Towards Better Navigation: Optimizing Algorithms for Mapping, Localization and Planning

by

Vikas Dhiman

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in the University of Michigan
2019

Vikas Dhiman

dhiman@umich.edu

ORCID iD: 0000-0003-0078-3677

ACKNOWLEDGMENTS

I am grateful to my advisor, Prof Jason Corso, for his valueable guidance. I thank Prof Jeffrey Siskind, Prof Matthew Roberson-Johnson and Prof Satinder Singh for being on my thesis committee.
I thank my lab members and collaborators without whom this dissertation would not have been possible, namely Dr Julian Ryde, Dr Jeffrey Delmerico, Dr Manmohan Chandraker, Dr Chenliang Xu, Dr Suren Kumar, Dr Brent Griffin, Shurjo Banerjee, Madan Ravi Ganesh.
I am grateful to my parents, sister and friends for their support, love and guidance.
I thank Jordan Turkomani for proofreading parts of this dissertation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Navigation is the problem of going from one place to another. It is an important problem in the areas of autonomous driving, mobile robotics and robotic manipulation. When navigation algorithms realize their potential, autonomous driving will lead to a new era of mobility, enabling greater independence for the disabled and greater convenience for all. It will reduce car ownership, increase fuel efficiency and reduce traffic jams.

The navigation problem needs to be solved efficiently. It is well understood that slow reaction time in driving can be fatal. For self-driving cars to be safe, the navigation algorithms have to be optimized without compromising accuracy. In this thesis, we focus on optimizing algorithms in the navigation domain.

Navigation is often addressed in three parts: mapping, localization and planning. Mapping is the problem of building a representation (a map) of the environment from noisy observations. Mapping is the slowest step in the navigation pipeline because the set of possible maps grows exponentially with the size of the map. This makes optimizing mapping crucial for optimizing navigation. We focus on grid based mapping algorithms that divide the space into grid cells. The state of the art (SOTA) grid based mapping algorithms have either of two limitations. They either make a limiting assumption of each grid-cell being independent of its neighboring cells or they use slow sampling based methods like Gibbs sampling and Metropolis Hastings. Although the sampling based methods are guaranteed to converge, they are slow in practice because they do not fully utilize the relationships among random variables. We avoid the independent cell assumption and use modern inference methods like Belief Propagation and Dual Decomposition, instead of sampling based methods. These modern methods not only converge up to two times faster but also lead to more accurate maps.

Localization, another part of navigation, is the problem of finding the robot's position with respect to the environment or the map. It is usually carried out under two restrictive assumptions: (1) there is only one robot in the environment (2) the map (or its estimate) is known. We relax the former assumption by recognizing the fact that robots can cooperatively map the environment faster. We propose a polynomial root-finding based mutual localization algorithm that uses observations from two robots. Our algorithm depends upon only a few fiducial markers instead of external landmarks, used by methods like Bundler, which makes it faster.

The final step of navigation, called planning, is the problem of taking actions based on the map, the robot's position and desired destination. Reinforcement learning (RL) is a popular algorithm for planning, when the effect of actions on the environment are not easily modeled. A special class of RL algorithms, called Goal-conditioned RL, is applied to cases when the goal location can change for every trial. The SOTA algorithms in Goal-conditioned RL specify the goal location in multiple, redundant ways. Due to this redundant information, algorithms like Hindsight Experience Replay re-samples rewards which makes them slow. We propose a deep extension to Floyd-Warshall Reinforcement Learning which removes of this redundant information thus avoiding rewards re-sampling. The resultant algorithm requires only half the reward samples as required by the baselines.

CHAPTER 1

Introduction

Self-driving technology has attracted USD 80 billion in investment from 2014 to 2017, with over USD 2 billion going to start-ups [56]. With current growth rates, the gross revenue of the autonomous car industry is predicted to exceed USD 10 trillion globally by the early 2030s [55]. An IHS report predicts that 21 million cars will be on the road globally by 2035 which is about 10% of the predicted global car sales [63]. Apart from financial incentives, the adoption of self-driving cars is motivated by safety. According to ARK research, at anticipated levels of self-driving car adoption, “nearly five million fatal accidents are likely to be avoided by 2035” [55]. Furthermore, this technology can enable greater independence for the disabled, reduce car ownership, increase in fuel efficiency and reduction in traffic jams [63, 5]. All these potential benefits to society highlight the importance of improving self-driving technology.

Deciding when to steer, brake or accelerate the car to take it from one place to another is called navigation. Though the self-driving cars are currently getting attention, the navigation problem has been well-studied for decades in the fields of mobile robotics and robotic manipulation. Many goal-oriented tasks can be modeled as navigation problems. Hence, navigation is one of the core problems in the field of artificial intelligence. For example, to make a sandwich, the robot has to locate the ingredients and plan an efficient strategy to move them from their starting location into the desired configuration.

The navigation problem needs to be solved efficiently. It is well understood that slow reaction time in driving is highly correlated to driving errors [1, 82] and is thus a matter of life and death. For the self-driving cars to be safe, the navigation algorithms have to be optimized without compromising accuracy. In the thesis, we focus on the optimization of different algorithms in the navigation domain.

Navigation is often addressed in three parts: mapping, localization and planning. To understand these parts, let us take an example of a rent-able home cleaning robot. In order to fetch cleaning supplies, the robot has to not only find them but also remember their

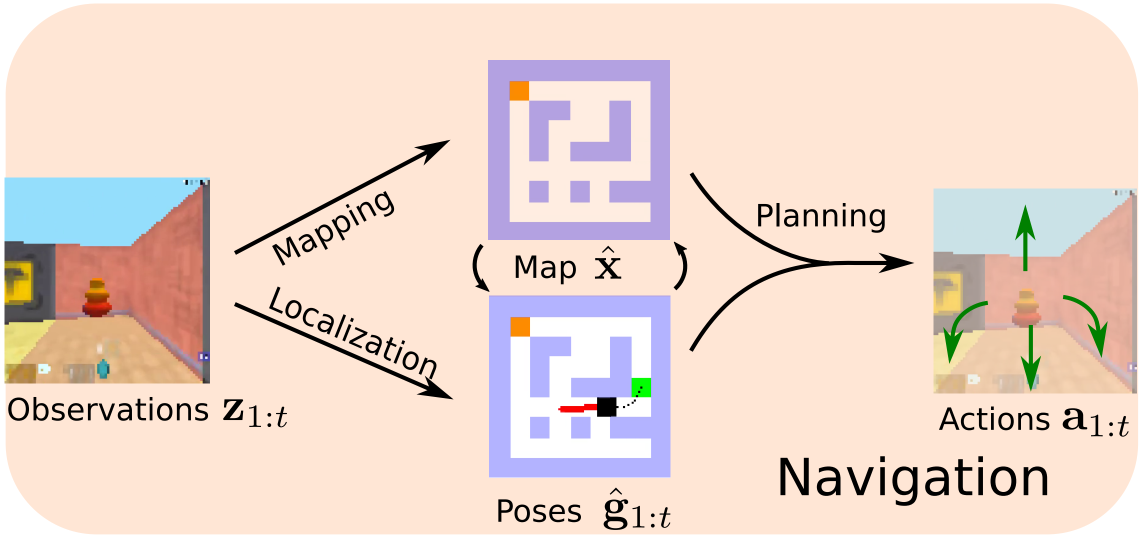

Mapping
Localization
Map $\hat{\mathbf{x}}$
Planning
Observations $\mathbf{z}_{1:t}$
Poses $\hat{\mathbf{g}}_{1:t}$
Actions $\mathbf{a}_{1:t}$
Navigation

Figure 1.1: Navigation is the problem of taking action that move the agent from one place to another. It is often addressed in three parts: mapping (building consistent representation of environment from noisy observations), localization (estimating the state of the agent in the map) and planning (using the location and map to decide sequence of actions in order to accomplish the task)

location. This information that relates the locations of obstacles, objects and places with the corresponding robot observations is called a *map* and the process is called *mapping*. In this example, the relative location of the kitchen, the cleaning supplies and other rooms along with their respective images form the map. This map can then be used to estimate the robot's location in the map, by comparing current robot observations with those in the map, in a process called *localization*. Often mapping and localization are done simultaneously because they depend upon each other. Increasing the accuracy or speed of one makes the other more accurate.

The map and the robot's location are used to plan a sequence of actions to be taken to reach the desired goal location. This step is called *planning*. Planning needs to be fast enough that the actions on the environment are taken before the environment changes. For example, a cleaning robot should clean bits of papers on the ground before they move.

1.1 Formal definition

Let us define the navigation problem more concretely. Let $\mathbf{x} \in \mathcal{X}$ denote a map of the static environment around a robot, where $\mathcal{X}$ is the set of possible maps selected for the problem. When the robot observes the environment $\mathbf{x}$ from a pose (position and orientation) $g_t \in \mathcal{G}$ (usually SE(3)) at time t, its sensor reads the noisy observation $z_t \in Z$ while taking an action $a_t \in \mathcal{A}$. We assume the observation probability $P_o(z_t|g_t, a_t, \mathbf{x})$ is a stationary distribution, i.e. it does not change with time. This commonly used assumption [139] is reasonable provided that the $\mathbf{x}$ and g_t contain enough information to generate the observation z_t. We also assume that the motion of the robot is constrained by a stationary transition probability $P_T(g_{t+1}|g_t, a_t, \mathbf{x})$. Transition probability is stationary when the factors affecting the robot motion are completely captured by the action a_t and the map $\mathbf{x}$. The stationarity assumptions do not hold when the static $\mathbf{x}$ is unable to capture the parts of the environment that affect robot observations and motion. This is especially true when large parts of the environment are dynamic. In this thesis, we focus on static environments. The problem of estimating the map $\mathbf{x}_t^*$ as function of past observations $\mathbf{z}_{1:t} = [z_1, z_2, \dots, z_t]$, and poses $\mathbf{g}_{1:t} = [g_1, g_2, \dots, g_t]$ is called mapping:

$$\begin{aligned} \mathbf{x}_t^* &= \arg\max_{\mathbf{x}\in\mathcal{X}} P(\mathbf{x}|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}, \mathbf{a}_{1:t}) \\ \text{s.t.} \qquad z_t &\sim P_o(z_t|g_t, a_t, \mathbf{x}) \\ g_{t+1} &\sim P_T(g_{t+1}|g_t, a_t, \mathbf{x}). \end{aligned} \tag{1.1}$$

The possible set of maps $\mathcal{X}$ is usually large. Consider occupancy grid mapping (explained in Chapter 2), in which the space is divided into grid cells and each cell models the probability of the cell being occupied, as opposed to being free. The space of occupancy grid maps $\mathcal{X} = \{0,1\}^{w\times h}$ grows exponentially with the number of grid-cells $w \times h$ in the map. Hence, the efficiency of the mapping algorithms is critical to the feasibility of solving the problem. In Chapter 2, we use factor graphs [64] and modern inference algorithms [105, 60] to speed up mapping algorithms.

Recall that localization is the problem of estimating the pose of the robot g_t:

$$\begin{aligned} g_t^* &= \arg\max_{g_t\in\mathcal{G}} P(g_t|\mathbf{z}_{1:t}, \mathbf{a}_{1:t}, \mathbf{x}) \\ \text{s.t.} \qquad z_t &\sim P_o(z_t|g_t, a_t, \mathbf{x}) \\ g_{t+1} &\sim P_T(g_{t+1}|g_t, a_t, \mathbf{x}). \end{aligned} \tag{1.2}$$

Furthermore, the mapping can be made faster by employing multiple robots working cooperatively. To this end we address the problem of localization of two robots with respect to each other, called mutual localization. We present a fast mutual-localization algorithm in Chapter 3. The solution is based on a polynomial root-finding that works even with only three fiducial markers distributed on the two robots. This approach is an order of magnitude faster then external-landmark based approaches like Bundler [128].

Recall that *navigation* includes exploring the environment to build a map and using the map to take actions $a_t \in \mathcal{A}$ that accomplish given tasks. The task is defined in terms of a *reward* function $R(s_t, a_t) \in \mathbb{R}$ that captures the desirability of a state-action pair, where $s_t \in \mathcal{S}$ is the Markovian state such that $P(s_{t+1}|a_t, s_t, \dots, s_1) = P(s_{t+1}|a_t, s_t)$. The state, in the context of mapping, is modeled as a combination of the map $\mathbf{x}$ and the robot pose $s_t = \left[\begin{smallmatrix}\mathbf{x}\\ g_t\end{smallmatrix}\right]$. To find the right sequence of actions, we estimate a *policy* $\chi(a_t|s_t)$ that gives the probability of choosing an action for every state. With this terminology, we can formalize the navigation problem as finding the policy that maximizes the expected cumulative reward:

$$\begin{aligned} \chi^* &= \arg\max_{\chi} \mathbb{E}\left[\sum_{k=t}^{\infty} \gamma^{k-t} R(s_k, a_k) \middle| \mathbf{z}_{1:t}, \mathbf{g}_{1:t}\right] \\ \text{s.t.} \qquad z_k &\sim P_o(z_k|g_k, a_k, \mathbf{x}) \\ g_{k+1} &\sim P_T(g_{k+1}|g_k, a_k, \mathbf{x}) \\ s_k &= \left[\begin{smallmatrix}\mathbf{x}\\ g_k\end{smallmatrix}\right] \\ a_k &\sim \chi(a_k|s_k) \quad . \end{aligned} \tag{1.3}$$

This formulation shows that reward depends upon state $s_t = \left[\begin{smallmatrix}\mathbf{x}\\ g_t\end{smallmatrix}\right]$, which is never fully observed. To estimate this unobserved state, the navigation algorithm usually depends on a mapping algorithm that maintains a running estimate of the map. However, the map only needs to be estimated up to a precision proportional to the effect on the rewards. For example, consider the cleaning robot carrying a bucket of water. Although estimating the state of the environment precisely requires modeling computationally expensive fluid dynamics, a reward driven approach can model the state of water to a precision just enough to avoid causing a spillage thus making the algorithm faster. End-to-end navigation methods like Deep Reinforcement Learning [89, 43] (DRL) are a promising step in this direction. In Chapter 5, we investigate one such algorithm [89] to find the strengths and weaknesses of the state-of-the-art in these approaches.

In Equation (1.3), the reward function fully defines the task which is assumed to be static. However, the desired goal locations are often dynamic. To allow dynamic goals the state is expanded to include desired goal location d_t, $s_t = [\mathbf{x}^\top, g_t^\top, d_t^\top]^\top$. In Chapter 6, we show that this extension leads to redundant information and unnecessary computations which can be eliminated by formulating the problem as a type of Floyd-Warshall Reinforcement Learning [51].

1.2 Contributions

This dissertation makes four contributions under the broad problems of mapping, localization and navigation.

1.2.1 Forward sensor models for accurate mapping

Recall that observation probability $P_o(z_t|g_t, \mathbf{x})$ models the sensor observation z_t given the map $\mathbf{x}$ and robot pose g_t. Since observation probability is equivalent to simulating a sensor, it is also called a *forward sensor model*. This is in contrast with *inverse sensor models* $P_{\text{inv}}(\mathbf{x}[z_t]|g_t, z_t)$ that estimate a part of the map $\mathbf{x}[z_t]$ given the sensor observations. We found that most of the state of the art mapping algorithms used inverse sensor models despite them being less accurate and requiring a restrictive assumption. We proposed an algorithm that instead uses forward sensor models and employs modern inference algorithms such as Belief Propagation [105] and Dual decomposition [60] (discussed in Chapter 2). Our algorithms are faster than the sole existing precedent [83] that used forward sensor models because [83] uses slower sampling based inference methods like Metropolis Hastings [79].

1.2.2 Mutual observations for accurate localization

The robots can map more efficiently if they work in teams. Two or more robots can divide and conquer the task of mapping an environment. This requires them to know each other's pose (position and orientation) with high accuracy to reliably merge their maps. We call this problem *mutual localization*. Mathematically, if two robots have pose g_t and h_t and the corresponding observations are z_t and y_t, the problem of jointly estimating $\hat{g}_t, \hat{h}_t = f_{\text{mutloc}}(z_t, y_t)$ is called mutual localization where f_{mutloc} is the estimator to be designed. We proposed a solution to this problem that works with a minimal number of fiducial markers visible to both the robots in our IROS 2013 paper [26]. We solve this problem by formulating it as a generalized version of the Perspective-3-Points (P3P) problem where the perspective image of points are observed in two different cameras. This work is discussed in Chapter 3.

1.2.3 Continuous occlusion models for accurate localization

In unstructured environments it is more common for objects to obstruct views of other objects. This is called occlusion and it makes the problem of mapping the environment harder. Recognizing this as a problem in the pressing domain of autonomous driving, we proposed a novel method of handling occlusions in a road scene. The occlusion model is evaluated on the localization task. This work was published in CVPR 2016 [27]. The core contribution of our work was a continuous occlusion model that represented cars as translucent ellipsoid where transparency was used as a proxy for occlusion probability.

1.2.4 Goal-rewards free reinforcement learning for navigation

More recently, with the advances in Deep Reinforcement Learning (DRL) algorithms it has become possible to approach navigation as a single optimization problem that includes mapping. The problem of navigation has been traditionally handled by breaking it down into mapping and path-planning. However, more recently, work by Mirowski et al. 2017 [89], shows that their proposed DRL algorithm learns to navigate from just pixel inputs. To understand and test their algorithm, we performed an extended set of experiments on their algorithm. We were able to replicate the result of Mirowski et al. 2017 [89] and show that DRL algorithms do learn to find the shortest path to the goals when trained and tested on the same map with the same goal location. However, when trained and tested on different sets of maps, the DRL algorithms fail to perform better than random exploration with wall-following. We also found that when the goal location is changed during training and testing, the algorithm finds sub-optimal paths to reach the previously visited goal locations. We discuss these experiments in Chapter 5. From these experiments, we conclude that the algorithm learns the training map but does not learns *to* navigate on new maps or new goal locations.

In order to improve the algorithm for the case when goal location is changed every episode, we investigate Goal-Conditioned Reinforcement Learning (GCRL). We found that the state-of-the-art GCRL algorithms model these problems such that the reward formulation depends on the goals. We argue that this dependence of reward on goals is duplicate information because the goal is specified at the start of every episode. Furthermore, this dependence introduces additional reward resampling steps in algorithms like Hindsight Experience Replay (HER) that reuse trials in which the agent fails to reach the goal by recomputing rewards as if reached states were psuedo-desired goals. We propose a reformulation of goal-conditioned value functions for GCRL that yields a similar algorithm, while removing the dependence of reward functions on the goal. Our formulation thus obviates the requirement of reward-recomputation that is needed by HER and its extensions. We also extend a closely related algorithm, Floyd-Warshall Reinforcement Learning, from tabular domains to deep neural networks for use as a baseline. Our results are competitive with

HER while substantially improving reward sample efficiency. We discuss this algorithm in Chapter 6. We discuss related work corresponding to each chapter as a section of the chapter.

1.3 Thesis statement

Mapping, localization and planning are optimized by the following observations:

1. Modern inference algorithms, like Belief Propagation and Dual Decomposition, lead to faster mapping than sampling based methods.
2. Mutual observations using fiducial markers lead to a faster polynomial roots based solution than iterative bundle adjustment.
3. Continuous occlusion models lead to more accurate localization than treating occlusion as noise.
4. Not using goal-rewards in goal conditioned navigation leads to faster learning.

1.4 Relevant Publications

This thesis includes work from following publications

- V. Dhiman, A. Kundu, F. Dellaert, and J. J. Corso. Modern MAP inference methods for accurate and faster occupancy grid mapping on higher order factor graphs. In *Proceedings of International Conference on Robotics and Automation*, 2014
- V. Dhiman, J. Ryde, and J. J. Corso. Mutual localization: Two camera relative 6-dof pose estimation from reciprocal fiducial observation. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2013
- V. Dhiman, Q. Tran, J. Corso, and M. Chandraker. A continuous occlusion model for road scene understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016
- V. Dhiman, S. Banerjee, B. Griffin, J. Mark Siskind, and J. J. Corso. A critical investigation of deep reinforcement learning for navigation. *CoRR*, abs/1802.02274, 2018
- V. Dhiman, S. Banerjee, J. M. Siskind, and J. J. Corso. Learning goal-conditioned value functions with one-step path rewards rather than goal-rewards. In *Open Review*, 2019

CHAPTER 2

Occupancy grid mapping[1]

2.1 Introduction

Problems in mobile robotics like navigation, path planning, localization and collision avoidance require an estimate of the robot's spatial environment; this underlying problem is called robotic mapping [137]. Even in environments where maps are available, the problem remains challenging as the environment may change over time, necessitating a mapping ability on the mobile robot. Furthermore, robotic mapping is an important problem in application areas like indoor autonomous navigation, grasping, reconstruction and augmented reality. Hence it remains an active field of research [85, 97, 84].

Although robot mapping can be performed in many ways—metric or topological; with range sensors, like sonar [138], laser scanners [138] and RGBD [99], or bearing-only sensors [20, 67]—metric mapping with range sensors is the most common. Bearing-only sensors provide estimates up to scale; topological maps still require local metric estimates for certain problems like navigation. We hence focus on metric mapping with range sensors, specifically, laser scanners.

Occupancy grid mapping (OGM) is a popular and useful range-based mapping method [29, 96]. It affords a simple implementation and avoids a need to explicitly seek and match landmarks in the environment [133, 9]. In contrast, it discretizes the environment into cells, squares (2D) or cubes (3D), and associates a random variable with each cell that represents the probability of the cell being occupied or free. Also, unlike surface-based approaches [121, 122], OGM makes it easier to query obstacles and therefore to predict and avoid collisions, which is critical for applications like robot navigation.

OGM methods vary in how cell occupancy is estimated, but most methods make use of an inverse sensor model that assumes the occupancy of each cell can be estimated in-

[1]This material originally appeared as: V. Dhiman, A. Kundu, F. Dellaert, and J. J. Corso. Modern MAP inference methods for accurate and faster occupancy grid mapping on higher order factor graphs. In *Proceedings of International Conference on Robotics and Automation*, 2014

dependently of the other cells in the map [29, 96, 95, 99]. The main reason for using this independence assumption is computational efficiency. However, the assumption is inaccurate and can lead to overconfident estimates of occupancy in noisy observations [138, 84].

To overcome this limitation, Thrun [138] proposes the use of a forward sensor model and expectation maximization to estimate occupancy. Following this line of work, more recently, Merali et al. [84] defines a Gibbs sampling algorithm based on a conditional estimate of cell occupancy given the rest of the map. However, it is widely known that Gibbs sampling algorithm can get caught in local maxima [77]. Additionally, these methods remain computationally expensive which limits their application.

In this chapter, we explore the use of modern inference algorithms for more efficient occupancy grid mapping with forward sensor models. Our contribution in this chapter is twofold. Firstly, we introduce the factor graph approach to occupancy grid mapping, which, to the best of our knowledge, has not been applied to this problem. This factor graph formalism makes it plausible to apply modern fast inference algorithms, such as loopy belief propagation [64] and dual decomposition [131].

Secondly, we introduce a class of higher order factors for our factor graph approach. Factor graph inference is exponential in neighborhood size, which requires us to focus on a certain sub-class of factors for tractability, such as the linear constraint-nodes [111] or pattern-based factors [61]. We extend the pattern-based factors, which explicitly compute the potential only for certain factors matching a given set of patterns and otherwise assign a constant. Whereas the pattern-based factors in [61] define each pattern with a fixed value for each node, we generalize these pattern-based factors by allowing for *free* nodes whose value does not impact the computed marginal.

We implement these algorithms for effective occupancy grid mapping with a forward sensor model and test our work on both simulated and real data. Our experiments demonstrate the effectiveness of our novel OGM approach, especially dual decomposition.

2.2 Background and Related work

A main contribution of this chapter is the application of modern MAP algorithms to occupancy grid mapping. Although there are many MAP inference algorithms [53] that work well for various problems, in this chapter, we focus on belief propagation [64] and dual decomposition [131] mainly because of their ability to handle higher order factors.

Belief propagation (BP) [105] was introduced as an algorithm to compute marginals over trees. Surprisingly, it was found to work well on graphs with loops. Later it was found that the convergent solution to belief propagation corresponds to the minima of the so-called Bethe free energy [153], which not only provided a theoretical justification for application of belief propagation to graphs with loops, but also solves the convergence problem by providing an objective function which can be minimized directly. Later, Fractional BP [146] was introduced. Inspired by the Bethe free energy formulation of BP, it suggested using a better free energy approximation by scaling the terms appropriately in the message update equation.

In an independent work, Wainwright et al. [144] introduce the Tree Re-Weighted (TRW) message passing algorithm, which uses re-weighting of edges and messages similar to Fractional BP. They also formulate the MAP estimation problem as a linear program over the so-called marginal polytope. The Langrangian dual of this LP problem is convex and provides the upper bound to the original problem. The family of algorithms that optimize the Lagrangian dual of the original combinatorial problem is called dual decomposition (DD). The dual of the problem can be decomposed in different ways, for example, as a set of spanning trees in TRW [144] or one problem per factor [131]. More recently, accelerated dual decomposition [50] was introduced that provably converges the upper bound faster than earlier approaches by smoothing the Lagrangian dual of the problem. This is the algorithm that we use for our experiments.

Another contribution of this chapter is the way we perform efficient inference with higher order factors (or potentials). Many researchers approach this problem by considering a class of functions for which higher order factors can be used efficiently, for example, Potetz et al. introduce a class of potentials called linear constraint nodes [111] and Komodakis et al. approach pattern-based class of potentials [61]. Another approach to handle higher order potentials [72] is to adaptively restrict the sample space of nodes by using initial estimates. Our work proposes a generalization of the pattern-based potentials of [61] allowing for *free* nodes to appear within a pattern.

2.3 Problem definition

Consider a robot—equipped with a laser scanner—moving in a static environment, and assume that the position of the robot associated with each laser measurement is given. Our task is to estimate occupied regions and free regions, so that the robot can avoid collisions and plan its movement in free regions. We divide the area to be mapped into N discrete cells. Let x_i denote the state of cell i, which can take values from label set $L_i = \{0, 1\}$, where 0 (resp. 1) denotes that the cell is free (resp. occupied). For convenience, we denote the full map (the state for all N cells) as $\mathbf{x} = [x_i]^\top_{1 \leq i \leq N}$ taking values from sample space $\Omega = \prod_{1 \leq i \leq N} L_i$.

Let z_f denote the f^{th} laser range measurement when captured from (known) pose g_f. The problem is to find the probability of all cells of in the map being occupied given all t observations, $\mathbf{z}_{1:t} = [z_f]^\top_{1\le f\le t}$ and $\mathbf{g}_{1:t} = [g_f]^\top_{1\le f\le t}$:

$$p(x_i = 1|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = \sum_{\mathbf{x}\in\Omega : x_i=1} p(\mathbf{x}|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}) \qquad \forall 1 \le i \le N \ . \tag{2.1}$$

Alternatively, we can focus on the maximum posterior map:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\Omega} p(\mathbf{x}|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}). \tag{2.2}$$

Problems (2.1) and (2.2) are related but yield different results. While (2.1) is useful to keep track of uncertainty in an incremental fashion, (2.2) provides a more meaningful result in the joint occupancy configuration by maximizing posterior probability. We use both the equations (2.1) and (2.2) with their corresponding algorithms. Clearly, a naïve solution to either problem would have complexity that is exponential in the number of cells. We hence focus on an approximate solution to this problem.

2.3.1 Mapping with an inverse sensor model

Commonly used occupancy grid mapping algorithms [29, 96, 99] make the simplifying assumption that each grid cell is independent of all other map cells:

$$p(\mathbf{x}|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = \prod_{1\le i\le N} p(x_i|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}) \ . \tag{2.3}$$

The probability of each cell can be easily computed independent of each other, by a simple Bayes formulation:

$$p(x_i|\mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = \frac{p(z_t, g_t|x_i, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})p(x_i|\mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})}{p(z_t, g_t|\mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})} \ . \tag{2.4}$$

Assuming a static world, $p(z_t, g_t | x_i, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) = p(z_t, g_t | x_i)$, as is commonly done, the above equation (2.4) can be simplified [84] to:

$$p(x_i | \mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = p(z_t, g_t | x_i) \frac{p(x_i | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})}{p(z_t, g_t | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})} \tag{2.5}$$

$$= \frac{p(x_i | z_t, g_t) p(z_t, g_t)}{p(x_i)} \frac{p(x_i | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})}{p(z_t, g_t | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})} \tag{2.6}$$

$$= \frac{1}{Z'} \frac{p(x_i | z_t, g_t)}{p(x_i)} p(x_i | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) \tag{2.7}$$

$$= \frac{1}{Z' p^{t-1}(x_i)} \prod_{1 \le f \le t} p(x_i | z_f, g_f) \ , \tag{2.8}$$

where Z' is a normalizing factor that is independent of x_i, $p(x_i)$ is the prior probability for cell x_i and $p(x_i | z_f, g_f)$ is called the *inverse sensor model*.

2.3.2 Mapping with a forward sensor model

The independent cell assumption is inaccurate and can lead to overconfident estimates of occupancy in noisy observations [138, 84]. In the absence of the independent cell assumption, we can still factorize the posterior probability in terms of a forward sensor model.

In this formulation, we make two assumptions, a) static world assumption, $p(z_t | \mathbf{x}, \mathbf{g}_{1:t}) = p(z_t | \mathbf{x}, g_t)$ and b) pose-map independence, $p(g_t | \mathbf{x}, \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) = p(g_t | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1})$. With these assumptions, the posterior evaluates [84] to

$$p(\mathbf{x} | \mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = \frac{1}{Z} p(z_t | \mathbf{x}, g_t) p(\mathbf{x} | \mathbf{z}_{1:t-1}, \mathbf{g}_{1:t-1}) \tag{2.9}$$

$$= \frac{1}{Z} p(\mathbf{x}) \prod_{1 \le f \le t} p(z_f | \mathbf{x}, g_f) \ , \tag{2.10}$$

where Z is a normalizing constant independent of $\mathbf{x}$ and $p(\mathbf{x})$ is the prior. For the rest of the chapter we assume no prior information about the maps is known, hence the prior probability is a uniform distribution.

The problem of estimating the occupancy map is intractable with the above formulation, as it still depends on the entire map, which has an exponential sample space. However, we can make use of the fact that a laser measurement f depends only on a small portion of the map $\mathbf{x}_f \subseteq \mathbf{x}$, hence simplifying the formulation to:

$$p(\mathbf{x} | \mathbf{z}_{1:t}, \mathbf{g}_{1:t}) = \frac{1}{Z} \prod_{1 \le f \le t} p(z_f | \mathbf{x}_f, g_f) \ . \tag{2.11}$$

The term $p(z_f|\mathbf{x}_f, p_f)$ is called the *forward sensor model*. With this formulation, we are in a position to describe the problem as *factor graph*. The above simplification is necessary to keep the factor graph sparsely connected and hence tractable.

2.3.3 Representation as a factor graph

The occupancy grid mapping problem can be expressed as energy minimization over a factor graph. Let all cells in the map be the variable nodes V and all the laser measurements be factor nodes F. There exists an undirected edge (i, f), if and only if the laser range measurement $p(z_f|\mathbf{x}_f, g_f)$ depends on the cell occupancy x_i. In this chapter, we assume that each laser range measurement depends on only those cells that the laser passes through for given pose g_f. Let E be set of all such edges:

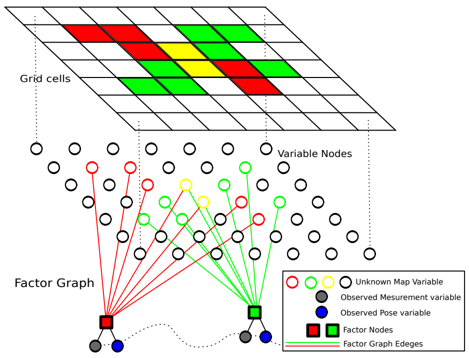


Figure 2.1: Representation of occupancy grid mapping as factor graph

$$E = \{(i, f) : i \in V, f \in F, \text{laser } f \text{ passes through cell } i\} \ . \tag{2.12}$$

The bipartite graph $G = (V, F, E)$ represents the structure of factorization in (2.11) and is hence called a factor graph [64]. Note that neighborhood $n(i)$ of any variable node i only consists of the factors nodes, $n(i) \subseteq F \, \forall i \in V$ and vice versa. Fig. 2.1 shows the factor graph diagrammatically. Note that observed nodes form the part of factor and are not part of the factor graph.

In terms of factor graph G the problems (2.1) and (2.2) along with factorization obtained in (2.11) can be written as:

$$P_i(x_i = l_i) = \sum_{\mathbf{x} \in \Omega : x_i = l_i} P(\mathbf{x}) \qquad \forall i \in V \tag{2.13}$$

$$\mathbf{x}* = \arg\max_{\mathbf{x} \in \Omega} P(\mathbf{x}) \tag{2.14}$$

$$P(\mathbf{x}) = \frac{1}{Z} \prod_{f \in F} P_f(\mathbf{x}_f) \ , \tag{2.15}$$

where $l_i \in L_i$ denotes an element from the label set L_i of node $i \in V$. By representation of occupancy grid mapping as factor graph the following equivalence holds true, $P_i(x_i = l_i) \equiv p(x_i = l_i|\mathbf{z}_{1:t}, \mathbf{g}_{1:t})$, $P(\mathbf{x}) \equiv p(\mathbf{x}|\mathbf{z}_{1:t}, \mathbf{g}_{1:t})$ and $P_f(\mathbf{x}_f) \equiv p(z_f|\mathbf{x}_f, g_f)$.

2.4 Markov Chain Monte Carlo methods (Metropolis Hastings)

We implement and evaluate a generalization of Merali's [84] Gibbs sampling algorithm for estimating maps. Metropolis Hastings is another popular MCMC algorithm used for sampling from complex probability distributions. Gibbs sampling can be shown to be a specialization of Metropolis Hastings [79].

Algorithm 1: Metropolis Hastings

```
Data: Factor Graph $G = (V, F, E)$;
Maximum number of iterations $N$;
Result: $\mathbf{x}^r$
Initialize the map $\mathbf{x}^0$ randomly.;
$r = 0$;
while $r < N$ do
    Randomly choose a cell $i : i \in V$;
    Flip its state $x'_i = \neg x^r_i$ in $\mathbf{x}^r$ to get $\mathbf{x}'$;
    Compute acceptance probability $a$ by (2.18);
    Sample random number $q : 0 \leq q \leq 1$;
    if $a \geq 1$ or $a \geq q$ then
        Accept proposed point, $\mathbf{x}^{r+1} = \mathbf{x}'$;
    else
        Reject proposed point, $\mathbf{x}^{r+1} = \mathbf{x}^r$;
    $r \leftarrow r + 1$;
```

Metropolis Hastings requires a *transition probability* $Q(\mathbf{x}'|\mathbf{x}^r)$, that depends on current sample $\mathbf{x}^r$ and guides the random walk in the high-dimensional space. We randomly sample a point $\mathbf{x}'$ from from $Q(.)$ and it is either accepted or rejected based on the *acceptance probability* a:

$$a = \frac{P(\mathbf{x}')Q(\mathbf{x}^r|\mathbf{x}')}{P(\mathbf{x}^r)Q(\mathbf{x}'|\mathbf{x}^r)} \quad . \tag{2.16}$$

If $a \geq 1$, then the new point $\mathbf{x}'$ is accepted otherwise it is accepted with probability a. Here acceptance means that the point in the next iteration is taken as the sampled point otherwise the earlier point is retained. The interested reader is referred to [79, 77] for further details about Metropolis Hastings.

For our experiments, we choose a symmetric uniform transition probability. We uniformly sample a cell from the map and flip the state of the sampled cell to get the proposal point $\mathbf{x}'$. This is equivalent to sampling from the probability density:

$$Q(\mathbf{x}'|\mathbf{x}^r) = \begin{cases} \frac{1}{N} & \text{if } \|\mathbf{x}' - \mathbf{x}^r\|_1 = 1 \\ 0 & \text{otherwise} \end{cases} , \tag{2.17}$$

where N is the number of cells in the map and $\|.\|_1$ is the L1-norm. The first case in (2.17) enforces that only one cell (or dimension) in the map can change its state. Since this is a symmetric transition probability, the acceptance ratio is just the ratio of target probabilities. Also note that the ratio of probability distributions can be efficiently computed because of the factorization obtained in (2.15):

$$a = \frac{P(\mathbf{x}')}{P(\mathbf{x}^r)} = \frac{\prod_{f\in n(i)} P_f(\mathbf{x}'_f)}{\prod_{f\in n(i)} P_f(\mathbf{x}^r_f)} \ , \tag{2.18}$$

where i is the (sampled) cell whose state is different in $\mathbf{x}'$ and $\mathbf{x}^r$, and $n(.)$ denotes the neighborhood of a vertex in graph G. The above simplification uses the fact that only those terms that depend on the state of i^{th} cell need to be computed. By definition of graph G, only the neighboring factors $f \in n(i)$ depend on the state of cell i.

It is in general difficult to detect when the sampling algorithm has converged. In practice, we often run the sampling algorithm for a fixed number of iterations. Alg. 1 lists the pseudo code for Metropolis Hastings algorithm.

2.4.1 Heat map

As we uniformly sample cells from the map, we notice that not all cells are equally important in mapping. There are three kinds of regions in an occupancy map: occupied, free and unexplored. Sampling and analyzing a cell in an unexplored region is not very helpful as we do not have any evidence for the region. On the other hand, the central regions of free areas are not very interesting as all factors usually agree on their state. The uncertainty tends to lie along the boundaries of free and occupied regions. This is the region we want to focus on.

We hence employ a *heat map* to bias our sampling along the boundaries of free and occupied regions. We maintain a vector of cells $\mathbf{x}_h$ that form the "interesting" region of the map. In our experiments, we take the last cell spanned by each laser measurement as an "interesting" cell and add it to the *heat map*, $\mathbf{x}_h$. We use a sampling bias of $1:4$ for cells outside the heat map to cells within the heat map. We compare both Metropolis Hastings with and without the heat map in our experiments.

2.5 Modern inference algorithms

As discussed in Sec. 2.2, last decade gave rise to faster and more accurate MAP inference algorithms [53]. Because of their ability to handle higher order factors [111, 61], we explore belief propagation and dual decomposition in the problem of occupancy grid mapping.

2.5.1 Belief Propagation

The sum product algorithm over factor graphs [64] is a powerful yet simple algorithm to compute marginals of expression, of the form (2.1), that can be decomposed into factors of the form (2.11). The algorithm provides exact marginals in the case when the graphs have no loops. For graphs with loops the algorithm has been shown to converge in most of practical problems.

The sum product algorithm works by sending messages along the edges of the factor graph. The messages can be understood as the *beliefs* of the source node about destination states. Mathematically, these beliefs are the probabilities of the destination states marginalized over the neighbours of the source except the destination itself. These messages are defined on a directed edge, with a different message value for each state of the variable node involved.

Let $\mu^r_{f\to i}(l_i)$ represent the message from node $i \in V$ to node $f \in F$ for state $x_i = l_i$ at any iteration r of the algorithm. With a similar convention we take $\mu^r_{i\to f}(l_i)$ to denote an update in the opposite direction. We use the following equations to update the messages on an edge depending on whether the direction of the edge is from variable node to factor node or vice versa:

$$\mu^{r+1}_{f\to i}(l_i) = \sum_{\mathbf{x}_f \in \Omega_f : x_i = l_i} P_f(\mathbf{x}_f) \prod_{j \in n(f)\setminus i} \mu^r_{j\to f}(x_j) \tag{2.19}$$

$$\mu^{r+1}_{i\to f}(l_i) = \prod_{h \in n(i)\setminus f} \mu^r_{h\to i}(l_i) \ , \tag{2.20}$$

where $\Omega_f = \prod_{i\in n(f)} L_i$ denotes the sample space of the neighborhood of factor f in graph G. On convergence, the belief of variable nodes can be computed by the product of incoming messages:

$$P(x_i = l_i) = \prod_{f \in n(i)} \mu^r_{f\to i}(l_i) \ . \tag{2.21}$$

This is called the sum product belief propagation (BP) algorithm.

One can compute the maximizing assignment instead of marginals by computing the max product instead of sum product in (2.19) and finally choosing the maximizing assignment of incoming messages:

$$\mu^{r+1}_{f\to i}(l_i) = \max_{\mathbf{x}_f\in\Omega_f : x_i = l_i} P_f(\mathbf{x}_f) \prod_{j\in n(f)\setminus i} \mu^r_{j\to f}(x_j) \tag{2.22}$$

$$x^*_i = \arg\max_{x_i\in L_i} \prod_{f\in n(i)} \mu^r_{f\to i}(l_i). \quad . \tag{2.23}$$

This form of the algorithm is called max product BP.

Belief propagation was initially designed to work on factor graphs without loops. In such a case one can start message updates from the leaf nodes and a node can be "triggered" to pass on the messages when messages from all but one neighbors are available. However, for graphs with loops various update sequences have been suggested that vary from problem to problem. For example, in vision problems, where the factor graph is a 2D grid, horizontal and then vertical sweeps have been shown to produce good results. For our implementation, we choose random update sequence, i.e., a random edge is selected from the graph for each iteration of message update.

2.5.2 Subgradient Dual decomposition

The dual decomposition algorithm employs the theory of Lagrangian duals to find a convex upper bound of the original combinatorial optimization problem. Here we explain the implementation of the algorithm without going into mathematical proofs. The interested reader is referred to [131, 50, 61] for proofs and more variations of the algorithm.

The underlying idea for dual decomposition is to split the maximization problem into *slave* problems that can be efficiently maximized. In a factor graph formulation the natural slave problem is one corresponding to each factor:

$$\mathbf{x}^f = \arg\max_{\mathbf{x}^f} P_f(\mathbf{x}^f) \prod_{i\in n(f)} \exp\left(-\mu_{if}(x^f_i)\right) \quad , \tag{2.24}$$

where $\mathbf{x}^f = \{x^f_i\}_{i\in n(f)}$ is the optimum assignment as determined by the corresponding slave problem. And $\mu_{if}(x^f_i)$ is the message (also the Lagrangian multiplier) from node i to f about state x^f_i. The above slave problem is usually written in the form of negative log likelihood:

$$\mathbf{x}^f = \arg\min_{\mathbf{x}^f} \theta_f(\mathbf{x}^f) + \sum_{i\in n(f)} \mu_{if}(x^f_i) \quad , \tag{2.25}$$

where $\theta_f(\mathbf{x}^f) = -\log P_f(\mathbf{x}^f)$ is the negative log likelihood corresponding to the factor.

In each iteration of the algorithm all slave problems are allowed to choose their optimum assignment independently. If all the factors agree on the assignments, then we have reached the global optimum. Often this is not the case. In case of disagreement, we decrease the belief of all the disagreeing slave problems about their respective optimums by sending appropriate messages. It can be shown that as long as we can increment the messages by decreasing step size in each iteration, the algorithm is guaranteed to converge to an approximate solution of the original problem [131].

Pseudocode for dual decomposition (DD) is provided in Alg. 2. Apart from input factor graph $G = (V, F, E)$ and label set $\{L_i\}_{i \in V}$ introduced in Sec 2.3.3, dual decomposition depends on a step size α. We note that step size is an important attribute and affects the speed of the algorithm. For illustration, we show the convergence of dual decomposition with different step sizes on *cave* dataset in Fig. 2.2. Note that erring on the higher side causes oscillations, while erring on the lower side can cause convergence to be too slow.

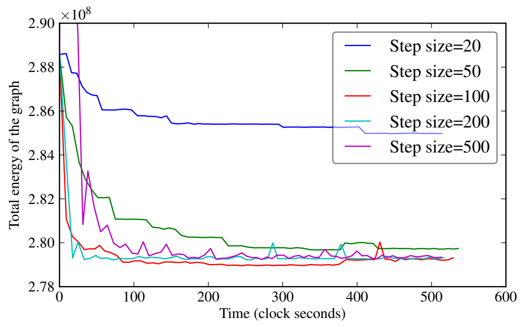


Figure 2.2: The rate of convergence in subgradient dual decomposition depends on step size

Upon convergence or completing a maximum number of iterations, we can compute the optimal assignment for variable nodes with disagreeing slaves from the messages:

$$x_i = \underset{x_i \in L_i}{\arg\max} \sum_{f \in n(i)} \mu_{if}(x_i). \qquad (2.26)$$

Note that the above equation is only valid for disagreeing slaves. When the slaves agree, we can simply pick the agreed upon value.

Dual decomposition is an optimization algorithm; hence it only solves the MAP problem (2.14), but not the marginal problem (2.13), which may be considered a limitation.

Algorithm 2: Subgradient Dual Decomposition

```
Data:
Factor Graph $G = (V, F, E)$
Step size $\alpha > 0$
Maximum number of iterations $N$
Result: Labels $\{x_i^f\}_{(i,f) \in E}$, Messages $\{\mu_{if}(x_i)\}$
$\mu_{if}(x_i) \leftarrow 0 \qquad \forall (i,f) \in E, x_i \in L_i$
$r \leftarrow 1$
while $r < N$ do
    for $f \in F$ do
        $\mathbf{x}^f \leftarrow \underset{\mathbf{x}^f}{\arg\min} \left( \theta_f(\mathbf{x}^f) + \sum_{i \in n(f)} \mu_{if}(x_i^f) \right)$
    // For disagreeing nodes
    for $i \in V : \exists f, f' \in n(i) : x_i^{f'} \neq x_i^f$ do
        for $f \in n(i)$ do
            $\mu_{if}(x_i^f) \leftarrow \mu_{if}(x_i^f) + \frac{\alpha}{r}$
    $r \leftarrow r + 1$
```

2.6 Higher order factors and efficiency

The message update equation (2.19) in the BP algorithm and the slave minimization (2.25) in the DD algorithm are, in general, exponential in the size of neighborhood of factor f, which is computationally expensive for higher order factor graphs. This motivates us to seek a generic class of factors that can be efficiently applied to the belief propagation (BP) and dual decomposition (DD) algorithms. We begin by introducing common forward sensor models that need to fit our class of factors followed by their generalization and then their application to the BP and DD algorithms.

2.6.1 Forward sensor models

Forward sensor models estimate the sensor reading given the environment. In this section, we describe two commonly used sensor models a) Gaussian and b) piecewise constant sensor models.

2.6.1.1 Gaussian sensor model (GSM)

Assuming Gaussian noise σ, the range measurement by a laser sensor is given by:

$$p(z_f|\mathbf{x}_f, g_f) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\bar{z}_f(\mathbf{x}_f, g_f) - z_f)^2}{2\sigma^2}\right), \tag{2.27}$$

where $\bar{z}_f(\mathbf{x}_f, g_f)$ is the distance of first occupied cell in $\mathbf{x}_f$ starting from pose g_f.

2.6.1.2 Piecewise constant sensor model (PCSM)

It is common in graphical models to have simple factors that assign high probability to expected configurations and low probability to all other states. Hence, we propose the following factor, which we use in all our experiments,

$$p(z_f|\mathbf{x}_f, g_f) = \frac{1}{Z}\begin{cases} 1 & \text{if } \mathbf{x}_f = \mathbf{R}_{f1} \\ \exp(-900) & \text{if } \mathbf{x}_f = \mathbf{R}_{f2} \\ \exp(-1000) & \text{otherwise} \end{cases}, \tag{2.28}$$

where Z is the normalization constant, $\mathbf{R}_{f1} = [0, 0 \ldots 0, 1]^\top$ denotes the all-free-but-last-occupied cell pattern and $\mathbf{R}_{f2} = [0, 0 \ldots 0, 0]^\top$ denotes all free cells. The second case indicates that we are more averse to estimating the reflecting cell closer to the robot as compared to estimating it away from the robot.

2.6.2 Generalization to pattern-based factors

We define a class of *pattern-based* higher order factors that are a generalization of those discussed by Komodakis et al. [61]. These are factors of the form:

$$p(z_f|\mathbf{x}_f, g_f) = \begin{cases} \psi_m & \text{if } \mathbf{x}_f \sim \mathbf{R}_m \ \ \forall 1 \leq m \leq M \\ \psi_{\text{min}} & \text{otherwise} \end{cases} \quad , \tag{2.29}$$

where $\mathbf{R}_m$ is one of the mutually exclusive M patterns and expression $\mathbf{x}_f \sim \mathbf{R}_m$ denotes that vector $\mathbf{x}_f$ "matches" pattern $\mathbf{R}_m$. A pattern, $\mathbf{R}_m = (n_0^m(f), \mathbf{r}^m)$, is defined by a non-empty set of *fixed* nodes $n_0^m(f) \subseteq n(f)$ that are expected to have desired values $\mathbf{r}^m$, while the state of remaining *free* nodes can take any value from the label set. A configuration $\mathbf{x}_f$ "matches" pattern $\mathbf{R}_m$ if the state of *fixed* nodes is the same as the desired values, $x_i = r_i^m \forall i \in n_0^m(f)$.

It is clear that PCSM (2.28) is an instance of pattern-based factors. It is also possible to represent GSM (2.27) as a pattern-based factor. We use the fact that the term $\bar{z}_f(\mathbf{x}_f, g_f)$, just depends on the first occupied cell. We define pattern $\mathbf{R}_m$ such that the first occupied cell is the m^{th} cell in $n(f)$:

$$n_0^m(f) = n(f)_{1:m} \tag{2.30}$$

$$\mathbf{r}^m = \{r_i^m = 0\}_{i \in n(f)_{1:m-1}} \text{ and } r^m_{n(f)_m} = 1 \tag{2.31}$$

$$R_m = (n_0^m, \mathbf{r}^m) \ , \tag{2.32}$$

where $n(f)_k$ is the k^{th} cell that is traced by the laser. With this formulation, we will have $M = n(f)$ patterns to describe the GSM in the form (2.29) with ψ_m given by

$$\psi_m = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\delta(m) - z_f)^2}{2\sigma^2} \right) \ , \tag{2.33}$$

where $\delta(m)$ is the distance of m^{th} cell from the robot starting point g_f. Also note that in this formulation the only case for *otherwise* is the one when all cells are 0 (or free).

2.6.3 Efficient sum product

To efficiently compute (2.19) for the pattern-based factors defined we need the messages to be normalized. We assume that the messages $\mu^r_{i\to f}(l_i)$ are normalized over the label set L_i to sum up to one, $\sum_{l_i \in L_i} \mu^r_{i \to f}(l_i) = 1$. Using the messages as a probability measure for the state of source nodes, we can compute the probability of each pattern being true:

$$p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i) = \begin{cases} 0 & \text{if } i \in n_0^m(f) \text{ and } l_i \neq r_i^m \\ \prod_{j \in n_0^m(f) \setminus i} \mu^r_{j \to f}(r_j^m) & \text{otherwise} \end{cases} . \tag{2.34}$$

The message update equation (2.19) can be written in terms of probability of patterns:

$$\mu^{r+1}_{f \to i}(l_i) = \sum_{m \leq M} \psi_m p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i) + \psi_{\text{min}} p_{\text{otherwise}} \ , \tag{2.35}$$

where $p_{\text{otherwise}} = 1 - \sum_{m \leq M} p(\mathbf{x_f} \sim \mathbf{R}_m | x_i = l_i)$.

The message update can be computed in $O(M|n(f)|)$ using (2.35), instead of $O(L_i^{|n(f)|})$ in the general case. Note that for GSM the number of patterns is same as the size of neighborhood, while for PCSM the number is fixed. Hence, the message update step is quadratic in neighborhood size for GSM and linear for PCSM.

2.6.4 Efficient dual decomposition

We note that slave function (2.25) is composed of two terms: the factor itself and the messages. While minimizing for each pattern we can make use of the fact that the value of θ_f is constant for a pattern and hence we need to focus only on minimizing the messages. Minimizing the messages is trivial as each term can be minimized independently.

While minimizing the *otherwise* case of patterns, we must ensure the exclusivity from explicit patterns. If the space for the otherwise case is small (like in the Gaussian sensor model), we simply go over the entire space to find the minimum value. Otherwise, we keep finding the n-best minimizations of messages until we get a minimization that does not match any of the patterns already considered. For example, for the piecewise constant sensor model, there are only two patterns that need to be checked for exclusion. Hence, the otherwise term can be easily minimized by finding the three best assignments that minimize the messages term. Note that message update for max product BP (2.22), can be done by following exactly the same logic.

2.7 Experiments

We run experiments on simulated as well real data. The simulated data is generated using *Player/Stage* [38] project. We use multiple map bitmaps bundled along with player/stage

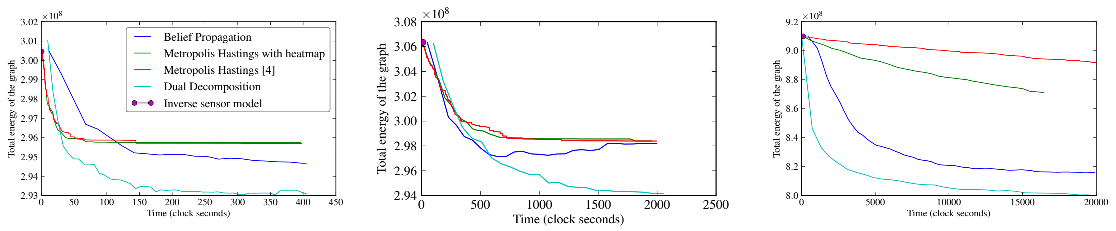


Figure 2.3: Comparison of convergence rate of different algorithms on occupancy grid mapping. From left to right, convergence rate on three datasets is shown: 1) *cave*, 2) *hospital section* and 3) *albert-b* [48] dataset. While sampling methods like Metropolis hastings converge quickly they stay far from optimum energy. On the other hand modern inference algorithms like belief propagation and dual decomposition reach closer to an optimum value. The legends are listed only once for clarity.

library. The robot motion is generated using the wander driver. The robot is allowed to wander in the map for 2 minutes aggregating approximately 270,000 laser measurements.

For real data, we have used the *albert-b-laser* dataset provided by C. Stachniss from University of Freiburg. The dataset was captured by a B21r robot with a SICK PLS moving through a lab at University of Freiburg. This data set was obtained from the Robotics Data Set Repository (Radish) [48].

To evaluate the convergence rate of each algorithm, we plot total energy (negative log likelihood) of the graph with respect to CPU ticks used by the algorithm. The plots of energy convergence with respect to time for cave dataset is shown in Fig. 2.3. This data clearly show the improvement from moving to belief propagation and dual decomposition, which, in all cases, leads to lower energies faster than our baselines. DD outperforms BP in typical cases.

In all our experiments we do not use any occupancy prior, although Merali et al. [84] suggest using an occupancy prior of 0.3 for better convergence. We use a step size of 50 for dual decomposition and piecewise constant sensor model. Also, we prefer piecewise constant sensor model over Gaussian sensor model because of the former being faster which is a consequence of having fewer patterns in the pattern-based factor formulation. We have implemented the algorithms in C++ and the code is available at the authors' websites.

2.7.1 Discussion

As is evident from the convergence comparison in Fig. 2.3, sampling algorithms (Metropolis Hastings with/without heatmap) are liable to getting stuck in a local minima. This is also an artifact of the simple transition probability where we flip only one cell at a time. Even from the qualitative results for sampling algorithms (Fig. 2.4), we see that the walls are thinner than the corresponding results in other algorithms which shows the inability of sampling-based algorithms to form lower energy and thicker walls for the piecewise constant sensor model. The downside of being biased towards thinner walls is evident in the

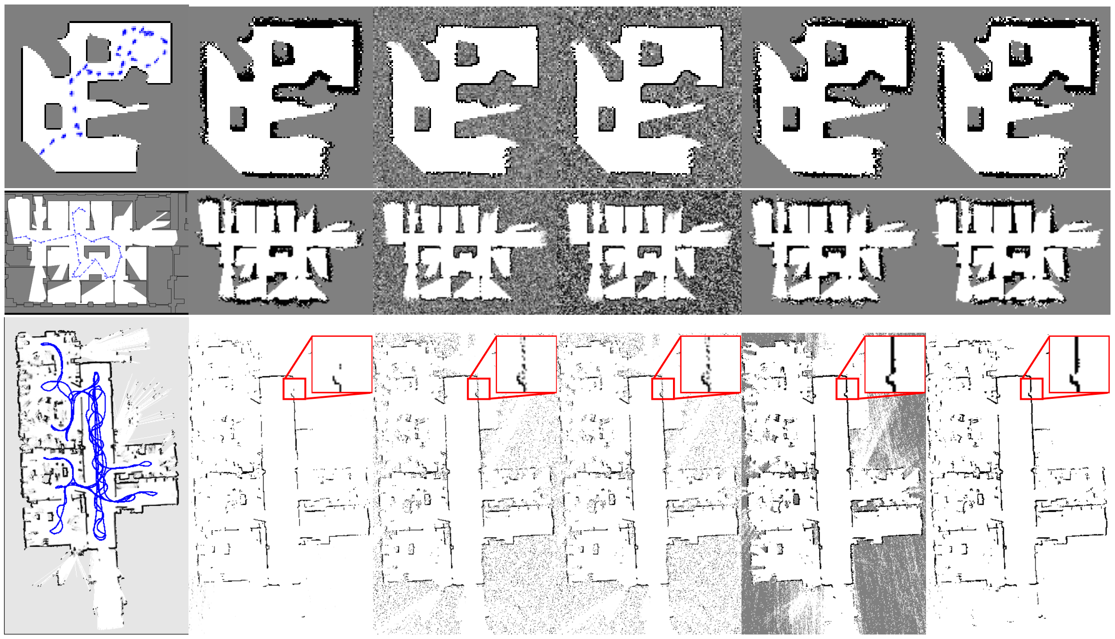


Figure 2.4: Qualitative results on different datasets. Each row represents a different dataset while each column represents a different algorithm. The columns correspond to the following algorithms (from left to right): 1) Ground truth with the trajectory of the robot 2) Inverse sensor model 3) Metropolis Hastings without heat map 4) Metropolis Hastings with heat map 5) Belief Propagation (BP) 6) Dual decomposition (DD). The rows correspond to the following datasets (from top to bottom): 1) cave 2) hospital section 3) albert-b. The grainy-ness in columns (3) and (4) is an artifact of sampling algorithms, when we sample over finite number of sample to compute expected state of a cell. Also note the missing or ragged walls in first 3 algorithms, while BP and DD are able to converge to thick solid walls. [48].

albert-b dataset (see Fig 2.4), as we get ragged walls for the sampling algorithms.

2.8 Conclusion and Future work

Dual decomposition is faster because it focuses on disagreeing nodes. However, step size is a crucial parameter that affects the speed of convergence. On the other hand, sum product belief propagation does not depend on any parameter, but has no preference for the disagreeing nodes. This combination obviously hints towards an algorithm where we perform belief propagation over disagreeing nodes only. The state of art variations of these algorithms, like Sequential Tree Re-Weighted (TRW-S) belief propagation [59], accelerated dual decomposition [50], are steps in this direction. Also, a recent comparative study [53] points towards other candidate methods, e.g. polyhedra based methods, that outperform than the dual decomposition class of methods. Even without using these more recent algorithmic developments, we get stronger performance than methods used so far. This only serves to prove our assertion that modern inference methods should be used for occupancy grid mapping.

CHAPTER 3

Mutual Localization[1]

3.1 Introduction

Cooperative localization is the problem of finding the relative 6-DOF pose between robots using sensors from more than one robot. Various strategies involving different sensors have been used to solve this problem. For example, Cognetti et al. [18, 34] use multiple bearning-only observations with a motion detector to solve for cooperative localization among multiple anonymous robots. Trawny et al. [140] and lately Zhou et al. [156, 155] provide a comprehensive mathematical analysis of solving cooperative localization for different cases of sensor data availability. Section 3.2 covers related literature in more detail.

To the best of our knowledge, all other cooperative localization works (see Section 3.2) require estimation of egomotion. However, a dependency on egomotion is a limitation for systems that do not have gyroscopes or accelerometers, which can provide displacement between two successive observations. Visual egomotion, like MonoSLAM [20], using distinctive image features estimates requires high quality correspondences, which remains a challenge in machine vision, especially in cases of non-textured environments. Moreover, visual egomotion techniques are only correct up to a scale factor. Contemporary cooperative localization methods that use egomotion [156, 140, 81] yield best results only with motion perpendicular to the direction of mutual observation and fails to produce results when either observer undergoes pure rotation or motion in the direction of observation. Consequently, in simple robots like Turtlebot, this technique produces poor results because of absence of sideways motion that require omni-directional wheels.

[1]This material originally appeared as: V. Dhiman, J. Ryde, and J. J. Corso. Mutual localization: Two camera relative 6-dof pose estimation from reciprocal fiducial observation. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2013

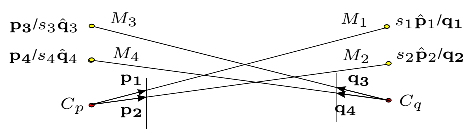

$\mathbf{p_3}/s_3\hat{\mathbf{q}}_3$
M_3
M_1
$s_1\hat{\mathbf{p}}_1/\mathbf{q_1}$
$\mathbf{p_4}/s_4\hat{\mathbf{q}}_4$
M_4
M_2
$s_2\hat{\mathbf{p}}_2/\mathbf{q_2}$
$\mathbf{p_1}$
$\mathbf{q_3}$
C_p
$\mathbf{p_2}$
$\mathbf{q_4}$
C_q

Figure 3.1: Simplified diagram for the two-camera problem. Assuming the length of respective rays to be $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ respectively, each marker coordinates can be written in both coordinate frames $\{p\}$ and $\{q\}$. For example M_1 is $\gamma_1\hat{\mathbf{p}}_1$ in frame $\{p\}$ and $\mathbf{q_1}$ in $\{q\}$, where $\hat{\mathbf{p}}_1$ unit vector parallel to $\mathbf{p}_1$.

To obviate the need for egomotion, we propose a method for relative pose estimation that leverages distance between fiducial markers mounted on robots for resolving scale ambiguity. Our method, which we call *mutual localization*, depends upon the simultaneous mutual/reciprocal observation of bearing-only sensors. Each sensor is outfitted with fiducial markers (Fig. 3.1) whose position within the host sensor coordinate system is known, in contrast to assumptions in earlier works that multiple world landmarks would be concurrently observable by each sensor [162]. Since our method does not depend on egomotion, hence it is instantaneous, which means it is robust to false negatives and is not susceptible to the errors in egomotion estimation.

The main contribution of our work is a generalization of *Perspective-3-Points* (P3P) problem where the observer and the observed points are distributed in different reference frames unlike conventional approaches where the observer's reference frame does not contain any observed points and vice versa. In this chapter we present an algebraic derivation to solve for the relative camera pose (rotation and translation) of the two bearing-only sensors in the case that each can observe two known fiducial points in the other sensor; essentially giving an algebraic system to compute the relative pose from four correspondences (only three are required in our algorithm but we show how the fourth correspondence can be used to generate a set of hypothesis solutions from which best solution can be chosen). Two fiducial points on each robot (providing four correspondences) are preferable to one on one and two on the other, as it allows extension to multi-robot (> 2) systems ensuring that any pair of similarly equipped robots can estimate their relative pose. In this chapter, we focus on only the two robot case as an extension to the multi-robot case as pairwise localization is straightforward yet practically effective.

Our derivation, although inspired by the linear pose estimation method of Quan and Lan [113], is novel since all relevant past works we know on the P3P problem [44], assume all observations are made in one coordinate frame and observed points in the other. In contrast, our method makes no such assumption and concurrently solves the pose estimation problem for landmarks sensed in camera-specific coordinate frames.

We demonstrate the effectiveness of our method, by analyzing its accuracy in both synthetic, which affords quantitative absolute assessment, and real localization situations by deployment on Turtlebots. We use 3D reconstruction experiments to show the accuracy of our algorithm. Our experiments demonstrate the effectiveness of the proposed approach.

3.2 Related Work

Cooperative localization has been extensively studied and applied to various applications. One of the latest works in this area comes from Cognetti et al. [18], [34] where they focus on the problem of cooperatively localizing multiple robots anonymously. They use multiple bearing-only observations and a motion detector to localize the robots. The robot detector is a simple feature extractor that detects vertical cardboard squares mounted atop each robot in the shadow zone of the range finder. One of oldest works come from Karazume et. al. [69] where they focus on using cooperative localization as a substitute to dead reckoning by suggesting a "dance" in which robots act as mobile landmarks. Although they do not use egomotion, but instead assume that position of two robots are known while localizing the third robot. Table 3.1 summarizes a few closely related works with emphasis on how our work is different from each of them. The rest of the section discusses those in detail.

Howard et al. [47] coined the CLAM (Cooperative Localization and Mapping) where they concluded that as an observer robot observes the explorer robot, it improves the localization of robots by the new constraints of observer to explorer distance. Recognizing that odometry errors can accumulate over time, they suggest using constraints based on cooperative localization to refine the position estimates. Their approach, however, does not use the merits of mutual observation as they propose that one robot explores the world and whilst the other robot watches. We show in our experiments, by comparison to ARToolKit [54] and Bundler [128], that mutual observations of robots can be up to 10 times more accurate than observations by a single robot.

A number of groups have considered cooperative vision and laser based mapping in outdoor environments [80, 123] and vision only [76, 119]. Localization and mapping using heterogeneous robot teams with sonar sensors is examined extensively by [40, 57]. Using more than one robot enables easier identification of previously mapped locations, simplifying the loop-closing problem [62].

Fox et al. [33] propose a Monte-Carlo based cooperative localization technique. The method uses odometry measurements for egomotion. Chang et al. [14] uses depth and visual sensors to localize Nao robots in the 2D ground plane. Roumeliotis and Bekey [120] focus on sharing sensor data across robots, employing as many sensors as possible which include odometry and range sensors. Rekleitis et al. [117] provide a model of robots moving in 2D equipped with both distance and bearing sensors.

Zou and Tan [162] proposed a cooperative simultaneous localization and mapping

Related work \ Tags	NoEM	BO	NoSLAM	MO
Mutual localization	✓	✓	✓	✓
Howard et al.[47]	✗	✓	✓	✓
Zou and Tan [162]	✓	✓	✗	✗
Cognetti et al.[18]	✗	✓	✓	✓
Trawny et al.[140]	✗	✓	✓	✓
Zhou and Roumeliotis [156, 155]	✗	✓	✓	✓
Roumeliotis et al.[120]	✗	✗	✗	✓

where

Tag	meaning	Tag	meaning
NoEM	Without Ego-Motion. All those works that use egomotion are marked as ✗.	BO	Localization using bearing only measurements. No depth measurements required. All those works that require depth measurements are marked with ✗.
NoSLAM	SLAM like tight coupling. Inaccuracy in mapping leads to cumulating interdependent errors in localization and mapping. All those works that use SLAM like approach are marked with a ✗.	MO	Utilizes mutual observation, which is more accurate than one-sided observations. All those works that do not use mutual observation, and depend on one-sided observations are marked as ✗

Table 3.1: Comparison of related work with Mutual localization

method, CoSLAM, in which multiple robots concurrently observe the same scene. Correspondences in time (for each robot) and across robots are fed into an extended Kalman filter and used to simultaneously solve the localization and mapping problem. However, this and other "co-slam" approaches such as [58] remain limited due to the interdependence of localization and mapping variables: errors in the map are propagated to localization and vice versa.

Recently Zhou and Roumeliotis [156, 155] have published a set of 14 minimal solutions that cover a wide range of robot-to-robot measurements. However, they use egomotion for their derivation and assume that observable fiducial markers coincide with the optical center of the camera. Our work does not make any of these two assumptions.

3.3 Problem Formulation

We use the following notation in this chapter, see Fig. 3.1. C_p and C_q represent two robots, each with a camera as a sensor. The corresponding coordinate frames are $\{p\}$ and $\{q\}$ respectively with origin at the optical center of the camera. Fiducial markers M_1 and M_2 are fixed on robot C_q and hence their positions are known in frame $\{q\}$ as

$\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^3$. Similarly, $\mathbf{p}_3, \mathbf{p}_4 \in \mathbb{R}^3$ are the positions of markers M_3 and M_4 in coordinate frame $\{p\}$. Robots are positioned such that they can observe each others markers in their respective camera sensors. The 2D image coordinates of the markers M_1 and M_2 in the image captured by the camera $\{p\}$ are measured as $\bar{\mathbf{p}}_1, \bar{\mathbf{p}}_2 \in \mathbb{R}^2$ and that of M_3 and M_4 is $\bar{\mathbf{q}}_3, \bar{\mathbf{q}}_4 \in \mathbb{R}^2$ in camera $\{q\}$. Let $K_p, K_q \in \mathbb{R}^{3\times 3}$ be the intrinsic camera matrices of the respective camera sensors on robot C_p, C_q. 2D image coordinates are denoted by a *bar* ($\bar{\mathbf{p}}$) and unit vectors that provide bearing information are denoted by a *caret* ($\hat{\mathbf{p}}$).

Since the real life images are noisy, the measured image positions $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{q}}_i$ will differ from the actual positions $\bar{\mathbf{p}}_{i0}$ and $\bar{\mathbf{q}}_{i0}$ by Gaussian noise η_{pi}, η_{qi}.

$$\bar{\mathbf{p}}_i = \bar{\mathbf{p}}_{i0} + \eta_{pi} \quad \forall i \in \{1, 2\} \tag{3.1}$$

$$\bar{\mathbf{q}}_i = \bar{\mathbf{q}}_{i0} + \eta_{qi} \quad \forall i \in \{3, 4\} \tag{3.2}$$

The problem is to determine the rotation $R \in \mathbb{R}^{3\times 3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ from frame $\{p\}$ to frame $\{q\}$ such that any point $\mathbf{p}_i$ in frame $\{p\}$ is related to its corresponding point $\mathbf{q}_i$ in frame $\{q\}$ by the following equation.

$$\mathbf{q}_i = R\mathbf{p}_i + \mathbf{t} \tag{3.3}$$

The actual projections of markers M_i into the camera image frames of the other robot are governed by following equations,

$$\bar{\mathbf{p}}_{i0} = \pi(K_p R^{-1}(\mathbf{q}_i - \mathbf{t})) \quad \forall i \in \{1, 2\} \tag{3.4}$$

$$\bar{\mathbf{q}}_{i0} = \pi(K_q(R\mathbf{p}_i + \mathbf{t})) \quad \forall i \in \{3, 4\} \tag{3.5}$$

where π is the projection function defined over a vector
$\mathbf{v} = \begin{bmatrix} v_x, v_y, v_z \end{bmatrix}^\top \in \mathbb{R}^3$ as

$$\pi(\mathbf{v}) = \begin{bmatrix} \frac{v_x}{v_z}, \frac{v_y}{v_z} \end{bmatrix}^\top \tag{3.6}$$

To minimize the effect of noise we must compute the optimal transformation, R^* and $\mathbf{t}^*$.

$$(R^*, \mathbf{t}^*) = \arg\min_{(R,\mathbf{t})} \left(\sum_{i\in\{1,2\}} \|\bar{\mathbf{p}}_i - \pi(K_p R^{-1}(\mathbf{q}_i - \mathbf{t}))\|^2 + \sum_{i\in\{3,4\}} \|\bar{\mathbf{q}}_i - \pi(K_q(R\mathbf{p}_i + \mathbf{t}))\|^2 \right) \tag{3.7}$$

To solve this system of equations we start with exact equations that lead to a large number of polynomial roots. To choose the best root among the set of roots we use the above minimization criteria.

Let $\hat{\mathbf{p}}_i, \hat{\mathbf{q}}_i \in \mathbb{R}^3$ be the unit vectors drawn from the camera's optical center to the image projection of the markers. The unit vectors can be computed from the position of markers in camera images $\bar{\mathbf{p}}_i, \bar{\mathbf{q}}_i$ by the following equations.

$$\hat{\mathbf{p}}_i = \frac{K_p^{-1}\left[\bar{\mathbf{p}}_i^\top, 1\right]^\top}{\left\|K_p^{-1}\left[\bar{\mathbf{p}}_i^\top, 1\right]^\top\right\|} \quad \forall i \in \{1,2\}, \qquad \hat{\mathbf{q}}_i = \frac{K_q^{-1}\left[\bar{\mathbf{q}}_i^\top, 1\right]^\top}{\left\|K_q^{-1}\left[\bar{\mathbf{q}}_i^\top, 1\right]^\top\right\|} \quad \forall i \in \{3,4\} \tag{3.8}$$

Further let γ_1, γ_2 be the distances of markers M_1, M_2 from the optical center of the camera sensor in robot C_p. And γ_3, γ_4 be the distances of markers M_3, M_4 from the camera optical center in robot C_q. Then the points $\mathbf{q}_1$, $\mathbf{q}_2$, $\gamma_3\hat{\mathbf{q}}_3$, $\gamma_4\hat{\mathbf{q}}_4$ in coordinate frame $\{q\}$ correspond to the points $\gamma_1\hat{\mathbf{p}}_1$, $\gamma_2\hat{\mathbf{p}}_2$, $\mathbf{p}_3$, $\mathbf{p}_4$ in coordinate frame $\{p\}$,

$$\mathbf{q}_1 = \mathbf{t} + \gamma_1 R\hat{\mathbf{p}}_1, \qquad \mathbf{q}_2 = \mathbf{t} + \gamma_2 R\hat{\mathbf{p}}_2, \tag{3.9}$$

$$\gamma_3\hat{\mathbf{q}}_3 = \mathbf{t} + R\mathbf{p}_3, \qquad \gamma_4\hat{\mathbf{q}}_4 = \mathbf{t} + R\mathbf{p}_4 \ . \tag{3.10}$$

These four vector equations provide us 12 constraints (three for each coordinate in 3D) for our 10 unknowns (3 for rotation R, 3 for translation $\mathbf{t}$, and 4 for γ_i). We first consider only the first three equations, which allows an exact algebraic solution of the nine unknowns from the nine constraints.

Our approach to solving the system is inspired by the well studied problem of *Perspective-3-points* [44], also known as *space resection* [113]. However, note that the method cannot be directly applied to our problem as known points are distributed in both coordinate frames as opposed to the space resection problem where all the known points are in the one coordinate frame.

The essential flow of our approach is to first solve for the three range factors, γ_1, γ_2 and γ_3 (Section 3.3.1). Then we set up a classical absolute orientation system on the rotation and translation (Section 3.3.2), which is solved using established methods such as Arun et al. [4] or Horn [46]; finally, since our algebraic solution will give rise to many candidate roots, we develop a root-filtering approach to determine the best solution (Section 3.3.4).

3.3.1 Solving for γ_1, γ_2 and γ_3

The first step is to solve the system for γ_1, γ_2 and γ_3. We eliminate R and $\mathbf{t}$ by considering the inter-point distances in both coordinate frames.

$$\begin{aligned} \|\gamma_1\hat{\mathbf{p}}_1 - \gamma_2\hat{\mathbf{p}}_2\| &= \|\mathbf{q}_1 - \mathbf{q}_2\| \\ \|\gamma_2\hat{\mathbf{p}}_2 - \mathbf{p}_3\| &= \|\mathbf{q}_2 - \gamma_3\hat{\mathbf{q}}_3\| \\ \|\mathbf{p}_3 - \gamma_1\hat{\mathbf{p}}_1\| &= \|\gamma_3\hat{\mathbf{q}}_3 - \mathbf{q}_1\| \end{aligned} \tag{3.11}$$

Squaring both sides and representing the vector norm as the dot product gives the following system of polynomial equations.

$$\gamma_1^2 + \gamma_2^2 - 2\gamma_1\gamma_2\hat{\mathbf{p}}_1^\top\hat{\mathbf{p}}_2 - \|\mathbf{q}_1 - \mathbf{q}_2\|^2 = 0 \tag{3.12a}$$
$$\gamma_2^2 - \gamma_3^2 - 2\gamma_2\hat{\mathbf{p}}_2^\top\mathbf{p}_3 + 2\gamma_3\mathbf{q}_2^\top\hat{\mathbf{q}}_3 + \|\mathbf{p}_3\|^2 - \|\mathbf{q}_2\|^2 = 0 \tag{3.12b}$$
$$\gamma_1^2 - \gamma_3^2 - 2\gamma_1\hat{\mathbf{p}}_1^\top\mathbf{p}_3 + 2\gamma_3\mathbf{q}_1^\top\hat{\mathbf{q}}_3 + \|\mathbf{p}_3\|^2 - \|\mathbf{q}_1\|^2 = 0 \tag{3.12c}$$

This system has three quadratic equations implying a Bezout bound of eight (2^3) solutions. Using the Sylvester resultant we sequentially eliminate variables from each equation. Rewriting (3.12a) and (3.12b) as quadratics in terms of γ_2 gives

$$\gamma_2^2 + \underbrace{(-2\gamma_1\hat{\mathbf{p}}_1^\top\hat{\mathbf{p}}_2)}_{a_1}\gamma_2 + \underbrace{(\gamma_1^2 - \|\mathbf{q}_1 - \mathbf{q}_2\|^2)}_{a_0} = 0 \tag{3.13}$$

$$\gamma_2^2 + \underbrace{(-2\hat{\mathbf{p}}_2^\top\mathbf{p}_3)}_{b_1}\gamma_2 - \underbrace{(\gamma_3^2 - 2\gamma_3\mathbf{q}_2^\top\hat{\mathbf{q}}_3 - \|\mathbf{p}_3\|^2 + \|\mathbf{q}_2\|^2)}_{b_0} = 0 \tag{3.14}$$

The Sylvester determinant [12, p. 123] of (3.13) and (3.14) is given by the determinant of the matrix formed by the coefficients of γ_2.

$$\mathrm{SD}(\gamma_1, \gamma_3) = \begin{vmatrix} 1 & a_1 & a_0 & 0 \\ 0 & 1 & a_1 & a_0 \\ 1 & b_1 & b_0 & 0 \\ 0 & 1 & b_1 & b_0 \end{vmatrix} \tag{3.15}$$

This determinant is a quartic function in γ_1, γ_3. By definition, the resultant is zero if and only if the parent equations have at least a common root[12]. Thus we have eliminated variable γ_2 from (3.12a) and (3.12b). We can repeat the process for eliminating γ_3 by rewriting SD(γ_1, γ_3) and (3.12c) as:

$$\begin{aligned} \text{SD}(\gamma_1, \gamma_3) = c_4\gamma_3^4 + c_3\gamma_3^3 + c_2\gamma_3^2 + c_1\gamma_3 + c_0 = 0 \\ -\gamma_3^2 + \underbrace{(2\mathbf{q}_1^\top \hat{\mathbf{q}}_3)}_{d_1}\gamma_3 + \underbrace{\gamma_1^2 - 2\gamma_1\hat{\mathbf{p}}_1^\top \mathbf{p}_3 + \|\mathbf{p}_3\|^2 - \|\mathbf{q}_1\|^2}_{d_0} = 0 \end{aligned} \tag{3.16}$$

The Sylvester determinant of (3.16) would be

$$\text{SD}_2(\gamma_1) = \begin{vmatrix} c_4 & c_3 & c_2 & c_1 & c_0 & 0 \\ 0 & c_4 & c_3 & c_2 & c_1 & c_0 \\ 1 & d_1 & d_0 & 0 & 0 & 0 \\ 0 & 1 & d_1 & d_0 & 0 & 0 \\ 0 & 0 & 1 & d_1 & d_0 & 0 \\ 0 & 0 & 0 & 1 & d_1 & d_0 \end{vmatrix} = 0. \tag{3.17}$$

Solving (3.17) gives an 8 degree polynomial in γ_1. By Abel-Ruffini theorem [6, p. 131], a closed-form solution of the above polynomial does not exist.

The numeric solution to (3.17) gives eight roots for γ_3. We compute γ_1 and γ_2 using (3.12c) and (3.12b) respectively. Because the camera cannot see objects behind it, only real positive roots are maintained from the resultant solution set.

3.3.2 Solving for R and t

With the solutions for the scale factors, $\{\gamma_1, \gamma_2, \gamma_3\}$ we can compute the absolute location of the markers $\{M_1, M_2, M_3\}$ in both the frames $\{p\}$ and $\{q\}$.

$$\mathbf{p}_i = \gamma_i\hat{\mathbf{p}}_i \quad \forall i \in \{1, 2\}, \qquad \mathbf{q}_i = \gamma_i\hat{\mathbf{q}}_i \quad \forall i \in \{3\}$$

These exact correspondences give rise to the classical problem of absolute orientation i.e. given three points in two coordinate frames find the relative rotation and translation between the frames. For each positive root of γ_1, γ_2, γ_3 we use the method in Arun et. al [4] method (similar to Horn's method [46]) to compute the corresponding rotation R and translation value $\mathbf{t}$.

3.3.3 Absolute orientation [4]

We provide a short summary of the Arun et al.'s algorithm. The algorithm computes R and $\mathbf{t}$ by the following optimization function,

$$(R*, \mathbf{t}*) = \arg\min_{R,\mathbf{t}} \sum_{i=1}^{3} \|\mathbf{q}_i - (R\mathbf{p}_i + \mathbf{t})\| \tag{3.18}$$

where $\mathbf{q}_i, \mathbf{p}_i$ are absolute positions of the markers in respective coordinate frames. We can eliminate the translation $\mathbf{t}$ by moving the origin to the centroids of all the points in both the coordinate frames. Let us call the new points $\mathbf{q}'$ and $\mathbf{p}'$.

$$\mathbf{q}'_i = \mathbf{q}_i - \sum_{j=1}^{3} \mathbf{q}_j \tag{3.19}$$

$$\mathbf{p}'_i = \mathbf{p}_i - \sum_{j=1}^{3} \mathbf{q}_j \tag{3.20}$$

Eliminating the translation variable $\mathbf{t}$,

$$R^* = \arg\min_{R \in SO(3)} \sum_{i=1}^{3} \|\mathbf{q}'_i - R\mathbf{p}'_i\|. \tag{3.21}$$

It can be proved that minimizing the above error function is the same as the maximizing the function

$$R^* = \arg\max_{R \in SO(3)} \mathrm{Tr}\left(RH\right), \tag{3.22}$$

where $H = \sum_{i=1}^{3} \mathbf{p}'_i \mathbf{q}'_i$ is known as the covariance matrix and $\mathrm{Tr}(M)$ is the Trace of matrix M. The optimal R is obtained by

$$R^* = VU^\top \tag{3.23}$$

where U, V are the left-singular vectors and right-singular vectors of H such that $H = U\Lambda V^\top$.

3.3.4 Choosing the optimal root

Completing squares in (3.12) yields important information about redundant roots.

$$(\gamma_1 + \gamma_2)^2 - 2s_1\gamma_2(1 + \hat{\mathbf{p}}_1^\top \hat{\mathbf{p}}_2) - \|\mathbf{q}_1 - \mathbf{q}_2\|^2 = 0 \tag{3.24a}$$

$$(\gamma_2 - \hat{\mathbf{p}}_2^\top \mathbf{p}_3)^2 - (\gamma_3 - \mathbf{q}_2^\top \hat{\mathbf{q}}_3)^2 + (\mathbf{p}_3 - \hat{\mathbf{p}}_2)^\top \mathbf{p}_3 - \mathbf{q}_2^\top(\mathbf{q}_2 - \hat{\mathbf{q}}_3) = 0 \tag{3.24b}$$

$$(\gamma_1 - \hat{\mathbf{p}}_1^\top \mathbf{p}_3)^2 - (\gamma_3 - \mathbf{q}_1^\top \hat{\mathbf{q}}_3)^2 + (\mathbf{p}_3 - \hat{\mathbf{p}}_1)^\top \mathbf{p}_3 - \mathbf{q}_1^\top(\mathbf{q}_1 - \hat{\mathbf{q}}_3) = 0 \tag{3.24c}$$

Equations (3.24) do not put any constraints on positivity of terms $(\gamma_2 - \hat{\mathbf{p}}_2^\top \mathbf{p}_3)$, $(\gamma_3 - \mathbf{q}_2^\top \hat{\mathbf{q}}_3)$, $(\gamma_1 - \hat{\mathbf{p}}_1^\top \mathbf{p}_3)$ or $(\gamma_3 - \mathbf{q}_1^\top \hat{\mathbf{q}}_3)$. However, all these terms are positive as long as the markers of the observed robot are farther from the camera than the markers of the observing robot. Also, the distances γ_i are assumed to be positive. Assuming the above, we filter the *real* roots by the following criteria:

$$\gamma_1 \geq \|\mathbf{p}_3\|, \qquad \gamma_2 \geq \|\mathbf{p}_3\|, \qquad \gamma_3 \geq \max(\|\mathbf{q}_1\|, \|\mathbf{q}_2\|) \tag{3.25}$$

These criteria not only reduce the number of roots significantly, but also filter out certain degenerate cases.

For all the filtered roots of (3.17), we compute the corresponding values of R and $\mathbf{t}$, choosing the best root that minimizes the error function, (3.7).

3.3.5 Extension to four markers

Even though the system is solvable by only three markers, we choose to use four markers for symmetry. We can fall back to the three marker solution in situations when one of the markers is occluded. Once we extend this system to 4 marker points, we obtain 6 bivariate quadratic equations instead of the three in (3.12) that can be reduced to three 8-degree univariate polynomials. The approach to finding the root with the least error is the same as described above.

3.3.6 Extension to more than four markers

Assume that the five markers are distributed on two robots such that the equations are formulated as:

$$\mathbf{q}_1 = \mathbf{t} + \gamma_1 R\hat{\mathbf{p}}_1, \qquad \mathbf{q}_2 = \mathbf{t} + \gamma_2 R\hat{\mathbf{p}}_2, \qquad \gamma_3\hat{\mathbf{q}}_3 = \mathbf{t} + R\mathbf{p}_3 \tag{3.26}$$

$$\gamma_4\hat{\mathbf{q}}_4 = \mathbf{t} + R\mathbf{p}_4, \qquad \gamma_5\hat{\mathbf{q}}_5 = \mathbf{t} + R\mathbf{p}_5 \tag{3.27}$$

Compare this system of equations to the system when two cameras observe an unknown set of corresponding points i.e. in both the coordinate frames we only know the projection of the points.

$$\gamma_1\hat{\mathbf{q}}_1 = \mathbf{t} + \gamma_2 R\hat{\mathbf{p}}_1, \qquad \gamma_3\hat{\mathbf{q}}_2 = \mathbf{t} + \gamma_4 R\hat{\mathbf{p}}_2, \qquad \gamma_5\hat{\mathbf{q}}_3 = \mathbf{t} + \gamma_6 R\hat{\mathbf{p}}_3 \tag{3.28}$$

$$\gamma_7\hat{\mathbf{q}}_4 = \mathbf{t} + \gamma_8 R\hat{\mathbf{p}}_4, \qquad \gamma_9\hat{\mathbf{q}}_5 = \mathbf{t} + \gamma_{10} R\hat{\mathbf{p}}_5 \tag{3.29}$$

Clearly, our problem under consideration (3.27) is easier than solving (3.29) as we already know five of the ten unknown scale factors. Fortunately, this problem of finding relative camera pose as they observe unknown set of five or more corresponding points has been well studied [132, 101, 108, 78]. However, these methods result in a solution that is correct up to a scale factor. By additionally using the distance relations from (3.11) we can solve for the relative camera pose along with scale factor.

Assuming the unknown essential matrix to be E, the system of equations (3.27) can be rewritten in the following form:

$$\mathbf{q}_i^\top E\hat{\mathbf{p}}_i = 0 \quad \forall i \in \{1, 2\}, \qquad \hat{\mathbf{q}}_i^\top E\mathbf{p}_i = 0 \quad \forall i \in \{3, 4, 5\} \tag{3.30}$$

With this notation, it is easy to follow the solution outlined in [101] for five or six points (markers). For even more points refer to [45] for the widely known 7-point and linear 8-point algorithms.

3.4 Implementation

We implement our algorithm on two Turtlebots with fiducial markers. One of the Turtlebots with markers is shown in Fig. 3.2. We have implemented the algorithm in Python using the Sympy [13], OpenCV [10] and Numpy [21] libraries. As the implementing software formulates and solves polynomials symbolically, it is generic enough to handle any reasonable number of points in two camera coordinate frames. We have tested the solver for the

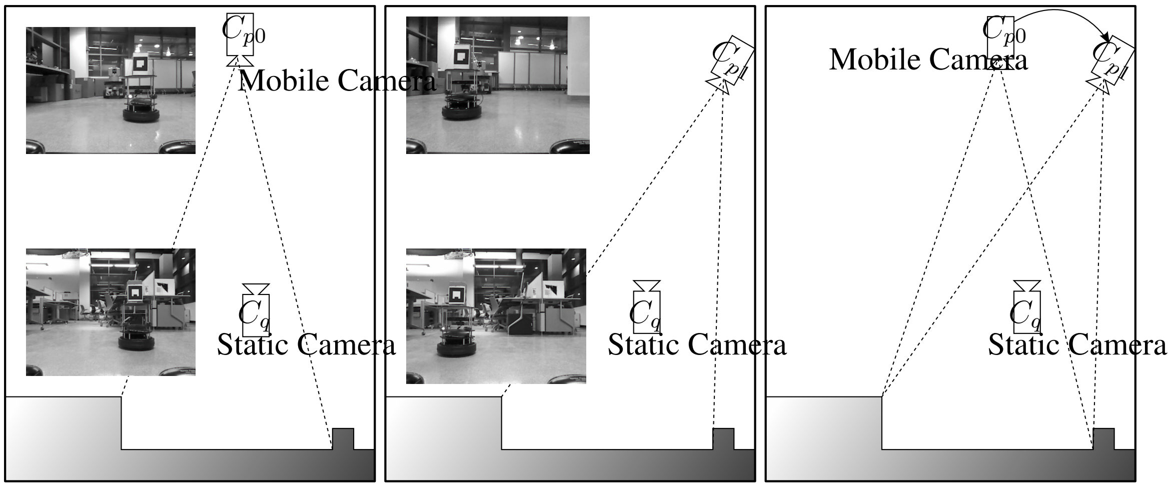


Figure 3.3: Diagram of the two camera setup for mutual localization 3D metric reconstruction, along with images from each camera for two poses of the mobile camera. Cameras have distinctive cylindrical barcode-like markers to aid detection in each others image frames. Also depicted is the triangulation to two example feature points.

following combination of points: 0-3, 1-2, 2-2, where 1-2 means that 1 point is known in the first coordinate frame and 2 points are known in the second.

We use blinking lights as fiducial markers on the robots and barcode-like cylindrical markers as for the 3D reconstruction experiment.

The detection of blinking lights follows a simple thresholding strategy on the time differential of images. This approach coupled with decaying confidence tracking produces satisfactory results for simple motion of robots and relatively static backgrounds. Fig. 3.3 shows the cameras mounted with blinking lights as fiducial markers. The robots shown in Fig. 3.3 are also mounted with ARToolKit[54] fiducial markers for the comparison experiments.

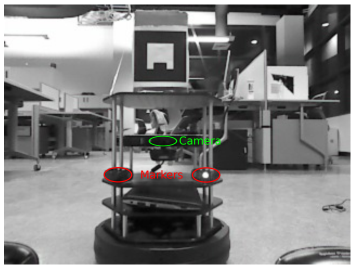


Figure 3.2: The deployment of markers on the Turtlebot used for experiments

3.5 Experiments

To assess the accuracy of our method we perform a localization experiment in which we measure how accurately our method can determine the pose of the other camera. We compare our localization results with the widely used fiducial-based pose estimation in ARToolKit [54] and visual egomotion and SfM framework Bundler [128]. We also generate a semi-dense reconstruction to compare

	Median Trans. err.	Median Rot. err.
ARToolKit[54]	0.57m	9.2°
Bundler[128]	0.20m	0.016°
Mutual Localization	0.016m	0.33°

Table 3.2: Table showing mean translation and rotation error for ARToolKit, Bundler and Mutual Localization

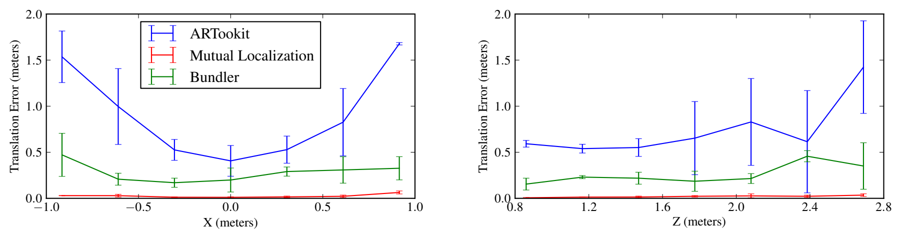


Figure 3.4: Translation error comparison between the ARToolKit and our mutual localization. The translation error is plotted to ground truth X and Z axis positions to show how error varies with depth (Z) and lateral (X) movements. We get better results in localization by a factor of ten. Also note how the translation error increases with Z-axis (inter-camera separation).

the mapping accuracy of our method to that of Bundler. The reconstruction quality is a measure of the mutual localization accuracy of the two cameras used in the reconstruction.

3.5.1 Localization Experiment

Setup Two turtlebots were set up to face each other. One of the turtlebot was kept stationary and the other moved in 1 ft increments in an X-Z plane (Y-axis is down, Z-axis is along the optical axis of the static camera and the X-axis is towards the right of the static camera). We calculate the rotation error by extracting the rotation angle from the differential rotation $R_{gt}^{\top} R_{est}$ as follows:

$$E_\theta = \frac{180}{\pi} \arccos\left(\frac{\mathrm{Tr}(R_{gt}^{\top} R_{est}) - 1}{2}\right) \tag{3.31}$$

where R_{gt} is the ground truth rotation matrix, R_{est} is the estimated rotation matrix and Tr is the matrix trace. The translation error is simply the norm difference between two translation vectors.

Results in comparison with ARToolKit [54] The ARToolKit is an open source library for detecting and determining the pose of fiducial markers from video. We use a ROS[114] wrapper – *ar_pose* – over ARToolKit for our experiments. We repeat the relative camera localization experiment with the ARToolKit library and Bundler[128] and compare to our results. The corresponding results are shown in Table 3.2. The results show a tenfold improvement in translation error over Bundler.

3.5.2 Simulation experiments with noise

A simple scene was constructed in Blender to verify the mathematical correctness of the method. Two cameras were set up in the blender scene along with a target object 1m from

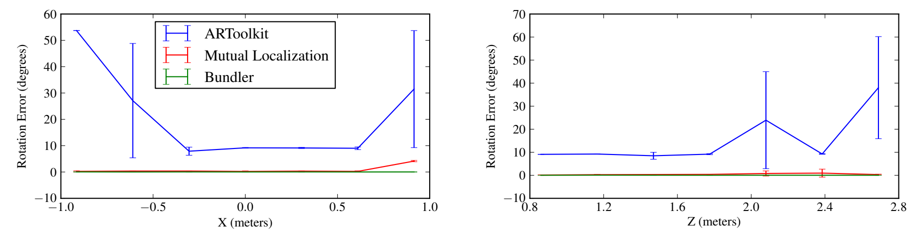


Figure 3.5: Rotation error comparison between the ARToolKit and Mutual localization. Rotation error decreases with Z-axis (ground truth inter-camera separation). See (3.31) for computation of rotation error.

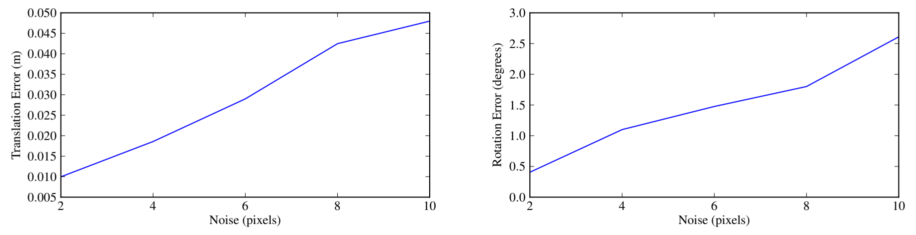
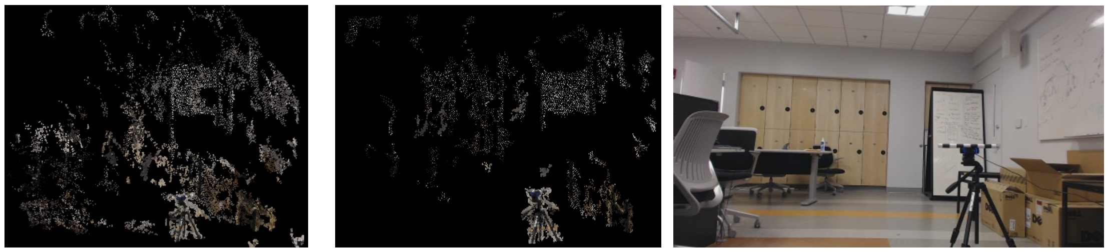


Figure 3.6: Rotation and translation error as noise is incrementally added to the detection of markers.

the static camera. Camera images were rendered at a resolution of 960×540. The markers were simulated as colored balls that were detected by simple hue based thresholding. The two cameras in the simulated scene were rotated and translated to cover maximum range of motion. After detection of the center of the colored balls, zero mean Gaussian noise was added to the detected positions to investigate the noise sensitivity of our method. The experiment was repeated with different values of noise covariance. Fig. 3.6 shows the translation and rotation error in the experiment with variation in noise. It can be seen that our method is robust to noise as it deviates only by 5cm and 2.5° when tested with noise of up to 10 pixels.

3.5.3 3D Reconstruction experiment

The position and orientation obtained from our method is inputted into the patch based multi-view stereo (PMVS-2) library [35] to obtain a semi-dense reconstruction of an indoor environment. Our reconstruction is less noisy when compared to that obtained by Bundler [128]. Fig. 3.7 shows a side-by-side snapshot of the semi-dense map from Bundler-PMVS and, our method, Mutual Localization-PMVS. To compare the reconstruction accuracy, we captured the scene as a point cloud with an RGB-D camera (Asus-Xtion). The Bundler and Mutual Localization output point clouds were manually aligned (and scaled) to the Asus-Xtion point cloud. We then computed the nearest neighbor distance from each point in the Bundler/Mutual localization point clouds discarding points with nearest neighbors further than 1m as outliers. With this metric the mean nearest neighbor distance for our method



(a) Bundler-PMVS (b) Mutual Localization-PMVS (c) Actual scene

Figure 3.7: The semi-dense reconstruction produced by our method, Mutual Localization, is less noisy (0.18m) when compared to that produced by Bundler (0.33m).

was 0.176m while that for Bundler was 0.331m.

3.6 Conclusion

We have developed a method to cooperatively localize two cameras using fiducial markers on the cameras in sensor-specific coordinate frames, obviating the common assumption of sensor egomotion. We have compared our results with the ARToolKit showing that our method can localize significantly more accurately, with a tenfold error reduction observed in our experiments. We have also demonstrated how the cooperative localization can be used as an input for 3D reconstruction of unknown environments, and find better accuracy (0.18m versus 0.33m) than the visual egomotion-based Bundler method. We plan to build on this work and apply it to multiple robots for cooperative mapping. Though we achieve reasonable accuracy, we believe we can improve the accuracy of our method by improving camera calibration and measurement of the fiducial marker locations with respect to the camera optical center. We will release the source code (open-source) for our method upon publication.

Acknowledgments

This material is based upon work partially supported by the Federal Highway Administration under Cooperative Agreement No. DTFH61-07-H-00023, the Army Research Office (W911NF-11-1-0090) and the National Science Foundation CAREER grant (IIS-0845282). Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the views of the FHWA, ARO, or NSF.

CHAPTER 4

Occlusion-aware localization[1]

4.1 Introduction

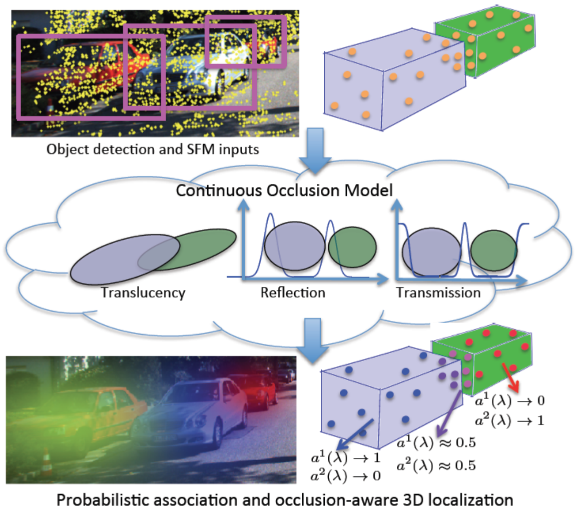
As a two-dimensional (2D) projection of the three-dimensional (3D) world, image formation is associated with a loss of information. This is especially significant when objects in 3D space occlude each other with respect to the camera viewpoint. In recent years, we have seen remarkable progress in various aspects of scene understanding, such as structure from motion (SFM) and object detection. However, occlusions still present a challenge, with the difficulty of physically modeling them being a major bottleneck.

Our main contribution is a novel theoretical model for occlusion handling that is continuous and fully 3D. Our model is motivated by insights from computer graphics, whereby we represent objects as translucent 3D ellipsoids. In Section 4.3, we develop novel continuous models for representing *transmission* and *reflection* probabilities for each ray emanating from the camera. This allows assigning probabilities for each point in space belonging to an object, which can explicitly explain image observations and reason about occlusions. This is in contrast to prior works that consider occlusions in 2D, or through discrete occluder patterns or models that are not physically interpretable [107, 106, 150, 161, 160, 159].

A key advantage afforded by our occlusion model is unification. While previous approaches to handling occlusions are application-dependent, ours is physically-inspired, thus, flexible enough to be used in a variety of scenarios. In this chapter, we show that our theory can be used for uniformly modeling the association of SFM point tracks with static or dynamic objects (Section 4.4.1), as well as modeling object detection scores in applications like 3D localization (Section 4.4.2). We demonstrate the application of our formulations for road scenes from the KITTI raw dataset [37].

[1]This material originally appeared as: V. Dhiman, Q. Tran, J. Corso, and M. Chandraker. A continuous occlusion model for road scene understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016

In particular, assigning 2D point tracks to independent, but potentially occluding, objects is a fundamental challenge in computer vision problems such as multibody SFM [103]. Recent works use motion segmentation [11, 115] as a precursor to localizing objects, which often suffices for moving objects [141] and has also been considered for multibody SFM [68]. However, motion-based segmentation is not always applicable in road scenes, due to static parked cars, or dynamic cars that move with similar velocities. Occlusions make the problem more severe since point tracks get clustered together for static objects and may frequently appear to change association among dynamic objects in 2D. Indeed, we show in Section 4.5 that our point track association outperforms state-of-the-art motion segmentation methods, as well as a baseline that uses detection bounding boxes but does not consider occlusions.


Object detection and SFM inputs
Continuous Occlusion Model
Translucency
Reflection
Transmission
$a^1(\lambda) \to 0$
$a^2(\lambda) \to 1$
$a^1(\lambda) \approx 0.5$
$a^2(\lambda) \approx 0.5$
$a^1(\lambda) \to 1$
$a^2(\lambda) \to 0$
Probabilistic association and occlusion-aware 3D localization
Figure 4.1: We propose an occlusion model in 3D that is physically-inspired and continuous. Given object detection and SFM point tracks, our unified model probabilistically assigns point tracks to objects and reasons about object detection scores and bounding boxes. It uniformly handles static and dynamic objects, thus, outperforms motion segmentation for association problems. We also demonstrate occlusion-aware 3D localization in road scenes.

Another potential application of our proposed model is towards 3D localization in road scenes. Prior works such as [130] combine information from point tracks and detection bounding boxes, but do not consider occlusions for either. In contrast, our unified occlusion model allows a probabilistic soft assignment of point tracks to objects, as well as an occlusion-aware interpretation of object detection outputs. Our model is continuous, so it remains amenable to the use of continuous optimization tools.

To summarize, our main contributions are:

- A novel theoretical model for handling occlusions that is continuous and formulated in 3D.
- Unified occlusion handling for point tracks in SFM and bounding boxes and detection scores in object detection.
- Application of our model to association of point tracks with both static and moving objects, improving over motion segmentation and occlusion-unaware baselines.
- Application of our unified formulation to 3D localization of traffic participants in road scenes.

4.2 Related Work

Occlusion handling in detection Several works in object detection consider occlusion by training a detector on visible parts of the object [36]. Occlusion reasoning based on 2D image silhouettes is used to improve detection performance in [49]. On the other hand, our occlusion reasoning is based on 3D entities. In recent years, object detectors have also considered occlusion reasoning using 3D cues, often learned from a dataset of CAD models [107, 106, 150]. By necessity, such frameworks are often a discrete representation of occlusion behavior, for example, in the form of a collection of occlusion masks derived from object configurations discretized over viewpoint. In contrast to these works, our occlusion modeling is also fully 3D, but allows for a continuous representation. Further, to derive 3D information, we do not use CAD models, rather we derive a probabilistic formulation based on physical insights.

Occlusion handling in tracking Occlusions have also been handled in tracking-by-detection frameworks by considering occluder patterns in the image [70, 149]. A notable exception is the work of Milan et al. [86] that explicitly models occlusions in the continuous domain to determine a visibility ratio for each object in multi-target tracking. However, the occlusion model in [86] is essentially the overlap of image projections of a Gaussian representation of the object. Our occlusion modeling, on the other hand, is fully 3D, based on physical modeling of object-ray intersections and much more general in determining the probability of a point in space as belonging to an object. While our model can also be used to determine a visibility ratio similar to [86], it has far more general applications and can be quantitatively evaluated, as shown by our experiments on point track associations.

Motion segmentation and multibody SFM An application for our occlusion modeling is to determine point track associations in scenes with multiple objects. For moving objects, this is within the purview of motion segmentation, which has been approached through algebraic factorization methods [19, 143, 142], statistical methods [52, 41, 116] and clustering methods [151, 39]. Some recent efforts include robust algebraic segmentation with hybrid perspective constraints [115] and spectral clustering with point track spatial affinities [11]. Unlike our work, such methods cannot handle static objects, or dynamic objects with little relative motion. Closer to our application, motion segmentation is also used within multibody SFM frameworks [68, 98, 103]. In contrast to these works, our formulation does not distinguish between moving and static objects and also explicitly reasons about occlusions due to 3D object geometries for associating point tracks to individual objects.

3D localization One of the vital goals of 3D scene understanding is to localize 3D objects in complex scenes. Monocular frameworks like ours have also reasoned about occlusions, for instance, partial object detectors are considered in [148]. A detailed part-based representation of objects based on annotated CAD models is used for monocular scene understanding in [161, 160, 159], which also allows reasoning about mutual occlusions between objects. In contrast to these works, our monocular framework uses a physical modeling of occlusion in continuous space and derives unified representations for SFM points and object detection bounding boxes. This makes our model more general, extensible and amenable for continuous optimization.

4.3 Continuous Occlusion Model

A common parametric modeling for objects, especially traffic participants in road scene understanding, is as opaque cuboids.[2] However, such models introduce discontinuities in the problem formulation and do not adequately account for uncertainties in pose and dimensions. With this motivation, we introduce our representation of 3D objects and our modeling of object-object relationships, which lead to a continuous occlusion model that correctly accounts for uncertainties in position and dimensions. We refer the reader to Figure 4.2 for an illustration of the proposed concepts.

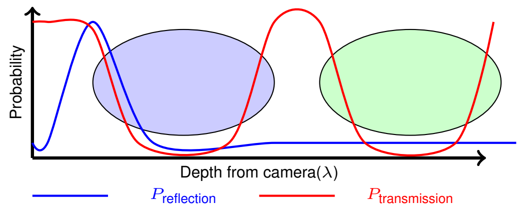


Figure 4.2: We represent objects as translucent ellipsoids, which leads to the formulation of transmission and reflection probabilities. This figure shows the reflection probability for the first object (in violet), which has high values around the camera-facing side of the object. Also, note that the transmission probability is inversely proportional to occupancy.

Occupancy model for traffic participants Intuitively, we consider traffic participants to be regions of 3D space with a high probability of occupancy. We model the uncertainty in occupancy as a translucency function, with regions more likely to be occupied by an object considered more opaque and regions more likely to be free space considered more transparent. Based on this intuition, we model objects as translucent 3D ellipsoids whose opacity is maximum at the center and falls off towards the edges. In particular, we model the occupancy at 3D location $\mathbf{p}$ corresponding to an object C_i centered at $\mathbf{m}_i$ as

$$f_{\text{occ}}(\mathbf{p}) = \mathcal{L}(\mathbf{p}; \mathbf{m}_i, \mathbf{\Sigma}_i), \tag{4.1}$$

[2]Notable exceptions exist, such as [160, 159], but we note that such models are expensive, application-specific and still discontinuous.

where $\mathcal{L}(\cdot)$ is the logistic function given by

$$\mathcal{L}(\mathbf{p}; \mathbf{m}, \mathbf{\Sigma}) = \frac{1}{1 + e^{-k(1-\zeta(\mathbf{p},\mathbf{m}))}}, \tag{4.2}$$

with $\zeta(\mathbf{p}, \mathbf{m}) = (\mathbf{p} - \mathbf{m})^\top \mathbf{\Sigma}^{-1}(\mathbf{p} - \mathbf{m})$ being the Mahalanobis distance. We set $k = 10 \ln(49)$ as the value that allows the logistic function $\mathcal{L}$ to drop to 0.98 at a distance $d = 0.9$ from the object center. The spread of the ellipsoid, determined by $\mathbf{\Sigma}_i$, depends on the dimensions of the object. Please refer to the supplementary material for the computation of $\mathbf{\Sigma}_i$ from object dimensions.

Image formation Given the above occupancy representation of the scene, a point on an object is observed in the camera when precisely two conditions are satisfied. First, the backprojected ray from the observed image pixel is transmitted through free space until it reaches the object. Second, the ray encounters an opaque enough object surface and is reflected. More formally, the probability of observation of a point $\mathbf{p}_j$ on object C_i is given by

$$P^{ij}_{\text{observation}} = P^{ij}_{\text{reflection}} P^{j}_{\text{transmission}} \tag{4.3}$$

The reflection probability ensures the presence of an object to constitute the observation, while the transmission probability allows us to model occlusions. The forms of these two functions are described next.

Reflection probability Consider a 3D point $\mathbf{p}_j$ observed in the image at pixel $\bar{\mathbf{p}}_j$. Let K be the intrinsic calibration matrix for the camera and $\hat{\mathbf{p}}_j = \dfrac{K^{-1}\bar{\mathbf{p}}_j}{\|K^{-1}\bar{\mathbf{p}}_j\|}$ be the unit vector along the backprojected ray from the camera center passing through $\bar{\mathbf{p}}_j$. Then, the probability of reflection at depth γ along the ray $\hat{\mathbf{p}}_j$, by an object C_i, is determined by the gradient of the object's occupancy function f^i_{occ} as

$$P^{ij}_{\text{reflection}}(\lambda) = \frac{1}{Z}(\max\{0, \nabla f^i_{\text{occ}}(\mathbf{p}_j)^\top \hat{\mathbf{p}}_j\})^2. \tag{4.4}$$

The gradient $\nabla f^i_{\text{occ}}(\mathbf{p}_j)$ encourages the reflection probability to be high near object boundaries, the $\max$ ensures that negative probability due to the gradient in the direction opposite to the ray is clipped off and squaring allows the function to be smooth near zero. Here, Z denotes the normalization factor. We note that in the extreme case of an object being fully opaque (that is, $\nabla f^i_{\text{occ}}(\mathbf{p}_j) \in \{0, 1\}$), the above model reverts to a (squared) Lambertian reflection. Figure 4.2 shows an example of the reflection probability.

Transmission probability Since we are modeling occupancy as transparency, we derive inspiration from optics for the modeling of translucent objects. A model for transmission of light across a distance α, through a medium of density ρ and opacity β is given by the Beer-Lambert law as

$$I(\alpha) = I_0 e^{-\beta\rho\alpha}. \tag{4.5}$$

In our formulation of scene occupancy, both opacity and density at a scene point $\mathbf{p}_j$ are encapsulated within the total occupancy function summed over all objects, $f_{\text{occ}}(\mathbf{p}_j) = \sum_i f^i_{\text{occ}}(\mathbf{p}_j)$. Further, the domain of our occupancy function $f_{\text{occ}}(\mathbf{p}_j)$ is $[0, 1]$ instead of $[0, \infty)$ for opacity β. Thus, we replace $e^{-\beta\rho}$ by the transparency function $1 - f_{\text{occ}}(\mathbf{p}_j)$ and consequently, the transmission probability over a small distance $d\gamma$ is given by

$$P^j_{\textit{transmission}}(\gamma + d\gamma) = P^j_{\textit{transmission}}(\gamma)(1 - f_{\text{occ}}(\mathbf{p}_j))^{d\gamma}. \tag{4.6}$$

Thus, for an image point $\bar{\mathbf{p}}_j$ to correspond to a 3D point $\mathbf{p}_j$ at depth γ along the backprojected ray $\hat{\mathbf{p}}_j$, the ray must be transmitted through space with the probability

$$P^j_{\text{transmission}}(\gamma) = \prod_c^\gamma (1 - f_{\text{occ}}(\gamma\hat{\mathbf{p}}_j))^{d\gamma}. \tag{4.7}$$

Here, $\prod_c^\gamma$ represents a *product integral* from c to γ, where c is the position of camera screen, considered here to be equivalent to the focal length of the camera .[3]

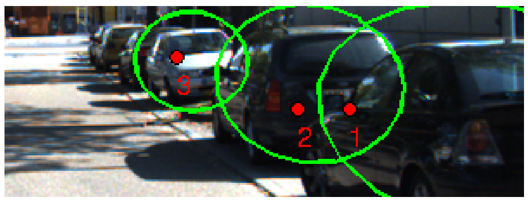
In practice, the integral for transmission probability (4.7) is difficult to compute even numerically. So we choose a parameterization in the form of a product of sigmoid functions, which is a reasonable approximation to the behaviour of the transmission probability, as follows:

$$P^j_{\text{transmission}}(\gamma) = \prod_i (1 - \mathcal{L}_u(\bar{\mathbf{p}}; \boldsymbol{\mu}_i, \boldsymbol{\Gamma}_i)\mathcal{L}_\gamma(\gamma; \nu_i)), \tag{4.8}$$

where $\mathcal{L}_u(.)$ is sigmoid in the image domain, with $\boldsymbol{\mu}_i$ and $\boldsymbol{\Gamma}_i$ representing the elliptical projection of object C_i in the image and $\mathcal{L}_\gamma(.)$ is sigmoid in the depth domain, with ν_i

[3]A product integral is a simple integral in the log domain

$$\prod_c^\gamma (1 - f_{occ}(\gamma\hat{\mathbf{p}}_j))^{d\gamma} = e^{\int_c^\gamma \ln(1 - f_{occ}(\gamma\hat{\mathbf{p}}_j))d\gamma}.$$



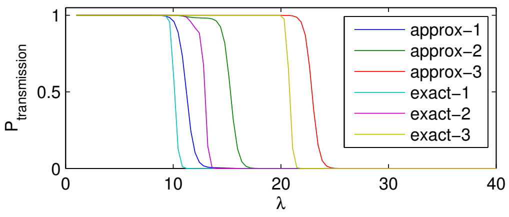


Figure 4.3: Comparisons between the approximate and exact formulations of $P^j_{\text{transmission}}(\lambda)$. The drop in the approximate version is delayed because we assume drop at the object center rather than the camera-facing face of the object.

being the mean depth of object C_i. That is,

$$\mathcal{L}_u(\bar{\mathbf{p}};\boldsymbol{\mu}_i,\boldsymbol{\Gamma}_i) = \frac{1}{1+e^{-k_u(1-(\bar{\mathbf{p}}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Gamma}_i^{-1}(\bar{\mathbf{p}}-\boldsymbol{\mu}_i))}}, \tag{4.9}$$

$$\mathcal{L}_u(\gamma;\nu_i) = \frac{1}{1+e^{-k_d(\gamma-\nu_i)}}. \tag{4.10}$$

In Figure 4.3, we compare the exact and approximate formulations of transmission probability given by (4.7) and (4.8), respectively. Note that the choice of mean depth of the object causes some deviation from the exact transmission probability. However, the shift of transmission probability anywhere through the object is still a reasonable approximation as occluded points can only lie outside the object. On the other hand, it yields significant computational savings since ray intersections with an ellipsoid are expensive to evaluate densely.

Thus, we have modeled the transmission probability to effectively capture the effect of occlusion due to all traffic participants in a scene that lie along a particular ray. We reiterate that our reflection and transmission probabilities are continuous functions, which allows us to keep the problem formulation in the continuous domain.

4.4 Unified Occlusion Models

In this section, we highlight the versatility of our occlusion modeling by demonstrating its unified application to two different problems: associating point tracks with objects and 3D object localization using objects and point tracks. Table 4.1 summarizes inputs and outputs for these problems.

4.4.1 Object-Point Association

Given 2D image points $\{\bar{\mathbf{p}}_j\}$ that are tracked between consecutive frames and a set of objects $\{C_i\}$ appearing in the frames, we aim to associate $\bar{\mathbf{p}}_j$ with C_i. Based on our continuous occlusion model in Section 4.3, the association probability $a^{ij}(\lambda)$ between point track

$\bar{\mathbf{p}}_j$ and object C_i at depth λ can be defined as

$$a^{ij}(\lambda) = P^{ij}_{\text{reflection}}(\lambda) P^{j}_{\text{transmission}}(\lambda), \tag{4.11}$$

where $P^{ij}_{\text{reflection}}(\lambda)$ and $P^{j}_{\text{transmission}}(\lambda)$ are from (4.4) and (4.8) respectively. Note that the fraction $a^{ij}(\lambda)$, although called association probability, does not capture the entire information that we have available for computing the association of point tracks to objects.

Rather, to compute the association probability between point track $\bar{\mathbf{p}}_j$ and object C_i, we should also use the reprojection error. When the association of point track $\bar{\mathbf{p}}_j$ and object C_i is correct and the point of reflection is at depth λ, the corresponding reprojection error $E^{ij}_{\text{reproj}}(\lambda)$ must be zero, otherwise the error becomes a measure of distance from the true solution. The error $E^{ij}_{\text{reproj}}(\lambda)$ can be used for associating point tracks and objects by converting it to the probability domain as

$$P^{ij}_{\text{reproj}}(\lambda) = \frac{1}{Z'} \exp(-E^{ij}_{\text{reproj}}(\lambda)), \tag{4.12}$$

where Z' is the normalization coefficient.

Using both of the evidence terms in (4.11) and (4.12), we can define the new association probability P^{ij}_{assoc}, as follows:

$$P^{ij}_{\text{assoc}} = \frac{1}{Z''} \int_c^{\infty} a^{ij}(\lambda) \exp(-E^{ij}_{\text{reproj}}(\lambda)) d\lambda, \tag{4.13}$$

where Z'' is the new normalization coefficient.

Once we have computed the association probability P^{ij}_{assoc} for every pair of point tracks and objects, we can assign each point track to the object with the highest association probability. The point tracks having very small association probabilities are assigned to the background.

	Symbol	Description
Input	$\bar{\mathbf{p}}_j(t)$	2D feature track j at time t
	$\bar{\mathbf{B}}^i(t)$	2D detection bounding box of object C_i at time t
Initialization with [129]	$\mathbf{p}^c(t)$	Position of camera at time t
	$\boldsymbol{\omega}^c(t)$	Orientation of camera at time t
	$\mathbf{m}^i_0(t)$	Initial position of object C_i at time t
	$\boldsymbol{\omega}^i_0(t)$	Initial orientation of object C_i at time t
	$\mathbf{B}^i_0$	Initial 3D dimensions of object C_i
Output	P^{ij}_{assoc}	Probability of assigning feature track j to object C_i
	$\mathbf{m}^i(t)$	Position of object C_i at time t
	$\boldsymbol{\omega}^i(t)$	Orientation of object C_i at time t
	$\mathbf{B}^i$	3D dimensions of object C_i

Table 4.1: Notation of inputs and outputs for object-point association and 3D object localization. Note that object dimensions are independent of time.

In contrast to the principled approach above, a heuristic baseline may simply assign a point track to the detection bounding box enclosing it (and the background if outside all bounding boxes). For regions where bounding boxes overlap, it may assign point tracks to the object that has the smallest mean depth among the competing bounding boxes. As we demonstrate in our experiments, such heuristics are sub-optimal compared to using (4.13) from our occlusion model.

4.4.2 3D Object Localization

In this section, we exploit our continuous occlusion model for another application, namely, 3D object localization in road scenes, which further demonstrates its versatility. Given a set of 2D tracked feature points $\{\bar{\mathbf{p}}_j(t)\}$ and 2D detection bounding boxes $\{\bar{\mathbf{B}}^i(t)\}$ at frame t, the goal is to localize 3D traffic participants. In particular, for each traffic participant, we wish to estimate its position $\mathbf{m}^i(t)$ and orientation $\boldsymbol{\omega}^i(t)$ on the ground plane and its 3D dimensions $\mathbf{B}^i(t)$. Please refer to Table 4.1 for a summary of inputs and outputs.

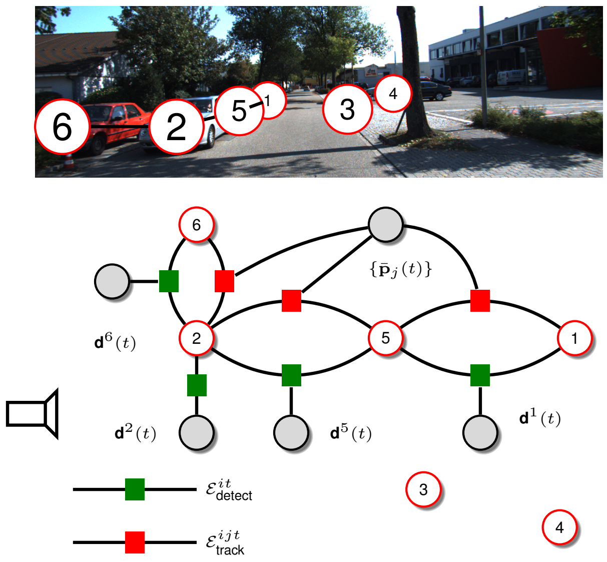

6
2
5
1
3
4
6
$\{\bar{\mathbf{p}}_j(t)\}$
$\mathbf{d}^6(t)$
2
5
1
$\mathbf{d}^2(t)$
$\mathbf{d}^5(t)$
$\mathbf{d}^1(t)$
$\mathcal{E}^{it}_{\text{detect}}$
3
$\mathcal{E}^{ijt}_{\text{track}}$
4

Figure 4.4: (Top) A sample road scene with occlusions, where the unknowns of each object are modeled as random variables. (Bottom) The graphical model corresponding to the above frame. In particular, the numbered nodes denote the unknown state variables of each object (position, orientation, and dimensions), the shaded nodes are observed variables (detection bounding boxes and point tracks), and the colored squares represent various energies that capture object-object interactions.

We construct a graphical model for representing relationships among objects, as well as between objects and point tracks. Figure 4.4 illustrates an example of the graph and energies. The negative log likelihood is decomposed as follows:

$$
\begin{aligned}
-\log P(\{\mathbf{m}^i(t), \boldsymbol{\omega}^i(t), \mathbf{B}^i(t)\} | \{\bar{\mathbf{p}}_j(t)\}, \{\bar{\mathbf{B}}^i(t)\}) = \\
-\tilde{Z} + \sum_{t=s_i}^{e_i} \lambda_{\text{track}} \mathcal{E}^{ijt}_{\text{track}} + \\
\sum_{t=s_i}^{e_i} \sum_{i=1}^{N} \left(\lambda_{\text{detect}} \mathcal{E}^{it}_{\text{detect}} + \lambda_{\text{dyn}} \mathcal{E}^{it}_{\text{dyn}} + \lambda_{\text{size}} \mathcal{E}^{it}_{\text{size}} \right) \quad ,
\end{aligned}
$$

where $\mathcal{E}^{ijt}_{\text{track}}$ and $\mathcal{E}^{it}_{\text{detect}}$ reason about image observations such as point tracks and bounding boxes, while $\mathcal{E}^{it}_{\text{dyn}}$ and $\mathcal{E}^{it}_{\text{size}}$ impose smoothness constraints and size priors respectively. Here, λ_{track}, λ_{detect}, λ_{dyn}, λ_{size} are energy weights, N is the number of objects in the sequence, s_i and t_i are respectively the starting and ending frames of object C_i, and $\tilde{Z}$ is the normalization coefficient. Next, we present our unified continuous occlusion modeling for both point track and bounding box energies. Due to space constraints, we present the details of other energies in the supplementary material.

Continuous point track energy with occlusion Let $g^i(t)$ be the pose of object C_i at time t in world coordinates, which is computed using the camera pose at time t and the relative pose of object C_i with respect to the camera at time t. We denote $\pi_{g^i(t)}(.)$ and $\pi^{-1}_{g^i(t-1)}(.)$ as the forward and inverse projection functions that project a 3D point to the 2D image and vice versa. Then, the reprojection error for 2D point $\bar{\mathbf{p}}_j(t)$ with hypothesized depth λ, is given by

$$E^{ij}_{\text{reproj}}(\lambda) = \left\| \bar{\mathbf{p}}_j(t) - \pi_{g^i(t)}\left(\pi^{-1}_{g^i(t-1)}\left(\bar{\mathbf{p}}_j(t-1), \lambda\right)\right)\right\|^2 . \qquad (4.14)$$

Note that the inverse projection $\pi^{-1}_{g^i(t)}(.)$ depends on both the 2D point $\bar{\mathbf{p}}_j(t)$ and the unknown depth λ. Also note that the inverse projection relies on the object pose at time $t-1$ while the forward projection relies on the object pose at time t, which can be different.

For an object C_i, let $\{g(t)\}_i$ be the poses of all occluding objects at time t (inclusive of object C_i) and $\{\mathbf{B}\}_i$ be their corresponding 3D dimensions. Then, we model the continuous point track energy with explicit occlusion reasoning as the expected reprojection error over the association probability

$$\mathcal{E}^{ijt}_{\text{track}}(\{g(t)\}_i, \{g(t-1)\}_i, \{\mathbf{B}\}_i) = \sum_{i=1}^{N}\sum_{j=1}^{M}\int_0^\infty a^{ij}(\lambda) E^{ij}_{\text{reproj}}(\lambda) d\lambda, \qquad (4.15)$$

where N and M are, respectively, the number of objects and points and $a^{ij}(\lambda)$ is the association probability of point $\bar{\mathbf{p}}_j(t)$ with object C_i at depth λ, given by (4.11).

Continuous bounding box energy with occlusion Object detection is usually followed by non-maximal suppression that results in discarding similar bounding boxes. When we are jointly optimizing detections with other cues, it is usually not desirable to use a single bounding box. To retain the entire detection output while maintaining the continuous form of our energies, we approximate the distribution of detection scores with a multi-modal sum of Gaussian-like logistic functions. In particular, let 2D bounding box $\bar{\mathbf{B}}^i(t)$ be parameterized as a 4D vector $[x_{\text{min}}, y_{\text{min}}, x_{\text{max}}, y_{\text{max}}]^\top$. We fit a parametric function to the detection scores, of the form

$$S(\bar{\mathbf{B}}^i(t)) = \sum_k A_k \exp\left(-\boldsymbol{\epsilon}^i_k(t)^\top \boldsymbol{\Lambda}^{-1}_k \boldsymbol{\epsilon}^i_k(t)\right), \qquad (4.16)$$

where A_k is an amplitude and $\boldsymbol{\epsilon}^i_k(t) = \bar{\mathbf{B}}^i(t) - \boldsymbol{\mu}_k$, with $\boldsymbol{\mu}_k$ the mean and $\boldsymbol{\Lambda}_k$ the covariance. We use a non-linear solver to minimize the above, with initialization from non-maximal suppressed outputs. The optimization is constrained by the symmetry and positive definiteness of $\mathbf{\Lambda}_k$, $x_{\text{max}} \geq x_{\text{min}}$ and $y_{\text{max}} \geq y_{\text{min}}$.

Detection scores with occlusion reasoning With our model of $P^j_{\text{transmission}}(\lambda)$ described in Section 4.3, we compute the probability of a point $\bar{\mathbf{p}}$ in the image to be occluded, assuming the point is on object C_i with mean depth ν_i, as

$$\Theta_i(\bar{\mathbf{p}}, \nu_i) = 1 - P^j_{\text{transmission}}(\nu_i, \bar{\mathbf{p}}). \tag{4.17}$$

If a portion of the proposed detection bounding box is known to be occluded, one would like to decrease the confidence in the detection score about the localization of that end of the object. Assuming that occlusions are more likely on the boundaries of the detection bounding box, we wish to decrease the confidence on the mean detection boundaries around the occluded boundaries. To re-model detection scores scaled by continuous occlusion, we sample $\Theta_i(\bar{\mathbf{p}}, \nu_i)$ at the hypothesized detection boundaries from the Gaussian mixture model (GMM) $S(.)$ in (4.16) and augment the detection boundary covariance matrix by $\mathcal{P}_i = \rho_i \rho_i^\top$, where $\rho_i = \Theta_i(\mathbf{u}, \nu_i)$. The new covariance matrix for the detection scores is given by $\mathbf{\Lambda}'_k = \mathcal{P}_i + \mathbf{\Lambda}_k$ for all k. The detection score GMM $S'(.)$ with explicit occlusion reasoning is given by replacing the covariance matrix, as follows:

$$S'(\bar{\mathbf{B}}^i(t)) = \sum_k A_k \exp\left(-\boldsymbol{\epsilon}^i_k(t)^\top \mathbf{\Lambda}'^{-1}_k \boldsymbol{\epsilon}^i_k(t)\right). \tag{4.18}$$

The energy of detection scores is simply taken to be the inverse of the above detection score, that is,

$$\mathcal{E}^{it}_{\text{detect}}(\{g^i(t)\}_i, \{\mathbf{B}^i\}_i) = \frac{1}{S'(\bar{\mathbf{B}}^i(t))}. \tag{4.19}$$

Inference on graphical model We apply the Metropolis-Hastings method [79] to perform inference on the graphical model. Since we optimize over continuous variables, we use the Gaussian distribution as the proposal function. We choose this over alternatives such as block-coordinate descent since they are slower in our experiments.

4.5 Experiments

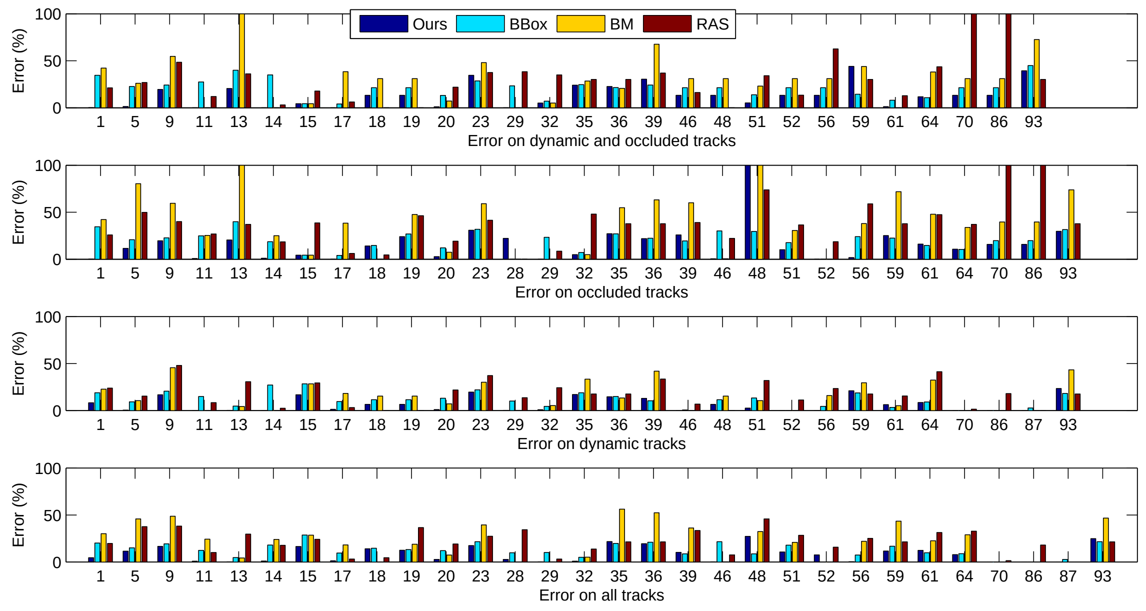


Figure 4.5: Association errors on different sets of input point tracks. Numbers on the x-axis represent sequence numbers in the KITTI raw dataset. Errors are in terms of average fractions of foreground points incorrectly associated to objects per sequence.

Point tracks	Ours	BBox	BM	RAS
Dynamic & occluded	**13.2**	21.3	30.9	30.1
Occluded	**15.7**	19.8	39.5	37.8
Dynamic	**6.6**	11.4	15.3	17.7
All	**8.6**	12.6	21.9	21.5

Table 4.2: Mean association errors on different sets of input point tracks over all sequences. Errors are in terms of average fractions of foreground points incorrectly associated to objects per sequence.

In this section, we benchmark our continuous occlusion model for point-to-object association against the baseline method using detection bounding boxes and state-of-the-art methods for motion segmentation [11, 115]. We then show how the proposed model may be applied for 3D object localization in road scenes. For our experiments, we use 35 sequences of the KITTI raw dataset [37], which are recorded under a variety of driving conditions and include 10,088 frames and 634 object tracks in total.

4.5.1 Association Experiments

Setup We first perform the association experiment that compares the accuracy of point-to-object association using our proposed model against a heuristic baseline and state-of-the-art motion segmentation methods. The detection bounding box baseline method (BBox) is as described at the end of Section 4.4.1. For motion segmentation, we use robust algebraic segmentation with hybrid perspective constraints (RAS) [115] and spectral clustering with point track spatial affinities (BM) [11].

For each sequence, the methods of [31] and [16] are used for computing detection bounding boxes and object tracklets, respectively. We then apply [154] to extract point

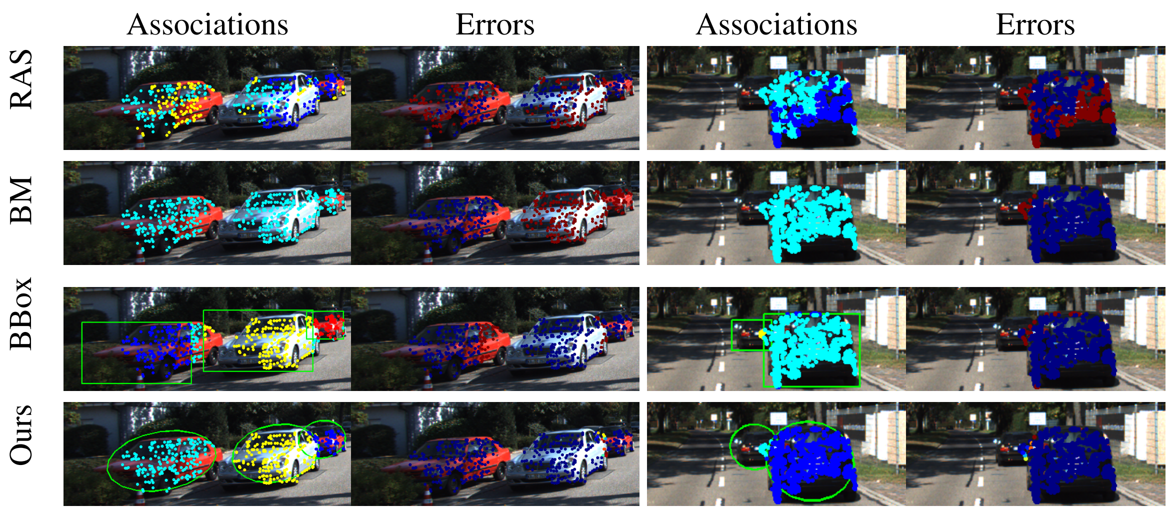


Figure 4.6: Qualitative results of the association experiment. The "Associations" columns show the point track assignments to appropriate objects. Each color represents a different object to which point tracks can be associated to. The "Errors" columns show the probabilistic errors in association: low error points are in blue while high error points are in red. Note that our method changes smoothly at the object boundaries with intermediate probabilities, while the baseline method has merely 0 and 1 errors.

tracks. Note that our method can handle occlusions in both static and dynamic scenes, but motion segmentation focuses on dynamic scenes. For a complete evaluation, we organize the point tracks into four sets: all point tracks, occluded point tracks, dynamic point tracks and dynamic as well as occluded point tracks. The parameters (position, orientation, and dimensions) of all objects (cars) estimated by the method of [129] are provided to our method (for computing association probability) and the baseline BBox method (for depth ordering). The number of objects is known a priori in our model (from object tracking [16]) and is also provided to other methods such as BBox and RAS.

Results Figure 4.5 shows the association errors – the percentages of point tracks incorrectly assigned to objects – for all methods on the four sets of input point tracks, for each sequence. The mean results over all sequences are summarized in Table 4.2. From Figure 4.5, our method is usually the most accurate among all methods, leading to the best mean error on all sets of input point tracks in Table 4.2, which is followed by the bounding box baseline method. This clearly shows the advantage of our continuous occlusion model over the simple baseline method for resolving occlusions. RAS and BM often have the highest errors in Figure 4.5, thus, the highest mean errors on all sets of input point tracks in Table 4.2.

Method	t	dim
Point cloud fitting	6.87	4.02
Initialization by [129]	5.61	3.23
$\mathcal{E}^{ijt}_{\text{trackNoOcc}} + \mathcal{E}^{it}_{\text{detectNoOcc}} + \mathcal{E}^{it}_{\text{size}} + \mathcal{E}^{it}_{\text{dyn}}$	3.95	1.72
$\mathcal{E}^{ijt}_{\text{trackNoOcc}} + \mathcal{E}^{it}_{\text{detect}} + \mathcal{E}^{it}_{\text{size}} + \mathcal{E}^{it}_{\text{dyn}}$	4.81	2.16
$\mathcal{E}^{ijt}_{\text{track}} + \mathcal{E}^{it}_{\text{detectNoOcc}} + \mathcal{E}^{it}_{\text{size}} + \mathcal{E}^{it}_{\text{dyn}}$	4.05	**1.59**
$\mathcal{E}^{ijt}_{\text{track}} + \mathcal{E}^{it}_{\text{detect}} + \mathcal{E}^{it}_{\text{size}} + \mathcal{E}^{it}_{\text{dyn}}$	**3.24**	2.16

Table 4.3: Localization experiment results with different combinations of energies. We report translation error (t) and dimension error (dim) in meters per car. Yaw angles for static objects are not optimized by our model. These experiments use the set of occluded tracks to demonstrate the effect of our modeling.

More importantly, both RAS and BM rely on motions of objects for clustering point tracks, therefore they cannot work well with static point tracks (for example, point tracks that belong to parked cars). This fact can be observed in Table 4.2, where there are large differences in the mean errors of both methods on data containing static point tracks (rows 2 and 4) and data consisting of dynamic point tracks only (rows 1 and 3). In contrast, our method and the baseline method are relatively independent of object motions, resulting in smaller performance gaps. Further, our method also outperforms motion segmentation on dynamic objects (row 3), which shows the effect of detection bounding boxes and by a more significant margin when occlusions are present (row 1), which shows the effect of our occlusion modeling.

Qualitative comparisons of point track associations from various methods are shown in Figure 4.6. We note the low errors using our occlusion model and the smooth transition of assignment across object boundaries.

4.5.2 Localization Experiments

We report errors in translation and dimension estimates, measured in meters per car, in Table 4.3. The average depth of cars in the dataset is approximately 20 meters. We compare four combinations of energies against the initialization using [129] and a simple baseline which fits a 3D cuboid on the 3D point cloud reconstructed using SFM within detection bounding boxes in consecutive frames (for unobservable dimensions, such as when only the back of a car is visible, we rely on 3D size priors). The energy $\mathcal{E}^{ijt}_{\text{trackNoOcc}}$ represents the point track energy without accounting for occlusions, that is, we model $\mathcal{E}^{ijt}_{\text{track}}$ in the absence of $a^{ij}(\lambda)$. Similarly, $\mathcal{E}^{it}_{\text{detectNoOcc}}$ is the bounding box energy without the modification of $\mathbf{\Lambda}_k$ that accounts for occlusion. We use $\lambda_{\text{track}} = 1$, $\lambda_{\text{detect}} = 1$, $\lambda_{\text{dyn}} = 10$, $\lambda_{\text{size}} = 7$. Please refer to the supplementary material for a detailed list of parameter settings.

From Table 4.3, the baseline method has the highest errors, which is likely due to lack of point tracks and incorrect point-to-object associations (using detection bounding boxes). Moreover, minimizing different combinations of energies yields lower errors than the initialization with [129], which shows the advantage of our energy minimization. Finally, we observe that the use of the continuous occlusion model improves the localization accuracy in terms of the translation error, which is the most significant metric affected by all cues. Occlusion modeling for detection increases dimension error since we explicitly allow greater uncertainty in occluded edges of the bounding box. Note that none of our energies optimize yaw angles for static objects, which can be handled in practice through either the detector orientation or external information such as lane geometry.

4.6 Conclusions and Future Work

We have presented a theoretically novel continuous model for occlusion reasoning in 3D. A key advantage is its physical inspiration that lends flexibility towards occlusion reasoning for varied elements of scene understanding, such as point tracks, object detection bounding boxes and detection scores. We demonstrate unified modeling for different applications such as object-point track associations and 3D localization. Our occlusion model can uniformly handle static and dynamic objects, which is an advantage over motion segmentation methods for object-point association. A challenge is that inference for 3D localization is currently slow, requiring a few minutes per window of frames, which prevents exhaustive cross-validation for tuning of weights. Our future work will explore speeding up the inference, for example, by approximating the graph with a tree using the Chow-Liu method [17], which will allow belief propagation for fast inference. Another direction for future work is to replace a single ellipsoid by a set of spheres for modelling a translucent object [118], which will better capture object boundary and appearance while remaining a continuous model.

Acknowledgments This work was part of V. Dhiman's internship at NEC Labs America, in Cupertino. V. Dhiman and J. J. Corso were also supported by NSF NRI IIS 1522904.

CHAPTER 5

Deep reinforcement learning for navigation[1]

5.1 Introduction

Navigation remains a fundamental problem in mobile robotics and artificial intelligence ([127, 30]). The problem is classically addressed by separating the task of navigation into two steps, exploration and exploitation. In the exploration stage, the environment is represented as some kind of *map*. In the exploitation stage, the map is used to *plan a path* to a given destination based on some optimality criterion. This classical approach has been quite successful in navigation using a variety of sensors. However, navigation in general unstructured environments, especially with texture-less [152], transparent and reflective surfaces [71], remains a challenge.

Recently, end-to-end navigation methods—which attempt to solve the navigation problem without breaking it down into separate parts of mapping and path-planning—have gained traction. With the recent advances in Deep Reinforcement Learning (DRL), these end-to-end navigation methods, such as [90, 126, 73, 89, 102], forego decisions about the details that are required in the intermediate step of mapping. The potential for simpler yet more capable methods is rich; for example, the resulting trained agents can potentially optimize the amount of map information required for navigation tasks. One such algorithm by [89] has shown promise in exploring and finding the goal efficiently within complex environments. Notably, this is done using only monocular first-person views.

Despite such potential advances, DRL-based navigation remains a relatively unexplored field with its own limitations. The black-box nature of these methods make them difficult to study, and the patterns captured by the methods are not well understood. Recent work analyzing neural networks has shown that deep learning-based object detection methods can be easily fooled by introducing noise that is imperceptible to humans ([100]); this level

[1]This material originally appeared as: V. Dhiman, S. Banerjee, B. Griffin, J. Mark Siskind, and J. J. Corso. A critical investigation of deep reinforcement learning for navigation. *CoRR*, abs/1802.02274, 2018

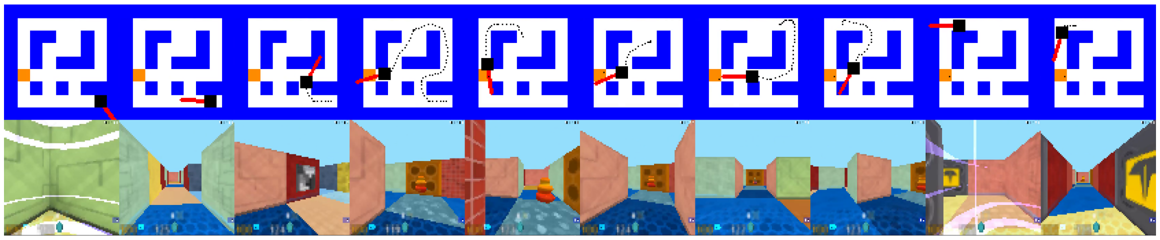


Figure 5.1: Snapshots of the path taken by the agent while evaluating the model trained on the same random map with random goal and random spawn. The first row shows the top view of the robot moving through the maze with the goal location marked orange, the agent marked black and the agent's orientation marked red. The second row shows the first person view, which, besides reward, is the only input available to the agent and the top view is available only for human analysis.

of sensitivity motivates why it is particularly important to analyze DRL methods across a wide variety of experiments: we need to understand their strengths and limitations.

In this work, we develop a better understanding of recent DRL-based methods. In particular, we thoroughly explore and analyze the state-of-the-art [89] methods across hundreds of maps with increasing difficulty levels. We set up the environment as a randomly generated map, as shown in Fig 5.1, with an agent and a goal. The agent is provided only with the first-person view and is tasked to find the goal as many times as possible within a fixed amount of time, re-spawning its location each time it reaches the goal. We train and evaluate the algorithms with increasing difficulty. In the easiest stage, we keep the goal location, spawn location and map constant over the training and testing. We call this set up *static goal, static spawn, and static map*. To increase the difficulty, we incrementally randomize the spawn locations, goal locations and map structures until all three are random. We discuss the design of experiments in Section 5.4.1 in more detail.

[89] do train and test their algorithms with randomized goals and spawns and show that their algorithm is able to exploit the knowledge of the goal location at evaluation time to maximize reward. However, following training and testing on constant map structures, this state-of-the-art result is shown to be successful on only one map, which brings into question the repeatability of the results. It is also unclear whether these results generalize to unseen maps.

Although disjoint training and testing sets are standard practice in machine learning, to the best of our knowledge, we are the first to evaluate any DRL-based navigation method on maps with unseen structures. We expand on the analysis in [89] to address its limitations and ask whether DRL-based algorithms such as NavA3C+D_1D_2L perform any mapping followed by shortest path planning. Our experiments show no evidence of mapping in cases where algorithms are evaluated on unseen maps and no evidence of optimal path planning, even when the map is constant and only the goal is randomized.

To better understand navigation, we compute attention-maps for models to show which portions of the input image are being used. We find that the models discard most of the image information, focusing attention on a small band in the middle of the image except around junctions, in which case the attention is distributed evenly throughout the image.

These findings result from training and testing on multiple maps that were randomly selected from a set of 1100 randomly generated maps. We provide experimental results on ten randomly selected maps and a testing set of 100 unseen maps to ensure results are independent of map choice. We will make our code and data available online [2].

5.2 Related Work

Localization and mapping Localization and mapping for navigation is a classic problem in mobile robotics and sensing. [127] introduced the idea of propagating spatial uncertainty for robot localization while mapping, and [30] popularized Occupancy Grids. In the three decades since these seminal works, the field has exploded with hundreds of algorithms for many types of sensors (e.g., cameras, laser scanners, sonars and depth sensors). These algorithms vary by how much detail is captured in their respective maps. For example, topological maps, like [65], aim to capture as little information as possible while occupancy grid maps, [30], aim to capture metrically accurate maps in resolutions dependent upon the navigation task.

All these approaches require significant hand-tuning for the environment, sensor types and navigation constraints of the hardware. In contrast, end-to-end navigation algorithms optimize the detail of map storage based on the navigation task at hand, which makes them worth exploring.

Deep reinforcement learning DRL gained prominence recently when used by [91, 93] to train agents that outperform humans on Atari games; agents that trained using only the games' visual output. More recently, DRL has been applied to end-to-end navigation ([102, 89, 15]). It is common for agents to be trained and tested on the same maps with the only variation being the agent's initial spawn point and the map's goal location ([89, 157, 66]).

In contrast, [102] test their algorithm on random unseen maps, but their agents are trained to choose between multiple potential goal locations based on past observations. The episodes end when the agent collects the goal, so there is no requirement for the algorithm to store map information during their exploration. Thus, their agents decide to avoid a goal of a particular color while seeking other colors rather than remembering the path to the goal. [15] test their method on unseen maps in the VizDoom environment, but only vary

[2] github.com/umrobotslang/does-drl-learn-to-navigate

the maps with unseen textures. Thus, their agents are texture invariant, but train and test on maps with the same geometric structure.

In this work, we extend the study of these methods in a more comprehensive set of experiments to address the question of whether DRL-based agents remember enough information to obviate mapping algorithms or may in fact need to be augmented with mapping for further progress.

5.3 Background

Our problem formulation is based on the work of [89]. For completeness, we summarize the technical setup here. For additional details regarding the setup, we refer the reader to [90, 89].

The problem of navigation is formulated as an interaction between an environment and an agent. At time t the agent takes an action $a_t \in \mathcal{A}$ and observes observation $z_t \in \mathcal{O}$ along with a reward $r_t \in \mathbb{R}$. We assume the environment to be a Partially Observable Markov Decision Process (POMDP). In a POMDP, the future state of the environment, $s_{t+1} \in \mathcal{S}$, is conditionally independent of all the past states, $s_{1:t-1}$, given the current state s_t. It is further assumed that z_t and r_t are independent of previous states given current state s_t and last action a_{t-1}. Formally, a POMDP is defined as a six tuple $(\mathcal{O}, C, \mathcal{S}, \mathcal{A}, T, R)$ that is composed of an observation space $\mathcal{O}$, an observation function $C(s_t, a_t) \rightarrow z_t$, a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $T(s_t, a_t) \rightarrow s_{t+1}$ and a reward function $R(s_t, a_t) \rightarrow r_{t+1}$. For our problem setup, the observation space $\mathcal{O}$ is the space of an encoded feature vector that is generated from input image along with previous action and reward. Action space $\mathcal{A}$ contains four actions: rotate left, rotate right, move forward and move backward and reward function R is defined for each experiment so that reaching the goal location leads to high reward with auxiliary rewards to encourage certain kinds of behavior.

For DRL algorithms, the state space $\mathcal{S}$ is not hand tuned, but it is modeled as a vector of floats. Additionally, instead of modeling observation function $C(s_t, a_t) \rightarrow z_t$ and $T(s_t, a_t) \rightarrow s_{t+1}$, a combined transition function $T_c(s_t, z_t, a_t, r_t; \theta_T) \rightarrow s_{t+1}$ is modeled to estimate the next state s_{t+1} and directly take previous observation and reward into account. For policy-based DRL a policy function $\pi(a_{t+1}|s_t, z_t, a_t, r_t; \theta_\pi) \rightarrow \pi_t(a_{t+1}; \theta_\pi)$ and a value function $V(s_t, z_t, a_t, r_t; \theta_V) \rightarrow V_t(\theta_V)$ are also modeled. All three functions T_c, π_t and V_t share most of the parameters such that $\theta_T \subseteq \theta_\pi \cap \theta_V$

The DRL objective is to estimate unknown weights $\theta = \theta_T \cup \theta_\pi \cup \theta_V$ that maximizes the expected future reward $R_t = \sum_{k=t}^{t_{end}-t} \gamma^{k-t} r_k$ (where γ is the discount factor) and is

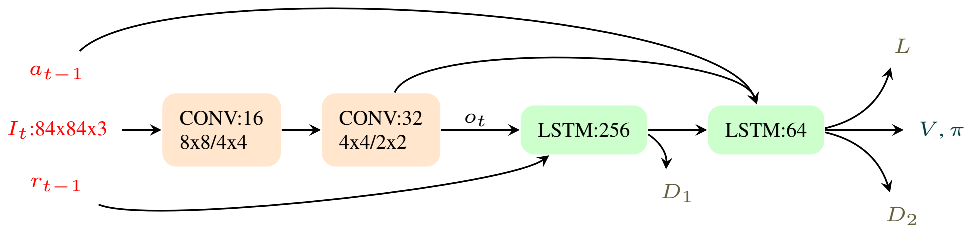


Figure 5.2: Modified NavA3C+D$_1$D$_2$L ([89]) architecture. The architecture is has three inputs the current image I_t and previous action a_{t-1} and previous reward r_{t-1}. As shown by [89], the architecture improves upon vanilla A3C architecture by using auxiliary outputs of loop-closure signal L and predicted depth D_1 and D_2. Since we use a smaller action space than [89] and our agent moves with constant velocity, we do not use velocity at previous time step as input signal.

expressed as

$$\theta^* = \arg\max_{\theta} \mathbb{E}[R_t]\,, \tag{5.1}$$

where $\mathbb{E}[.]$ denotes the expected value.

Asynchronous Advantage Actor-Critic In this work, we use the policy-based method called Asynchronous Advantage Actor-Critic (A3C) ([90]), which allows weight updates to happen asynchronously in a multi-threaded environment. It works by keeping a "shared and slowly changing copy of target network" that is updated every few iterations by accumulated gradients in each of the threads. The gradients are never applied to the local copy of the weights; instead, a local copy of weights is periodically synced from the shared copy of target weights. The gradient for the weight update is proportional to the product of *advantage*, $R_t - V_t(\theta_V)$, and *characteristic eligibility*, $\nabla_{\theta_\pi} \ln \pi_t(a_{t+1};\theta_\pi)$ ([147]), which update the weights as

$$\theta_\pi \leftarrow \theta_\pi + \sum_{t\in\text{episode}} \alpha_\pi \nabla_{\theta_\pi} \ln \pi_t(R_t - V_t(\theta_V)) \quad \text{and} \quad \theta_V \leftarrow \theta_V + \sum_{t\in\text{episode}} \alpha_V \frac{\partial (R_t - V_t(\theta_V))^2}{\partial \theta_V}\,. \tag{5.2}$$

For additional details of the A3C algorithm, we refer the reader to [90].

NavA3C+D$_1$D$_2$L In this work, we use the NavA3C+D$_1$D$_2$L architecture as proposed by [89], which builds modifying the network architecture to have two LSTMs and with auxiliary outputs of depth predictions along with loop-closure predictions. The schematic of the architecture is shown in Fig 5.2. The architecture has three inputs: the current image I_t, previous action a_{t-1} and previous reward r_{t-1}. As shown by [89], the architecture improves

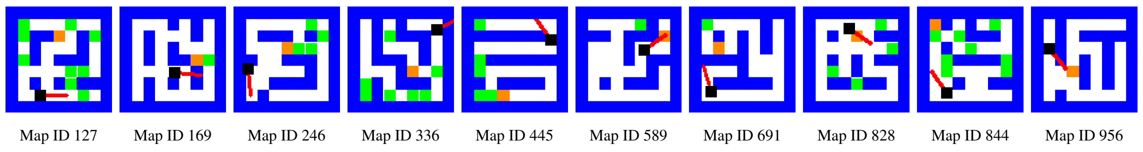


Figure 5.3: The ten randomly chosen mazes for evaluation. We generate 1100 random mazes and choose ten to evaluate our experiments that require testing and training on the same maps.

upon vanilla A3C architecture by optimizing predictions for the auxiliary outputs of loop closure signal L and predicted depth D_1 and D_2. Since we use a smaller action space than [89] and our agent moves with constant velocity, we do not use velocity at the previous time step as an input signal.

5.4 The DRL Navigation Challenge

Since deep reinforcement learning algorithms need millions of iterations to train, in the absence of thousands of robotic replicas like [73], we evaluate the algorithms on a simulated environment. We use the same game engine as [89], called Deepmind Lab ([7]). The game is setup such that an agent is placed within a randomly generated maze containing a *goal* at a particular location. On reaching the goal, the agent *re-spawns* within the same maze while the goal location remains unchanged. Following [89], we scatter the maze with randomly placed smaller apple rewards (+1) to encourage initial explorations and assign the goal a reward of +10. The agent is tasked to find the goal as many times as possible within a fixed amount of time (1200 steps for our experiments), re-spawning within the maze, either statically or randomly, each time it reaches the goal.

Unlike [89], we include a small wall penalty (-0.2) that pushes the agent away from the wall. The wall penalty is useful to prevent agents from moving along the walls, thereby discarding vision input for exploration. We also use a discrete 4-action space (move forward/backward, rotate left/right)which is different from the 8-action space one used by [89]. A smaller action space helps the algorithm train faster while achieving similar reward values.

We generate 1100 random maps using depth-first search based maze generation methods. More information on maze generation can be found in the appendix. Of the first 1000 maps, 10 are randomly selected for our static-map experiments (Fig. 5.3). For our unseen map experiments, agents are trained on increasing subsets of the first 1000 maps and tested on the remaining 100. Unlike [89] and similar to [15], we use randomly textured walls in our mazes so that the policies learned are texture-independent.

5.4.1 Experiments

We evaluate the NavA3C+D_1D_2L algorithm on maps with 5 stages of difficulty. While the algorithm works smoothly on the easier stages, it does not perform better than wall-following methods on the hardest stage. We propose these experiments as a 5-stage benchmark for all end-to-end navigation algorithms.

1. **Static goal, static spawn, and static map** To perform optimally on this experiment, the agent needs to find and learn the shortest path at training time and repeat it during testing.
2. **Static goal, random spawn and static map** This is a textbook version of the reinforcement learning problem, especially in grid-world [134], with the only difference being that the environment is partially observable instead of fully observable. This problem is more difficult than Problem 1 because the agent must find an optimal policy to the goal from each possible starting point in the maze.
3. **Random goal, static spawn, and static map** In this setup, we keep the spawn location and the map fixed during both training and testing but choose a random goal location for each episode. Note that the goal location stays constant throughout an episode. The agent can perform well on this experiment by remembering the goal location after it has been discovered and exploiting the information to revisit the goal faster.
4. **Random goal, random spawn, and static map** In this version of the experiment both the spawn point and the goal location is randomized. To perform optimally, the agent must localize itself within the map in addition to being able to exploit map-information.

   This is the problem that is addressed by [89] with limited success. They evaluate this case on two maps and report Latency $1 :> 1$ to be greater than 1 in one of the two maps. We evaluate the same metric on ten other maps.
5. **Random goal, random spawn, and random map** We believe that any proposed algorithms on end-to-end navigation problems, should be evaluated on unseen maps. To our knowledge, this is the first paper to do so in the case of deep reinforcement learning based navigation methods. We train agents to simultaneously learn to explore 1, 10, 100, 500 and 1000 maps and test them on the same 100 unseen maps. The relevant results can be found in Fig 5.5 and discussed in Section 5.5.

The comparative evaluation of the different the stages of this benchmark are shown in Fig 5.4 and expanded upon in the Section 5.5.

5.4.2 Evaluation Metrics

We evaluate the algorithms in terms of three metrics: rewards, *Latency* $1 :> 1$ and *Distance-inefficiency*. Following [89], we report *Latency* $1 :> 1$, a ratio of the time taken to hit the goal for the first time (exploration time) versus the average amount of time taken to hit goal subsequently (exploitation time). The metric is a measure of how efficiently the agent exploits map information to find a shorter path once the goal location is known. If this ratio is greater than 1, the agent is doing better than random exploration and the higher the value, the better its map-exploitation ability. Note that the metric is meaningful only when the goal location is unknown at evaluation time.

Distance-inefficiency is defined to be the ratio of the total distance traveled by the agent versus the sum of approximate shortest distances to the goal from each spawn point. The metric also disregards goals found during exploitation time as the agent must first find the goal before it can traverse the shortest path to it. Note that the shortest distance between the spawn and goal locations is computed in the top-down block world perspective and hence is only an approximation.

While the Latency $1 :> 1$ measures the factor by which planned path to the goal is shorter than the exploration path, the Distance-inefficiency measures the length of this path with respect to the shortest possible path.

5.5 Results and Analysis

In this section we discuss the results for experiments as discussed in Section 5.4.1 over ten randomly chosen maps shown in Fig 5.3. The results are shown in Fig 5.4 and are discussed for each case below.

1. **Static goal, static spawn, static maze** For this case, the reward is consistently high, and Distance-inefficiency is close to 1 with small standard deviations implying the path chosen is the shortest available. Please note that Latency $1 :> 1$ should be close to 1 for the static goal case because the goal location is known at training time.

2. **Static goal, random spawn, static map** Again, note that Distance-inefficiency is close to 1 implying that when the goal is found, the shortest path is traversed. This is because the agent can learn the optimal policy for the shortest path to the goal at training time.

3. **Random goal, static spawn, static map** In this case, the mean of the Latency $1 :> 1$ is more than 1 show that in general the agent is able to exploit map information.

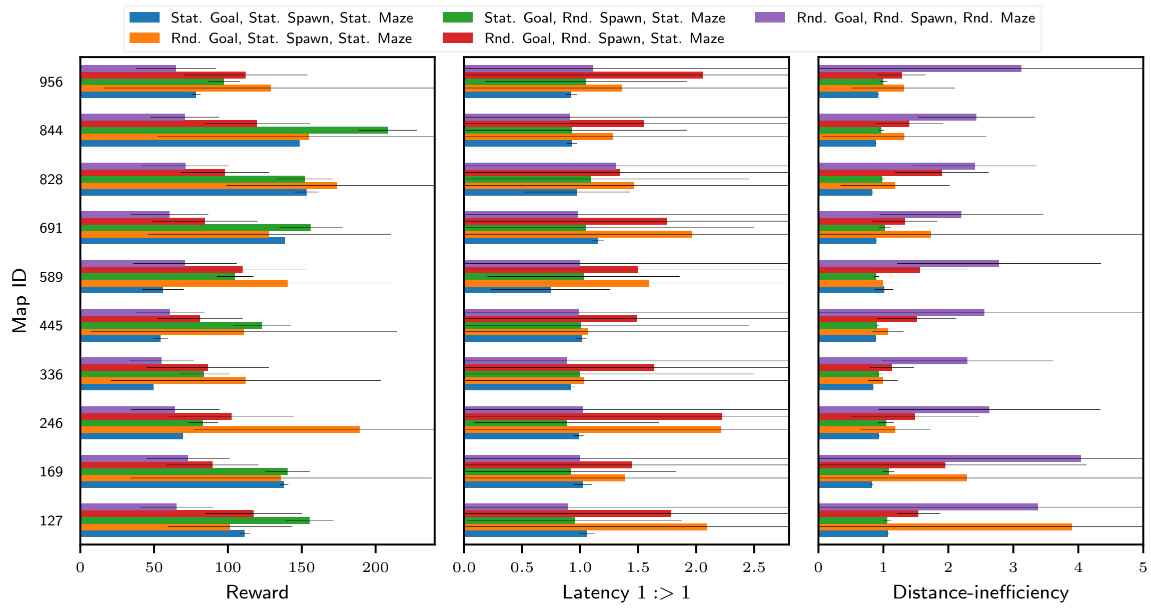


Figure 5.4: We evaluate the NavA3C+D_1D_2L [89] algorithm on ten randomly chosen maps, shown in Fig. 5.3, with increasing difficulty as described in Sec. 5.4.1. The figure is best viewed in color. Vertical axis is one of the ten map ID's on which the agent was trained (except for Rnd. Maze) and evaluated. Horizontal axis are different evaluation metrics. We note that when the goal is static then rewards are consistently higher as compared to random goal. With static goals, the metric Distance-inefficiency is close to 1, indicating that the algorithms are able to find shortest path. However, with random goals, the agents struggle to find the shortest path. From the Latency $1 :> 1$ results we note that the algorithm do well when trained and tested on the same map but fail to generalize to new maps when evaluated on ability to exploit the information about goal location. Also note that Latency $1 :> 1$ metric for cases of static goals is expected to be close to one because the location of goal is learned at train time.

However the large standard deviations in this metric and the reward values show that this exploitation is not consistent through episodes. For most of the experiments, the Distance-inefficiency is close to one within error metrics, again implying that the shortest path is taken when the goal is found.

4. **Random goal, Random spawn, static map** Similar to the previous experiment, the Latency $1 :> 1$ is more than 1 but with a large standard deviation implying inconsistent performance from episode to episode. The Distance-inefficiency is larger than 1 showcasing the paths traversed to the goal are not necessarily the shortest.

5. **Random goal, Random spawn, Random map** For this experiment, agents trained on a 1000 maps are tested individually on the 10 chosen maps that are a subset of the 1000 maps. The Latency $1 :> 1$ is close to 1 implying that map-exploitation is not taking place. The large Distance-inefficiency numbers seem to confirm this statement. We present, qualitative results in Sec. 5.5.1 on very simple maps to show that the agents trained on 1000 maps are only randomly exploring the maze rather than utilizing any form of shortest path planning.

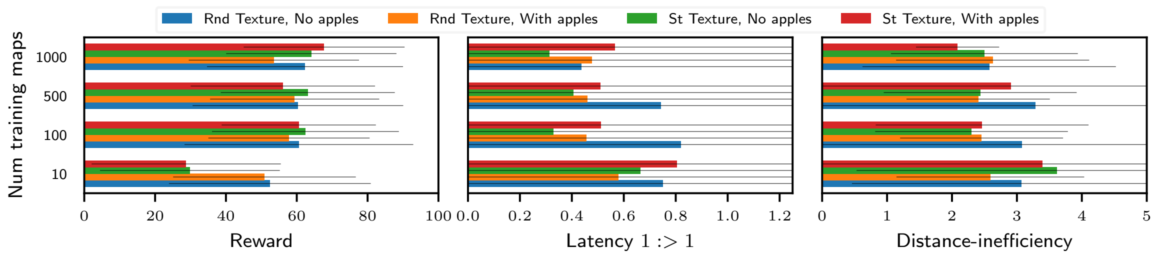


Figure 5.5: Plots showing the effect of number of training maps with random texture (Rnd Texture) and presence of apples (With apples), when evaluated on unseen maps. We note that although the difference between mean metrics is negligible as compared to standard deviation of the metrics. Hence we say that the effect of apples or textures can be ignored. The only clear trend is apparent Latency $1 :> 1$ metric which suggest that random texture along without apples is advantageous in exploiting goal location while finding the goal second time on-wards.

Evaluation on unseen maps The results for training on N maps, where $N \in \{10, 100, 500, 1000\}$, and testing on 100 unseen maps are shown in Fig 5.5. We observe that there is a significant jump of average reward and average goal hits when the number of training maps is increased from 10 to 100 but no significant increase when the number of training maps are increased from 100 to 500 to 1000. This is due to the fact that the wall-following strategy learned by the algorithm, is learned with enough variation in 100 maps and training on additional maps does not add to the learned strategy.

Effect of apples and texture We evaluate the effect of apples and texture during evaluation time in Fig 5.5. We train the algorithm on randomly chosen training maps with random texture and evaluate them no maps with and without random texture and also on maps with and without apples. When the apples are present, we place the apples with probability 0.25 in each block of the map. We find that the algorithm, being trained on random textures and random placement of apples, is robust to presence or absence of textures and apples.

5.5.1 Qualitative evaluation on simple maps

To evaluate what strategies the algorithm is employing to reach the goal we evaluate the algorithm on very simple maps where there are only two paths to reach the goal. The qualitative results for the evaluation are shown in Fig 5.6.

Square map A Square map (Fig 5.6) is the simplest possible map with two paths to the goal. We evaluate the algorithm trained on 1000 random maps on this map. We observe that the agent greedily moves in the direction of initialization. This may be because of the initial learning which is motivated by small rewards of getting apples. We compute the percentage of times the agent takes the shortest path over a trial of 100 episodes. We find the agent takes the shortest path only 50.4% (±12.8%) of the times which is no better than random.

Wrench map To eliminate the dependency on initial orientation, we evaluate the algorithm on Wrench map as shown in Fig 5.6. We fix in the spawn point at the bottom of the tail so that shortest path is independent of the spawn orientation. The decision about the shortest path is made at the junction where the agent can either chose to go left or right. We find that the agent takes the shortest path only 32.9% (±25.1%) of the times which is in fact worse than random.

Goal map Similar to the wrench map, the goal map (Fig 5.6) provides a decision point independent of the initial orientation but penalizes the wrong decision more than the wrench map. The shortest path is chosen 42.6% (±35.1%) of the times which is again no better than random.

These experiments show that NavA3C+D_1D_2L algorithm, even when trained on 1000 maps, do not generalize to these very simple maps. Again note that even in cases when there are only two possible paths to the goal, the agent is unable to chose the shorter path with more than 50% probability. This shows that the models trained on 1000 maps have learned only a wall-following strategy rather than a learning to plan-path based one.

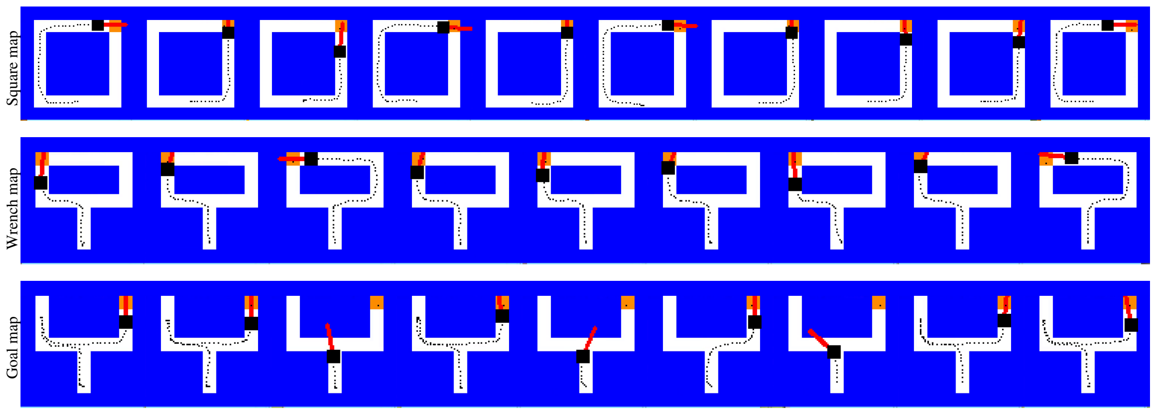


Figure 5.6: Snapshots of path taken by the agent to reach the goal in a single episode when model trained on 1000 maps is evaluated Square, Wrench and Goal map. The top row shows an evaluation example on Square map, the agent takes the shortest path 6/10 times but when averaged over 100 episodes, the percentage of shortest path taken is not better than random 50.4% (±12.8%). Although for the example of Wrench map the agent takes the shortest path 8/10 times but when averaged over 100 episodes, the percentage of shortest path taken is reduced to 32.9% (±25.1%). For the Goal map, the example chosen here shows that the shortest path is only taken 1/6 times, on an average over 100 episodes, the shortest path is taken 42.6% (±35.1%) times.

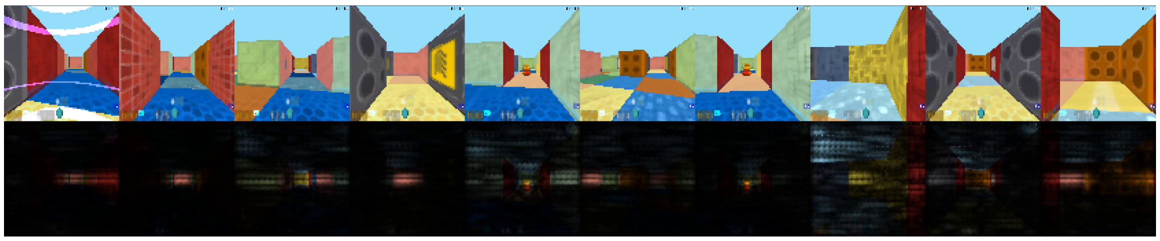


Figure 5.7: Visualizing attention for a sequence. The second row shows the corresponding attention mask of the algorithm, when the model is trained on 1000 maps and evaluated on map ID 127. We observe that the attention is uniformly distributed on the image when the agent spawns. The attention narrows down few pixels in the center when the agent is navigating through the corridor. It spreads to the entire image around turns and junctions. The algorithm also pays close attention to important objects like goal, apples and unique decals.

5.5.2 Attention heat maps

We use the normalized sum of absolute gradient of the loss with respect to the input image as a proxy for attention in the image. The gradients are normalized for each image so that the maximum gradient is one. The attention values are then used as a soft mask on the image to create the visualization as shown in Fig 5.7

We observe that the attention is uniformly distributed on the image when the agent spawns. The attention narrows down to a few pixels in the center when the agent is navigating through the corridor. It spreads to the entire image around turns and junctions. The attention also pays close attention to important objects like goal, apples and unique decals.

5.6 Conclusion

In this work, we comprehensively evaluate NavA3C+D_1D_2L ([89]), a DRL-based navigation algorithms, through systematic set of experiments that are repeated over multiple randomly chosen maps. Our experiments show that DRL-based navigation models are able to perform some degree of path-planning and mapping when trained and tested on the same map even when spawn locations and goal locations are randomized. However the large variation in the evaluation metrics show that how such behaviour is not consistent across episodes. We also train and test these methods on disjoint set of maps and show that such trained models fail to perform any form of path-planning or mapping in unseen environments.

In this work, we begin by asking: do DRL-based navigation algorithms really “learn to navigate”? Our results answer this question negatively. At best, we can say that DRL-based algorithms learn to navigate in the exact same environment, rather than general technique of navigation which is what classical mapping and path planning provide. We hope that the systematic approach to the experiments in this work serve as a benchmark for future DRL-based navigation methods.

CHAPTER 6

Goal-conditioned RL without goal-rewards[1]

6.1 Introduction

Many tasks in robotics require the specification of a *goal* for every trial. For example, a robotic arm can be tasked to move an object to an arbitrary goal position on a table [42]; a mobile robot can be tasked to navigate to an arbitrary goal landmark on a map [158]. The adaptation of reinforcement learning to such goal-conditioned tasks where goal locations can change is called Multi-Goal Reinforcement Learning (MGRL) [109]. State-of-the-art MGRL algorithms [3, 110] work by estimating *goal-conditioned value functions* (GCVF) which are defined as expected cumulative rewards from start states with specified goals. GCVFs, in turn, are used to compute *policies* that determine the actions to take at every state.

To learn GCVFs, MGRL algorithms use *goal-reward*, defined as the relatively higher reward recieved on reaching the desired goal state. This makes the reward function dependent on the desired goal. For example, in the Fetch-Push task [109] of moving a block to a given location on a table, every movement incurs a "-1" reward while reaching the desired goal returns a "0" goal-reward. This dependence introduces additional reward resampling steps in algorithms like Hindsight Experience Replay (HER) [3], where trials in which the agent failed to reach the goal are reused by recomputing rewards as if the reached states were pseudo-desired goals. Due to the dependence of the reward function on the goal, the relabelling of every pseudo-goal requires an independent *reward-recomputation* step, which can be expensive.

In this chapter, we demonstrate that *goal-rewards are not needed to learn GCVFs*. For the Fetch-Push example, the "0" goal-reward does not need to be achieved to learn its GCVF. Specifically, the agent continues to receive "-1" reward even when the block is in

[1]This material originally appeared as: V. Dhiman, S. Banerjee, J. M. Siskind, and J. J. Corso. Learning goal-conditioned value functions with one-step path rewards rather than goal-rewards. In *Open Review*, 2019

the given goal location. This reward formulation is atypical in conventional RL because high reward is used to specify the desired goal location. However, this goal-reward is not necessary in goal-conditioned RL because the goal is already specified at the start of every episode. We use this idea, to propose a goal-conditioned RL algorithm which learns to reach goals without goal-rewards. This is a counter-intuitive result which is important for understanding goal-conditioned RL.

Let us consider another example to motivate the redundancy of goal-rewards. Consider a student who has moved to a new campus. To learn about the campus, the student explores it randomly with no specific goal in mind. The key intuition here is that the student is not incentivized to find specific goal locations (i.e. no goal-rewards) but is aware of the effort required to travel between points around the university. When tasked with finding a goal classroom, the student can chain together these path efforts to find the least-effort path to the classroom. Based on this intuition of least-effort paths, we redefine GCVFs to be the expected *path-reward* that is learned for all possible start-goal pairs. We introduce a *one-step loss* that assumes one-step paths to be the paths of maximum reward between pairs wherein the state and goal are adjacent. Under this interpretation, the *Bellman equation* chooses and chains together one-step paths to find longer maximum reward paths. Experimentally, we show how this simple reinterpretation, which does not use goal rewards, performs as well as HER while outperforming it in terms of reward computation.

We also extend a closely related algorithm, Floyd-Warshall Reinforcement Learning (FWRL) [51] (also called Dynamic Goal Reinforcement learning) to use parametric function approximators instead of tabular functions. Similar to our re-definition of GCVFs, FWRL learns a goal-conditioned Floyd-Warshall function that represents path-rewards instead of future-rewards. We translate FWRL's compositionality constraints in the space of GCVFs to introduce additional loss terms to the objective. However, these additional loss terms do not show improvement over the baseline. We conjecture that the compositionality constraints are already captured by other loss terms.

In summary, the contributions of this work are twofold. Firstly, we reinterpret goal-conditioned value functions as expected path-rewards and introduce one-step loss, thereby removing the dependency of GCVFs on goal-rewards and reward resampling. We showcase our algorithm's improved sample efficiency (in terms of reward computation). We thus extend algorithms like HER to domains where reward recomputation is expensive or infeasible. Secondly, we extend the tabular Floyd-Warshal Reinforcement Learning to use deep neural networks.

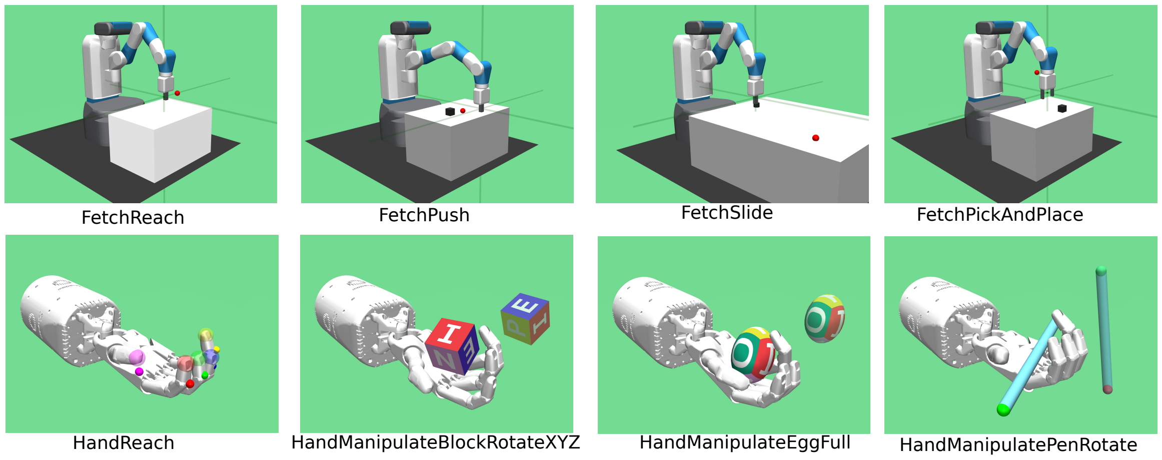


Figure 6.1: [109] introduce challenging tasks on the Fetch robot and the Shadow Dextrous hand. We use these tasks for our experiments. Images are taken from the technical report. (Images from [109])

6.2 Related Work

Goal-conditioned tasks in reinforcement learning have been approached in two ways, depending upon whether the algorithm explicitly separates state and goal representations. The first approach is to use vanilla reinforcement learning algorithms that do not explicitly make this separation [88, 28, 43, 104, 87]. These algorithms depend upon neural network architectures to carry the burden of learning the separated representations.

The second approach makes this separation explicit via the use of goal-conditioned value functions [32, 136]. *Universal Value Function Appoximators* [125] propose a network architecture and a factorization technique that separately encodes states and goals, taking advantage of correlations in their representations. *Temporal Difference Models* combine model-free and model-based RL to gain advantages from both realms by defining and learning a horizon-dependent GCVF. All these works require the use of goal-dependent reward functions and define GCVFs as future-rewards instead of path-rewards, contrasting them from our contribution.

Unlike our approach, [3] propose *Hindsight Experience Replay*, a technique for resampling state-goal pairs from failed experiences; which leads to faster learning in the presence of sparse rewards. In addition to depending on goal rewards, HER also requires the repeated recomputation of the reward function. In contrast, we show how removing goal-rewards removes the need for such recomputations. We utilize HER as a baseline in our work.

[51] also use the structure of the space of GCVFs to learn. This work employs compositionality constraints in the space of these functions to accelerate learning in a tabular domain. While their definition of GCVFs is similar to ours, the terminal condition is different. We describe this difference in Section 6.4. We also extend their tabular formulation

to deep neural networks and evaluate it against the baselines.

6.3 Background

A reinforcement learning (RL) problem is formalized as a Markov Decision Process (MDP) [135]. A MDP is defined by a five tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$, that governs a sequence of state-action-reward triples $[(s_0, a_0, r_0), \ldots, (s_T, a_T, r_T)]$. $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s, a) : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the system dynamics, $R(s, a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and γ is the discount factor. In a typical RL problem the transition function T is not given but is known to be static. In RL, the objective is to find a policy $\chi(s) : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative reward over time, $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$, called the *return*. The discount factor, $\gamma < 1$, forces the return to be finite for infinite horizons. Reinforcement learning is typically formulated in single-goal contexts. More recently there has been interest in multi-goal problems [3, 110, 109], which is the focus of this work.

6.3.1 Deep Reinforcement learning

A number of reinforcement learning algorithms use parametric function approximators to estimate the return in the form of an action-value function, $Q(s, a)$:

$$Q_\chi(s, a) = \mathbb{E}_\chi \left[\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k) \middle| s_t = s, a_t = a \right], \tag{6.1}$$

where T is the episode length. When the policy χ is optimal, the Q-function satisfies the *Bellman equation* [8].

$$Q_*(s_t, a_t) = \begin{cases} R(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q_*(s_{t+1}, a) & \text{if } t < T \\ R(s_T, a_T) & \text{if } t = T \end{cases}. \tag{6.2}$$

The $Q_*()$ function can be learned using Q-learning algorithm [145]. The optimal policy can be computed from Q_* greedily, $\chi_*(s_t) = \arg\max_{a \in \mathcal{A}} Q_*(s_t, a)$. In *Deep Q-Networks* (DQN), [92] formulate a loss function based on the Bellman equation to approximate the optimal Q_*-function using a deep neural network, Q_m:

$$\mathcal{L}(\theta_{Q_m}) = \mathbb{E}_{a_t \sim \chi(s_t; \theta_{\chi m})} \left[(Q_m(s_t, a_t; \theta_{Q_m}) - y_t)^2 \right], \tag{6.3}$$

where $y_t = R(s_t, a_t) + \max_a \gamma Q_{\text{tgt}}(s_{t+1}, a; \theta_{Q_{\text{tgt}}})$, is the *target* and Q_{tgt} is the target network [93]. The target network is a slower-changing copy of the main network that stabilizes learning. [93] also employ *replay buffers* [75] that store transitions from episodes. During training, these transitions are sampled out of order to train the networks in an off-policy manner, avoiding correlation in the samples and thus leading to faster learning.

In this work, we use an extension of DQN to continuous action spaces called deep deterministic policy-gradients (DDPG) [74]. DDPG approximates the policy with a policy network $\chi_{\text{tgt}}(s; \theta_\chi)$ that replaces the $\max$ operator in y_t. The target thus becomes $y_t = R(s_t, a_t) + \gamma Q_{\text{tgt}}(s_{t+1}, \chi_{\text{tgt}}(s_{t+1}; \theta_\chi); \theta_{Q_{\text{tgt}}})$ and the loss function changes accordingly:

$$\mathcal{L}(\theta_Q, \theta_\chi) = \mathbb{E}_{a_t \sim \chi(s_t; \theta_\chi)}[(Q_m(s_t, a_t; \theta_Q) - y_t)^2]. \tag{6.4}$$

6.3.2 Multi-goal Reinforcement learning

Multi-Goal Reinforcement Learning [109] focuses on problems where the desired goal state can change for every episode. State-of-the-art MGRL algorithms learn a goal-conditioned value function (GCVF), $Q\,(s, a, g)$, which is defined similar to the Q-function (6.1), but with an additional dependence on the desired goal specification $g \in \mathcal{G}$:

$$Q_\chi\,(s, a, g) = \mathbb{E}_\chi \left[\sum_{k=t}^{T} \gamma^{k-t} R(s_k, a_k, g) \middle| s_t = s, a_t = a \right]. \tag{6.5}$$

The structure of the goal specification, $g \in \mathcal{G}$, can be arbitrary. For example, in a robotic arm experiment, possible goal specifications include the desired position of the end-effector and the desired joint angles of the robot. The states and *achieved goals* are assumed to be an observable part of the Goal-MDP to enable the agent to learn the correspondences between them, $[(s_0, a_0, g_0, r_0), \ldots, (s_T, a_T, g_T, r_T)]$. Consequently, this Goal-MDP is fully governed by the six tuple $(\mathcal{S}, \mathcal{A}, \mathcal{G}, T, R, \gamma)$. The reward, $R(s, a, g) : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$, and policy $\chi(s, g) : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ are also in turn conditioned on goal information.

Hindsight Experience Replay HER [3] builds upon this definition of GCVFs (6.5). The main insight of HER is that there is no valuable feedback from the environment when the agent does not reach the goal. This is further exacerbated when goals are sparse in the state-space. HER solves this problem by reusing these failed experiences for learning. It recomputes a reward for each reached state by relabeling them as pseudo-goals.

In our experiments, we employ HER's *future* strategy for pseudo-goal sampling. More specifically, two transitions from the same episode in the replay buffer for times t and $t + f$

are sampled. The achieved goal g_{t+f} is then assumed to be the pseudo-goal. The algorithm generates a new transition for the time step t with the reward re-computed as if g_{t+f} was the desired goal, $(s_t, a_t, s_{t+1}, R(s_t, a_t, g_{t+f}))$. HER uses this new transition as a sample.

6.4 Path Reward-Based GCVFs

In our definition of the GCVF, instead of making the reward function depend upon the goal, we count accumulated rewards over a path, *path-rewards*, only if the goal is reached. This makes the dependence on the goal explicit instead of implicit to the reward formulation. Mathematically,

$$Q^P_\chi(s, a, g^*) = \begin{cases} \mathbb{E}_\chi \left[\sum_{k=t}^{l-1} \gamma^{k-t} R^P(s_k, a_k) \middle| s, a, g_l = g^* \right] & \text{if } \exists\, l \text{ such that } g_l = g^* \quad (6.6a) \\ -\infty & \text{otherwise,} \quad (6.6b) \end{cases}$$

where l is the time step when the agent reaches the goal. If the agent does not reach the goal, the GCVF is defined to be negative infinity. This first term (6.6a) is the expected cumulative reward over paths from a given start state to the goal. This imposes the constraint that cyclical paths in the state space must have negative cumulative reward for (6.6a) to yield finite values. For most practical physical problems, this constraints naturally holds if reward is taken to be some measure of negative energy expenditure. For example, in the robot arm experiment, moving the arm must expend energy (negative reward). Achieving a positive reward cycle would translate to generating infinite energy . In all our experiments with this formulation, we use a constant reward of -1 for all states, $R^P(s, a) = -1\, \forall s, a$.

For the cases when the agent does reach the goal at time step l (6.6a), the Bellman equation takes the following form:

$$Q^P_*(s_t, a_t, g^*) = \begin{cases} R^P(s_t, a_t) + \gamma \max_{a \in \mathcal{A}} Q^P_*(s_{t+1}, a, g^*) & \text{if } t < l - 1 \quad (6.7a) \\ R^P(s_{l-1}, a_{l-1}) & \text{if } t = l - 1\ . \quad (6.7b) \end{cases}$$

Notice that terminal step in this equation is the step to reach the goal. This differs from Equation (6.3), where the terminal step is the step at which the episode ends. This formulation is equivalent to the end of episode occuring immediately when the goal is reached. This reformulation does not require goal-rewards, which in turn obviates the requirement for pseudo-goals and reward recomputation.

One-Step Loss To enable algorithms like HER to work under this reformulation we need to recognize when the goal is reached (6.7b). This recognition is usually done by the reception of high goal reward. Instead, we use (6.7b) as a *one-step loss* that serves this purpose which is one of our main contributions:

$$\mathcal{L}_{\text{step}}(\theta_Q) = (Q^P_* (s_{l-1}, a_{l-1}, g_l; \theta_Q) - R(s_{l-1}, a_{l-1}))^2. \tag{6.8}$$

This loss is based on the assumption that one-step reward is the highest reward between adjacent start-goal states and allows us to estimate the one-step reward between them. Once learned, it serves as a proxy for the reward to the last step to the goal (6.7b). The Bellman equation (6.7), serves as a one-step rollout to combine rewards to find maximum reward paths to the goal.

One-step loss is different from the terminal step of Q-Learning because one-step loss is applicable to every transition unlike the terminal step. However, one-step loss can be thought of as Q-Learning where every transition is a one-step episode where the achieved goal is the pseudo goal.

One-step loss also different from the terminal condition in [51]. [51] defines $Q^P_* (., ., .)$ similar to Eq (6.7) except the terminal condition is defined as $Q^P_* (s_t, a, g*) = 0$ when $g_t = g^*$. Under the stated assumptions, the two definitions are equivalent but one-step loss is advantageous as it can be applied to every transition unlike the [51] terminal condition which can be only applied when $g_t = g^*$.

We modify an implementation of HER to include the step-loss term and disable goal rewards for our experiments. As in HER, we use the DDPG loss $\mathcal{L}_{\text{ddpg}}$ while using the "future" goal sampling strategy described in the chapter. The details of the resulting algorithm are shown as pseudo-code in Algorithm 3 in the Appendix.

6.4.1 Deep Floyd-Warshall Reinforcement Learning

The GCVF redefinition and one step-loss introduced in this chapter are inspired by the tabular formulation of Floyd-Warshall Reinforcement Learning (FWRL) [51]. We extend this algorithm for use with deep neural networks. Unfortunately, the algorithm itself does not show significant improvement over the baselines. However, the intuitions gained in its implementation led to the contributions of this chapter.

The core contribution of FWRL is a compositionality constraint in the space of GCVFs. This constraint states that the optimal Q_* value from any state s_t to any goal g_{t+f} is greater than or equal to the sum of optimal Q_* values via any intermediate state-goal pair (s_w, g_w):

$$Q_*(s_t, a_t, g_w) + Q_*(s_w, \chi_*(s_w, g_{t+f}; \theta_\chi), g_{t+f}) \geq Q_*(s_t, a_t, g_{t+f}) . \tag{6.9}$$

We translate these constraints into loss terms and add them to the DDPG loss $\mathcal{L}_{\text{ddpg}}$ and one-step loss $\mathcal{L}_{\text{step}}$. Taking cue from [94], we do not repeat the the main online network Q_m in the loss term. We use a target network Q_{tgt} and split the constraint into two loss terms. One loss term acts as a lower bound $\mathcal{L}_{\text{lo}}$ and the other acts as an upper bound $\mathcal{L}_{\text{up}}$:

$$\mathcal{L}_{\text{lo}} = \text{ReLU}[Q_{\text{tgt}}(s_t, a_t, g_w) + Q_{\text{tgt}}(s_w, \chi_t(s_w, g_{t+f}; \theta_\chi), g_{t+f}) - Q_m(s_t, a_t, g_{t+f})]^2 \tag{6.10}$$

$$\mathcal{L}_{\text{up}} = \text{ReLU}[Q_m(s_t, a_t, g_w) + Q_{\text{tgt}}(s_w, \chi_t(s_w, g_{t+f}; \theta_\chi), g_{t+f}) - Q_{\text{tgt}}(s_t, a_t, g_{t+f})]^2 . \tag{6.11}$$

Note that the above terms differ only by choice of the target and main network.

FWRL Sampling We augment HER sampling to additionally get the intermediate state-goal pair (s_w, g_w). Once a transition (s_t, a_t, r_t, s_{t+1}) and a future goal g_{t+f} have been sampled from the same episode, we sample another intermediate state and goal pair (s_w, g_w) such that $t \leq w \leq t + f$.

6.5 Experiments

We use the environments introduced in [109] for our experiments. Broadly the environments fall in two categories, Fetch and Hand tasks. Our results show that learning is possible across all environments without the requirement of goal-reward. More specifically, the learning happens even when reward given to our algorithm is agent is always “-1” as opposed to the HER formulation where a special goal-reward of “0” is needed for learning to happen.

The Fetch tasks involve a simulation of the Fetch robot’s 7-DOF robotic arm. The four tasks are Reach, Push, Slide and PickAndPlace. In the Reach task the arm’s end-effector is tasked to reach the a particular 3D coordinate. In the Push task a block on a table needs to be pushed to a given point on it. In the Slide task a puck must be slid to a desired location. In the PickAndPlace task a block on a table must be picked up and moved to a 3D coordinate.

The Hand tasks use a simulation of the Shadow’s Dexterous Hand to manipulate objects of different shapes and sizes. These tasks are HandReach, HandManipulateBlock-

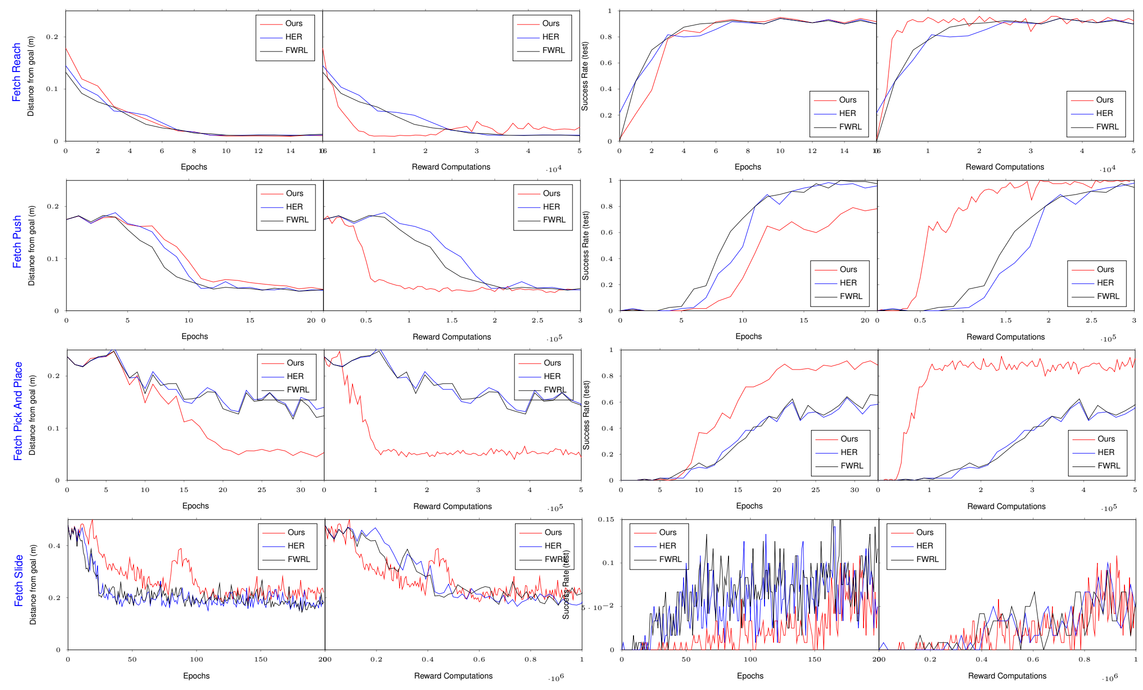


Figure 6.2: For the Fetch tasks, we compare our method (red) against HER (blue) [2] and FWRL (green) [51] on the distance-from-goal and success rate metrics. Both metrics are plotted against two progress measures: the number of training epochs and the number of reward computations. Except for the Fetch Slide task, we achieve comparable or better performance across the metrics and progress measures.

RotateXYZ, HandManipulateEggFull and HandManipulatePenRotate. In HandReach the hand's fingertips need to reach a given configuration. In the HandManipulateBlockRotateXYZ, the hand needs to rotate a cubic block to a desired orientation. In HandManipulateEggFull, the hand repeats this orientation task with an egg, and in HandManipulatePenRotate, it does so with a pen.

Snapshots of all these tasks can be found in Figure 6.1. Note that these tasks use joint angles, not visual input.

6.5.1 Metrics

Similar to prior work, we evaluate all experiments on two metrics: the success rate and the average distance to the goal. The success rate is defined as the fraction of episodes in which the agent is able to reach the goal within a pre-defined threshold region. The metric *distance of the goal* is the euclidean distance between the achieved goal and the desired goal in meters. These metrics are plotted against a standard progress measure, the number of training epochs, showing comparable results of our method to the baselines.

To emphasize that our method does not require goal-reward and reward re-computation, we plot these metrics against another progress measure, the number of reward computations used during training. This includes both the episode rollouts and the reward recomputations during HER sampling.

6.5.2 Hyper-parameters choices

Unless specified, all our hyper-parameters are identical to the ones used in the HER implementation [22]. We note two main changes to HER to make the comparison more fair. Firstly, we use a smaller *distance-threshold*. The environment used for HER and FWRL returns the goal-reward when the achieved goal is within this threshold of the desired goal. Because of the absence of goal-rewards, the distance-threshold information is not used by our method. We reduce the threshold to 1cm which is reduction by a factor of 5 compared to HER.

Secondly, we run all experiments on 6 cores each, while HER uses 19. The batch size used is a function of the number of cores and hence this parameter has a significant effect on learning.

To ensure fair comparison, all experiments are run with the same hyper-parameters and random seeds to ensure that variations in performance are purely due to differences between the algorithms.

6.5.3 Results

All our experimental results are described below, highlighting the strengths and weaknesses of our algorithm. Across all our experiments, the distance-to-the-goal metric achieves comparable performance to HER *without requiring goal-rewards*.

Fetch Tasks The experimental results for Fetch tasks are shown in Figure 6.2. For the Fetch Reach and Push tasks, our method achieves comparable performance to the baselines across both metrics in terms of training epochs and outperforms them in terms of reward recomputations. Notably, the Fetch Pick and Place task trains in significantly fewer epochs. For the Fetch Slide task the opposite is true. We conjecture that Fetch Slide is more sensitive to the distance threshold information, which our method is unable to use.

Hand Tasks For the Hand tasks, the distance to the goal and the success rate show different trends. We show the results in Figure 6.3. When the distance metric is plotted against epochs, we get comparable performance for all tasks; when plotted against reward computations, we outperform all baselines on all tasks except Hand Reach. The baselines perform well enough on this task, leaving less scope for significant improvement. These trends do

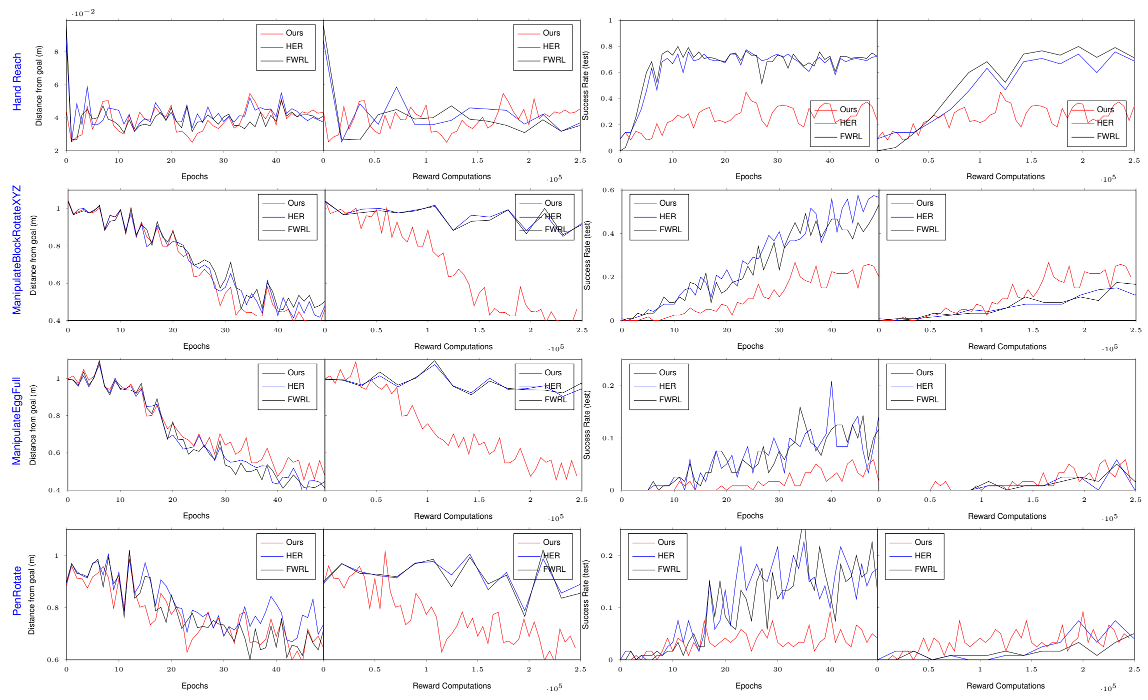
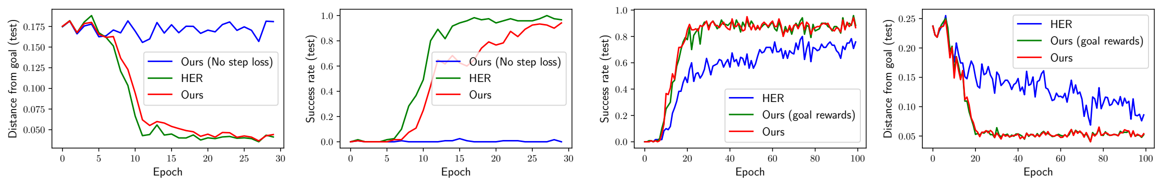


Figure 6.3: For the hand tasks, we compare our method (red) against HER (blue) [2] and FWRL (green) [51] for the distance-from-goal and success rate metrics. Furthermore, both metrics are plotted against two progress measures, the number of training epochs and the number of reward computations. Measured by distance from the goal, our method performs comparable to or better than the baselines for both progress measurements. For the success rate, our method underperforms against the baselines.


(a) Do we really need the step-loss?

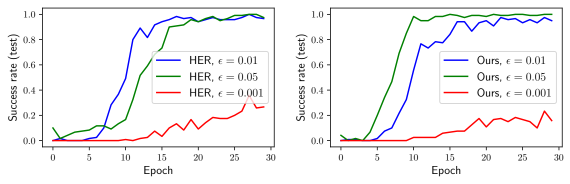
(b) Effect of goal-rewards

Figure 6.4: (a) Effects of removing the step-loss from our methods. Results show that it is a critical component to learning in the absence of goal-rewards. (b) Adding goal-rewards to our algorithm that does have an effect further displaying how they are avoidable.

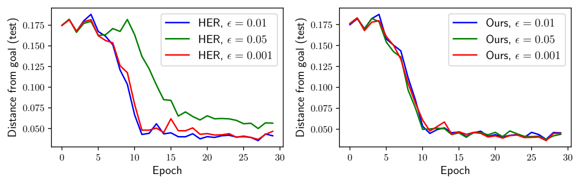
not hold for the success rate metric, on which our method consistently under-performs compared to the baselines across tasks. This is surprising, as all algorithms average equally on the distance-from-goal metric. We conjecture that this might be the result of high-distance failure cases of the baselines, i.e. when the baselines fail, they do so at larger distances from the goal. In contrast, we assume our method's success and failure cases are closer together.

6.6 Analysis

To gain a deeper understanding of the method we perform three additional experiments on different tasks. We ask the following questions: (a) How important is the step loss? (b)



(a) Sucesss rate



(b) Distance from goal

Figure 6.5: We measure the sensitive of HER and our method to the dsitance-threshold (ϵ) with respect to the success-rate and distance-from-goal metrics. Both algorithms success-rate is sensitive the threshold while only HER's distance-from-goal is affected by it.

What happens when the goal-reward is also available to our method? (c) How sensitive is HER and our method to the distance-threshold?

How important is the step loss? We choose the Fetch-Push task for this experiment. We run our algorithm with no goal reward and without the step loss on this task. Results show that our algorithm fails to reach the goal when the step-loss is removed (Fig. 6.4a) showing its necessity.

What happens when the goal-reward is also available to our method? We run this experiment on the Fetch PickAndPlace task. We find that goal-rewards do not affect the performance of our algorithm further solidifying the avoidability of goal-reward (Fig 6.4b).

How sensitive is HER and our method to the distance-threshold? In the absence of goal-rewards, our algorithm is not to able capture distance threshold information that decides whether the agent has reached the goal or not. This information is available to HER. To understand the sensitivity of our algorithm and HER on this parameter, we vary it over 0.05 (the original HER value), 0.01 and 0.001 meters (Fig. 6.5). Results show that for the success-rate metric, which is itself a function of this parameter, both algorithms are affected equally (Fig. 6.5a). For the distance-from-goal, only HER is affected (Fig. 6.5b). This fits our expectations as set up in section 6.5.2.

6.7 Conclusion

In this work we pose a reinterpretation of goal-conditioned value functions and show that under this paradigm learning is possible in the absence of goal reward. This is a surprising result that runs counter to intuitions that underlie most reinforcement learning algorithms. In future work, we will augment our method to incorporate the distance-threshold information to make the task easier to learn when the threshold is high. We hope that the experiments and results presented in this chapter lead to a broader discussion about the assumptions actually required for learning multi-goal tasks.

CHAPTER 7

Conclusion and Future work

I present different ways to improve mapping and navigation. Mapping and navigation problem is so fundamental to artificial intelligence that even after years of progress in the field we do not have a set of adaptive algorithm that works in all kinds of environment with different sensor types.

In Chapters 2-4, I show that the algorithms can be improved by identifying the structure in the environment and explicitly modeling that structure like mutual observations, occlusion modeling and forward sensor models. Our work has added the knowledge of building mapping algorithms that are more accurate, robust to occlusions and can deploy on multiple robots by including mutual observations.

In Chapters 5-6, I investigate where Deep Reinforcement Learning based navigation algorithms stand and how they can be improved. However, as we saw in Chapter 5, we do not have a navigation driven DRL algorithm that learns to build a map in novel environments.

Lately, a few works have focused on this problem [124, 43]. But these works leave enough to be desired. Gupta et al. 2017 [43] build a occupancy grid map as a part of the pipeline which is not scalable to high dimensional use cases of navigation algorithms. Savinov et al. 2018 [124] use a topological memory for navigation that works on topological memory for navigation. An ideal navigation algorithm that includes mapping will seamlessly transition between the topological memory and the learned mapping from observation to action. More recent works on instance specific control [112] show one possible direction to approach this problem. One can first use the learned mapping from observation to action and depending upon accuracy of prediction, switch to topological memory when the prediction accuracy falls below a threshold.

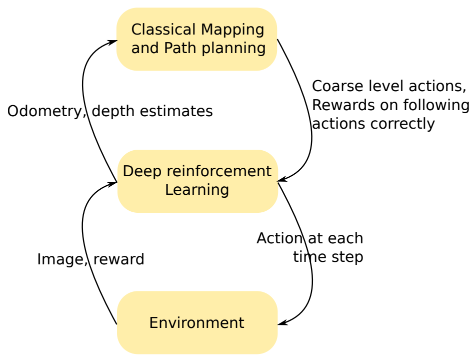

Classical Mapping and Path planning
Odometry, depth estimates
Coarse level actions, Rewards on following actions correctly
Deep reinforcement Learning
Image, reward
Action at each time step
Environment

Figure 7.1: Hierarchical hybrid scheme to augment DRL algorithms with classical mapping and path-planning.

Alternatively, we can combine DRL algorithms and classical navigation in a hierarchical hybrid setup where the long term coarse-level mapping and path-planning is handled by a classical approach and fine grained low level control is left to DRL algorithm (Fig. 7.1). We hypothesize that the DRL-based navigation algorithms are good at learning decisions that correspond based on few (may be distant) past observations but are not able to learn mapping and shortest path-planning algorithms. In hindsight, to expect generic Recurrent Neural Networks (RNNs) to learn complex algorithms like mapping and path-planning seems like wishful thinking. In future, the classical approach level in the hierarchy can be replaced by a trainable neural network model that learns the tune-able aspects of the algorithm from data.

BIBLIOGRAPHY

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *arXiv preprint arXiv:1711.07280*, 2017.

[2] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[4] K.S. Arun, T.S. Huang, and S.D. Blostein. Least-squares fitting of two 3-D point sets. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (5):698–700, 1987.

[5] Danielle Attias. *The Autonomous Car, a Disruptive Business Model?*, pages 99–113. Springer International Publishing, Cham, 2017.

[6] E.J. Barbeau. *Polynomials*. Problem Books in Mathematics. Springer, 2003.

[7] Charles Beattie, Joel Z Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, et al. Deepmind lab. *arXiv preprint arXiv:1612.03801*, 2016.

[8] Richard Bellman. The theory of dynamic programming. Technical report, RAND Corp Santa Monica CA, 1954.

[9] M. Betke and L. Gurvits. Mobile robot localization using landmarks. *Robotics and Automation, IEEE Transactions on*, 13(2):251–263, 1997.

[10] G. Bradski. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[11] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *ECCV*, 2010.

[12] V.I. Bykov, A.M. Kytmanov, M.Z. Lazman, and M. Passare. *Elimination methods in polynomial computer algebra*, volume 448. Kluwer Academic Pub, 1998.

[13] O. Certik et al. Sympy python library for symbolic mathematics. Technical report, Technical report (since 2006), http://code. google. com/p/sympy/(accessed November 2009), 2008.

[14] Chun-Hua Chang, Shao-Chen Wang, and Chieh-Chih Wang. Vision-based cooperative simultaneous localization and tracking. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 5191 –5197, may 2011.

[15] Devendra Singh Chaplot, Guillaume Lample, Kanthashree Mysore Sathyendra, and Ruslan Salakhutdinov. Transfer deep reinforcement learning in 3d environments: An empirical study. In *NIPS*, 2016.

[16] Wongun Choi and Silvio Savarese. Multi-target tracking in world coordinate with single, minimally calibrated camera. In *ECCV*, pages 553–567, 2010.

[17] C Chow and C Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

[18] M. Cognetti, P. Stegagno, A. Franchi, G. Oriolo, and H.H. Bulthoff. 3-D mutual localization with anonymous bearing measurements. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 791 –798, may 2012.

[19] JooPaulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998.

[20] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[21] N. Developers. Scientific computing tools for python-numpy, 2010.

[22] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. `https://github.com/openai/baselines`, 2017.

[23] V. Dhiman, S. Banerjee, B. Griffin, J. Mark Siskind, and J. J. Corso. A critical investigation of deep reinforcement learning for navigation. *CoRR*, abs/1802.02274, 2018.

[24] V. Dhiman, S. Banerjee, J. M. Siskind, and J. J. Corso. Learning goal-conditioned value functions with one-step path rewards rather than goal-rewards. In *Open Review*, 2019.

[25] V. Dhiman, A. Kundu, F. Dellaert, and J. J. Corso. Modern MAP inference methods for accurate and faster occupancy grid mapping on higher order factor graphs. In *Proceedings of International Conference on Robotics and Automation*, 2014.

[26] V. Dhiman, J. Ryde, and J. J. Corso. Mutual localization: Two camera relative 6-dof pose estimation from reciprocal fiducial observation. In *Proceedings of International Conference on Intelligent Robots and Systems*, 2013.

[27] V. Dhiman, Q. Tran, J. Corso, and M. Chandraker. A continuous occlusion model for road scene understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[28] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[29] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[30] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[31] 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(9):1627–1645, 2010.

[32] David Foster and Peter Dayan. Structure in the space of value functions. *Machine Learning*, 49(2-3):325–346, 2002.

[33] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. Collaborative multi-robot localization. In Wolfram Burgard, Armin Cremers, and Thomas Cristaller, editors, *KI-99: Advances in Artificial Intelligence*, volume 1701 of *Lecture Notes in Computer Science*, pages 698–698. Springer Berlin / Heidelberg, 1999.

[34] A. Franchi, G. Oriolo, and P. Stegagno. Mutual localization in a multi-robot system with anonymous relative position measures. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3974–3980. IEEE, 2009.

[35] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(8):1362–1376, 2010.

[36] Tianshi Gao, B. Packer, and D. Koller. A segmentation-aware object detection model with occlusion handling. In *CVPR*, pages 1361–1368, 2011.

[37] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*, pages 3354–3361, 2012.

[38] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, 2003.

[39] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *CVPR*, pages 1–6, 2007.

[40] R. Grabowski and P. Khosla. Localization techniques for a team of small robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[41] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *CVPR*, pages 707–714, 2004.

[42] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3389–3396. IEEE, 2017.

[43] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] B.M. Haralick, C.N. Lee, K. Ottenberg, and M. Nölle. Review and analysis of solutions of the three point perspective pose estimation problem. *International Journal of Computer Vision*, 13(3):331–356, 1994.

[45] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000.

[46] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *JOSA A*, 4(4):629–642, 1987.

[47] A. Howard and L. Kitchen. Cooperative localisation and mapping. In *International Conference on Field and Service Robotics (FSR99)*, pages 92–97. Citeseer, 1999.

[48] Andrew Howard and Nicholas Roy. The robotics data set repository (radish), 2003.

[49] Edward Hsiao and Martial Hebert. Occlusion reasoning for object detection under arbitrary viewpoint. In *CVPR*, pages 3146–3153, 2012.

[50] Vladimir Jojic, Stephen Gould, and Daphne Koller. Accelerated dual decomposition for MAP inference. In *ICML*, 2010.

[51] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[52] K. Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV*, pages 586–591, 2001.

[53] Jörg H Kappes, Bjoern Andres, Fred A Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X Kausler, Jan Lellmann, Nikos Komodakis, et al. A comparative study of modern inference techniques for discrete energy minimization problems. In *CVPR*, 2013.

[54] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR 99)*, Oct 1999.

[55] Tasha Keeney. Mobility-as-a-service: Why self-driving cars could change everything. *ARK Investment Management Research*, 2017.

[56] Cameron F Kerry and Jack Karsten. Gauging investment in self-driving cars. *Brookings Institution, October*, 16, 2017.

[57] P. Khosla, R. Grabowski, and H. Choset. An enhanced occupancy map for exploration via pose separation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[58] Gab-Hoe Kim, Jong-Sung Kim, and Ki-Sang Hong. Vision-based simultaneous localization and mapping with two cameras. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1671 – 1676, aug. 2005.

[59] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(10):1568–1583, 2006.

[60] N. Komodakis, N. Paragios, and G. Tziritas. Mrf optimization via dual decomposition: Message-passing revisited. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.

[61] Nikos Komodakis and Nikos Paragios. Beyond pairwise energies: Efficient optimization for higher-order MRFs. In *CVPR*, 2009.

[62] K. Konolige and S. Gutmann. Incremental mapping of large cyclic environments. *International Symposium on Computer Intelligence in Robotics and Automation (CIRA)*, pages 318–325, 2000.

[63] K Korosec. Autonomous car sales will hit 21 million by 2035, ihs says. *Fortune*, 6(07):2016, 2016.

[64] F. R. Kschischang, B. J. Frey, and H. . Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, Feb 2001.

[65] Benjamin Kuipers. Modeling spatial knowledge. *Cognitive science*, 2(2):129–153, 1978.

[66] Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[67] A. Kundu, K.M. Krishna, and C.V. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *ICCV*, 2011.

[68] Abhijit Kundu, K Madhava Krishna, and C. V. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *ICCV*, pages 2080–2087, 2011.

[69] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on*, pages 1250 –1257 vol.2, may 1994.

[70] Suha Kwak, Woonhyun Nam, Bohyung Han, and Joon Hee Han. Learning occlusion with likelihoods for visual tracking. In *ICCV*, pages 1551–1558, 2011.

[71] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[72] Xiangyang Lan, Stefan Roth, Daniel Huttenlocher, and Michael J Black. Efficient belief propagation with learned higher-order markov random fields. In *ECCV*. 2006.

[73] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. *Learning Hand-Eye Coordination for Robotic Grasping with Large-Scale Data Collection*, pages 173–184. Springer International Publishing, Cham, 2017.

[74] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[75] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[76] J. Little, C. Jennings, and D. Murray. Vision-based mapping with cooperative robots. In *Sensor Fusion and Decentralized Control in Robotic Systems*, volume 3523, pages 2–12, October 1998.

[77] J. S. Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2002.

[78] HC Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62, 1987.

[79] David JC MacKay. Introduction to monte carlo methods. In *Learning in graphical models*, pages 175–204. Springer, 1998.

[80] R. Madhavan, K. Fregene, and L. Parker. Distributed cooperative outdoor multirobot localization and mapping. *Autonomous Robots*, 17:23–39, 2004.

[81] A. Martinelli. Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination. *Robotics, IEEE Transactions on*, (99):1–17, 2012.

[82] A.James McKnight and A.Scott McKnight. Multivariate analysis of age-related driver ability and performance deficits. *Accident Analysis & Prevention*, 31(5):445 – 454, 1999.

[83] R. S. Merali and T. D. Barfoot. Occupancy grid mapping with markov chain monte carlo gibbs sampling. In *IEEE International Conference on Robotics and Automation*, 2013.

[84] R.S. Merali and T.D. Barfoot. Occpancy grid mapping with markov chain monte carlo gibbs sampling. In *ICRA*, 2013.

[85] Daniel Meyer-Delius, Maximilian Beinhofer, and Wolfram Burgard. Occupancy grid models for robot mapping in changing environments. In *AAAI*, 2012.

[86] A. Milan, S. Roth, and K. Schindler. Continuous energy minimization for multi-target tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(1):58–72, 2014.

[87] Piotr Mirowski, Matthew Koichi Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Koray Kavukcuoglu, Andrew Zisserman, and Raia Hadsell. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*, 2018.

[88] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[89] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[90] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[91] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*. NIPS, 2013.

[92] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[93] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[94] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[95] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121, Mar 1985.

[96] Hans P Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2):61, 1988.

[97] KS Nagla, Moin Uddin, and Dilbag Singh. Improved occupancy grid mapping in specular environment. *Robotics and Autonomous Systems*, 60(10):1245 – 1252, 2012.

[98] R.K. Namdev, A. Kundu, K.M. Krishna, and C.V. Jawahar. Motion segmentation of multiple objects from a freely moving monocular camera. In *ICRA*, pages 4092–4099, 2012.

[99] R.A. Newcombe, A.J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[100] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, June 2015.

[101] D. Nister. An efficient solution to the five-point relative pose problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(6):756–770, 2004.

[102] Junhyuk Oh, V. Chockalingam, S. Singh, and H. Lee. Control of memory, active perception, and action in minecraft. In *International Conference on Machine Learning*, 2016.

[103] Kemal Egemen Ozden, Konrad Schindler, and Luc Van Gool. Multibody structure-from-motion in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1134–1141, 2010.

[104] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *arXiv preprint arXiv:1702.08360*, 2017.

[105] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial intelligence*, 29(3):241–288, 1986.

[106] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Occlusion patterns for object class detection. In *CVPR*, pages 3286–3293, 2013.

[107] Bojan Pepik, Peter Gehler, Michael Stark, and Bernt Schiele. 3D2PM – 3D deformable part models. In *ECCV*, pages 356–370, 2012.

[108] J. Philip. A non-iterative algorithm for determining all essential matrices corresponding to five point pairs. *The Photogrammetric Record*, 15(88):589–599, 1996.

[109] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[110] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

[111] Brian Potetz. Efficient belief propagation for vision using linear constraint nodes. In *CVPR*, 2007.

[112] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. *arXiv preprint arXiv:1703.01988*, 2017.

[113] L. Quan and Z. Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774–780, 1999.

[114] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[115] Shankar R. Rao, Allen Y. Yang, S. Shankar Sastry, and Yi Ma. Robust algebraic segmentation of mixed rigid-body and planar motions from two views. *IJCV*, 2010.

[116] S.R. Rao, R. Tron, R. Vidal, and Yi Ma. Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In *CVPR*, pages 1–8, 2008.

[117] Loannis M. Rekleitis, Gregory Dudek, and Evangelos E. Milios. Multi-robot exploration of an unknown environment, efficiently reducing the odometry error. In *In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1340–1345, 1997.

[118] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *ICCV*, pages 765–773, 2015.

[119] R. Rocha, J. Dias, and A. Carvalho. Cooperative multi-robot systems: a study of vision-based 3-D mapping using information theory. *Robotics and Autonomous Systems*, 53:282–311, April 2005.

[120] S.I. Roumeliotis and G.A. Bekey. Distributed multirobot localization. *Robotics and Automation, IEEE Transactions on*, 18(5):781 – 795, oct 2002.

[121] M. Ruhnke, R. Kummerle, G. Grisetti, and W. Burgard. Highly accurate maximum likelihood laser mapping by jointly optimizing laser points and robot poses. In *ICRA*, pages 2812–2817. IEEE, 2011.

[122] J. Ryde, V. Dhiman, and R. Platt. Voxel planes: Rapid visualization and meshification of point cloud ensembles. In *IROS*, November 2013.

[123] J. Ryde and H. Hu. Mutual localization and 3D mapping by cooperative mobile robots. In *Proceedings of International Conference on Intelligent Autonomous Systems (IAS)*, The University of Tokyo, Tokyo, Japan, March 2006.

[124] Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

[125] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[126] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[127] Randall C Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The international journal of Robotics Research*, 5(4):56–68, 1986.

[128] N. Snavely, S.M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 835–846. ACM, 2006.

[129] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular SFM for autonomous driving. In *CVPR*, 2014.

[130] Shiyu Song and Manmohan Chandraker. Joint SFM and detection cues for monocular 3D localization in road scenes. In *CVPR*, pages 3734–3742, 2015.

[131] David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1, 2011.

[132] Henrik Stewnius, Christopher Engels, and David Nistr. Recent developments on direct relative orientation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 60(4):284 – 294, 2006.

[133] Kokichi Sugihara. Some location problems for robot navigation using a single camera. *Computer Vision, Graphics, and Image Processing*, 42(1):112 – 129, 1988.

[134] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[135] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

[136] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[137] Sebastian Thrun. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millenium*, 2002.

[138] Sebastian Thrun. Learning occupancy grid maps with forward sensor models. *Autonomous Robots*, 15(2):111–127, 2003.

[139] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[140] N. Trawny, X.S. Zhou, K. Zhou, and S.I. Roumeliotis. Interrobot transformations in 3-D. *Robotics, IEEE Transactions on*, 26(2):226–243, 2010.

[141] R. Tron and R. Vidal. A benchmark for the comparison of 3D motion segmentation algorithms. In *CVPR*, pages 1–8, 2007.

[142] R. Vidal and R. Hartley. Motion segmentation with missing data using powerfactorization and gpca. In *CVPR*, pages 310–316, 2004.

[143] R. Vidal, Yi Ma, and S. Sastry. Generalized principal component analysis (gpca). In *CVPR*, pages 621–628, 2003.

[144] Martin J Wainwright, Tommi S Jaakkola, and Alan S Willsky. MAP estimation via agreement on trees: message-passing and linear programming. *Information Theory, IEEE Transactions on*, 51(11):3697–3717, 2005.

[145] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[146] Wim Wiegerinck, Tom Heskes, et al. Fractional belief propagation. In *NIPS*, 2003.

[147] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.

[148] Christian Wojek, Stefan Walk, Stefan Roth, Konrad Schindler, and Bernt Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(4):882–897, 2013.

[149] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, 2007.

[150] Yu Xiang and Silvio Savarese. Object detection by 3D aspectlets and occlusion reasoning. In *ICCV Workshops*, pages 530–537, 2013.

[151] Jingyu Yan and Marc Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *ECCV*, pages 94–106, 2006.

[152] Shichao Yang, Yu Song, Michael Kaess, and Sebastian Scherer. Pop-up SLAM: Semantic monocular plane slam for low-texture environments. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 1222–1229. IEEE, 2016.

[153] Jonathan S Yedidia, William T Freeman, Yair Weiss, et al. Generalized belief propagation. In *NIPS*, 2000.

[154] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM on Pattern Recognition*, pages 214–223, 2007.

[155] X. S. Zhou and S. I. Roumeliotis. Determining 3-D relative transformations for any combination of range and bearing measurements. *Robotics, IEEE Transactions on*, PP(99):1–17, 2012.

[156] Xun S Zhou and Stergios I Roumeliotis. Determining the robot-to-robot 3D relative pose using combinations of range and bearing measurements: 14 minimal problems and closed-form solutions to three of them. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 2983–2990. IEEE, 2010.

[157] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017.

[158] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3357–3364. IEEE, 2017.

[159] M. Zeeshan Zia, Michael Stark, and Konrad Schindler. Towards scene understanding with detailed 3d object representations. *International Journal of Computer Vision*, 112(2):188–203, 2014.

[160] Muhammad Zia, Michael Stark, and Konrad Schindler. Are cars just 3D boxes? - Jointly estimating the 3D shape of multiple objects. In *CVPR*, pages 3678–3685, 2014.

[161] M.Z. Zia, M. Stark, and K. Schindler. Explicit occlusion modeling for 3D object class representations. In *CVPR*, pages 3326–3333, 2013.

[162] D. Zou and P. Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

APPENDIX A

Deep Reinforcement Learning

A.1 Hyperparameters

We use the Deepmind Lab environment to train our experiments. As mentioned previously, apple rewards are scattered throughout the maze and constitute a +1 reward. Goals constitute a +10 reward. An included wall penalty linearly penalizes the agent as it moves closer to the wall with the penalty being capped off at -.2 per frame. Our episodes are of fixed time length ending at 40 seconds each. The agent interacts with the environment at a rate of 30 frames per second. Each episode thus consists of 1200 frames of data coupled with the corresponding reward signals. Our mazes constitute an area of $900units \times 900units$ though we provide the tools to generate mazes to arbitrary dimensions.

Our A3C implementation is a modified version of OpenAIs open-sourced *universe-starter-agent*. RGB images are fed in to the network of dimensions $84 \times 84 \times 3$. 16 threaded agents are used for all experiments. We use a learning rate of 10^{-4} along with the *AdamOptimizer* to train our network. Our models train for a maximum of 10^8 iterations though we end them early if maximum reward saturates.

A.2 Benchmarking code

To motivate more comprehensive experimental evaluations of DRL-based navigation methods, we will be releasing all our trained models coupled with corresponding reward curves and videos of performance online. This will include completely reproducible evaluation sets wherein we display metric scores for all the trained models on the follow environments:

- the original training conditions
- the training conditions in the absence of apples and textures
- the 100 unseen testing maps
- the planning maps i.e. the square, wrench and goal map

We hope our work can also be utilized as a stepping stone for the creation of better generalized DRL navigation methods bypassing the needless amounts of time spent engineering the infrastructure necessary for these experiments. All our work will be available on github after the blind-review process is over.

A.3 Reward curves with different number of training maps

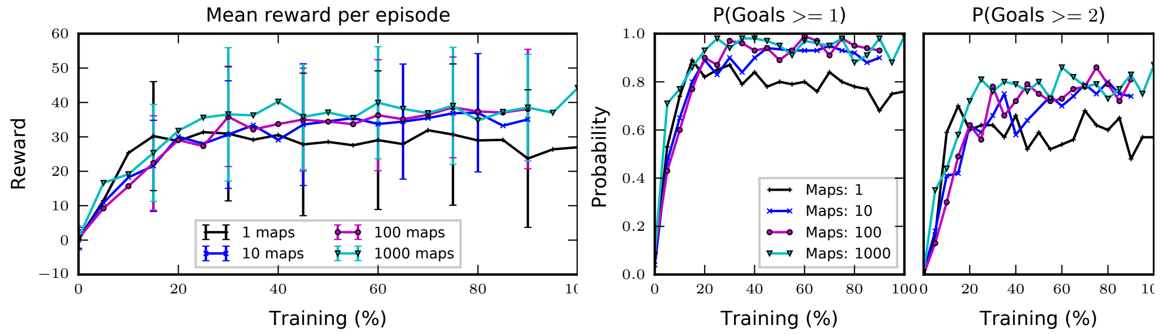


Figure A.1: Mean reward while tested on 100 unseen maps, while being trained on different number of training maps. Note that while training on 1000 maps eventually achieves high reward, it is only higher mean reward (44.2), training on 1 map hits the maximum (31) much faster.

APPENDIX B

Goal-conditioned Reinforcement Learning

Our algorithm 3 is different from HER [3] because it contains additional step-loss term $\mathcal{L}_{\text{step}}$ at line number 17 which allows the algorithm to learn even when the rewards received are independent of desired goal. Also in HER sampling (line 13), the algorithm recomputes the rewards because the goal is replaced with a pseudo-goal. Our algorithm does not need reward recomputation because the reward formulation does not depend on the goal and is not affected by choice of pseudo-goal.

Our algorithm is also different from Floyd-Warshall Reinforcement learning because it does not contain $\mathcal{L}_{\text{up}}$ and $\mathcal{L}_{\text{lo}}$ terms and contains the additional $\mathcal{L}_{\text{step}}$.

Algorithm 3: Path-reward reinforcement learning

```
/* By default all states are unreachable                                  */
Initialize networks $Q_m(s_i, a_i, g_j; \theta_Q)$ and $\chi(s_i, s_g; \theta_\chi)$ ;
Copy the main network to target network $Q_{\text{tgt}}(s_i, a_i, g_j; \theta_Q) \leftarrow Q_m(s_i, a_i, g_j; \theta_Q)$ ;
Initialize replay memory $M$ ;
for $e \leftarrow 1$ to $E$ do
    Sample $g_e \in \mathcal{G}$ ;
    Set $t \leftarrow 0$;
    Observe state $s_t$ and achieved goal $g_t$ ;
    /* Episode rollout                                                    */
    for $t \leftarrow 1$ to $T$ do
        Take action $a_t \leftarrow \epsilon$-greedy$(\chi_m(s_t, g; \theta_\chi))$ ;
        Observe $s_{t+1}, g_{t+1}, r_t$ ;
        Store $(s_t, g_t, a_t, s_{t+1}, g_{t+1}, r_t; g_e)$ in memory $M[e]$ ;
    /* Train                                                              */
    for $t \leftarrow 1$ to $T$ do
        HER sample batch $B = [(s_i, g_i, a_i, s_{i+1}, g_{i+1}, r_i; g_{i+f_i}), \dots, (s_b, g_b, a_b, s_{b+1}, g_{b+1}, r_b; g_{b+f_b})]$ from $M$ ;
        $\mathcal{L}(\dots) = 0$ ;
        for $b \in 1$ to $|B|$ do
            $(s_b, g_b, a_b, s_{b+1}, g_{b+1}, r_b, g_{b+f_b}) = B[b]$ ;
            /* Step loss                                                  */
            $\mathcal{L}(\dots) += (Q_m(s_b, a_b, g_{b+1}) - r_b)^2$ ;
            /* DDPG loss                                                  */
            $\mathcal{L}(\dots) += (Q_m(s_b, a_b, g_{b+f_b}) - r_b - \gamma Q_{\text{tgt}}(s_{b+1}, \chi_{\text{tgt}}(s_{b+1}, g_{b+f_b}; \theta_\chi), g_{b+f_b}))^2$ ;
        Update gradients for $Q_m$ and $\chi_m$ using loss $\mathcal{L}(\dots)$;
Result: $Q_m, \chi_m$
```

APPENDIX C

Notation

Probabilities $P(a|b, c)$ We use $P(a|b, c)$ as a compact notation of the form $P(A = a|B = b, C = c)$, which is the conditional probability of the random variable A taking value a given random variable B has value b and random variable C has value c. We assume an implicit definition of the random variable unless specified otherwise. If $a \in \mathcal{A}$, then the random variable A is defined define as $(\mathcal{A}, \sigma(\mathcal{A}), P_A)$, with $\sigma(\mathcal{A})$ is trivial sigma algebra over $\mathcal{A}$ such as a power set for discrete random variable and Borel set for a continuous random variable.

Sampling $z_t \sim P_o(z_t)$ is used to denote that z_t has been sampled from $P_o(z_t)$.

Stationary process and distributions A stochastic process generating a sequence of random variables X_t is called stationary if the joint probability distribution of a particular sequence is independence of the shift in time index i.e. $P(X_{t_1} = x_1, X_{t_2} = x_2, \ldots, X_{t_n} = x_n) = P(X_{\tau+t_1} = x_1, X_{\tau+t_2} = x_2, \ldots, X_{\tau+t_n} = x_n)$ for all $\tau, t_1, t_2, \ldots, t_n$ and n. The probability distributions that govern the stationary stochastic process are also stationary.