq b$ as

$$p(x_T) \propto \exp\left(-d(x_T, R)^2 / \sqrt{(2\sigma^2)}\right) \tag{4.23}$$

where $d\left(x, R\right) = \min |x - x'|,\ x' \in R$ is a distance function, and σ controls the *sharpness* of the approximation.

Once inference over the trajectory distributions converges, we must select a single trajectory estimate to execute. A common approach to accomplish this is by taking the mean, or weighted mean, of the particle set. This method is ineffective when the trajectory distribution is multi-modal. An alternative approach is to pick the maximum weighted particle. Our proposed set-based terminal losses yield a single score over the whole distribution, which does not enable weighing individual particles based on terminal loss. To select our final sample, we instead select the lowest cost trajectory. In practice, we also include the prior in the weight computation to avoid local minima with very low-cost trajectories.

4.5 Experimental Results

We evaluate our method in a simulated 2D planar navigation problem and manipulation experiments both in simulation and on a physical robot. For all experiments, the goal set planner is run in MPC-style, where optimization is performed at each timestep, then the first action of the lowest-cost trajectory is executed.

Baselines. Each baseline involves inference using SVGD [92, 81] with the same running costs, but with different heuristic terminal costs based on previously published work, as defined below.

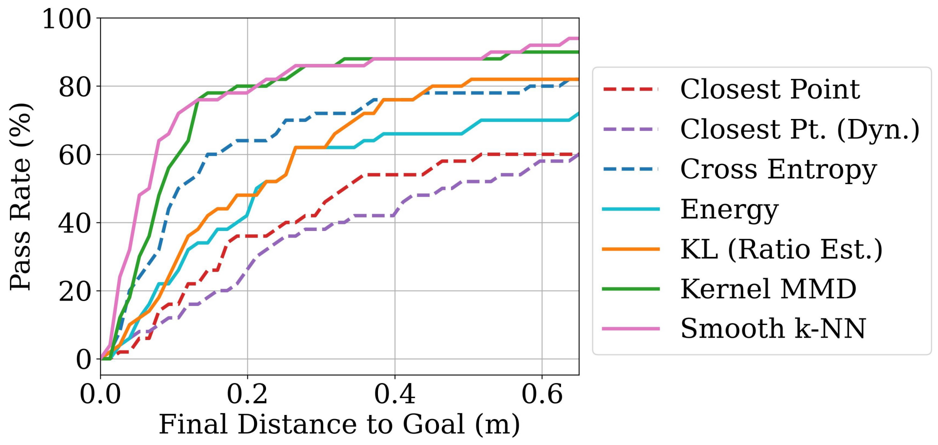


Figure 4.3: Pass rate for different distance thresholds between the final state and the nearest goal sample. Dashed lines represent baseline methods.

- *Closest Point:* The nearest goal sample to the start configuration is selected as the goal. The terminal cost is the squared Euclidean distance to this point. A *dynamic* version of this cost is obtained by recomputing the closest point at the beginning of each planning timestep [185].
- *True Goal Distribution:* Given the true goal distribution, the objective is formulated as maximizing the density of the goal distribution at the terminal point [34]. This method is only applicable to the 2D planar navigation task when the goal distribution is known.
- *Mixture Model:* We define a goal likelihood with a Gaussian mixture model obtained by placing each component centered on each goal sample, with a fixed covariance, akin to a kernel density estimator. The goal loss is then the cross entropy over the mixture model, which is equivalent to maximizing the likelihood [34].

4.5.1 Planar Navigation

We implement a 2D planar navigation task to characterize the behavior of our approach. We define goal distributions over 2D position for each scene using a parametric distribution (Gaussian, mixture of Gaussians, or uniform). The goal set supplied to the planner consists of a random set of goal samples drawn from the distribution.

The agent state x_t is composed of a 2D position and velocity, and the control signal u_t is a 2D acceleration. We use known, linear dynamics in a fully observed environment for the controller rollouts.

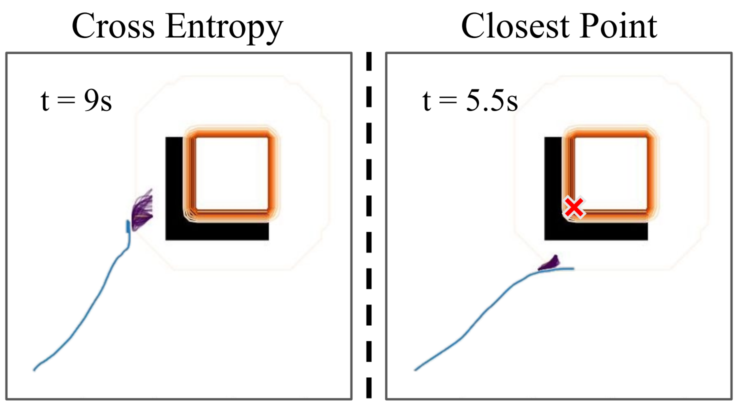


Figure 4.4: Cross Entropy (left) and Closest Point (right) baselines fall into local minima failing to reach the goal. The trajectory is shown in blue, the trajectory distribution in purple, and the goal distribution in orange contours. The red 'x' shows the closest point.

Losses. For each method, the running cost is summed over each timestep, where the cost for one timestep is:

$$c_t(x_t, u_t, z) = x_t^\top Q x_t + u_t^\top R u_t + \alpha\, c_{\text{SDF}}(x_t, z) \tag{4.24}$$

where Q and R are quadratic cost parameters for the state and action, and $c_{\text{SDF}}(x_t, z)$ is the obstacle avoidance term, computed using the Signed Distance Function (SDF) over the environment z. The goal loss $\mathcal{L}_{\text{goal}}(\tilde{X}_T, \tilde{G})$ is a set goal loss which is differentiable with respect to the trajectory τ.

Implementation Details. For each of our goal set planner ablations, $N = 50$ samples are randomly selected from goal distribution, except for *KL (Ratio Estimation)*, which uses $N = 100$ samples. This method involves training a learned classifier so is aided by a higher sample size. For the closest point methods, the goal sample with the smallest Euclidean distance from the start state is selected. For all the methods, $M = 50$ particles are used to represent the trajectory distribution. The particles represent the discrete control signals, u_t, which are 2D accelerations at each 0.1 second timestep over a horizon of 3 seconds. All planners are initialized with the distribution from the previous timestep, shifted to the current timestep, and run for 50 iterations. We use the Adam optimizer [76] to select the step size in the SVGD update rule.

The KL divergence uses a 3 layer fully-connected network as the classifier, retrained at each timestep. To mitigate computational complexity, we warm start the training with the weights from the previous timestep. The Kernel MMD uses an RBF kernel, with a bandwidth selected by applying the median heuristic over the goal samples [76]. The Smooth k-NN loss uses a value of $k = 1$.

We use the RBF kernel for SVGD, and set the bandwidth using the median heuristic, a popular technique for choosing the kernel bandwidth which yields a good estimate in many cases [55]. Without access to data consisting of trajectory samples, we assume that they are normally distributed with covariance $\sigma = 1$. Under these assumptions, it follows that the expected distance between samples drawn from the distribution is $2D$, where D is the trajectory dimension.

Results. To evaluate the goal set planner, we execute 10 runs per method per scene across 5 scenes. We measure the Euclidean distance between the terminal state in the trajectory and the nearest goal sample for each run. We present the pass rate for each method in Figure 4.3, where pass rate is defined as the percentage of runs for which the error is less than a given error threshold. The cross entropy baseline performs close to the set-based methods with respect to pass rate, but is prone to falling into local minima. The closest point baselines have a high failure rate on challenging environments, particularly those with collisions near the goal (see Figure 4.4), or those with the closest point in collision (see bottom-left in Figure 4.6). While it would be trivial to check for collisions with the closest point in the planar environment, we avoid this additional domain-dependent pre-processing step to better replicate environments in which collisions are expensive or intractable to compute.

We visualize the total Euclidean path length for each method in Figure 4.5. The baselines are shown with hatched bars. We exclude paths for which the terminal state is not within 40 cm of any goal sample. All methods achieve similar path lengths, with the closest point baselines consistently resulting in slightly shorter paths. This is unsurprising given the distance-based goal selection bias, but comes at the cost of being less robust to different environments.

Figure 4.6 shows example trajectory distributions for the goal set planner using the MMD cost. Both the Smooth k-NN and the Kernel MMD terminal losses display moment-matching behaviour, which we posit allows them to be more robust to challenging environments. We posit that the kernel embedding in the MMD loss enables more flexible representation of the goal region compared to the other losses. We select MMD for all manipulation experiments.

Distribution matching. We hypothesize that our goal set planning achieves higher success rate in challenging environments due to its ability to better approximate the underlying goal distribution. To verify this, we measure the Chamfer distance between the terminal states in the trajectory distribution at each timestep and the goal samples. The results are shown in Fig. 4.7. The cross entropy baseline, using the true goal distribution, achieves lower Chamfer distance than the other baselines, but performs worse than the goal set planner due to its mode-seeking behavior. Kernel MMD and smooth k-NN terminal set losses perform

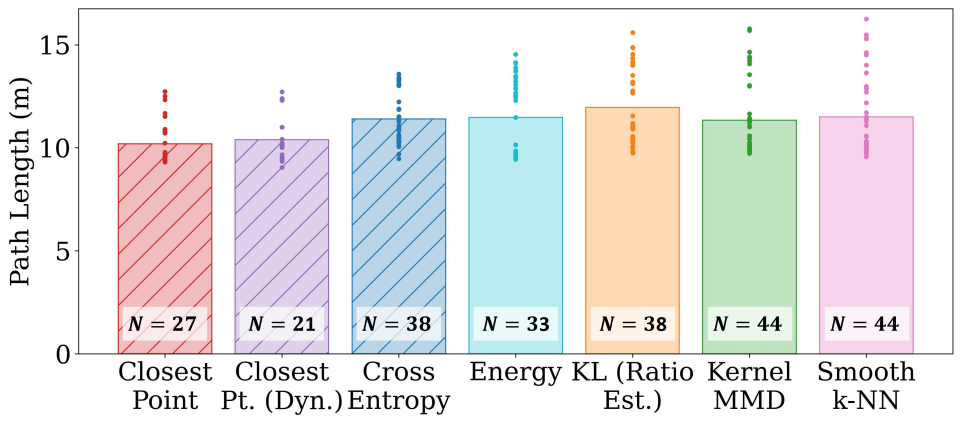


Figure 4.5: Average path length for each terminal cost for the planar navigation task. The points indicate results for individual runs. Only successful trajectories are included, where success is defined as getting to within 40 cm from any sample within the goal set. The numbers on the bars indicate the number of successful runs out of 50 in each category.

better on this metric than the energy statistic and the KL approximation.

4.5.2 Robotic Grasping

Stable grasping of arbitrary objects is a challenging problem since the goal distribution consisting of stable grasp poses is intractable to model. We evaluate our goal set planner on a grasping task using grasp samples generated in simulation [47]. We compare our method to the Closest Point and Mixture Model baselines.

We assume that the environment and target object are fully observed. Experiments are performed using the Franka Panda manipulator in the Isaac Gym simulator [99]. We use the STORM library for GPU-accelerated, fully differentiable dynamics and costs [17]. The goal set is represented by $N = 100$ samples of valid grasp configurations drawn from the Acronym dataset [47]. The goal set planner is used to plan to a pre-grasp pose to offset grasp locations in an MPC framework over a fixed horizon. The goal set planner optimizes over all goal samples at each planning iteration, and does not require pre-processing the set prior to planning. After convergence, we use inverse kinematics (IK) to move to the nearest grasp sample, and then lift the object.

We evaluate over 5 runs across 10 scenes containing 8 different objects and 5 environments. Grasp success results are shown in Table 4.1. We observe that a primary source of failure is due to failure in the IK stage, when moving from the pre-grasp pose to the grasp pose. IK errors can arise from collisions or singularities encountered during the grasp. To better understand the performance of the goal set planner independent from the grasp stage, we

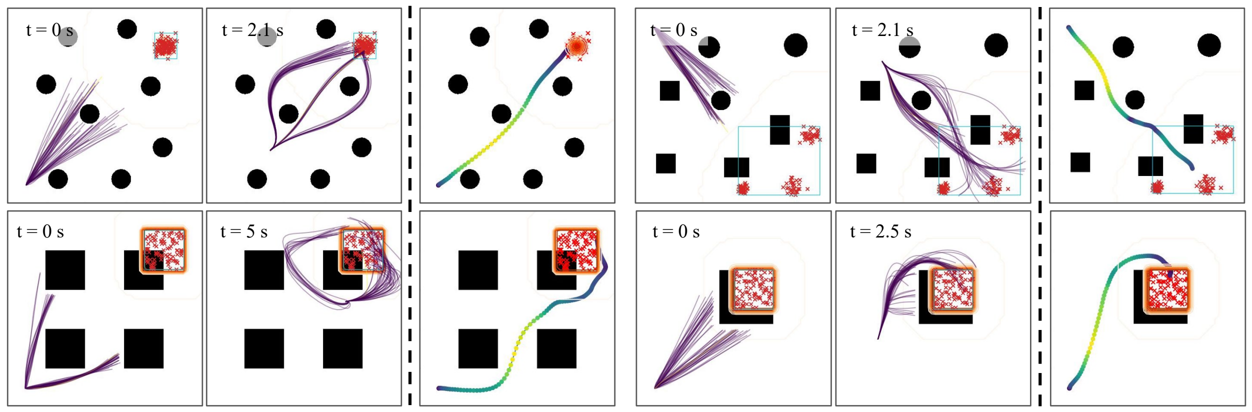


Figure 4.6: Trajectory distributions for the Goal Set Planner using the MMD terminal loss are represented as a set of samples (drawn in purple). The final trajectory is shown in the last pane for each environment, colored with respect to the velocity (yellow is fast, purple is slow). The red points represent the goal samples and the orange contours represent the true goal distribution.

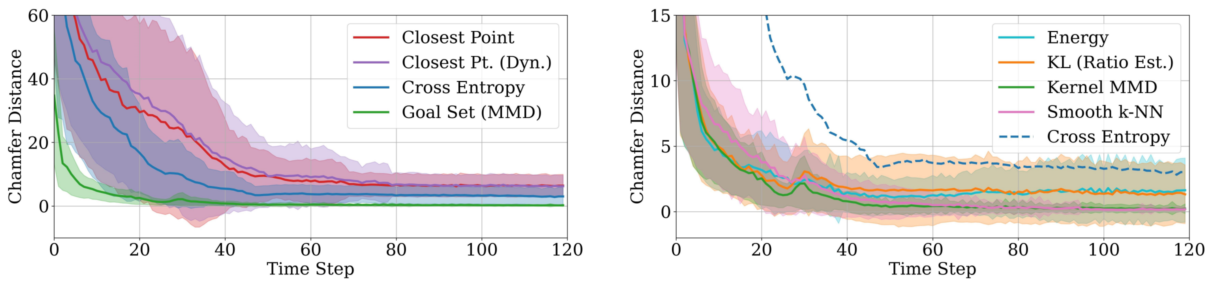


Figure 4.7: Chamfer Distance at each iteration for different terminal costs. (left) Comparison between the baseline methods and our goal set planner with MMD terminal loss. (right) Comparison between different terminal costs. The best performing baseline (Cross Entropy) is shown for reference.

measure the planning success rate of our planner to the pre-grasp pose, computed as the distance from the terminal end-effector pose to the nearest goal sample. We present the pass rate for each method in Figure 4.8. The goal set planner achieves a higher pass rate at thresholds over approximately 4 cm. The closest point method may select goal points which are in collision in some of the example scenes, increasing the distance error, which also accounts for the discrepancy in grasp success. The rotation error is lower for closest point because it matches the orientation of the goal consistently, including for unreachable poses. We posit that the mixture model poorly approximates the true goal distribution explaining its failure.

Figure 4.10 shows a case where the goal set planner finds a reachable point from the set, but the closest point method may select goal points which are in collision in some of the example scenes. This phenomenon helps explain the discrepancy in grasp success at the cost

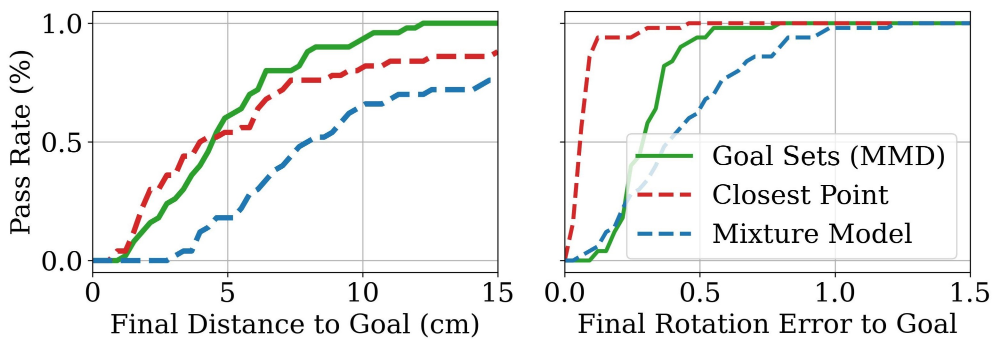


Figure 4.8: Pass rate data for the grasping experiments: the Euclidean distance (left) and the L2-norm of the rotation matrix (right) between the final end-effector orientation and the goal.

	Success (%)
Closest Pt.	50
Mixture Model	40
Goal Set (MMD)	54

Table 4.1: Grasp success rate

of increasing distance error.

4.5.3 Robot Demonstration

We demonstrate that our simulated manipulation experiments are transferable to the real robot, under the assumptions of known object and obstacle locations and geometries, through a robotic grasping demonstration. Figure 4.9 shows two successful grasping runs of the planner, one in an environment without obstacles and one with obstacles. Goal samples are drawn from positive stable grasp examples from the Acronym dataset [47]. We postulate

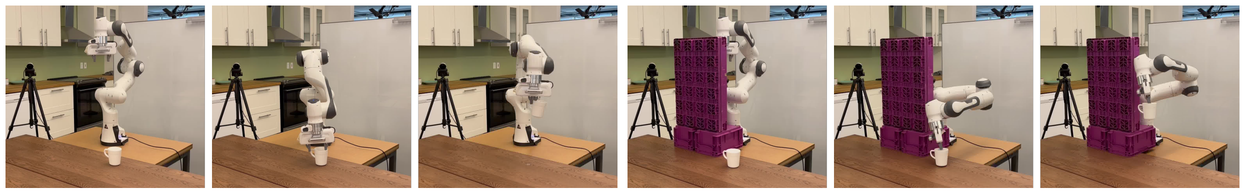


Figure 4.9: Execution runs of the goal set planner on the real robot platform in an environment with no collisions (left three) and with collisions (right three). The robot grasps a mug by planning over a set of successful sample grasp poses obtained from simulation [47]. Trajectories are planned offline and executed on the robot.

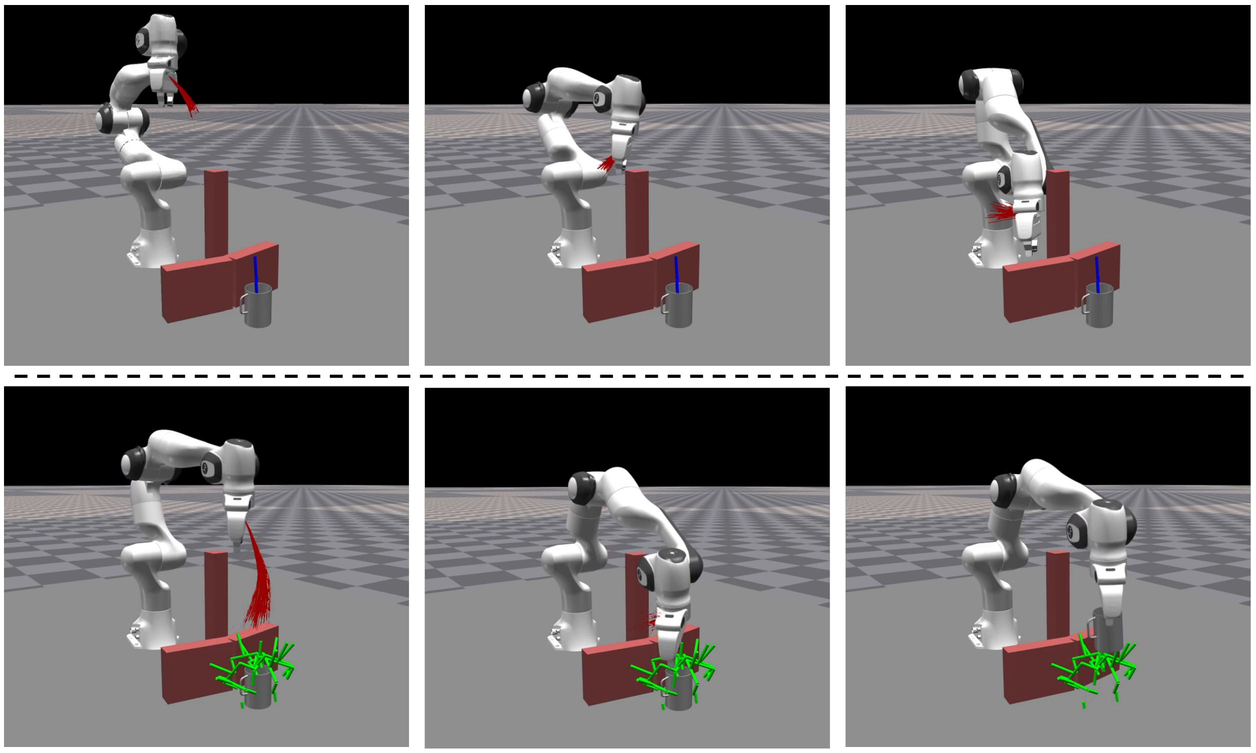


Figure 4.10: Example failure execution for the closest point planner (top) compared the the goal set planner (bottom). The closest point selected is shown in blue. The point is not reachable in the environment, causing the robot to fail to reach it (top right). The set planner considers all the grasp samples, shown in green. It finds a reachable grasp point and grasps successfully (bottom right).

that since the Isaac Gym environment has high-fidelity physics simulation, our results are transferable to the real robot environment. However, to deploy the planner presented in this work to a real robot environment, more investigation would be required into dealing with perception uncertainty. This is an interesting avenue for future work.

4.5.4 Placement

We posit that our planner is more effective than point selection heuristics at dealing with multi-modal goal regions and changing environments due to its ability to consider multiple possible goal regions within the trajectory distribution. To examine this claim, we design a table setting experiment where mugs are placed sequentially on a surface using demonstrations of valid configurations, inspired by demonstration-based arrangement tasks [189]. The goal region consists of three uniform distributions representing valid terminal gripper 3D positions (see Figure 4.11, left). We sample 60 points to form the goal set at the beginning of the task, and keep the goal set constant until all mugs are placed. Each subsequent

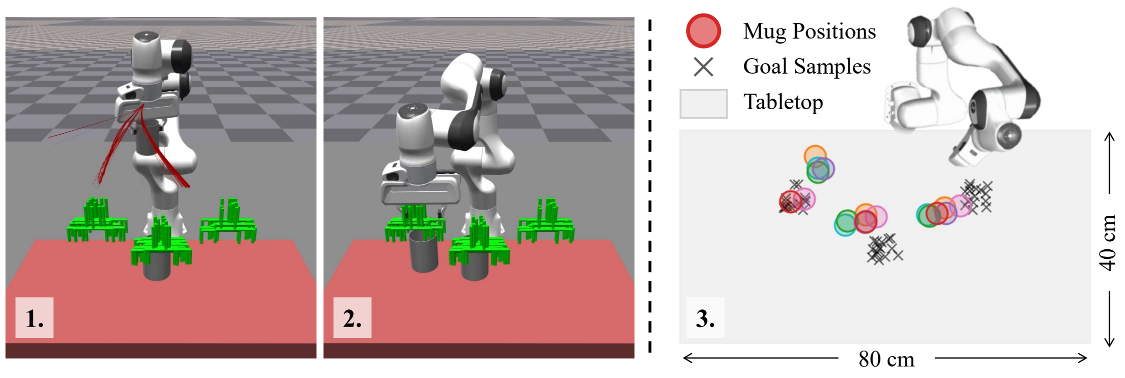


Figure 4.11: Given example demonstrations of mug placement as the goal set (green), the robot plans a path to place a mug. The trajectory distribution (red) considers both free goal distribution modes, and avoids the mode which has a collision due to the first mug (1), ultimately selecting a free mode (2). The final positions of the mugs is show relative to the goal samples over 6 runs (3). The mug colors correspond to runs.

placement operation requires the consideration of a new obstacle in the scene. We employ the MMD terminal loss over the goal samples compared to the 3D terminal positions of the end effector.

We perform six runs for this task, each involving placing three mugs. Figure 4.11 shows an example trajectory distribution for the goal set planner when placing the second of 3 mugs, which shows the multi-modality of the trajectory distribution. The final mug positions (Figure 4.11, right) are clustered corresponding to the modes of the demonstrations. We observe that the final mug placements may lie outside the sampled region. This can be caused by early termination, or in some cases, trajectory distributions which converge outside the sampled region. The latter could be caused by singularities in the robot kinematics or local minima.

4.6 Discussion & Conclusion

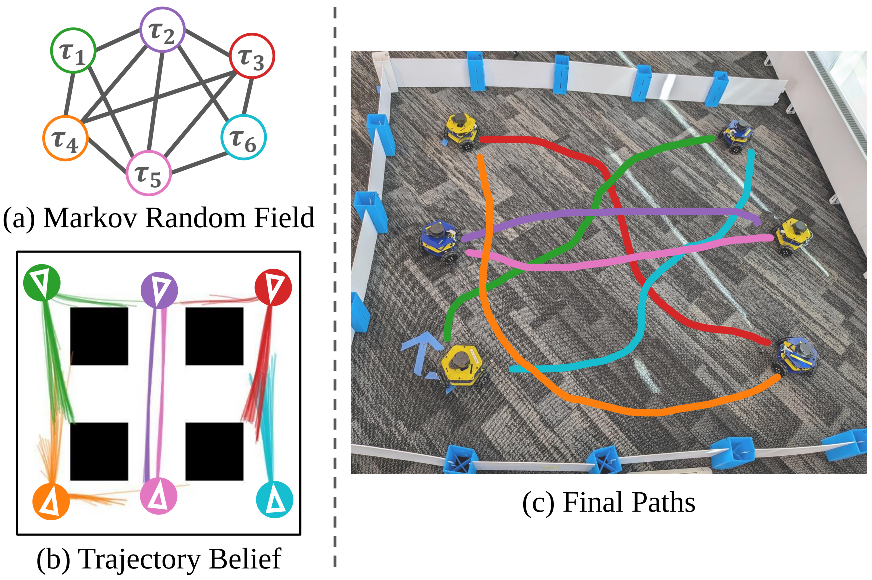
In this work, we present a formulation for a goal set planner, which considers the planning problem in which an uncertain goal region is represented as a set of samples. The planner is generalizable across goals which can be represented by implicit distributions, without the need for domain-specific, user-defined heuristics. We find that independent of the specific loss used, the GBI formulation outperforms previously proposed approaches for planning to goal sets and distributions. Our method outperforms all baselines on a 2D navigation problem and yields improved plan-to-grasp success on a robotic manipulation platform.

In order to remain true to the general planning problem, our method does not consider domain-specific techniques commonly used for the experiments considered. Our grasping results would likely be improved by integration of pre-planning steps for checking reachability or collisions [116], or pre-grasping refinement of the final pose [178]. We hypothesize that combining these approaches with goal set planning for the application of grasping would improve robustness.

Limitations. Maintaining modes in the trajectory distribution can lend added robustness in challenging environments, but it also raises challenges in selecting a single trajectory to execute. We attribute the larger errors in the final pre-grasp pose of our method to the challenge of detecting convergence given the multi-modality of the goal set planner. The terminal set losses considered work best as local measures, meaning gradient information is noisy if trajectories terminate too far from the goal. We mitigate this by applying smooth box priors around the goal samples. Furthermore, the terminal point is not guaranteed to be close to one of the points in the set. This could be mitigated by switching to a point-based cost once close to the goal set. To deploy our planner in real-world applications, more investigation is needed to explicitly handle perceptual uncertainty.

CHAPTER 5

Stein Variational Belief Propagation for Multi-Robot Coordination



(a) Markov Random Field

(b) Trajectory Belief

(c) Final Paths

Figure 5.1: Stein Variational Belief Propagation (SVBP) computes marginal trajectory distributions for each robot in a multi-robot system. SVBP represents the relationships between robots as a Markov Random Field (a) and maintains multi-modal distributions over each robot trajectory (b). The final trajectory generated by SVBP for each robot is shown in (c), where the colour represents time, and the robot is shown in its final position.

5.1 Introduction

Multi-robot coordination is an essential capability for applications involving teams of robots, such as industrial robots, delivery vehicles, and autonomous cars. Planning for multi-robot systems is challenging due to the high-dimensionality introduced by a large number of agents.

Decentralized algorithms enable each robot to perform local computations using information from neighboring robots. This distributed approach is well-suited to multi-robot systems since it involves solving lower-dimensional, local problems compared to the expensive high-dimensional centralized approach.

Decentralized control algorithms [53, 169] are prone to deadlock scenarios which arise from the multi-modality of the solutions that each robot must consider. Considering multiple possible trajectories as a *distribution* allows us to represent diverse solutions [8, 166]. This ability lends added robustness in dynamic environments, such as those with multiple mobile agents. We therefore consider the problem of multi-robot coordination as a probabilistic inference problem. We represent the robot swarm as a graphical model, where each robot is a node in the graph, and edges connect robots in communication range [140, 163, 124]. This representation enables distribution of possible trajectories for each robot to be inferred via graphical inference (see Figure 5.1).

We propose Stein Variational Belief Propagation (SVBP), an algorithm for performing probabilistic inference on a Markov Random Field (MRF) through message passing, and demonstrate its applicability to multi-robot coordination. SVBP employs Stein Variational Gradient Descent (SVGD) [92] to infer marginal posterior distributions as a set of particles through nonparametric belief propagation [152, 70]. Leveraging SVGD enables effective representation of multi-modal distributions, mitigating mode collapse compared to sampling-based methods. Our formulation extends SVGD to graphical models by leveraging the particle message update rules from Particle Belief Propagation (PBP) [69]. In contrast to SVGD, SVBP approximates the marginals rather than the full posterior, and can therefore scale to higher dimensional problems. The resulting algorithm is highly parallelizable since the particles are deterministically updated using gradient information, making it well-suited to efficient implementation on a GPU.

We demonstrate our approach on two applications: a simulated multi-robot perception task, and a multi-robot Model Predictive Control (MPC) task, both in simulation and on a real-world mobile robot swarm. We demonstrate how these problems can be formulated as MRFs [140, 23] and solved via SVBP. The belief propagation framework enables multi-hop information to be passed through the graph while only passing messages between immediate neighbors. The perception experiments show that SVBP can maintain multi-modal belief distributions in uncertain environments, leading to lower localization error compared to baselines. The planning experiments demonstrate that SVBP is more resilient to deadlock scenarios, and produces smoother trajectories resulting in faster time-to-goal. Our robot experiments show that our SVBP controller is robust to noisy localization and dynamics and asynchronous message passing.

5.2 Related Work

Decentralized multi-robot coordination algorithms are those in which each robot executes a controller to satisfy individual objectives considering local information from neighbors. This technique is highly scalable to large and dynamic swarms. Optimal Reciprocal Collision Avoidance (ORCA) [169], a variant of velocity obstacles [53], demonstrates real-time collision avoidance for thousands of agents with independent objectives but are highly prone to deadlock scenarios. We focus on decentralized Model-Predictive Control and graphical approaches in this section and refer the reader to existing surveys [58, 133] for broader coverage.

Multi-robot coordination with graphical models. Probabilistic graphical models present a natural formulation for decentralized multi-robot coordination, whereby individual robots are represented by nodes in a graph and edges connect communicating robots [140]. This formulation has been used to solve for robot localization and control with Gaussian Belief Propagation [124]. Graphical representations have also been used to learn factors for robot control via graph neural networks [163, 162]. This technique requires expert trajectory demonstrations from a centralized controller for training.

Multi-robot Model Predictive Control. Decentralized model-predictive control (DMPC) has been applied to multi-agent collision avoidance problems [112, 119, 171, 36, 95]. By planning over a horizon, these techniques mitigate deadlock scenarios issues but introduce complexities due to the higher dimensionality introduced. These works consider the problem of finding a single trajectory solution. In work most similar to ours, Patwardhan et al. use Gaussian Belief Propagation for collision avoidance in multi-robot planning [124]. This method restricts the trajectory distributions to Gaussian forms, and requires all factors to be linearized about an estimate. In contrast, our approach can be used with any differentiable factor and uses a more flexible nonparametric distribution.

5.3 Background

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denote a MRF with nodes $\mathcal{V}$ and edges $\mathcal{E}$. Let $\mathcal{X} = \{x_s\}_{s\in\mathcal{V}}$ denote the set of all hidden nodes in the graph, and $\mathcal{Z} = \{z_s\}_{s\in\mathcal{V}}$ denote the observed nodes corresponding to each hidden node. We seek to infer the marginal distribution of a node, $s \in \mathcal{V}$, given its corresponding observation, $p(x_s \mid z_s)$.

5.3.1 Belief Propagation

The marginal posteriors in an MRF can be computed using the Belief Propagation (BP) framework:

$$p(x_s \mid z_s) \propto \phi_s(x_s) \prod_{t \in \rho(s)} m_{t \to s}(x_s), \tag{5.1}$$

where $\rho(s)$ denotes the neighbors of s. The message from node t to node s, $m_{t \to s}$, is defined as:

$$m_{t \to s}(x_s) = \int \phi_t(x_t) \psi_{ts}(x_t, x_s) \prod_{u \in \rho(t) \setminus s} m_{u \to t}(x_t) \, dx_t. \tag{5.2}$$

We refer the reader to Section 2.5 for a full description of BP.

A number of belief propagation algorithms have been proposed in the literature. Gaussian Belief Propagation (GaBP) is an efficient algorithm when the node distributions and their corresponding factors can be represented as Gaussian [179, 37]. This method enables efficient computation and has been shown to be effective for multi-robot collision avoidance and localization [124, 115]. However, many applications in robotics are complex and multi-modal, and cannot be fully represented by unimodal Gaussian uncertainty.

Particle Belief Propagation. Nonparametric Belief Propagation (NBP) [152, 70] represents distributions nonparametrically as mixtures of Gaussians and are well-suited to cases where the integral in Equation (5.2) is intractable. NBP algorithms involves expensive product operations between mixture distributions. NBP has been applied to robotic perception of articulated objects, using an efficient sampling-based message product technique [42], learned unary factors [126], and end-to-end learned factors [120]. While these methods enable complex representations of belief distributions, they rely on expensive sequential sampling operations.

Particle Belief Propagation (PBP) defines a sampling-based algorithm for computing the messages in Equation (5.2) for cases where the integral is intractable due to the complexity of the state space [69]. PBP represents the belief at each node with a set of N particles, $\{x_s^{(i)}\}_{i=1}^N$. Given samples from node $t \in \rho(s)$, $\{x_t^{(i)}\}_{i=1}^M$ drawn from a candidate distribution W_t, PBP defines the approximate message:

$$\hat{m}_{t \to s}(x_s^{(i)}) = \frac{1}{M} \sum_{j=1}^{M} \frac{\phi_t(x_t^{(j)}, z_t)}{W_t(x_t^{(j)})} \psi_{ts}(x_t^{(j)}, x_s^{(i)}) \prod_{u \in \rho(t) \setminus s} m_{u \to t}(x_t^{(j)}). \tag{5.3}$$

This message definition is used to draw samples from the marginal posterior, $p(x_s \mid z_s)$, using importance sampling.

PBP relies on the definition of a sampling distribution, W_t, which later work proposed to estimate via expectation maximization [90]. Importance sampling is prone to mode collapse, which has been mitigated by drawing from multiple sampling distributions [121]. However, current solutions to PBP require careful selection of sampling distributions and sequential sampling operations.

5.3.2 Stein Variational Inference

Stein Variational Inference is an algorithm for approximating a distribution $p(x)$ using a candidate distribution, $q(x)$, in the form of a set of particles, $\{x^{(i)} \sim q\}_{i=1}^{N}$. SVGD [92] employs gradient-based optimization over the particle set to minimize the kernelized Stein discrepancy [91] between the true density and a candidate density represented by the particle set. SVGD is an iterative algorithm which applies the following update to each particle i at iteration k:

$$x^{(i)}[k] \leftarrow x^{(i)}[k-1] + \epsilon\gamma(x^{(i)}[k-1]) \tag{5.4}$$

$$\gamma(x) = \frac{1}{N}\sum_{j=1}^{N} \kappa(x^{(j)}, x)\nabla_{x^{(j)}} \log p(x^{(j)}) + \nabla_{x^{(j)}}\kappa(x^{(j)}, x) \tag{5.5}$$

where $\kappa(x^{(j)}, x)$ is a kernel function between particles. We can interpret the first term inside the summation of Equation (5.5) as an attractive force that moves particles according to the gradient of the log-density, while the second term acts as a repulsive term keeping particles from collapsing to a single point estimate. Thus SVGD leverages parallel gradient-based optimization to generate a diverse set of samples more efficiently than Markov chain Monte Carlo (MCMC) samplers. For a full discussion of Stein Variational Inference, see Section 2.4.3.

SVGD has proven useful in a number of robotic applications in recent years, including control, planning, and point cloud matching [81, 11, 98, 82]. SVGD has been applied to graphical models to approximate joint distributions using kernels over local node neighborhoods [176] and conditional distributions over nodes [190]. Both these methods rely on the conditional independence structure of MRFs and as such only pass messages between immediate neighbors in the graph. In contrast, our proposed method computes the *marginal* beliefs over nodes using belief propagation, which involves passing messages through the whole graph.

Algorithm 1 The SVBP Algorithm

```
procedure SVBP($\mathcal{G}$, $\mathcal{Z}$)
    Initialize particles $\{x_s^{(i)}\}_{i=1}^N$ for $s \in \mathcal{V}$
    for $k = 1, \dots, K$ do
        Update messages $(m_{t\to s}, m_{s\to t})$ for $(s,t) \in \mathcal{E}$
        for $s \in \mathcal{V}$ do
            for $i = 1, \dots, N$ do
                Compute $\gamma(x_s^{(i)})$                    ▷ Equations (5.5) and (5.6)
                $x_s^{(i)} \leftarrow x_s^{(i)} + \epsilon\gamma(x_s^{(i)})$
```

5.4 Stein Variational Belief Propagation

We propose Stein Variational Belief Propagation (SVBP), an algorithm for inferring marginal beliefs in an MRF using SVGD. The marginal distribution is represented nonparametrically using a particle set for each node in the graph, $\{x_s^{(i)}\}_{i=1}^N$. We use SVGD gradient updates to infer the density $p(x_s \mid z_s)$ for each node. We define the posterior likelihood term in Equation (5.5) using the marginal belief from Equation (5.1), $p(x_s \mid z_s)$, to obtain the SVBP likelihood gradient:

$$\nabla_{x_s} \log p(x_s \mid z_s) = \nabla_{x_s} \log \phi_s(x_s, z_s) + \sum_{t\in\rho(s)} \nabla_{x_s} \log \hat{m}_{t\to s}(x_s), \tag{5.6}$$

where $\hat{m}_{t\to s}(x_s)$ is defined via the PBP message rule from Equation (5.3). A distinct set of Stein particles represents the posterior belief at each node.

The inference process using SVBP is described in Algorithm 1. Particles are first initialized based on the problem domain. At each iteration, messages are updated with Equation (5.3). For each node, particles are updated using Equation (5.5), computed by evaluating the gradients in Equation (5.6) and the kernel function. The process is repeated for K iterations or until convergence.

SVBP provides several key advantages over other NBP techniques. First, it uses gradient-based, deterministic particle updates which can be efficiently parallelized on a GPU, without relying on sequential sampling operations. Second, SVGD is well-suited to multi-modal applications due to its ability to maintain diverse modes with fewer particles. SVBP also defines the kernel function in Equation (5.6) over individual nodes in the graph. This makes SVBP well-suited to high-dimensional problems which can be represented as a graph.

Computing Gradients. SVBP requires that potentials ϕ and ψ are differentiable. The message gradients can be computed as follows:

$$\nabla_{x_s} \log \hat{m}_{t\to s}(x_s) = \frac{\nabla_{x_s} \hat{m}_{t\to s}(x_s)}{\hat{m}_{t\to s}(x_s)} \tag{5.7}$$

$$\nabla_{x_s} \hat{m}_{t\to s}(x_s) = \frac{1}{M} \sum_{j=1}^{M} \frac{\phi_t(x_t^{(j)}, z_t)}{W_t(x_t^{(j)})} \left[\nabla_{x_s} \psi_{ts}(x_t^{(j)}, x_s)\right] \prod_{u\in\rho(t)\setminus s} \hat{m}_{u\to t}(x_t^{(j)}) \tag{5.8}$$

We note that the gradient update from Equation (5.6) only involves evaluating gradient information from immediate neighbors, since the messages $m_{u\to t}$ in Equation (5.3) are not a function of x_s. This enables efficient gradient updates since the algorithm only requires backpropagating through single-hop neighbors.

Sampling Distribution. In practice, we use the current belief of the neighboring node, $p(x_t)$, as the sampling distribution, W_t, where $p(x_t)$ is represented by Stein particles for node t with equal weights. This enables efficient computation of the messages since it eliminates the need to run expensive sampling algorithms like MCMC, as originally proposed.

Message Passing Schedule. We employ a synchronous message passing scheme in which all messages are computed prior to updating each node belief. This enables efficient batch computations of factors and messages suitable for execution on a GPU. However, our algorithm can be employed with other message passing schedules.

Selecting an Estimate. In practice, multiple estimates exist for drawing an estimate from the particle set. In practice, we select the highest weighted estimate for the experiments described. The weights for the particles can be computed after convergence using Equation (5.1), $w_s^{(i)} = \phi(x_s^{(i)}, z_s) \prod_{t\in\rho(s)} m_{t\to s}(x_s^{(i)})$, where the messages are computed using Equation (5.3).

5.5 SVBP for Multi-Robot Perception

The first application on which we validate our algorithm is a simulated multi-robot perception experiment. The objective is to infer the belief, $p(x_s \mid z_s)$, over the robot's 2D position, denoted x_s, for each robot s for a single timestep. We consider the challenging case in which the observation for each agent, z_s, is multi-modal. Specifically, the observation consists of a mixture of Gaussians which contains a component centered around the true position of the robot and randomly sampled noisy components. An example observation and the associated graphical model are shown in Figure 5.2. In addition to the observations, robots observe the displacement to neighboring robots within communication range, creating edges in the

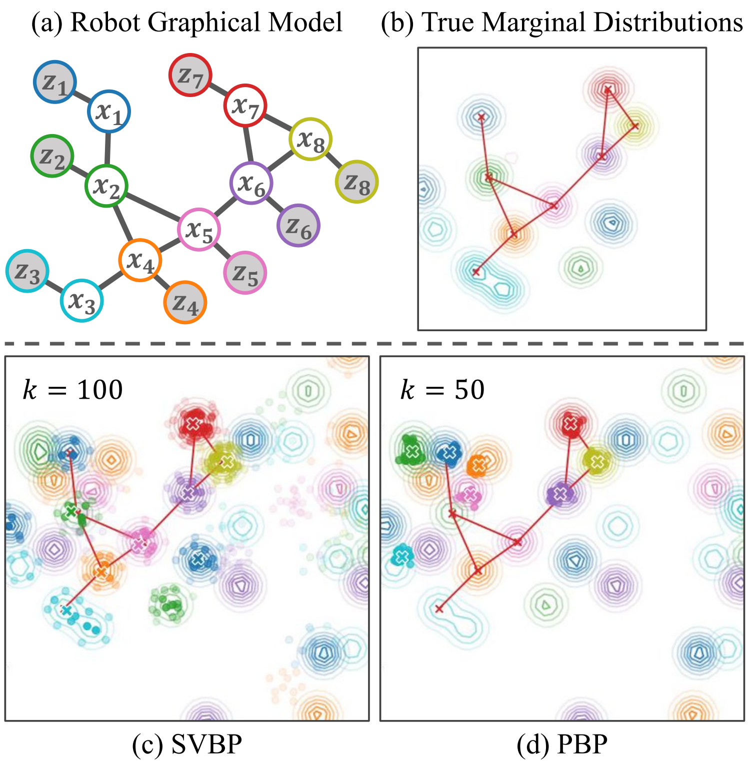


Figure 5.2: SVBP better represents the underlying distribution, avoiding mode collapse. (a) Graphical model of the multi-robot perception problem. The position of each node is denoted x_s, and the corresponding observation is denoted z_s. (b) The approximate true marginals for the graph in (a) and the observation shown in (c, d). Qualitative results for SVBP (c) and PBP (d) at the final iteration (k). The red lines represent the true position of the nodes, and the colored 'x' markers represent the maximum likelihood estimate for each node. Lower-weighted particles are shown with lower transparency. The distributions represent the noisy observations for each node of the corresponding color. Best viewed in color.

graph (shown in red). The resulting marginal distributions for each robot are multi-modal, as shown in Figure 5.2(b).

The MRF in Figure 5.2 requires the definition of the potentials in Equations (5.1) and (5.2). We define the unary potential for each robot to be the mixture of Gaussians corresponding to the robot observation. The pairwise potential is defined as a function of the observed translation L_{st} between neighboring robots:

$$\psi_{ts}(x_t, x_s) = \exp\Big(-\alpha\big(\|x_s - x_t\| - L_{st}\big)^2\Big). \tag{5.9}$$

where x_s and x_t are the 2D positions of neighboring robots and α is a user-selected coefficient.

Baseline. We implement PBP as a baseline approach. We employ iterative importance

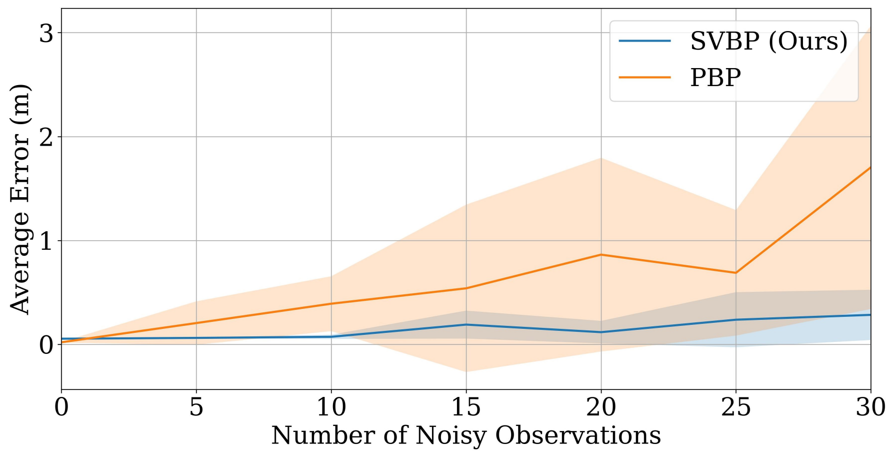


Figure 5.3: Average error for each node estimate for multi-robot localization. Results are shown for varying levels of noise, corresponding to the number of noisy components added to the observation.

sampling over the particles at each node, where each particle is weighted according to Equation (5.1) with the message definition of Equation (5.3). We use the current particle set at each neighboring node as the candidate distribution for message computation, with equal weights, as in SVBP. We apply random noise at the beginning of each iteration. The same factor definitions and parameters are used for PBP and SVBP.

5.5.1 Results

For each run, the position of each robot is randomly selected such that the graph is connected for a radius of 2 meters. To form the observation, a component is added at the true location of the robot. Noisy components with uniformly sampled means are then randomly assigned across nodes and added to the observation, making each observation a Gaussian mixture. Particles are initialized uniformly. SVBP ran for 100 optimization iterations, and PBP ran for 50 iterations. To generate an estimate for each node's position, we select the highest weighted particle.

The average error for 8 nodes over 10 runs for our SVBP algorithm against PBP is shown in Figure 5.3. The x-axis represents the total number of noisy Gaussian components added to the node observations. A visualization of the final belief distributions of SVBP and PBP for the highest noise observation is shown in Figure 5.2. SVBP performs comparatively to PBP for low observation noise, but significantly outperforms PBP in noisy cases. We observed that PBP tends to converge quickly but was subjected to mode collapse which results in locally optimal estimates. In contrast, SVBP maintains multiple modes, making it more likely that the global solution is represented in the candidate particle set.

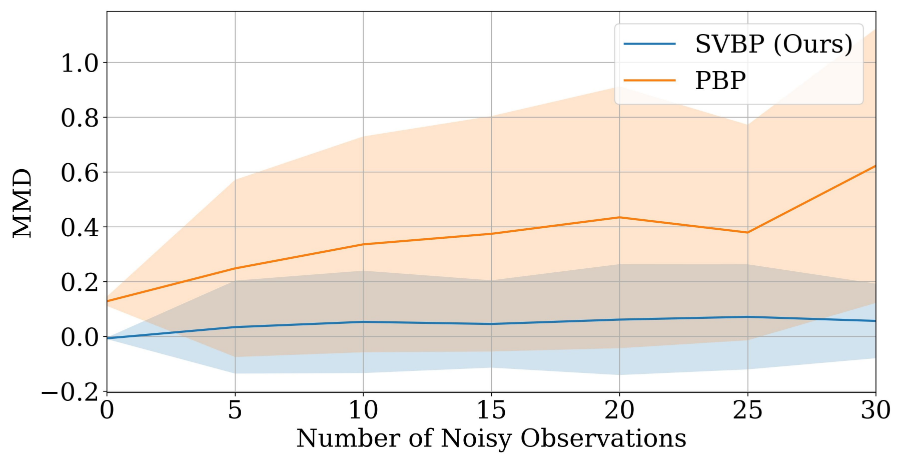


Figure 5.4: The average Maximum Mean Discrepency (MMD) between the samples from the true marginal distribution and the particle sets from SVBP and PBP. Both methods use 50 particles.

Comparison to True Marginals. We hypothesize that SVBP better represents the true marginal distributions. We perform an analysis of the particle distribution of each method compared to the true marginal beliefs. To obtain the true marginal beliefs, we run a Gibbs simulation [26] to sample from the marginal using the density from Equation (5.1). To evaluate the integral for the message in Equation (5.2), we employ Monte-Carlo integration over the full region of the observation with a high number of samples (1000). The ground truth sampled marginals are imperfect due to the sampling procedure but provide a reasonable baseline approximation. The visualization of the true marginal is shown in Figure 5.2(b).

We compute the kernelized Maximum Mean Discrepancy (MMD) [60] between the sampled particle set and the belief particles for SVBP and PBP. The kernel bandwidth is chosen using the median heuristic over the ground truth sample set [55]. Results are shown in Figure 5.4. SVBP obtains a lower MMD than PBP consistently across noisy environments. We observe that some particles in SVBP get caught in local minima in very noisy cases in areas where the unary potential is high, as in Figure 5.2(c). These particles are easily detected as they have very low overall weights and could be reset in practice. We therefore do not include any particles with weights less than 1% of the highest weight in the MMD computation.

Analysis of Number of Particles. We claim that SVBP can represent the marginal beliefs with fewer particles due to SVGD's ability to maintain modes of the distribution. We execute both SVBP and PBP with different particle set sizes and measure the average error across each node for the final estimate. The results are shown in Figure 5.5. For noisy environments, PBP benefits significantly when the size of the particle set is increased from 25 to 100, whereas SVBP finds a good estimate with only 25 particles.

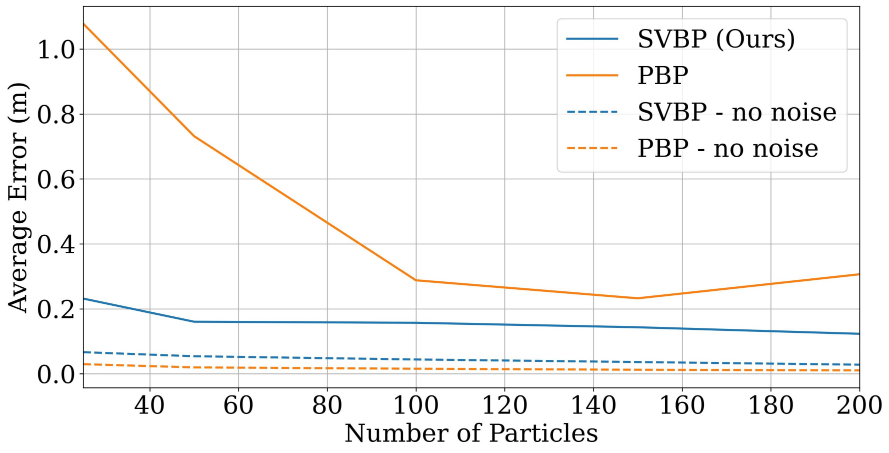


Figure 5.5: Average error for each node estimate for different numbers of particles. The solid lines correspond to experiments runs with noise added to the observation. The dashed lines correspond to experiments with no noise added to the observation.

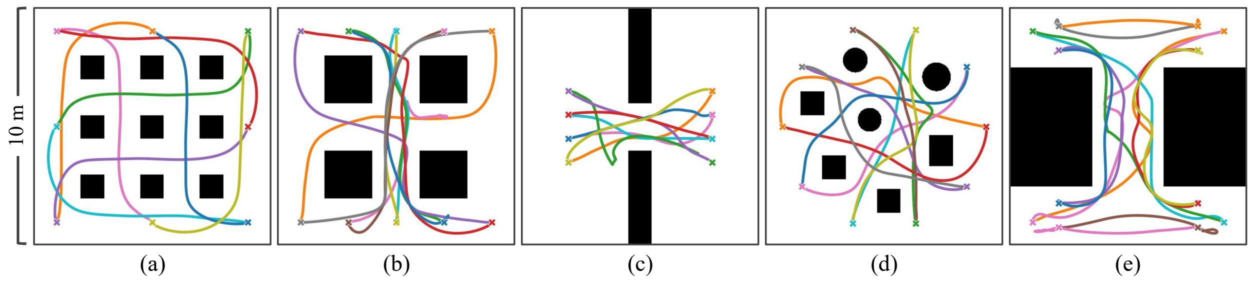


Figure 5.6: Testing environments for the multi-robot control experiments with randomly selected trajectories from SVBP. The goal positions for each robot are marked with an 'x'. Each environment is 10 meters by 10 meters.

5.6 SVBP for Multi-Robot Planning

Our second application involves decentralized Model Predictive Control (MPC) of a multi-robot system. Each robot must avoid obstacles and the other robots in its trajectory to the goal. We run experiments both in a 2D planar navigation simulation and on a decentralized real robot system with realistic sensor and action noise.

5.6.1 Problem Formulation

We consider the problem of finding a collision-free trajectory for each robot s, $\tau_s = \{u_{s,k}\}_{k=0}^{T-1}$, where $u_{s,k}$ are control commands for time k over a fixed horizon T. We take a planning as inference approach [8, 166] in which the nodes in the graph represent the trajectory distribution, $p(\tau_s)$ for each robot, and the edges in the graph represent robots in communication, as in Figure 5.1. We assume known dynamics $\theta_{s,k+1} = f_s(\theta_{s,k}, u_{s,k})$, where $\theta_{s,k}$ is the state of robot s at time k, and a known initial state $\theta_{s,0}$. At each timestep, we execute the first action in the trajectory and rerun the optimization, as in MPC. This approach is akin to a multi-robot version of Stein MPC [81].

For this experiment, we assume the graph is fully-connected. We employ a loopy version of belief propagation, in which the messages are initialized and iteratively updated. This approach does not provide exact marginals but has proven to be effective in practice [117].

Potential functions. The unary potential for each robot trajectory is defined with respect to the running cost $c(\theta_{s,k}, u_{s,k})$ and terminal cost $C(\theta_{s,T})$ for a trajectory:

$$\phi_s(\tau_s, \theta_{s,0}) = \exp - \left(C(\theta_{s,T}) + \sum_{k=1}^{T-1} \gamma_k \, c_s(\theta_{s,k}, u_{s,k}) \right) \tag{5.10}$$

where T is the time horizon and γ_k are constants, and the initial state replaces the "observation," z_s, from Equation (5.1). The running cost consists of a quadratic cost and an obstacle avoidance cost based on the signed-distance function for the obstacles. Intermediate state values $\theta_{s,k}$ needed to compute the costs are obtained by simulated rollouts using the dynamics, $f_s(\theta_{s,k}, u_{s,k})$.

The pairwise potential between communicating robots employs the following collision avoidance factor over the trajectory:

$$\log \; \psi_{ts}(\tau_t, \tau_s) = - \sum_{k=0}^{T} \begin{cases} \alpha_k \left(1 - \left(\frac{d(\theta_{s,k}, \theta_{t,k})}{r} \right)^{\beta} \right) & d(\theta_{s,k}, \theta_{t,k}) \leq r \\ 0 & d(\theta_{s,k}, \theta_{t,k}) > r \end{cases} \tag{5.11}$$

where $d(\theta_{s,k}, \theta_{t,k})$ is the distance between the robot positions at timestep k, r is the desired collision radius, and α_k and $0 < \beta \leq 1$ are constants. In our experiments, we use $r = 0.5$ and $\beta = 0.3$. We set α_k to decrease linearly over the horizon.

Given differentiable dynamics, the above potential definitions allow the gradients from Equation (5.6) to be computed with respect to the trajectories τ_s. We use a Gaussian kernel which employs a distance function computed as the sum of the Euclidean distance between states in two trajectories at corresponding times. Equation (5.5) is applied iteratively to obtain a set of trajectories comprising the belief for each robot, $\{\tau_s^{(i)}\}_{i=1}^{N}$.

5.6.2 Baselines

Two baselines are employed for this scenario: the well-established Optimal Reciprocal Collision Avoidance (ORCA) algorithm [169], and Gaussian Belief Propagation (GaBP), as in [124]. ORCA assumes that neighboring agent's velocity are known and calculates optimal reciprocally collision-avoiding velocities that are closest to the original preferred velocity. The scenario was implemented using the RVO2 library [170]. We assume full connectivity.

For GaBP, potentials are expressed as a linearized Gaussian factor [37] with a bias term that encodes the expected joint Gaussian to be observed. In contrast to the formulation by Patwardhan et al. [124], we represent the trajectory consisting of 2D acceleration commands for one robot as a single node, rather than inferring the state at individual timesteps. We use similar potential functions to our SVBP implementation for fair comparison. The factors in GaBP are restricted to the form:

$$E_s(\tau_s) = \frac{1}{2}(h_s(\tau_s) - b_s)^\top \Sigma_s^{-1}(h_s(\tau_s) - b_s), \tag{5.12}$$

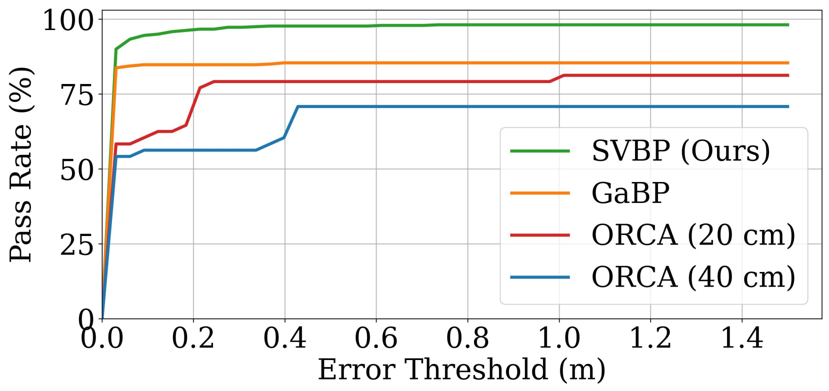
where $h_s(\tau_s)$ is an "observation function" over the trajectory τ_s, b_s is a bias term, and Σ_s is the covariance [37].

In order to use our non-linear, non-Gaussian costs, we set $h_s(\tau_s)$ to be the cost for each of our factors, with $b_s = 0$. Since our costs are non-linear, $h_s(\tau_s)$ must be linearized about an estimate via a first-order Taylor series expansion. As in SVBP, the linearization requires backpropagation through the dynamics $f_s(\theta_{s,k}, u_{s,k})$. Since the quadratic cost is already linear, we use $h_s(\tau_s) = \begin{bmatrix}\theta_s & \tau_s\end{bmatrix}$, where θ_s is the state vector from simulated trajectory rollouts using the dynamics model. Our GaBP implementation is able to infer optimal trajectories without the need of a trajectory planner by making use of the dynamics function, in contrast to the formulation by Patwardhan et al. [124].

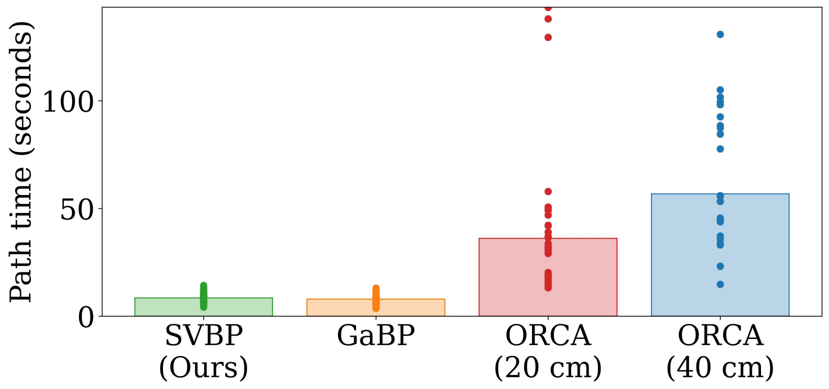
5.6.3 Simulated Robot Experiments

We perform the simulated experiments in acceleration space, where the state $\theta_{s,k}$ consists of 2D position and velocity, and the control commands $u_{s,k}$ are 2D accelerations. We use a time horizon of 20 discrete steps of 0.1 seconds each, making τ_s 40 dimensional for each robot. The first control command from the lowest cost trajectory, equivalent to the heighest weight particle, is executed at each timestep. The optimization is then rerun in MPC-style.

Results. We present the pass rate for ORCA, GaBP, and SVBP in Figure 5.7a. The pass rate represents the percentage of trajectories (y-axis) which reached the goal within a given error threshold (x-axis) across all robots for each run. Any robots that collided with another robot are not counted as passed for any threshold. Since ORCA is sensitive to the robot radius parameter, we show results for both a radius of 20 cm and 40 cm. We perform 10 runs on each of the environments in Figure 5.6. The total path time for each method is shown in Figure 5.7b. Path time is only computed for trajectories which terminated within



(a)



(b)

Figure 5.7: Pass rate (a) and path time (b) for each method considered for the multi-robot control example. The pass rate represents the percentage of trajectories which finished within a given error threshold. Only successful results are included for path time analysis. A trajectory is successful if it reaches the goal within 30 cm without collisions.

30 cm of the goal without collisions. While all methods result in similar path lengths, the robots move much more conservatively in the ORCA baseline, which results in higher path times.

We observe that the failure modes in SVBP can occur due to local minima, for example around large obstacles such as in the environments in Figure 5.6(c, e). GaBP is especially susceptible to getting caught in local minima in the presence of challenging obstacles. A subset of robots fail to reach their goals for every run in one environment, as illustrated in Figure 5.8(b). ORCA is particularly prone to deadlock scenarios when it must obey a collision tolerance (i.e. 40 cm collision radius case), failing for all runs in the environment shown in Figure 5.8(a).

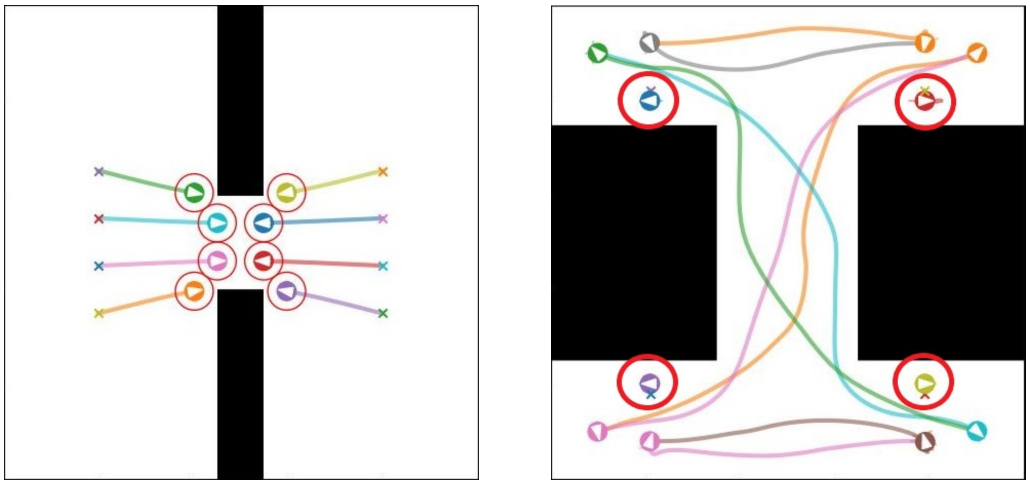

(a) ORCA (40 cm) (b) GaBP

Figure 5.8: Failure modes for the baselines considered for the planar navigation experiment. (a) ORCA is prone to deadlock, especially in the presence of obstacles. All run for this environment fail with a 40 cm radius (shown with red circles). (b) Gaussian Belief Propagation is prone to falling into local minima, especially around large objects. The four robots circled in red cannot get around the obstacles.

5.7 Real Robot Experiments

We run our controller on a real multi-robot system comprised of omni-directional MBots [56]. We perform a collision avoidance experiment with three robots where the robots must cross paths to reach their goal locations. The goal of this experiment is to determine the performance of our controller under 1) noisy perception and dynamics, 2) limited computing resources, 3) realistic asynchronous message passing schedules. The robots are equipped with a single-board computer with limited processing power (a Raspberry Pi) and pass messages through a custom websocket interface over a WiFi connection. Each robot performs SLAM using a 2D Lidar for state estimation. The swarm is assumed to be fully-connected.

Decentralized Message Passing. Each robot independently maintains a representation of the graph and updates messages within their local graph based on neighbor belief. At the beginning of each optimization iteration, the robots request the current trajectory distribution from their neighbors which is used to update the local messages in each robot's representation of the graph. The robots pass messages using a custom API which passes messages through websockets, inspired by *rosbridge* [35], which allows them to synchronously query data from robots on a shared network.

Baseline. ORCA is implemented on the robots as a baseline. The algorithm is run in

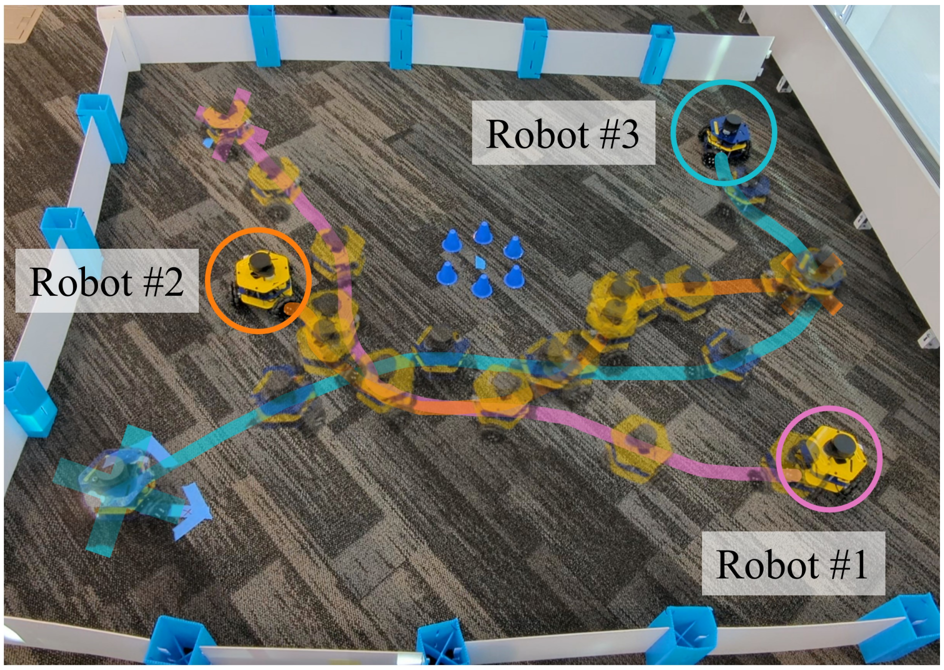


Figure 5.9: An example of a trajectory for the SVBP algorithm on a real robot. The circles highlight the final goal position for each robot.

a centralized manner on a single robot which broadcasts velocity commands to the whole fleet. ORCA outputs a velocity command for each robot rather than a trajectory, therefore we do not use an external trajectory tracker and execute the velocity command directly.

Implementation Details. We perform the simulated experiments in velocity space, where the state $\theta_{s,k}$ consists of 2D position and the control commands $u_{s,k}$ are 2D velocities. We plan over 10 discrete timesteps, with a 0.25 second timestep. We first perform 15 initialization iterations over random trajectory particles before beginning execution. The lowest cost trajectory is chosen and executed by a closed-loop velocity controller. The optimization is repeated until the goal is reached in MPC-style, initializing using particles from the previous timestep.

Results. We perform 5 runs on a scene with and without obstacles (10 runs total) for SVBP and ORCA. The time-to-goal results are shown in Figure 5.10 for each of the robots shown in Figure 5.9. On the scene with no obstacles, SVBP reaches the goal in all runs with no collisions except for in one run, in which one robot has a localization failure resulting in a collision. ORCA deadlocks at the start of the trajectory for all runs. To obtain meaningful comparisons, we manually perturb the robots from their start positions to escape deadlock. ORCA's built-in random perturb for deadlock prevention fails in practice as ORCA tends to select low speeds which are insufficient to displace the physical robots unless large

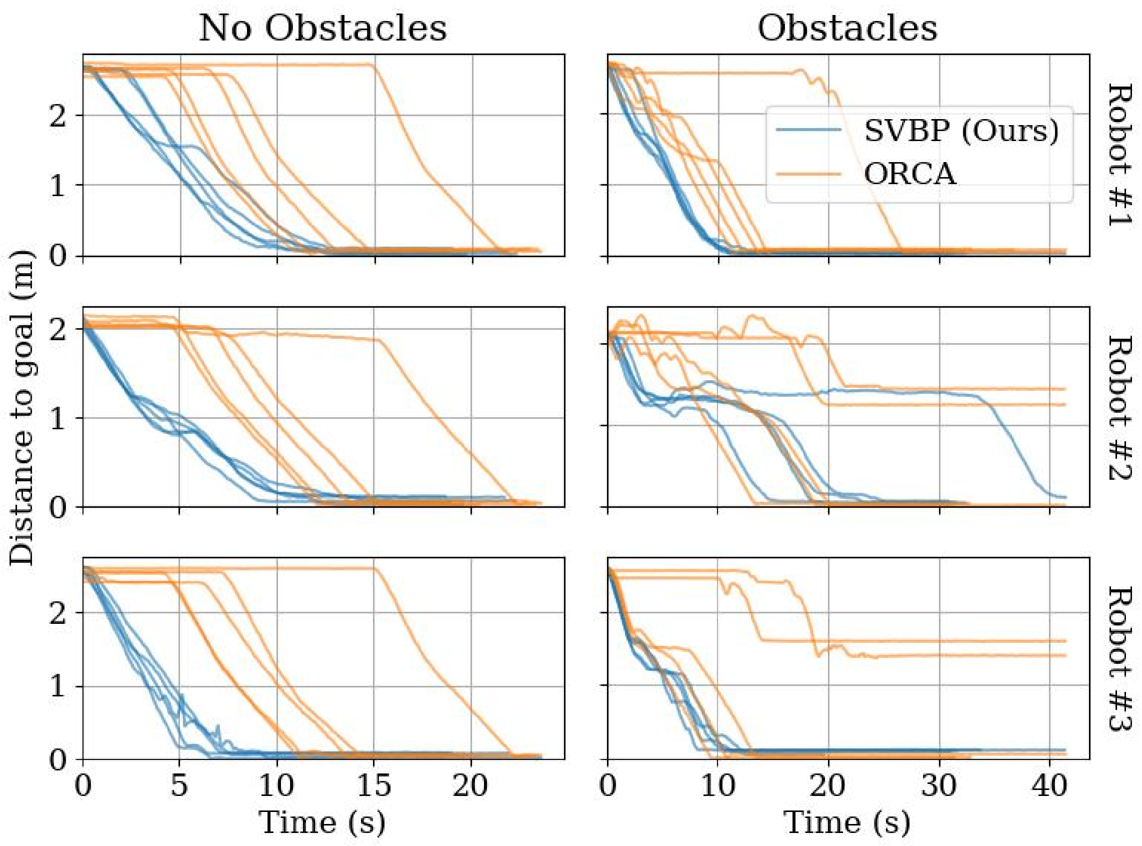


Figure 5.10: Distance to the goal over time for each robot for runs with no obstacles (left) and with obstacles (right).

clearances are available. Modification of the perturbation functionality for this application could mitigate this issue. After the deadlock is resolved, ORCA and SVBP achieve similar time-to-goal in scenarios without obstacles.

For the case with obstacles, ORCA deadlocks at the beginning of the run in two cases. The algorithm gets stuck in deadlock for 2 of 5 runs near the obstacle, and the deadlock results in a collision in one of the cases (robots #2 and #3 do not reach the goal in 2 cases in Figure 5.10, right). We only apply manual perturbation for deadlocks for ORCA at the beginning of the run. SVBP reaches the goals in all the cases with smoother paths. A visualization of an execution of SVBP with obstacles is shown in Figure 5.9.

5.8 Discussion & Conclusion

We present Stein Variational Belief Propagation (SVBP), an algorithm for inferring nonparametric marginal beliefs in graphs using Stein Variational Gradient Descent. We demonstrate the applicability of our algorithm on two applications: simulated multi-robot perception, and multi-robot planning both in simulation and on real robots. Through simulated perception experiments, we show that SVBP approximates the true marginal distributions better and is more particle efficient than sampling-based baselines. The planning experiments show that the algorithm is more effective at escaping local-minima and deadlock scenarios than baselines. The real-world planning experiments show that the method is robust to realistic noise.

A limitation of the proposed algorithm is that the computation time scales with the number of neighbors. We limited the robot experiments to three robots in order to achieve fast enough execution to run MPC on the single-board computers on the robots. Future work will involve improving the implementation efficiency in order to increase the size of the swarm. Another limitation for decentralized control is the need to time-synchronize incoming neighbor messages from other robots. We observe that the robots are prone to starting and stopping behavior when near other robots which we posit occurs due to lack of time synchronization. This could be mitigated by accounting for the time delays between messages. Further study is needed on the impact of message delays and packet loss [162].

Our robot experiments show that our method is robust to realistic perception and action uncertainty, despite the fact that we do not explicitly model this uncertainty. Integrating perception and action noise models into the model in order to deal with more challenging scenarios is an interesting avenue of investigation. We hypothesize that this could also help scale the algorithm to cases with nonlinear dynamics.

CHAPTER 6

Hello, Robot! An Introductory Course for Robotics Undergraduate Education

Figure 6.1: *Hello, Robot! Introduction to AI and Programming* is a course which teaches the fundamentals of robotics and programming using real robot platforms. The course is taught in a distributed teaching collaborative across institutions. *Photo: Brenda Ahearn/University of Michigan, College of Engineering, Communications and Marketing.*

6.1 Introduction

The rapid growth of robotics and AI in recent years has led to a growing demand for a labor force with specific skills relevant to these fields. This need has resulted in a corresponding demand for training in these fields in higher education. Robotics as a tool for computing education has a rich history, dating back to the seminal work *Mindstorms* by Papert [123]. Since then, Educational Robotics has been used extensively to engage students of all ages in computational thinking [105, 108, 48]. The study of robotics itself has a rich history within university education [180, 164, 14]. Robotics courses have typically been offered at the graduate level or as specialized technical electives at the upper undergraduate levels. Advancements in robotics and AI have resulted in the need to expand education in the field and establish *robotics as a discipline* within undergraduate education.

The development of robotics as a discipline is exemplified by the recent emergence of undergraduate programs in robotics [57, 71]. The demand for robotics undergraduate education created by these programs has resulted in the need for new curricula in robotics throughout the undergraduate degree. While robotics is a broad field encompassing the intersection of multiple domains of engineering, we focus our discussion in this chapter to treatment to the algorithmic elements of robotics. Computational robotics is distinct from computer science in that computer science is concerned with programming computers, while robotics is concerned with programming embodied robotic devices. The latter presents unique challenges which necessitate specialized algorithmic techniques.

This chapter is concerned with the development of the course *Hello, Robot! Introduction to AI and Programming (HelloRob)*, an introductory programming class presented through the lens of robotics and AI, offered as part of the first year of the Robotics major at the University of Michigan [71]. The development of introductory computer science courses has received extensive attention in the field of Computing Education Research (CER), commonly referred to as *CS1* [12]. In contrast, this chapter explores the design of an introductory *robotics* course, which we term *ROB1*. The *HelloRob* curriculum shares a common goal with CS1 courses in developing computational thinking and programming skills, but is concerned with introducing key components of a robotic system and the unique challenges of programming a robot.

The *HelloRob* course was first offered in 2021 at the University of Michigan, and has subsequently been offered at Berea College, Howard University, and Morehouse College. The course has been taught to over 100 students, in collaboration with seven instructors across four participating institutions as of the publication of this dissertation. In this chapter, we describe the design of this course contextualized within the existing literature on computing and robotics education. The treatment of the design of the course is three-fold: first, we describe the curriculum of a ROB1 course focused on computing. Second, we describe the design of the educational robotic platform and associated tools used in the instruction of the course. Third, we discuss the distribution of the curriculum to diverse institutions. Finally, we conclude with a discussion of the key takeaways from the development and instruction of a ROB1 course and recommendations of best practices for future work.

6.2 Related Work: Robotics in Higher Education

Robotics courses have been offered in undergraduate education for decades. In many cases, robotics courses involve hands-on learning activities, which introduce the challenge of selecting a real robot platform and associated tools. This section focuses on robotics curricula in higher education and the associated hardware and software employed. We first summarize efforts in computing education which leverage robots. Next, we describe curricula which focus on robotics as a discipline. Finally, we turn our attention to the platforms and tools which are an essential component of the design of curricula in robotics.

It is worth noting that many robotics education initiatives conducted in higher education are not published in official proceedings or made public online. As a result, the works listed here form an incomplete history of educational robotics.

6.2.1 Computing Education Through Robotics

Robots have a rich history within the field of CER, pioneered with the introduction of the programmable "Turtle" as part of the LOGO programming language [146]. Since then, robotics has been used extensively to teach principles of computing and engineering to students of all ages [105, 14]. Here, we focus on computing education through robotics as it applies to undergraduate education.

Introductory computer science (CS1) courses which employ robots typically involve programming assignments which are accomplished on physical robot hardware. The release of user-friendly platforms like the LEGO ® Mindstorms robot, which takes its name from Papert's original work [123], sparked the use of such platforms for computing education [49, 147]. Today, a vast array of robotic platforms and competitions exist for robotics as an educational tool [48, 52] (see Section 6.2.3). A pioneering work in using robots in computing education was done as part of the Institute for Personal Robotics, a multi-institution collaboration which aimed to provide a CS1 curriculum developed around a low-cost robot platform [9]. This initiative centered the notion that students could use personal robots as a tool for learning computer science similar to a personal computer or textbook.

Many of the efforts in robotics for computing education motivate the use of physical robots as a way of motivating and engaging students. The evidence to support this claim is largely anecdotal. Notably, one study found that using physical robots did not significantly affect student motivation [105]. The discrepancy in this result and positive anecdotal evidence from instructors could be explained by the fact that students enrolled in university courses which employ robots self-select based off their interest in these courses. Another key factor is in the *design* of and *support* for the robotic platform itself. The design of robotics for education is largely an open problem and more investigation is needed into how the design affects student experience [141]. We offer insights into this topic in Section 6.4.

6.2.2 Teaching Robotics as a Discipline

Robotics courses have been taught by pioneers in the field of robotics research for decades, primarily at the graduate or senior undergraduate levels. Early courses in robotics focused on theory and algorithms for robot modelling and control [180] and planning and localization [164]. These curricula primarily emphasized established robotics algorithms alongside cutting-edge research algorithms. While robotics courses are largely unpublished, the rise of education in topics in robotics can be tracked by the publication of foundational textbooks on robot control, planning, and localization [161, 84, 148, 46]. These textbooks and their associated curricula achieved wide-spread popularity and are instrumental to the education of modern roboticists.

Early efforts in robotics education have informed the creation of a vast array of modern courses in robotics which strive to keep pace with the rapidly evolving techniques in the field [125, 149, 25]. Alongside the development of such courses, particular attention has been placed in recent years on democratizing robotics education, following the spirit of the open-source community. This effort was enabled in part by the Robot Operating System (ROS) [130], an open-source framework for robot programming with an active online community in research and education. Robotics courses are offered through Massively Open Online Courseware (MOOC) platforms such as Coursera and Udacity. Duckietown is a multi-institution collaboration which teaches concepts in robotics through the lens of autonomous driving using real robot platforms, offering materials online to replicate the course at other institutions [125]. The course is suited to senior undergraduate or graduate students. Other work has focused on open-sourced, online textbooks or course material, often published alongside programming activities and recorded lectures [160, 38].

In recent years, the evolving maturity of robotics as a discipline has enabled the development of curricula offered throughout the undergraduate experience. Most significantly, robotics has begun to appear as a field of study in the form of undergraduate majors [57, 71]. These curricula follow the decades of robotics education initiatives and are a natural extension of graduate degrees in robotics. The increase in demand for undergraduate education in robotics as a discipline motivates the need to develop new curricula, platforms, and tools suitable for students starting in the early undergraduate years.

Table 6.1: Comparison of available robot platforms for education

	Sensors						Programming
Platform	⚙	📏	✱	📹	Drive Type	Computation	Skill Level
Pololu 3Pi+	✓	✓	–	–	DD	MC	CS1
SparkFun RedBot	✓	✓	–	–	DD	MC	CS1
Parallax Scribbler	✓	✓	–	–	DD	MC	Block / CS1
Vex V5 Kit	✓	✓	–	–	Multi	MC	Block / CS1
LEGO Mindstorms	✓	✓	–	–	Multi	MC	Block / CS1
DuckieBot	✓	✓	–	✓	DD	SBC	Adv.
JetBot	✓	✓	–	✓	DD	SBC	Adv.
TurtleBot 4 Lite	✓	✓	✓	✓	DD	SBC	Adv.
MuSHR	✓	✓	✓	✓	Ack	SBC	Adv.
MIT Racecar	✓	✓	✓	✓	Ack	SBC	Adv.
AgileX Limo	✓	✓	✓	✓	Multi	SBC	Adv.
MBot (Ours)	✓	✓	✓	✓	Multi	SBC	CS1 / Adv.

⚙ = Odometry; 📏 = 1D sensor; ✱ = Lidar; 📹 = Camera
DD = Differential; Ack = Ackermann; Multi = Reconfigurable
MC = Micro-Controller; SBC = Single-Board Computer

6.2.3 Platforms and Tools for Robotics

The era of scalable and programmable robot platforms was catalyzed by Martin's *Robotics Explorations* book [102] and the introduction of the LEGO Mindstorms. Following the success of LEGO Mindstorms in computing education [49, 147] were a number of highly impactful mobile robot platforms for education. These platforms included the Parallax Scribbler [9, 157], Sony AIBO [172], and modified versions of the iRobot Roomba [43, 32, 83, 168], as forerunners to the Willow Garage Turtlebot [5]. Robotics education also saw the development of its own robot programming environments, such as Pyro [20] and Tekkotsu [165], more amenable to undergraduate teaching than common robot middleware frameworks.

When selecting a robot for education, a number of factors can be considered. A selection of robotic platforms used for education are summarized in Table 6.1. We follow the definition from Fine et al. [52] for computing, and expand on the proposed sensor diversity criteria by the same authors to specify notable sensors of interest. When considering which tasks students can accomplish using the robotic platform, the sensors available play a significant role. Odometry (⚙) and 1D sensors (📏), which include distance sensors, touch sensors, and light sensors, are suitable for tasks involving simple feedback loops and state machines (e.g. wall following, line following). Higher-dimensional sensors, like Lidars (✱) and cameras

(📹), are suitable for advanced state estimation and perception, including Simultaneous Localization and Mapping (SLAM) and computer vision.

The computation capability available also impacts the algorithms that can be run. Microcontroller-based platforms (MC) are capable of running one single-threaded program at a time, whereas platforms with more advanced computing in the form of Single-Board Computers (SBC) (e.g. Raspberry Pi, NVIDIA Jetson Nano) are capable of running full operating systems. The former are therefore better suited to basic tasks like reactive control, whereas the latter can run high-level robotic algorithms like SLAM. Educational applications have heavily influenced development of commercially available microcontrollers [22, 10] and single-board computers [16, 6]. Morever, the configurability of the platform itself is also central to being customizable to a particular course objective. Some offer a configurable drive type, while some have a fixed hardware configuration.

Programming skill required is a critical characteristic when considering a platform for an undergraduate course. Evripidou et al. [48] categorized platforms as 'No-code', 'Basic-code', and 'Advanced-code' to describe coding level. We slightly modify these to capture the technical skills, besides coding, required. "Block" refers a platform that offers a block-based programming interface. A CS1 programming skill level is defined as the ability to write, compile, and execute single-threaded programs. Advanced (Adv.) programming skills include executing multiple, multi-threaded programs, familiarity with concepts like the Linux command line, and use of advanced frameworks (e.g. ROS [130]).

Based on the factors defined above, commercially available platforms, like the VEX Robotics Kit, the Pololu 3Pi+ or the SparkFun RedBot, are well-suited to K-12 or CS1 classes, concerned with learning fundamentals of programming through embedded systems. Platforms in this category have been used extensively in robotics for CS1 education [49, 9, 157]. These usually provide a simple way to program the robots, like with a custom library in the Arduino IDE, making the technical prerequisites required suitable for a beginner audience. However, the sophistication of behaviours students are able to program into the robot is typically limited. Standard robotic algorithms like SLAM, path planning, or computer vision cannot be supported by these platforms.

On the other end of the spectrum are robots like the NVIDIA JetBot, Turtlebot [5], Duckiebot [125], MIT RaceCar, and MuSHR [149], which are capable of interfacing with multiple high bandwidth sensors, cameras and Lidar, and utilize multi-threaded sophisticated robotics software and algorithms. These robots typically require advanced technical skills, as they are primarily designed for senior undergraduate- or graduate-level students, or robotics researchers. For example, in order to control the Duckiebot, users need a Linux computer and must use Docker and ROS via multiple command-line interfaces. These robots are therefore best suited to senior undergraduate or graduate curricula.

The MBot [56], a platform developed at the University of Michigan, was initially intended for the instruction of a graduate-level course as part of the Robotics graduate program. Its original design fell into the second category of high capability but advanced platforms. For the instruction of the *HelloRob* course, we expanded the initial design of the platform to include a beginner-friendly API and tools in order to make the platform accessible within a ROB1 curriculum, without loss of sophistication in the platform. The design is discussed in detail in Section 6.4.

6.3 Hello, Robot! Introduction to AI and Programming

Hello, Robot! Introduction to AI and Programming (HelloRob) is an introductory programming course through the lens of robotics and AI offered to first-year students as part of the University of Michigan's Robotics undergraduate degree [71]. The course is intended to introduce students to computational thinking and foundational concepts in robotics which they have the opportunity to explore in greater depth later in their studies. The course is composed as a collection of modules, each of which culminate in a project implemented on an omni-directional robot platform, a variant of the MBot ecosystem of educational robots [56] (shown in Figure 6.1).

When considering curriculum design for the course, we found that existing CS1 courses which utilized robots [9] did not include the breadth of robotics-specific algorithms that we planned to cover. Existing robotics courses [164, 125] contained technical tools and algorithms that were too advanced for a first-year undergratuate audience. See Sections 6.2.1 and 6.2.2 for an overview of these courses. We therefore elected to design a curriculum for *HelloRob* which met the following design criteria:

- **DC 1** Accessible to first-year students without prerequisites in programming or math beyond high school,
- **DC 2** Include modern robotic algorithms, such as feedback control and path planning,
- **DC 3** Scalable across diverse institutions.

In order to satisfy **DC 1**, the course begins with a preliminary module (Module 0) on introductory programming in C++. The modules are designed such that the first module involves practicing programming skills from Module 0 through execution of a project on a

Module	Contents	Project
0. Intro. to C++	Variables, operators, control flow, functions, arrays.	Pocket Calculator
1. Feedback Control	Bang-bang control, proportional control, sensor data, omnidrive geometry.	Wall Following
2. Autonomous Navigation	State machines, coordinate frames, odometry.	Bug Algorithm
3. Path Planning	Graph search, structs, occupancy grids.	Graph Search
4. Image Classification	Image data, nearest neighbors, linear classification, neural networks, gradient descent, cross-validation.	Robot Tour Guide

Table 6.2: Curriculum description for the course *Hello, Robot!*

real robot. The course begins with reactive control (wall following and bug navigation) which rely on computing concepts of feedback control and state machines, typical of CS1 courses which utilize robots. To satisfy **DC 2**, Modules 3 and 4 provide a high-level introduction to path planning and image recognition, relying on the concepts of graph search and machine learning. The modules and associated projects are detailed in Table 6.2.

The projects increase in complexity throughout the semester, however special care is taken to provide a level of abstraction appropriate for a ROB1 course. Our working definition of "appropriate level of abstraction" is that each project is implemented as a single-threaded program which uses only CS1 concepts. In order to accomplish the projects in Modules 3 and 4, which rely on mapping, localization, and vision capabilities, students program the robot through a custom, synchronous API. Details of the platform are described in Section 6.4.

Challenges & Opportunities. The abstraction provided in later modules proves challenging for some students, particularly for students with minimal prior programming experience. Another challenge commonly faced by students is using mathematical concepts (e.g. geometry, trigonometry, and linear algebra) within the context of robotics algorithms. More work is needed towards a standardized robotics curriculum to define the programming and mathematical skills and prerequisites for courses at each level of a robotics curriculum.

6.3.1 The Distributed Teaching Context

DC 3 is particularly challenging to achieve given the diversity of higher-learning institutions in the United States. To address it, *HelloRob* is taught in a *Distributed Teaching Collaborative (DTC)*, in which teaching partners teach the course in parallel to the University of Michigan offering, maintaining close collaboration between instructors. Students can also interact via a shared messaging forum between offerings. Between 2021 and 2024, *HelloRob* was offered as part of a DTC at Berea College (2021, 2023), Howard University (2023), and Morehouse College (2024).

Each offering provides customizations depending on the audience.[1] Berea and Morehouse Colleges offered the course to an audience made up primarily of second- and third-year engineering students. The introductory programming (Module 0) was offered as an optional module since students have prior programming knowledge. The Howard University course was offered to second- and third-year computer science students. In order to fit in to the existing computer science program at Howard University, the offering was taught entirely in Python and skipped the introductory programming module.

This demonstrates a key benefit of the distributed teaching collaborative approach: improvements and customizations made by another institution benefit the entire collaborative. For example, collaborators from Berea College contributed in-class activities based on the principles of active learning in which the college specializes. These new resources improve the quality of the course and offer greater flexibility to future instructors. Research into curriculum adoption has found that creating community around an educational innovation is key to its propagation and maintenance [67].

Challenges & Opportunities. It is equally important to consider the challenges across different institutions in teaching a ROB1 class. In teaching *HelloRob*, hardware issues were common for the institutions besides the University of Michigan, which caused delays in the course progression. This can be attributed to the fact that at other institutions, students and faculty were responsible for robot assembly, maintenance, and repair. At the University of Michigan, this was mitigated by allocating resources to offload hardware management through staff support to manage robots. Without these resources, students in the other offerings were expected to set up and maintain their robot hardware. The result is that these offerings include learning outcomes which focus on proficiency in robot hardware. By making hardware assembly and maintenance a key component of the course, students can develop problem-solving skills related to the unique challenges of programming a robot.

[1]We note that offering *HelloRob* as an introductory course at the University of Michigan was made possible by the undergraduate Robotics major. At other institutions, the course was offered as a special topic course to senior students.

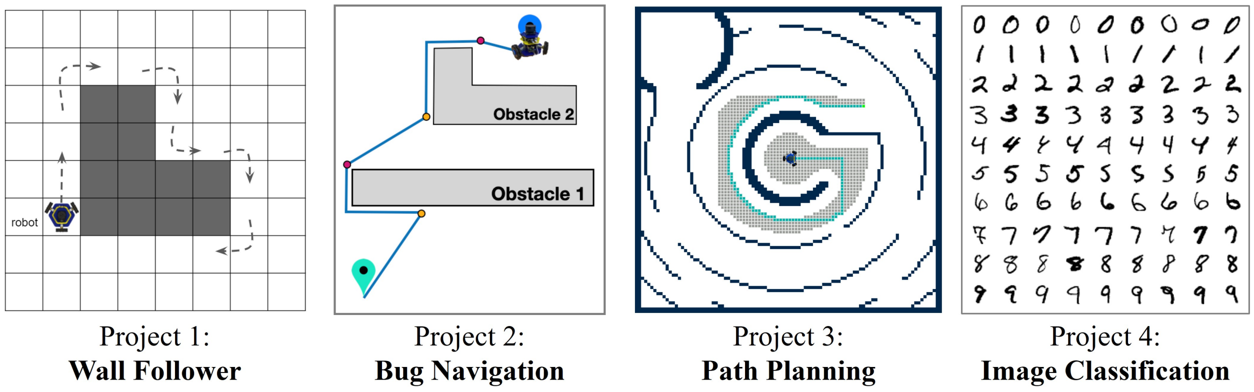


Figure 6.2: An illustration of the four primary projects students complete as part of the course *Hello, Robot! Introduction to AI and Programming.*

6.4 The MBot Ecosystem for Distributed Robotics Education

Central to the development of a robotics educational curriculum is the selection of robotics hardware, software, and infrastructure to support learning. The use of real robots in computing and robotics education is commonly considered to be motivational to students. Additionally, the challenges in dealing with real-world sensor data and challenges of real-world hardware are central to the field of robotics.

The design criteria for the robotic platform suitable for the ROB1 course described in Section 6.3 are as follows:

DC 1 Capable of running algorithms required for ROB1 (see Section 6.3 **DC 2**) (e.g. localization, mapping, vision),

DC 2 Relies on technical prerequisites appropriate for a CS1 student (see Section 6.2.3),

DC 3 Configurable for customizations across different courses.

When selecting a robot platform, a number of existing commercial and open-source platforms existed in the robotics education community at the time of the initial offering of the course in 2021. Despite the breadth of options, existing platforms were unable to satisfy the needs of the course. Microcontroller-based options [49, 147, 9] satisfy **DC 2**, but are limited in their ability to run multi-threaded processes needed to execute Projects 3 and 4 of the course. More sophisticated, custom options [125, 149] satisfy **DC 1**, but rely on advanced technical knowledge (e.g. Linux and ROS) in order to interface with them. An overview of these platforms is provided in Section 6.2.3.

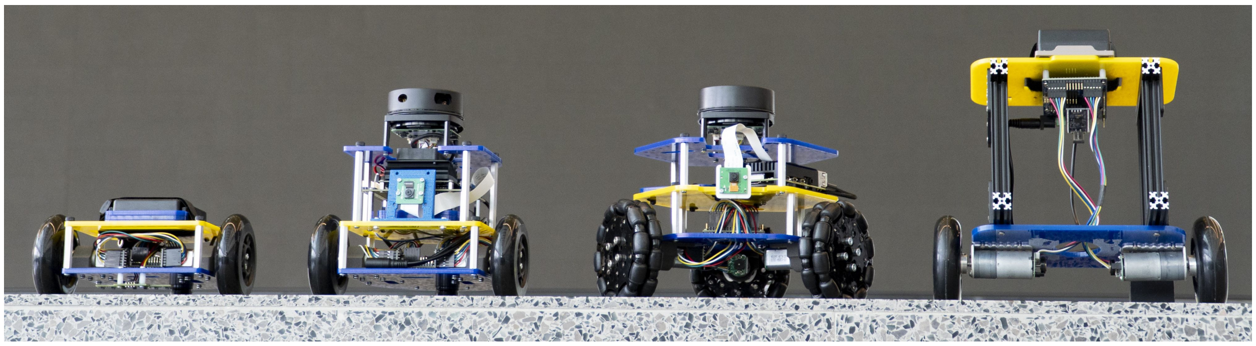


Figure 6.3: The MBot family of robotic education platforms.

We therefore elected to design a platform and suite of tools suitable for the ROB1 course described in Section 6.3. We extend the **MBot**, a low-cost, flexible platform for robotics education at the undergraduate and graduate levels designed at the University of Michigan [56] (see Figure 6.3). The platform was initially designed to teach a graduate-level robotics lab covering mapping, localization, and planning. The design of such a robotic platform for education is a critical component of the student experience within the course. Motivated by recent call for deeper discussion of the design elements which form a key component of computing education [141], this section examines the design considerations for a robot platform suitable for ROB1 education.

The *HelloRob* DTC initiative marks the first use of the MBot platform outside of the University of Michigan as part of the *HelloRob* DTC, and the first use of the MBot to teach an introductory course. In order to use the MBot in a first-year course, we develop a comprehensive set of open-source software tools and a custom API for high-level programming, alongside educational modules. This contribution is central to the versatility of the MBot as an education platform, making it suitable for use in both advanced and introductory courses.

The robot can be programmed through multiple modes depending on the needs of the user and on the application. The software supports advanced autonomy applications such as mapping, localization, and perception through message-passing frameworks. Alternatively, users can program the robot using the custom *MBot Bridge API*, which provides a simple, synchronous interface to the autonomy processes and is the primary form of programming the robot in the *HelloRob* course. The MBot platform also features a custom web application for remote control and visualization, which can be accessed from any personal computer with no dependencies. The MBot software and tools are further described in Section 6.4.1.

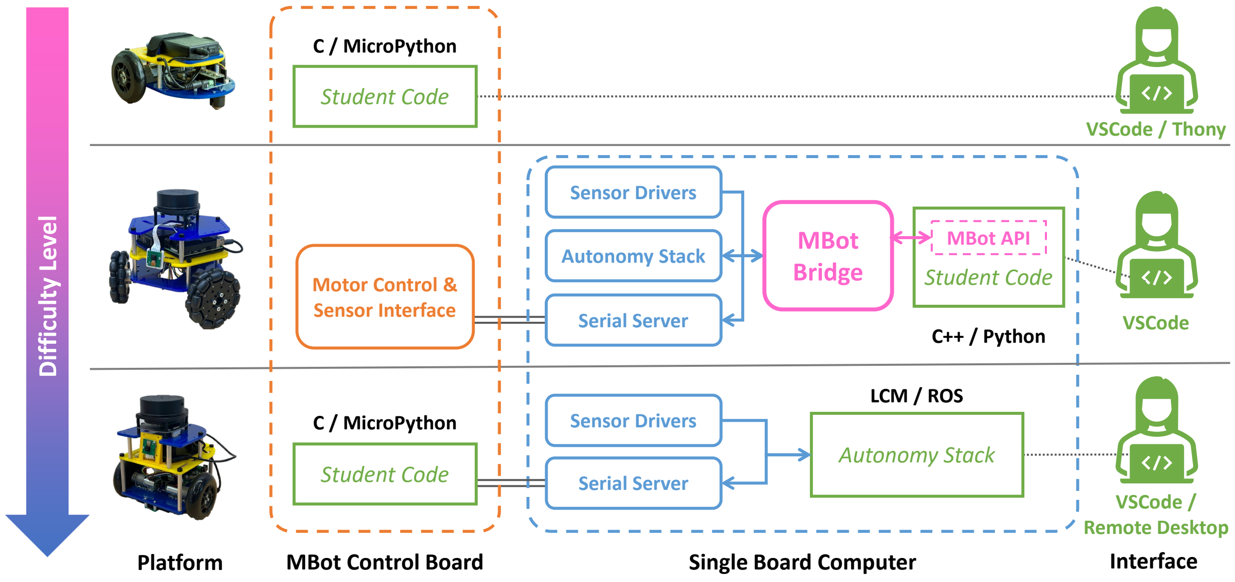


Figure 6.4: Possible software configurations for different levels of robotics courses. The Basic configuration with no single board computer can be programmed directly for basic applications using C or MicroPython (top). Classic and Omni versions can be programmed through the MBot Bridge API, which provides a simple interface for programming the MBot (middle). Advanced applications can interface directly with the message passing framework (bottom).

6.4.1 The MBot Software Stack

The MBot software architecture is designed to be both flexible and easy to use. The architecture enables users to select an interface based on the desired application and level of difficulty, making the platform suitable for introductory and advanced courses. The software for various configurations is shown in Figure 6.4. The MBot comes equipped with a full software stack, including sensor drivers, mapping and localization, and path planning, built on the asynchronous message-passing communication protocol Lightweight Communications and Marshalling (LCM) [68].[2] For single-threaded, synchronous applications relevant to a ROB1 course, the *MBot Bridge API* provides a simple interface to the MBot software stack (see the middle row of Figure 6.4). For advanced applications, the robot can be programmed by interfacing directly with the core software stack through an asynchronous message passing framework (e.g. LCM [68] or ROS [130, 96]). This addresses **DC 3** and enables the MBot to be used throughout an undergraduate curriculum.

The utility of this design for ROB1 is exemplified by *HelloRob* Projects 3 and 4, described in Section 6.3. These projects require students to use the MBot's SLAM system for mapping

[2]While the MBot software uses LCM, the platform is also compatible with other frameworks such as the Robot Operating System (ROS) [130, 96].

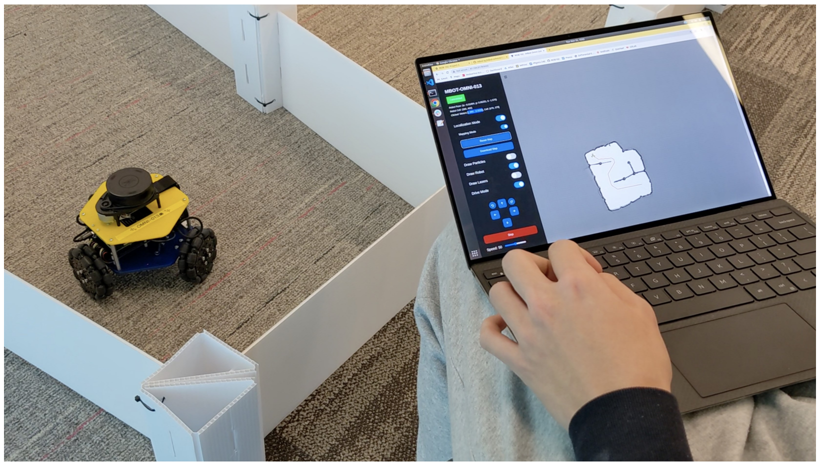


Figure 6.5: The MBot Web App in action. The web app can be accessed through a browser on any device connected to the robot's network, and enables teleoperation of the robot, control over the SLAM state, and visualization.

and localization as a prerequisite to path planning (**DC 1**). To make this capability accessible for a ROB1 audience (**DC 2**), students interface with the mapping module through the web app, then retrieve the localization information through the API. Students program the robot by connecting to the MBot's single-board computer using a remote session on a local IDE (e.g. VSCode's Remote extension). This enables the robots to be programmed with minimal configuration on a personal computer, making the platform ideal for a ROB1 course.

The MBot Bridge API. The MBot supports synchronous programming in C++ and Python through the MBot Bridge API. The API provides a simple interface for reading robot data and sending control commands in single-threaded programs. The API depends on the MBot Bridge Server, which manages incoming messages from the software stack and stores them in queues. The server exposes a websocket-based protocol using a custom JavaScript message definition inspired by ROS Bridge [35].

The MBot Web App. Core to the philosophy behind the MBot system is that the robots should be user-friendly from the perspective of a typical undergraduate student. The MBot Web App provides visualization of the robot's map, position, and Lidar scan. It also includes a driving interface and controls for the robot's mapping system. The app is hosted directly on the robot's single-board computer, and can be accessed from a browser from any device (e.g. personal computer or cell phone) connected to the same network as the robot. As such, interfacing with the robot through the web app requires no installation or technical prerequisites once installed on the robot. A visualization of the web app is shown in Figure 6.5.

6.5 Ingredients for a ROB1 Course

The growing maturity of the field of robotics has led to the need for undergraduate education in robotics. We argue that the consideration of *robotics as a discipline* requires that concepts in robotics be introduced throughout the undergraduate curriculum. Robotics as a field of undergraduate study is a relatively recent concept, which calls for further attention from the community to develop standards and best practices for robotics curricula. Here, we present key lessons learned from three years of instruction of the *HelloRob* course to lend insight into development of future ROB1 courses.

Development of a standardized Robotics curriculum. Robotics algorithms have primarily been taught at higher undergraduate or graduate levels, leaving a gap in the definition of appropriate curricula for robotics. In Section 6.3, we present a curriculum for a ROB1 course focused on building computational skills at a CS1 level through foundational robotics algorithms. We hope that this will spark more study study on the right curriculum for ROB1, similarly to the standards set for computing education [7], which paved the way for extensive development in the following decades [12].

Importance of collaborative development. Following the example of the computer science, AI, and robotics research communities, adopting an open-source philosophy for learning robotics is essential in order to democratize access to these courses across institutions. Our lecture materials and assignments are posted online[3] and students at each institution share common resources. A positive correlation between strong community and successful curriculum adoption has been demonstrated in previous successful cases of education innovation [67].

Robot hardware and tools that match the learning objectives. Many options exist in selecting a platform for a ROB1 course. In contrast to a robot for CS1, a ROB1 robot benefits greatly from the capability to support modern robotics algorithms like mapping and localization. It is important to keep in mind the technical complexity of programming the robot platforms selected for a ROB1 audience. In Section 6.4, we discuss the design of a custom platform and tools towards a platform which is accessible in a ROB1 context but can support courses beyond ROB1. A challenge in teaching with robot platforms is that even the most well-supported, reliable robot requires maintenance and repairs. Introducing educational modules on robot hardware into the curriculum directly enables students to address issues and avoid frustration. If resources are unconstrained, an alternative strategy is to offload hardware management, e.g. through dedicated course staff or through support offered by a robotics company.

[3] https://hellorob.org/

Course staff and community support. A class which centers robot platforms creates additional demands on instructors due to the added infrastructure involved. Thorough and actively maintained documentation for both students and instructors is essential for such a course. Larger course staffs, including graduate and undergraduate student assistants, are hugely beneficial. In *HelloRob*, offerings with smaller course staffs benefited from technical and instructional support from the broader teaching collaborative. As robotics education continues to grow, larger online communities (e.g. online forums) could further alleviate these challenges.

6.6 Ingredients for a ROB1 Educational Platform

Teaching robotics involves unique challenges in terms of tools and infrastructure requirements. The robots, simulators, and tools used can have a significant impact on student experience, which merits further study. Below, we summarize the key features a robot in a ROB1 course should have, drawing from our experience designing the MBot Omni platform to teach *HelloRob*.

Low floor, high ceiling. The call for a platform which is both accessible (low floor) and extendable to advanced tasks (high ceiling) is repeated throughout robotics education literature [9, 164, 56]. In practice, it is challenging to achieve both these goals. In *HelloRob*, we began with a "high ceiling" robot capable of running localization, mapping, planning, and vision applications on top of low-level control. To strive for a "low floor", we leverage web tools, a synchronous API, and a remote development environment. However, added complexity introduces higher development demands and more potential failure modes. As a community, we should strive to build platforms and tools than can adapt to different levels of the undergraduate and graduate curricula, from ROB1 to beyond.

Usability and reliability. User tools and interfaces which provide control, visualization, and diagnosis information are essential for robotics education at the undergraduate level. Platforms designed for K-12 place a significant emphasis on user-centered design, whereas platforms used in the undergraduate classroom tend to resemble research platforms intended for expert users. More effort is needed to build reliable and modular tools to adapt these advanced platforms for undergraduate education.

Resources and support. Another key factor that impacts student and faculty experience in a robotics course is the availability of quality instructional material and documentation on a platform. Additionally, availability of community support is critical for troubleshooting issues that arise. Strong examples of these communities include those around Arduino and Raspberry Pi. The *HelloRob* website includes guides directed towards staff and students, as well as a debugging guide with common issues. Adding more, diverse voices to the development community as robotics education scales will help grow these resources.

6.7 Conclusion

The emergence of *robotics as a discipline* requires that we rethink how robotics is taught in the undergraduate classroom, particularly in the introductory levels. In this chapter, we describe a *ROB1* course which covers introductory computing through the lens of robotics and AI, developed and taught at the University of Michigan as part of the recently introduced Robotics undergraduate major. The design of the curriculum exemplifies the opportunities and challenges of teaching robotics concepts at an early undergraduate level, a key component of the formalization of robotics as a field of study. The use of real robot platforms is integral to the curriculum. We present the MBot platform as a modular tool for robotics education. The platform is adapted specifically for teaching ROB1 through the design of tools and infrastructure. These tools include a synchronous API which can be used to write single-threaded programs at a CS1 level while interfacing with advanced systems like mapping and localization. This design enables modern robotic concepts like path planning to be taught in an introductory computing course. We end with key takeaways from our experience teaching a ROB1 course. These insights suggest opportunities and best practices for future efforts at the intersection of robotics and undergraduate education towards training the next generation of roboticists.

CHAPTER 7

Conclusion

This dissertation considers distributed probabilistic inference techniques and their applicability to robotic perception and planning. We argue that the consideration of uncertainty is critical to robust operation under challenging environmental conditions such as cluttered and dynamic scenes. Through works in perception and planning, we describe advancements in flexible, scalable techniques for inferring belief using nonparametric distributions and distributed inference. First, we consider pose estimation of articulated hand tools in cluttered scenes. Our key insight is that representing the tool as a collection of its component parts enables the use of distributed inference. This approach leverages known prior information about the object structure jointly with local information from the sensor observation and results in improved robustness in cluttered scenes. Second, we consider the problem of planning, particularly in the case where the goal is intractable to describe, resulting in uncertainty over the goal region. We introduce an approach which considers goal demonstrations as samples from a distribution and describe a novel framework for planning under such goals. The proposed method generalizes across different types of goals and outperforms heuristic approaches to handling goal regions. Third, we build on the previous techniques, considering the problem of planning for multiple coordinating robots in a decentralized framework. We introduce Stein Variational Belief Propagation (SVBP), an algorithm for graphical inference, and demonstrate that it is effective at maintaining multi-modal distributions.

Finally, inspired by the rapid growth of the field of robotics, we turn our attention to robotics education. We argue that the expansion of robotics courses and programs at the undergraduate level necessitates the formalism of a *ROB1* curriculum to meet the needs of the study of robotics as a discipline. We describe an intrpductory computing course with a robotics context as an example of a ROB1 course, and the associated robotics hardware and tools developed to support it. Furthermore, we propose a distributed approach to developing and teaching the course through a *Distributed Teaching Collaborative (DTC)*, drawing inspiration from distributed inference techniques described in this dissertation. This work aims to help train the next generation of roboticists with the skills to tackle to tackle essential open challenges in robotics. Some of these future directions are described in the following sections.

7.1 Future Directions

In this section, we present possible future directions based on the ideas explored in this dissertation. First, we describe future opportunities arising from the works presented in Chapters 3 to 5. We then propose directions for robotics education based on the work discussed in Chapter 6.

7.1.1 Opportunities in Robotic Perception and Planning under Uncertainty

The technical contributions described in Chapters 3 to 5 suggest multiple opportunities for future exploration towards the goal of versatile, robust robotic assistants. This dissertation has provided evidence that the consideration of uncertainty adds robustness in challenging conditions in both robotic perception and planning. We note, however, that we have largely considered the treatment of perception and planning individually. Chapter 3 considered uncertainty in pose estimation, while Chapters 4 and 5 considered uncertainty in the goal distribution and agent trajectories respectively. Each of these problems was shown to be computationally challenging individually.

The treatment of these problems jointly is an open challenge in robotics research which merits further advancements. To consider these sources of uncertainty jointly will involve both computational considerations to make this challenging problem tractable. There is particular opportunity for innovation in the investigation of novel frameworks and algorithms which encompass perception and planning jointly. We further describe specific future directions below.

Extended Applications of SVBP. The SVBP approach proposed in Chapter 5 is a generic algorithm for graph inference with multiple possible applications which could be explored in future work. For example, the algorithm could be used to solve SLAM, MPC, or planning over more complex and non-linear models as marginalizations through message passing. Another application is the parts-based perception challenge from Chapter 3. These applications provide opportunities for furthering the proposed algorithm.

Learning Parameters and Factors. Much of the work in this dissertation relies on the hand-coding of functions for probabilistic factors and the selection of parameters (e.g. size of particle set, kernel bandwidth, etc.). Chapter 3 takes steps towards learned likelihoods. Future work examined learning likelihoods further via implicit rendering models [86]. Stein Variational Inference, as discussed in Chapters 4 and 5, is well-suited to learning components of the problem due to its inherent differentiability. Exploring the intersection of this algorithm with data-driven approaches is an interesting avenue for future work.

7.1.2 Opportunities in Robotics Education

The ROB1 course discussed in Chapter 6 paves the way for multiple opportunities for advancement in both robotics education and robotics research. Most interesting is the overlap between the two problems. Perhaps most interesting are the opportunities that lie at the intersection of research and teaching. To make robots suitable for the undergraduate classroom requires significant technical innovations which in turn can aide the advancement of robotics research and related tools. As a specific case study, the MBot platforms used to teach *HelloRob*, as described in Chapter 6, proved the ideal platform for experiments in multi-robot coordination in Chapter 5. The same tools that enabled the platforms to be accessible to an undergraduate audience provided fast prototyping capability and easy interfaces for experiments. We hope this work will inspire future developments in robotics as a whole. Below, we enumerate specific avenues for future research which build on the ideas in Chapter 6.

The effectiveness of ROB1. Students in *HelloRob* have largely been successful in completing the course objectives, and feedback has been positive. More principled study is needed into the effectiveness of the course, and ROB1 in general, in preparing students for their future studies. One way to measure the effectiveness is a long-term study into student performance in future courses after having taken ROB1 in their first year.

The effectiveness of robots as educational tools. Another avenue of exploration is to investigate the factors which affect how the use of real robots impacts student learning. The use of concrete representations of abstract concepts, which Papert described as “objects-to-think-with” [123], has been studied in mathematics and computing [2]. Anecdotal evidence suggests that the effectiveness of robots as embodied representations of computing concepts on student learning depends heavily on platforms and tools. Future studies of particular interest would consider physical robots in the context of their practical use. Based on experience teaching with the MBots, we hypothesize that unreliable robots, lack of tools to diagnose hardware issues, or lack of resources to fix them could negate the educational benefit of the platform. Previous studies have also suggested that robots could impede learning if the constraint of programming on the robot reduces the amount of time students can spend practicing the skills [9]. Studying the effects of these and other factors would enable future development to improve educational platforms.

An Open-Source Robotics Education Platform. While the majority of educational materials and tools discussed in Chapter 6 are freely available online, customization of the curriculum is still mainly a task left to individual instructors based on existing case studies. An interesting avenue for future work is the creation of a collection of modules and tools for broad adoption of the curriculum. Investigations into curriculum adoption have shown that community around an innovation is beneficial for propagation and maintenance [67]. The creation of online resources and communities around the open-source ROB1 curriculum and tools could help scale robotics education to diverse institutions.

BIBLIOGRAPHY

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Joel C. Adams, Richard A. Brown, Jalal Kawash, Suzanne J. Matthews, and Elizabeth Shoop. Leveraging the Raspberry Pi for CS education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE '18, page 814–815, New York, NY, USA, 2018. Association for Computing Machinery.

[3] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic A*. *The International Journal of Robotics Research (IJRR)*, 35(1-3):224–243, 2016.

[4] Md Zahangir Alom, Tarek M Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S Awwal, and Vijayan K Asari. The history began from AlexNet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018.

[5] Robin Amsters and Peter Slaets. Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations 10*, pages 170–181. Springer, 2020.

[6] Jonathan A Ariza and Heyson Baez. Understanding the role of single-board computers in engineering and computer science education: A systematic literature review. *Computer Applications in Engineering Education*, 30(1):304–329, 2022.

[7] William F. Atchison, Samuel D. Conte, John W. Hamblen, Thomas E. Hull, Thomas A. Keenan, William B. Kehl, Edward J. McCluskey, Silvio O. Navarro, Werner C. Rheinboldt, Earl J. Schweppe, William Viavant, and David M. Young. Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science. *Commun. ACM*, 11(3):151–197, mar 1968.

[8] Hagai Attias. Planning by probabilistic inference. In *International Workshop on Artificial Intelligence and Statistics*, volume R4 of *Proceedings of Machine Learning Research*, pages 9–16. PMLR, 2003.

[9] Tucker Balch, Jay Summet, Doug Blank, Deepak Kumar, Mark Guzdial, Keith O'hara, Daniel Walker, Monica Sweat, Gaurav Gupta, Stewart Tansley, et al. Designing per-

sonal robots for education: Hardware, software, and curriculum. *IEEE Pervasive Computing*, 7(2):5–9, 2008.

[10] Massimo Banzi and Michael Shiloh. *Getting started with Arduino.* Maker Media, third edition edition, 2014.

[11] Lucas Barcelos, Alexander Lambert, Rafael Oliveira, Paulo Borges, Byron Boots, and Fabio Ramos. Dual online Stein variational inference for control and dynamics. In *Robotics: Science and Systems (RSS)*, 2021.

[12] Brett A Becker and Keith Quille. 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 338–344, 2019.

[13] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, Alvaro Collet, and James J Kuffner. Manipulation planning with workspace goal regions. In *International Conference on Robotics and Automation (ICRA)*, pages 618–624. IEEE, 2009.

[14] Carlotta A Berry, Sekou L Remy, and Tamara E Rogers. Robotics for all ages: A standard robotics curriculum for K-16. *IEEE Robotics & Automation Magazine*, 23(2):40–46, 2016.

[15] Paul J Besl and Neil D McKay. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[16] Thomas Bewley, James Strawson, Saam Ostovari, and Hugh C Briggs. Leveraging open standards and credit-card-sized Linux computers in embedded control & robotics education. In *56th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 0801, 2015.

[17] Mohak Bhardwaj, Balakumar Sundaralingam, Arsalan Mousavian, Nathan D. Ratliff, Dieter Fox, Fabio Ramos, and Byron Boots. STORM: An integrated framework for fast joint-space model-predictive control for reactive manipulation. In *Conference on Robot Learning (CoRL)*. PMLR, 2021.

[18] Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):1103–1130, 2016.

[19] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 2010.

[20] Douglas Blank, Lisa Meeden, and Deepak Kumar. Python robotics: An environment for exploring robotics beyond LEGOs. *ACM SIGCSE Bulletin*, 35(1):317–321, 2003.

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

[22] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. The LilyPad Arduino: Using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 423–432, New York, NY, USA, 2008. Association for Computing Machinery.

[23] Jesse Butterfield, Odest Chadwicke Jenkins, David M Sobel, and Jonas Schwertfeger. Modeling aspects of theory of mind with Markov random fields. *International Journal of Social Robotics*, 1:41–51, 2009.

[24] Berk Calli, Arjun Singh, James Bruce, Aaron Walsman, Kurt Konolige, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research (IJRR)*, 36(3):261–268, 2017.

[25] Luca Carlone, Kasra Khosoussi, Vasileios Tzoumas, Golnaz Habibi, Markus Ryll, Rajat Talak, Jingnan Shi, and Pasquale Antonante. Visual navigation for autonomous vehicles: An open-source hands-on robotics course at MIT. In *IEEE Integrated STEM Education Conference (ISEC)*, pages 177–184, 2022.

[26] George Casella and Edward I George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

[27] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University, Princeton University, Toyota Technological Institute at Chicago, 2015.

[28] Xiaotong Chen, Rui Chen, Zhiqiang Sui, Zhefan Ye, Yanqi Liu, R Iris Bahar, and Odest Chadwicke Jenkins. GRIP: Generative robust inference and perception for semantic robot manipulation in adversarial environments. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3988–3995. IEEE, 2019.

[29] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems (RSS)*, 2023.

[30] Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer, 2020.

[31] Cristina Garcia Cifuentes, Jan Issac, Manuel Wüthrich, Stefan Schaal, and Jeannette Bohg. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2(2):577–584, 2016.

[32] Morgan Conbere and Zachary Dodds. Toys and tools: Accessible robotics via laptop computers. American Association for Artificial Intelligence, 2007.

[33] Adam Conkey and Tucker Hermans. Active learning of probabilistic movement primitives. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids)*. IEEE, 10 2019.

[34] Adam Conkey and Tucker Hermans. Planning under uncertainty to goal distributions. *arXiv preprint arXiv:2011.04782*, 2020.

[35] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, and Odest Chadwicke Jenkins. Rosbridge: ROS for non-ROS users. In *Robotics Research: The 15th International Symposium ISRR*, pages 493–504. Springer, 2017.

[36] Li Dai, Qun Cao, Yuanqing Xia, and Yulong Gao. Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance. *Journal of the Franklin Institute*, 354(4):2068–2085, 2017.

[37] Andrew J Davison and Joseph Ortiz. FutureMapping 2: Gaussian belief propagation for spatial AI. *arXiv preprint arXiv:1910.14139*, 2019.

[38] Frank Dellaert and Seth Hutchinson. *Introduction to Robotics and Perception*. 2023.

[39] Charita Dellaporta, Jeremias Knoblauch, Theodoros Damoulas, and François-Xavier Briol. Robust Bayesian inference for simulator-based models via the MMD posterior bootstrap. In *International Conference on Artificial Intelligence and Statistics*, pages 943–970. PMLR, 2022.

[40] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao-Blackwellized particle filter for 6D object pose tracking. In *Robotics: Science and Systems (RSS)*, 2019.

[41] Karthik Desingh, Odest Chadwicke Jenkins, Lionel Reveret, and Zhiqiang Sui. Physically plausible scene estimation for manipulation in clutter. In *International Conference on Humanoid Robots (Humanoids)*, pages 1073–1080. IEEE, 2016.

[42] Karthik Desingh, Shiyang Lu, Anthony Opipari, and Odest Chadwicke Jenkins. Efficient nonparametric belief propagation for pose estimation and manipulation of articulated objects. *Science Robotics*, 4(30), 2019.

[43] Brendan Charles Dickinson, Odest Chadwicke Jenkins, Mark Moseley, David Bloom, and Daniel Hartmann. Roomba Pac-Man: Teaching autonomous robotics through embodied gaming. In *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, pages 35–39, 2007.

[44] Josip Djolonga and Andreas Krause. Learning implicit generative models using differentiable graph tests. *arXiv preprint arXiv:1709.01006*, 2017.

[45] Anca D Dragan, Nathan D Ratliff, and Siddhartha S Srinivasa. Manipulation planning with goal sets using constrained trajectory optimization. In *International Conference on Robotics and Automation (ICRA)*, pages 4582–4588. IEEE, 2011.

[46] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics.* Cambridge University Press, 2nd ed. edition, 2010.

[47] Clemens Eppner, Arsalan Mousavian, and Dieter Fox. ACRONYM: A large-scale grasp dataset based on simulation. In *International Conference on Robotics and Automation (ICRA).* IEEE, 2020.

[48] Salomi Evripidou, Kyriakoula Georgiou, Lefteris Doitsidis, Angelos A Amanatiadis, Zinon Zinonos, and Savvas A Chatzichristofis. Educational robotics: Platforms, competitions and expected learning outcomes. *IEEE Access*, 8:219534–219562, 2020.

[49] Barry Fagin. Using Ada-based robotics to teach computer science. In *Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education*, pages 148–151, 2000.

[50] Pedro F Felzenszwalb, Ross B Girshick, David Mcallester, and Deva Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1627–1645, 2009.

[51] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[52] Benjamin T Fine, Jory Denny, Nate Dix, and Ashley Frazier. Oh the robots that you can choose: A technical review of mobile robot platforms. *Journal of Computing Sciences in Colleges*, 35(8):126–135, 2020.

[53] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research (IJRR)*, 17(7):760–772, 1998.

[54] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte Carlo Localization: Efficient position estimation for mobile robots. *AAAI/IAAI*, 1999(343-349):2–2, 1999.

[55] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

[56] Peter Gaskell, Jana Pavlasek, Tom Gao, Abhishek Narula, Stanley Lewis, and Odest Chadwicke Jenkins. MBot: A modular ecosystem for scalable robotics education. In *International Conference on Robotics and Automation (ICRA).* IEEE, 2024.

[57] Michael A Gennert and Craig B Putnam. Robotics as an undergraduate major: 10 years' experience. In *2018 ASEE Annual Conference & Exposition*, 2018.

[58] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research (IJRR)*, 23(9):939–954, 2004.

[59] S. Godsill. Particle filtering: the first 25 years and beyond. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7760–7764. IEEE, 2019.

[60] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[61] J-S Gutmann and Dieter Fox. An experimental comparison of localization methods continued. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 454–459. IEEE, 2002.

[62] K. Hausman, S. Niekum, S. Osentoski, and G. S. Sukhatme. Active articulation model estimation through interactive perception. In *International Conference on Robotics and Automation (ICRA)*, pages 3305–3312. IEEE, 2015.

[63] Norbert Henze. A multivariate two-sample test based on the number of nearest neighbor type coincidences. *Annals of Statistics*, pages 772–783, 1988.

[64] Juan David Hernández, Mark Moll, and Lydia E. Kavraki. Lazy evaluation of goal specifications guided by motion planning. In *International Conference on Robotics and Automation (ICRA)*, page 944–950. IEEE, 2019.

[65] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In Kyoung Mu Lee, Yasuyuki Matsushita, James M. Rehg, and Zhanyi Hu, editors, *Computer Vision – ACCV 2012*, pages 548–562, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[66] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.

[67] Christopher Lynnly Hovey, David P Bunde, Zack Butler, and Cynthia Taylor. How do I get people to use my ideas? Lessons from successful innovators in CS education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, pages 841–847, 2023.

[68] Albert S. Huang, Edwin Olson, and David C. Moore. LCM: Lightweight communications and marshalling. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4057–4062. IEEE, 2010.

[69] Alexander Ihler and David McAllester. Particle belief propagation. In *Artificial Intelligence and Statistics*, pages 256–263, 2009.

[70] Michael Isard. PAMPAS: Real-valued graphical models for computer vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1. IEEE, 2003.

[71] Odest Chadwicke Jenkins, Jessy Grizzle, Ella Atkins, Leia Stirling, Elliott Rouse, Mark Guzdial, Damen Provost, Kimberly Mann, and Joanna Millunchick. The Michigan Robotics undergraduate curriculum: Defining the discipline of robotics for equity and excellence. *arXiv preprint arXiv:2308.06905*, 2023.

[72] Jack Jewson, Jim Q Smith, and Chris Holmes. Principles of Bayesian inference using general divergence criteria. *Entropy*, 20(6):442, 2018.

[73] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. STOMP: Stochastic trajectory optimization for motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 4569–4574. IEEE, 2011.

[74] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[75] Peter Karkus, David Hsu, and Wee Sun Lee. Particle filter networks with application to visual localization. In *Conference on Robot Learning (CoRL)*, pages 169–178. PMLR, 2018.

[76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[77] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

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

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 25, 2012.

[80] Oliver Kroemer, Scott Niekum, and George Konidaris. A review of robot learning for manipulation: Challenges, representations, and algorithms. *Journal of machine learning research*, 22(30):1–82, 2021.

[81] Alexander Lambert, Adam Fishman, Dieter Fox, Byron Boots, and Fabio Ramos. Stein variational model predictive control. In *Conference on Robot Learning (CoRL)*. PMLR, 2020.

[82] Alexander Lambert, Brian Hou, Rosario Scalise, Siddhartha S. Srinivasa, and Byron Boots. Stein variational probabilistic roadmaps. In *International Conference on Robotics and Automation (ICRA)*, pages 11094–11101. IEEE, 2022.

[83] Micah Lapping-Carr, Odest Chadwicke Jenkins, Daniel H Grollman, Jonas Schwertfeger, and Theodora Hinkle. Wiimote interfaces for lifelong robot learning. In *AAAI Spring Symposium: Using AI to Motivate Greater Participation in Computer Science*, pages 61–66, 2008.

[84] Steven M LaValle. *Planning algorithms.* Cambridge university press, 2006.

[85] S. Lenser and M. Veloso. Sensor resetting localization for poorly modelled mobile robots. In *International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1225–1232. IEEE, 2000.

[86] Stanley Lewis, Jana Pavlasek, and Odest Chadwicke Jenkins. NARF22: Neural articulated radiance fields for configuration-aware rendering. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[87] Chi Li, Jonathan Bohren, Eric Carlson, and Gregory D Hager. Hierarchical semantic parsing for object pose estimation in densely cluttered scenes. In *International Conference on Robotics and Automation (ICRA)*, pages 5068–5075. IEEE, 2016.

[88] Xiaolong Li, He Wang, Li Yi, Leonidas Guibas, A Lynn Abbott, and Shuran Song. Category-level articulated object pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020.

[89] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. DeepIM: Deep iterative matching for 6D pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[90] Thibaut Lienart, Yee Whye Teh, and Arnaud Doucet. Expectation particle belief propagation. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3609–3617, 2015.

[91] Qiang Liu, Jason D. Lee, and Michael Jordan. A kernelized Stein discrepancy for goodness-of-fit tests and model evaluation. In *International Conference on Machine Learning (ICML)*. PMLR, 2016.

[92] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[93] Cewu Lu, Hao Su, Yonglu Li, Yongyi Lu, Li Yi, Chi Keung Tang, and Leonidas J Guibas. Beyond holistic object recognition: Enriching image understanding with part states. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6955–6963. IEEE, 2018.

[94] Qingkai Lu, Mark Van der Merwe, Balakumar Sundaralingam, and Tucker Hermans. Multi-fingered grasp planning via inference in deep neural networks. *Robotics & Automation Magazine (Special Issue on Deep Learning and Machine Learning in Robotics)*, 27(2):55–65, 2020.

[95] Carlos E Luis, Marijan Vukosavljev, and Angela P Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):604–611, 2020.

[96] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[97] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems (RSS)*, 2017.

[98] Fahira Afzal Maken, Fabio Ramos, and Lionel Ott. Stein ICP for uncertainty estimation in point cloud matching. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):1063–1070, 2022.

[99] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac Gym: High performance GPU-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[100] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kPAM: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

[101] Pat Marion, Peter R Florence, Lucas Manuelli, and Russ Tedrake. LabelFusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes. In *International Conference on Robotics and Automation (ICRA)*, pages 3325–3242. IEEE, 2018.

[102] Fred G Martin. *Robotic explorations: A hands-on introduction to engineering*. Prentice Hall PTR, 2000.

[103] Roberto Martin Martin and Oliver Brock. Online interactive perception of articulated objects with multi-level recursive estimation based on task-specific priors. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2494–2501. IEEE, 2014.

[104] Takuo Matsubara, Jeremias Knoblauch, François-Xavier Briol, and Chris J Oates. Robust generalised Bayesian inference for intractable likelihoods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(3):997–1022, 2022.

[105] Monica M. McGill. Learning to program with personal robots: Influences on student motivation. *ACM Transactions on Computing Education (TOCE)*, 12(1), mar 2012.

[106] Aditya Menon and Cheng Soon Ong. Linking losses for density ratio and class-probability estimation. In *International Conference on Machine Learning (ICML)*, pages 304–313. PMLR, 2016.

[107] Frank Michel, Alexander Krull, Eric Brachmann, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Pose estimation of kinematic chain instances via object coordinate regression. In *BMVC*, pages 181–1, 2015.

[108] David P Miller and Illah Nourbakhsh. Robotics for education. In *Springer handbook of robotics*, pages 2115–2134. Springer, 2016.

[109] Chaitanya Mitash, Abdeslam Boularias, and Kostas Bekris. Robust 6D object pose estimation with stochastic congruent sets. In *British Machine Vision Conference (BMVC)*, 2018.

[110] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

[111] Thomas Moore and Daniel Stouch. A generalized extended Kalman filter implementation for the robot operating system. In *Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13*, pages 335–348. Springer, 2016.

[112] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, 2014.

[113] R. Moulton and Y. Jiang. Maximally consistent sampling and the Jaccard index of probability distributions. In *International Conference on Data Mining (ICDM)*, pages 347–356. IEEE, 2018.

[114] Arsalan Mousavian, Clemens Eppner, and Dieter Fox. 6-DoF GraspNet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[115] Riku Murai, Joseph Ortiz, Sajad Saeedi, Paul HJ Kelly, and Andrew J Davison. A robot web for distributed many-device localisation. *arXiv preprint arXiv:2202.03314*, 2022.

[116] Adithyavairavan Murali, Arsalan Mousavian, Clemens Eppner, Chris Paxton, and Dieter Fox. 6-DoF grasping for target-driven object manipulation in clutter. In *International Conference on Robotics and Automation (ICRA)*, pages 6232–6238. IEEE, 2020.

[117] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, UAI'99, page 467–475. Morgan Kaufmann Publishers Inc., 1999.

[118] Venkatraman Narayanan and Maxim Likhachev. Deliberative object pose estimation in clutter. In *International Conference on Robotics and Automation (ICRA)*, pages 3125–3130. IEEE, 2017.

[119] Hao Yi Ong and J. Christian Gerdes. Cooperative collision avoidance via proximal message passing. In *2015 American Control Conference (ACC)*, pages 4124–4130, 2015.

[120] Anthony Opipari, Jana Pavlasek, Chao Chen, Shoutian Wang, Karthik Desingh, and Odest C. Jenkins. DNBP: Differentiable nonparametric belief propagation. *ACM/IMS Journal of Data Science*, 1(1), 2024.

[121] Jason Pacheco, Silvia Zuffi, Michael Black, and Erik Sudderth. Preserving modes and messages via diverse particle selection. In *International Conference on Machine Learning (ICML)*, volume 32, pages 1152–1160. PMLR, 2014.

[122] Jason Pacheco, Silvia Zuffi, Michael Black, and Erik Sudderth. Preserving modes and messages via diverse particle selection. In *International Conference on Machine Learning (ICML)*, pages 1152–1160. PMLR, 2014.

[123] Seymour Papert. Mindstorms: Children, computers, and powerful ideas, 1980.

[124] Aalok Patwardhan, Riku Murai, and Andrew J Davison. Distributing collaborative multi-robot planning with Gaussian belief propagation. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):552–559, 2022.

[125] Liam Paull, Jacopo Tani, Heejin Ahn, Javier Alonso-Mora, Luca Carlone, Michal Cap, Yu Fan Chen, Changhyun Choi, Jeff Dusek, Yajun Fang, et al. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *International Conference on Robotics and Automation (ICRA)*, pages 1497–1504. IEEE, 2017.

[126] Jana Pavlasek, Stanley Lewis, Karthik Desingh, and Odest Chadwicke Jenkins. Parts-based articulated object localization in clutter using belief propagation. In *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[127] Jana Pavlasek, Stanley Lewis, Balakumar Sundaralingam, Fabio Ramos, and Tucker Hermans. Ready, set, plan! Planning to goal sets using generalized Bayesian inference. In *Conference on Robot Learning (CoRL)*, pages 3672–3686. PMLR, 2023.

[128] Jana Pavlasek, Joshua Jing Zhi Mah, Ruihan Xu, Odest Chadwicke Jenkins, and Fabio Ramos. Stein variational belief propagation for multi-robot coordination. *IEEE Robotics and Automation Letters (RA-L)*, 9(5):4194–4201, 2024.

[129] Thomas Power and Dmitry Berenson. Variational inference MPC using normalizing flows and out-of-distribution projection. In *Robotics: Science and Systems (RSS)*, 2022.

[130] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[131] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *International Conference on Robotics and Automation (ICRA)*, pages 489–494. IEEE, 2009.

[132] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In *Robotics: Science and Systems (RSS)*, 2012.

[133] Federico Rossi, Saptarshi Bandyopadhyay, Michael T Wolf, and Marco Pavone. Multi-agent algorithms for collective behavior: A structural and application-focused atlas. *arXiv preprint arXiv:2103.11067*, 2021.

[134] Stergios I Roumeliotis and George A Bekey. Bayesian estimation and Kalman filtering: A unified framework for mobile robot localization. In *International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2985–2992. IEEE, 2000.

[135] Dorsa Sadigh, Anca D Dragan, Shankar Sastry, and Sanjit A Seshia. Active preference-based learning of reward functions. In *Robotics: Science and Systems (RSS)*, 2017.

[136] Tanner Schmidt, Katharina Hertkorn, Richard Newcombe, Zoltan Marton, Michael Suppa, and Dieter Fox. Depth-based tracking with physical constraints for robot manipulation. In *International Conference on Robotics and Automation (ICRA)*, pages 119–126. IEEE, 2015.

[137] Tanner Schmidt, Richard A. Newcombe, and Dieter Fox. DART: Dense articulated real-time tracking. In *Robotics: Science and Systems (RSS)*, 2014.

[138] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research (IJRR)*, 33(9):1251–1270, 2014.

[139] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems (RSS)*, volume 9, pages 1–10, 2013.

[140] Jonas Nathan Schwertfeger and Odest Chadwicke Jenkins. Multi-robot belief propagation for distributed robot allocation. In *International Conference on Development and Learning*, pages 193–198. IEEE, 2007.

[141] R Benjamin Shapiro, Kayla DesPortes, and Betsy DiSalvo. Improving computing education research through valuing design. *Communications of the ACM*, 66(8):24–26, 2023.

[142] Guanya Shi, Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, Fabio Ramos, Animashree Anandkumar, and Yuke Zhu. Fast uncertainty quantification for deep object pose estimation. In *International Conference on Robotics and Automation (ICRA)*, pages 5200–5207. IEEE, 2021.

[143] Mark F. Shilling. Multivariate two-sample tests based on nearest neighbours. *Journal of the American Statistical Association*, 81(395):799–806, 1986.

[144] Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J Black, and Michael Isard. Tracking loose-limbed people. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 421–428. IEEE, 2004.

[145] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[146] Cynthia Solomon, Brian Harvey, Ken Kahn, Henry Lieberman, Mark L Miller, Margaret Minsky, Artemis Papert, and Brian Silverman. History of LOGO. *Proceedings of the ACM on Programming Languages*, 4(HOPL):1–66, 2020.

[147] Isabelle ML Souza, Wilkerson L Andrade, Lívia MR Sampaio, and Ana Liz Souto O Araujo. A systematic review on the use of LEGO® robotics in education. In *Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2018.

[148] Mark W Spong and Seth Hutchinson. *Robot modeling and control.* John Wiley & Sons, 2020.

[149] Siddhartha S. Srinivasa, Patrick Lancaster, Johan Michalove, Matt Schmittle, Colin Summers, Matthew Rockett, Joshua R. Smith, Sanjiban Chouhury, Christoforos Mavrogiannis, and Fereshteh Sadeghi. MuSHR: A low-cost, open-source robotic racecar for education and research. *CoRR*, abs/1908.08031, 2019.

[150] J. Sturm. *Approaches to Probabilistic Model Learning for Mobile Manipulation Robots.* Springer Tracts in Advanced Robotics (STAR). Springer, 2013.

[151] Jürgen Sturm, Cyrill Stachniss, and Wolfram Burgard. A probabilistic framework for learning kinematic models of articulated objects. *Journal of Artificial Intelligence Research*, 41:477–526, 2011.

[152] Erik B Sudderth, Alexander T Ihler, Michael Isard, William T Freeman, and Alan S Willsky. Nonparametric belief propagation. *Communications of the ACM*, 53(10):95–103, 2010.

[153] Erik B Sudderth, Michael I Mandel, William T Freeman, and Alan S Willsky. Visual hand tracking using nonparametric belief propagation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, pages 189–189, 2004.

[154] M Sugiyama, T Suzuki, and T Kanamori. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

[155] Zhiqiang Sui, Odest Chadwicke Jenkins, and Karthik Desingh. Axiomatic particle filtering for goal-directed robotic manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 4429–4436. IEEE, 2015.

[156] Zhiqiang Sui, Zheming Zhou, Zhen Zeng, and Odest Chadwicke Jenkins. SUM: Sequential scene understanding and manipulation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 3281–3288. IEEE, 2017.

[157] Jay Summet, Deepak Kumar, Keith O'Hara, Daniel Walker, Lijun Ni, Doug Blank, and Tucker Balch. Personalizing CS1 with robots. *ACM SIGCSE Bulletin*, 41(1):433–437, 2009.

[158] Martin Sundermeyer, Arsalan Mousavian, Rudolph Triebel, and Dieter Fox. Contact-GraspNet: Efficient 6-DoF grasp generation in cluttered scenes. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[159] Gábor J Székely and Maria L Rizzo. Energy statistics: A class of statistics based on distances. *Journal of statistical planning and inference*, 143(8):1249–1272, 2013.

[160] Russ Tedrake. *Underactuated Robotics*. 2023.

[161] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[162] Ekaterina Tolstaya, Landon Butler, Daniel Mox, James Paulos, Vijay Kumar, and Alejandro Ribeiro. Learning connectivity for data distribution in robot teams. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 413–420. IEEE, 2021.

[163] Ekaterina Tolstaya, Fernando Gama, James Paulos, George Pappas, Vijay Kumar, and Alejandro Ribeiro. Learning decentralized controllers for robot swarms with graph neural networks. In *Conference on Robot Learning (CoRL)*, pages 671–682. PMLR, 2020.

[164] David S Touretzky. Seven big ideas in robotics, and how to teach them. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, pages 39–44, 2012.

[165] David S Touretzky and Ethan J Tira-Thompson. Tekkotsu: A framework for AIBO cognitive robotics. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 1741. Citeseer, 2005.

[166] Marc Toussaint. Robot trajectory optimization using approximate inference. In *International Conference on Machine Learning (ICML)*, pages 1049–1056. PMLR, 2009.

[167] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. In *Conference on Robot Learning (CoRL)*. PMLR, 2018.

[168] Ben Tribelhorn and Zachary Dodds. Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education. In *International Conference on Robotics and Automation (ICRA)*, pages 1393–1399. IEEE, 2007.

[169] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium (ISRR)*, pages 3–19. Springer, 2011.

[170] Jur Van Den Berg, Stephen J Guy, Jamie Snape, Ming Lin, and Dinesh Manocha. RVO2 library: Reciprocal collision avoidance for real-time multi-agent simulation. https://gamma.cs.unc.edu/ORCA/, 2016.

[171] Ruben Van Parys and Goele Pipeleers. Distributed model predictive formation control with inter-vehicle collision avoidance. In *Asian Control Conference (ASCC)*, pages 2399–2404, 2017.

[172] Manuela M Veloso, Paul E Rybski, Scott Lenser, Sonia Chernova, and Douglas Vail. CMRoboBits: Creating an intelligent AIBO robot. *AI magazine*, 27(1):67–67, 2006.

[173] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[174] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[175] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. DenseFusion: 6D object pose estimation by iterative dense fusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3343–3352. IEEE, 2019.

[176] Dilin Wang, Zhe Zeng, and Qiang Liu. Stein variational message passing for continuous graphical models. In *International Conference on Machine Learning (ICML)*, pages 5219–5227. PMLR, 2018.

[177] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6D object pose and size estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2651. IEEE, 2019.

[178] Lirui Wang, Yu Xiang, and Dieter Fox. Manipulation trajectory optimization with online grasp synthesis and selection. In *Robotics: Science and Systems (RSS)*, 2020.

[179] Yair Weiss and William Freeman. Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Advances in Neural Information Processing Systems (NeurIPS)*, 12, 1999.

[180] Eric Welton, Seth Hutchinson, and Mark Spong. A modular, interdisciplinary approach to undergraduate robotics education. In *Frontiers in Education Conference (FIE)*, pages 714–719. IEEE, 1993.

[181] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[182] Jay M Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David MS Johnson, et al. SegICP: Integrated deep semantic segmentation and pose estimation. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 5784–5789. IEEE, 2017.

[183] Yu Xiang and Silvio Savarese. Estimating the aspect layout of object categories. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3410–3417. IEEE, 2012.

[184] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In *Robotics: Science and Systems (RSS)*, 2018.

[185] Wei Yang, Balakumar Sundaralingam, Chris Paxton, Iretiayo Akinola, Yu-Wei Chao, Maya Cakmak, and Dieter Fox. Model predictive control for fluid human-to-robot handovers. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022.

[186] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *SIGGRAPH Asia*, 2018.

[187] Fisher Yu, Vladlen Koltun, and Thomas Funkhouser. Dilated residual networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[188] Zhen Zeng, Yunwen Zhou, Odest Chadwicke Jenkins, and Karthik Desingh. Semantic mapping with simultaneous object detection and localization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 911–918. IEEE, 2018.

[189] Zhen Zeng, Zheming Zhou, Zhiqiang Sui, and Odest Chadwicke Jenkins. Semantic Robot Programming for goal-directed manipulation in cluttered scenes. In *International Conference on Robotics and Automation (ICRA)*, pages 7462–7469. IEEE, 2018.

[190] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing Stein variational gradient descent. In *International Conference on Machine Learning (ICML)*, pages 6018–6027. PMLR, 2018.