Improving Autonomous Navigation and Estimation in Novel Environments

by

Katherine Y. Liu

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2022

Author . . . .
Department of Aeronautics and Astronautics
October 5, 2021

Certified by . . . .
Nicholas Roy
Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by . . . .
Leslie Kaelbling
Professor of Computer Science and Engineering

Certified by . . . .
John Leonard
Professor of Mechanical and Ocean Engineering

Certified by . . . .
Kostas Daniilidis
Professor of Computer and Information Science, University of Pennsylvania

Accepted by . . . .
Jonathan How
R. C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Improving Autonomous Navigation and Estimation in Novel Environments

by

Katherine Y. Liu

Submitted to the Department of Aeronautics and Astronautics
on October 5, 2021, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Efficient autonomous navigation in novel environments is crucial to enable embodied agents to reach more sophisticated levels of autonomy. We are interested in improving autonomous navigation and estimation in unknown environments of vehicles carrying lightweight electro-optical sensor payloads. Due to sensing limitations, in non-trivial novel environments much of the geometric structure of the world has not yet been observed, leading to significant geometric ambiguity. Although collecting additional geometric information can reduce ambiguity, doing so is often at odds with the objectives of the mission. We propose to combine object-level semantic information and geometric information to tractably improve both navigation and estimation.

In this thesis, we present three contributions towards improving autonomous navigation in novel environments. We first improve navigation efficiency in novel environments by encoding useful navigation behaviors in a sampling distribution informed by partial occupancy and object-level maps. Recognizing that object-level estimation is challenging under the limited viewpoints available while navigating efficiently, we also develop two methods of building object-level representations online. In our second contribution, we improve the view-point efficiency of object-level SLAM with ellipsoid representations by introducing an additional texture measurement and semantic class shape prior. Finally, in our third contribution, we propose a novel method of deeply learned 3D object estimation that utilizes indirect image-space annotations and intra-class shape consistency to enable 3D object estimation from a single RGB image.

Thesis Supervisor: Nicholas Roy
Title: Professor of Aeronautics and Astronautics

Thesis Committee Member: Leslie Kaelbling
Title: Professor of Computer Science and Engineering

Thesis Committee Member: John Leonard
Title: Professor of Mechanical and Ocean Engineering

Thesis Committee Member: Kostas Daniilidis
Title: Professor of Computer and Information Science, University of Pennsylvania

Acknowledgments

I'd like to sincerely thank Nick Roy for building a lab whose research had an electrifying effect on me when I was still an undergraduate, and for giving me the opportunity to contribute to it. My desire to be a researcher with the Robust Robotics Group was something I was too intimidated to even admit out loud to my loved ones; that I was given a chance to follow through on my dream astonishes me to this day. Nick's mentorship has consistently challenged me to improve — not only as a researcher, but as a communicator and community member. His reminders to not let the perfect be the enemy of the good have also provided balance when it was sorely needed.

I'd also like to thank the members of my committee. Kostas Daniilidis, Leslie Kaelbling, and John Leonard have all provided incredibly valuable insights as members of my committee during my thesis journey, generously sharing both their time and immense combined expertise with me. Through all our meetings they have helped me to clarify and shape the core ideas in this thesis, and I have walked away from each and every meeting with new realizations. I'd also like to thank Kyel Ok and Valentin Peretroukhin for being thesis readers even after they became lab alumni.

I'd like to thank the Robust Robotics Group community. I have learned so much from you all. I must extend an especially warm thanks to Kyel Ok, who has spent countless hours with me in front of whiteboards, monitors, and hovering over quadrotors. I'm grateful for all the fun we had during our collaborations. I also want to thank Martina Stadler for her excellent contributions and methodical approach to research during our collaborations. Thank you also to Genevieve Flashpoher and Michael Noseworthy who were kind enough to provide thoughtful feedback on this thesis, in addition to being dear friends. I would also be remiss to not mention Bryt Bradley and Sophia Hasenfus for all of their practical help over the years.

I'd like to thank my friends and extended family, both in Boston and beyond. In particular, I am grateful for becoming roommates with Erin Looney by chance and gaining a lifelong friend. I am also grateful to Amanda Kelley, who has been on her own concurrent thesis journey in another state, for being a source of reliable cheer ever since we were in high school. The warm positivity radiated by the Ritopers and company has also meant so much to me over the years.

I'd like to thank my family for their never-ending support. My mother, who pulled off completing her Master's degree in Electrical Engineering while working full time and expecting her first child, has been an inspiration for balancing work ethic and making time for life's important things. My father, who is one of the smartest people I know, has always inspired me to tackle hard problems. Thanks to my brother and sister, Max and Lillian, for brightening my days.

Finally, I must thank Bryan for his role as a tireless motivator and steadfast partner. It's impossible to list all the ways that Bryan has supported me over the years ever since we met building an autonomous airplane the basement of the UCSD engineering building. He has been with me every step of the way, through five years of cross country long distance and a global pandemic, without falter. I am forever grateful for and in awe of his capacity for love, humor, and patience.

Contents

List of Figures

List of Tables

Abbreviations

ADI Average Distance Indistinguishable

BEV Bird's Eye View

CNN Convolutional Neural Network

CVAE Convolutional Variational Auto-Encoder

ICP Iterative Closest Point

IOU Intersection Over Union

MAB Multi-Arm Bandit

MAP *Maximum a Posteriori*

MLE Maximum Likelihood Estimate

PCA Principle Component Analysis

PnP Perspective-n-Point

PRM Probabilistic Roadmap

RRT Rapidly-Exporing Random Tree

SBMP Sampling Based Motion Planner

SLAM Simultaneous Localization and Mapping

TSDF Truncated Signed Distance Function

vSLAM Vision-Based Simultaneous Localization and Mapping

Chapter 1

Introduction

We would like to enable autonomous agents under size, weight, and power (SWaP) constraints to intelligently and efficiently model and navigate unknown, structured environments using electro-optical sensors. Such capabilities are crucial to empowering the meaningful operation of robotics into a diverse range of applications from search and rescue to last-mile autonomous delivery. Although there is tremendous interest in autonomous systems capable of operating in environments not known *a priori*, doing so efficiently remains an open research problem.

Electro-optical sensors such as color and depth cameras are attractive options for SWaP constrained agents compared to more conventional laser range finders, but also have significant sensing limitations. Color (RGB) cameras and combined color and depth cameras (RGB-D) have become increasingly available as a commercial off-the-shelf products, and can be used to build relatively lightweight sensing payloads. Both can provide detailed geometric and extra-geometric[1] information about the surrounding environment. Depth cameras in particular have been used with great success to build dense geometric representations that track occupied and free space [65, 149]. However, a property of electro-optical sensors is that they have limited ranges, fields

[1]We use the term *extra-geometric* to refer to properties that are not strictly geometric, such as object class. Loosely, we consider geometric information to correspond to the *where* of objects, and extra-geometric information to correspond to the *what*. We categorize properties such as size and shape as geometric as they can define the parts of the world the object occupies. In the remainder of this document, we will also interchangeably use the term *semantic* to align with previous work.

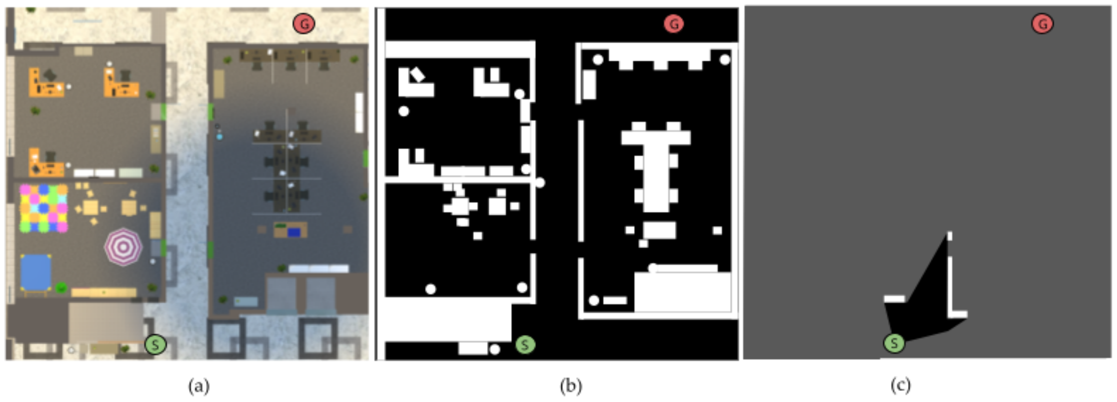


Figure 1-1: Example illustration of geometric ambiguity during navigation. Given detailed *a priori* geometric knowledge of the office environment shown in (a), an occupancy representation of the environment can be constructed (b), where black indicates free space and white indicates areas of the world where the robot would be in collision with the environment. However, when navigating in unknown environments, the geometric representations of the world must be built online. Due to the limited range and field of view of electro-optical sensors as well as their line-of-sight measurement properties, much of the environment relevant to navigation has not been observed yet, as indicated by the grey regions in (c).

of view, and are non-penetrative, i.e., they do not see through opaque structure in the environment. These sensing limitations result in portions of the environment remaining unobserved as the robot attempts to make progress to the goal. While RGB sensors generally have longer ranges than depth cameras, because they do not provide direct 3D measurements it is more challenging to use them to build dense geometric representations. In large-scale spaces [88], where the goal is far beyond the sensing range of the robot, a dense geometric representation built online by electro-optical sensors is likely to be quite incomplete. Figure 1-1 illustrates a scenario where the goal of a robot navigating an office environment is beyond the range of the known map, and very little dense geometry is available to the robot to make a navigation decision.

Incomplete dense geometric representations can introduce significant and often unmodelled geometric ambiguity into navigation, rendering efficient autonomous navigation particularly challenging in novel environments. Consider a robot, equipped with a single forward facing RGB-D sensor, tasked with navigating to a specific location in a novel environment. The robot can utilize a sequence of depth images to build an occupancy map that tracks occupied space, free space, and unobserved space [65]; such a map supports the planning of non-colliding trajectories. For planning purposes, when occupancy information is unavailable that part of the world is often optimistically assumed to be unoccupied. The optimistic free-space assumption causes many routes through the environment to appear equally promising due to the lack of any evidence that they may pass through occupied regions. As a result, the robot may attempt to follow trajectories only to find them very frequently unusable as more of the environment comes within range, resulting in inefficient navigation behavior. The consequences of geometric ambiguity are especially evident in large-scale spaces, where poor decisions made given little relevant geometric information early in the navigation process can add significant distance to the overall distance travelled. Furthermore, because many trajectories appear equally promising under geometric ambiguity, computational resources are wasted evaluating routes unlikely to be successful or low cost, leading to computational inefficiency and therefore higher latency. Although exploring to construct a more complete map online can resolve ambiguity, data collection usually negatively impacts the objective of efficient operation.

There is compelling evidence that humans are able to quickly estimate potential routes and navigate without focusing only on building highly accurate *a priori* geometric maps. Chase [26] found that expert taxi drivers operating in Chicago were not appreciably better than novice drivers at drawing accurate maps of the city, despite their ability to plan more efficient routes. Bonner and Epstein [15] presented human test subjects with 2D images of artificially generated rooms with traversable (doors) and untraversable (paintings) objects. Their experiments show that humans estimate navigation affordances even when performing auxiliary tasks. These examples suggest that looking beyond dense, accurate geometric models may be fruitful

We propose to mitigate the effects of geometric ambiguity by incorporating object-level abstractions that encode both the extra-geometric *what* as well as the geometric *where*, which have proven useful in several applications by providing sparse and compact structure. For example, reasoning about the world with knowledge of object class can improve the robustness of loop closures [171] and place recognition [48].

Knowledge of object class enables the prescription of object affordances (i.e., what actions can be applied to an object) for task and motion planning applications [75], and provides useful models for tracking [86]. The higher-level navigation implications of object-level elements in the environment can be quite intuitive. For example, the semantic bounding box detection of a door or an exit sign may help the robot navigating an office environment resolve significant geometric ambiguity by providing information about which modes of navigation are more likely to lie on the optimal trajectory; rooms are often exited by travelling through doors and if the robot is going outdoors it should prefer the door with the exit sign.

However, while the advent of low-cost semantic measurements such as 2D bounding box detections [68, 104, 135] as a commodity technology has made semantic measurements increasingly accessible to even SWaP-constrained vehicles, we argue that two key difficulties prevent wide-spread use of object-level semantics to inform navigation in novel environments. First, despite their intuitive usefulness, it is not immediately obvious how to integrate object-level representations within planners designed for unknown environments. Second, it is challenging to project readily available 2D image-space semantic detections to 3D world coordinates, especially under challenging sensor motions typical of efficient autonomous navigation and without extensive *a priori* geometric information.

In this thesis, we improve the efficiency of autonomous navigation by combining geometric and semantic information to develop algorithms for building and utilizing object-level representations online. In order to use object-level representations we must not only develop novel planning algorithms capable of incorporating object-level cues into navigation decisions, but also develop methods for building such representations online with limited information. In the following sections, we describe in more detail the key challenges we seek to make progress on, and outline our proposed approaches.

1.1 Incorporating semantics into navigation

In general, the goal of motion planning is to find a feasible trajectory from an initial state to a terminal state. Let $\mathcal{X} \subset \mathbb{R}^d$ denote the state space of the robot, where d is the dimension of the state. It is also commonly assumed that $\mathcal{X}$ is composed of occupied and unoccupied states, $\mathcal{X}_{free}$ and $\mathcal{X}_{occ}$, and that there exists a known mapping from every $\mathcal{X}$ to its occupancy value. In practice, when occupancy information is unavailable, the state is often assumed to be in $\mathcal{X}_{free}$. We denote a trajectory to be a continuous mapping through $\mathcal{X}$, i.e., $\lambda : [0,1] \to \mathbb{R}^d$, the start and goal as $x_s, x_g \in \mathcal{X}$, and the set of all paths as Λ. The planning problem can then be formally stated as the minimization of some cost c function that maps a trajectory to a scalar positive cost, i.e., $c : \Lambda \to \mathbb{R}_{\geqslant 0}$, subject to constraints:

$$\begin{aligned} & \lambda^* = \underset{\lambda \in \Lambda}{\arg\min}\, c(\lambda) \\ \text{s.t.} \quad & \lambda^*(t) \in \mathcal{X}_{free}\ \forall t \in [0,1] \\ & \lambda^*(0) = x_s, \lambda^*(1) = x_g. \end{aligned} \tag{1.1}$$

In this thesis, we will take cost function to be the length of the trajectory, although other objectives such as minimizing control effort are also common. To support real-world autonomous navigation, we are interested in not only the efficiency of the overall distance travelled during the mission, but also focusing the search for trajectories.

The formulation of Equation 1.1 is notably dependent on geometric constraints, and for good reason. Many of the properties of navigation that are of interest in autonomous navigation are fundamentally grounded in geometry, such as the non-collision constraint (intersection of the geometry of the robot and geometry of the environment) and the minimum distance objective (the total geometric distance covered by the robot over the course of navigation). However, unlike applications where *a priori* maps can be obtained by first exhaustively mapping an environment, and then planning collision free trajectories within that known map, navigation in unknown environments requires making decisions under incomplete information about the state of the world. We are interested in incorporating object-level representations to inform the planning process.

Although a first-order approach to incorporating object-level information into navigation algorithms would be to manually specify the effect of objects on navigation, such a method is difficult to scale. For example, a system designer might consider implementing a rule that doorways are useful for navigation. However, traversing through doors is not always the best policy. Consider a scenario where the robot is in a hallway and the goal is over a kilometer away. In such a case, it is likely that entering a room via a doorway will result in the robot having to turn around and eventually return to the hallway. However, the utility of object relationships can be a complex function that may change conditioned on the ultimate objective; if there is an exit sign adjacent to the doorway, the doorway likely leads outdoors. Therefore, if the goal is outdoors, going through the doorway is a strategy likely to lead efficiently to the goal. As the number of objects in the world increases, enumerating rules to explicitly cover all potential implications induced by the interaction between objects, geometry, and useful trajectories becomes increasingly impossible. Even if total enumeration is possible, constraining actions on noisy or mis-specified object information could lead to unnecessary inefficiencies. For example, a planning approach that plans solely using a topological map of doors would suffer if a door on the optimal trajectory was not detected and therefore not under consideration.

To improve autonomous navigation, we take inspiration from the rich literature of sampling-based motion planners (SBMPs) and planning in novel environments. Modern sampling-based motion planners can reduce computational burden by building potential trajectories via random sampling. It has been shown that the efficiency of SBMPs can be dramatically improved by intelligent sampling strategies [69, 70], but the development of efficient sampling algorithms to speed up planning has largely addressed problems given dense geometry in fully known environments. Various methods have been proposed to reduce the overall distance travelled when navigating in novel environments, but usually aim to predict quantities to allow for better planning decisions, rather than focusing on more efficient computation.

In Chapter 3, we show how to augment incomplete geometric information with partial object-level maps to improve autonomous navigation in novel environments. Our key insight is to combine the computational power of randomized motion planners with higher-level semantic information via a learned sampling distribution, enabling intelligent navigation in structured, unknown environments. We propose optimizing a predictive sampling distribution inferred from dense, local geometric representations that track unobserved space, explicit object-level contextual cues both within and beyond the range of dense geometry, and information about the goal. Our approach enables SBMPs to not only find plans quickly, but in certain environments plan trajectories that are more likely to reach the goal despite incomplete geometric information. Rather than pre-specifying navigation rules and heuristics over our representations, we show that a Convolutional Neural Network (CNN) can be leveraged to synthesize multimodal map information into a proper sampling distribution. Furthermore, we demonstrate that learning a sampling distribution over geometric and semantic information improves navigation results in unknown environments.

Our proposed approach to improving autonomous navigation relies in part on an object-level representation that encodes both the object class and geometric properties of the object in the world. In recent years, 2D object detections from monocular images have become increasingly accessible for use on robotic platforms [134, 68], providing measurements of objects in the image plane (*where*), but also object class predictions (*what*). However, to inform our navigation approach, we would like to use these 2D image-space measurements and monocular sensors to acquire 3D estimates of objects along with their class labels. Additionally, we would like for the method to provide estimates online as the vehicle executes efficient navigation maneuvers, which often limit the diversity of views of the object. In the next sections, we discuss the technical challenges induced by these requirements and our proposed approaches. In particular, we make improvements to two main thrusts of approaches to 3D object estimation to support providing contextual information for navigation: object-level SLAM and deep 3D object estimation.

1.2 Improving Object-Level SLAM with Limited Measurements

To provide object-level information for autonomous navigation, we first formulate object estimation as a simultaneous localization and mapping (SLAM) problem. Object-level SLAM provides a probabilistic framework to estimate 3D object landmarks by fusing sensor measurements of objects collected over time. Given a series of potentially heterogeneous noisy measurements, object-level SLAM aims to solve for the most probable robot and object states. For example, one may consider solving for the most probable J objects $\boldsymbol{O}$ and T robot states $\boldsymbol{x}$ conditioned on K observed measurements $\boldsymbol{y}$:

$$\boldsymbol{x}^*_{0:T}, \boldsymbol{O}^*_{0:J} = \underset{\boldsymbol{x}_{0:T}, \boldsymbol{O}_{0:J}}{\arg\max}\, p(\boldsymbol{x}_{0:T}, \boldsymbol{O}_{0:J} | \boldsymbol{y}_{0:K}). \tag{1.2}$$

It is common to take the state as $\boldsymbol{x} \in \mathrm{SE}(3)$, although other representations are certainly possible. In this thesis, we focus on building lightweight object representations from monocular images (i.e., $\boldsymbol{y} : \Omega \in \mathbb{N}^2 \to \mathbb{R}$, where Ω is the image pixel domain) and 2D object detections (parameterized by two corner pixel coordinates, i.e., $\boldsymbol{y} \in \Omega^2$). We let $\boldsymbol{O} \in \mathbb{R}^p$, where p depends on the dimension of the representation. For example, a 3D bounding box can be represented by the location of eight corner points in $\mathbb{R}^3$. In contrast, a mesh model object representation can be represented by a set of edges and vertices. To avoid the computational cost of building dense models of objects or requiring a pre-processing step where a detailed geometric model is built before online estimation, we consider an *approximate* geometric model that is expressive enough to capture general object properties such as pose and size.

A key challenge in monocular object-level SLAM is that limited diversity in views of an object can result in geometrically ambiguous estimation processes. Geometric ambiguity in object estimation when using approximate geometric models can manifest as many potential object configurations of size, position and rotation being possible due to perspective projective ambiguity. For example, if a bounding box

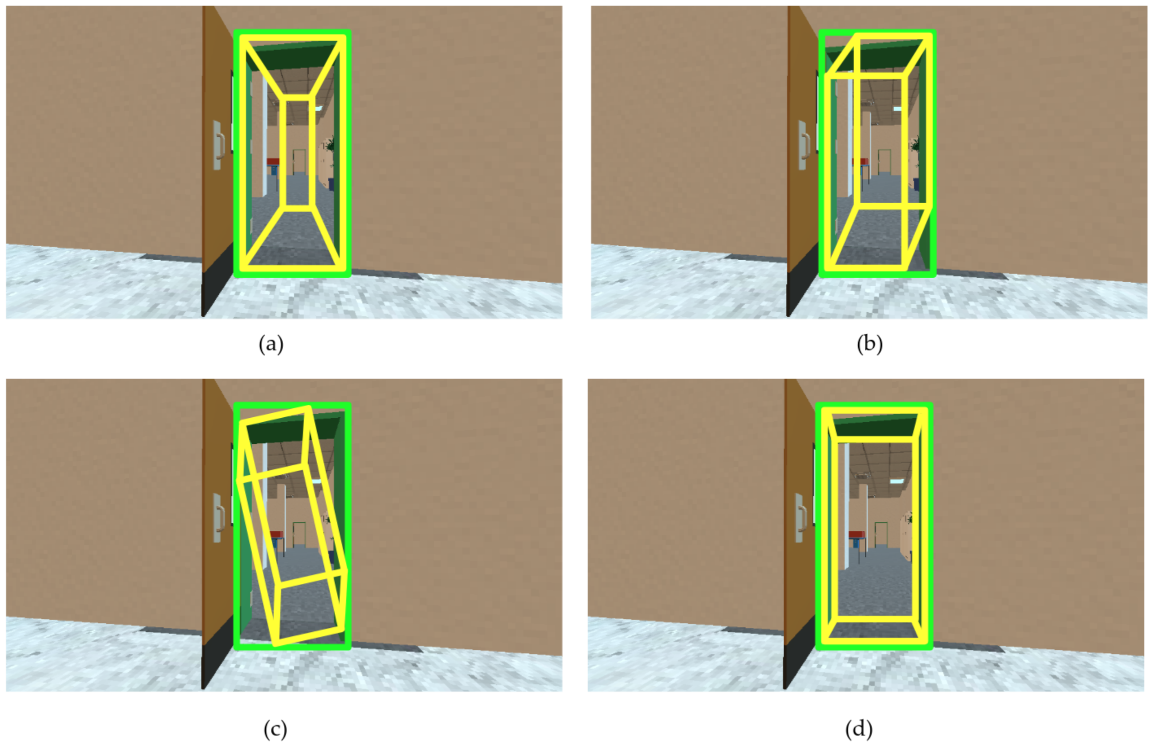


Figure 1-2: Examples of geometric ambiguity in fitting 3D bounding boxes to a detected 2D bounding box. There are many 3D bounding boxes (yellow) that may give rise to the same 2D image-space bounding box detection (green); in this example, (a)-(c) are poor estimates for the doorway compared to (d). There is significant geometric ambiguity when considering the 2D bounding box as a geometric measurement only.

measurement is considered purely as a 2D geometric measurement of a 3D object, there are many potential 3D bounding boxes[2] that satisfy the projection, as shown in Figure 1-2. Although the estimate may agree with the received measurements, it may not be accurate enough to be useful for online navigation.

Therefore, under the challenging camera motions typical of efficient navigation such as straight line, low baseline maneuvers, it can be difficult to obtain adequate measurements to sufficiently mitigate geometric ambiguity. When using only 2D bounding boxes to constrain all object models, which is exacerbated by common types of vehicle motions such as straight line motions that do not generate diverse viewpoints of the objects. The observability problem is similar to a recognized problem in point-based monocular SLAM [114]. Existing approaches to object-level SLAM

[2]3D bounding boxes are a common approximate geometric model.

often overcome the difficulty of geometric ambiguity by relying on *a priori* geometric models of the objects [150, 154] or camera trajectories with diverse views of objects [118].

Although collecting additional measurements can mitigate geometric ambiguity in the object-level estimation process, doing so is often at odds with the objectives of the mission. For example, various methods have been proposed in other contexts to determine the next-best-view to collect of an object [12, 175, 41, 8] to reduce uncertainty, but detours solely for the purpose of collecting additional measurements of objects can result in longer trajectories overall. In contrast to mapping-centric missions, where map accuracy and coverage is a priority, we would like to obtain object estimates with as little disruption to the navigation process as possible. We therefore focus our efforts on improving the performance of object-level SLAM under the types of trajectories typical of efficient autonomous navigation, rather than tightly coupling estimation and navigation.

In Chapter 4 we improve performance of online object-level SLAM with dual ellipsoid representations under challenging camera motions by incorporating semantic shape priors with bounding box and texture measurements. We choose the dual ellipsoid representation as our approximate geometric representation for its convenient mathematical form.[3] Our key insight is to extract additional measurements from RGB images: in addition to exploiting texture-based information, we also propose incorporating a shape prior based on object class. We show that by triangulating points on the surface of the object, we can induce a useful measurement to aid in ellipsoid estimation. The shape prior is a heuristic suitable for many object classes of interest such as people and cars and allows for single-shot initialization of new objects, rather than the delayed initialization of the previous state of the art approach [118]. To demonstrate the usefulness of our approach under challenging camera motions, we test on both simulated and real-world flight sequence data from an autonomously navigating quadrotor.

[3]More detailed discussion of the benefits of the dual ellipsoid representation can be found in Section 2.2.1.

1.3 Reducing the Data Burden of Deep Object Estimation

To enable 3D object estimation from a single view, we formulate object estimation as a deep learning problem. Rather than requiring several measurements fused over time, deep learning methods for 3D object estimation can potentially provide object estimates from a single sensor image. Deep learning based approaches to 3D object estimation achieve lower measurement latency by relying on large datasets of annotations to learn mapping functions from input images to object properties. Specifically, many deep learning approaches decompose object estimation into two stages: an offline training stage and an online query stage. Let Φ denote a hyper-parametric function approximator in the form of a convolutional neural network that takes as input a measurement $\boldsymbol{y}_I$, outputs object representation $\boldsymbol{O}$, and is parameterized by $\boldsymbol{\tau}$. In the offline training stage, given a dataset that pairs a series of N inputs $\{\boldsymbol{y}_{I_0}, ..., \boldsymbol{y}_{I_N}\}$ with annotations[4] $\{\boldsymbol{y}_{L_0}, ..., \boldsymbol{y}_{L_N}\}$, the parameters $\boldsymbol{\tau}$ of Φ are first optimized using training data, i.e.,

$$\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau}} \sum_{i=1}^{N} \boldsymbol{\mathcal{F}}(\Phi(\boldsymbol{y}_{I_i}, \boldsymbol{\tau}), \boldsymbol{y}_{L_i}) \tag{1.3}$$

where $\boldsymbol{\mathcal{F}}$ is a function that measures how well the predicted object agrees with the annotations and lower values indicate better agreement. The input is commonly taken to be an image (i.e., $\boldsymbol{y}_I : \Omega \in \mathbb{N}^2 \to \mathbb{R}$), while $\boldsymbol{y}_L$ varies more widely between approaches. At inference time, the optimized parameters $\boldsymbol{\tau}^*$ are used to define the model Φ and a prediction is simply defined by applying the resulting function to sensor measurements $\boldsymbol{y}_I$ acquired online, rather than an inference process as in SLAM. However, unlike SLAM formulations, where the form of the objective function is generally well understood (i.e., as the conditional probability), a significant branch in the study of deep learning approaches to object estimation involves the design of an appropriate objective function $\boldsymbol{\mathcal{F}}$.

[4]We use the terms *annotations* and *labels* interchangeably.

Deep learning approaches reduce geometric ambiguity at inference time by relying on large datasets of direct labels or *a priori* models. For example, one way to overcome the the ambiguity shown in Figure 1-2 is to build a dataset of ground-truth 3D bounding box annotations and images, and optimize the network to take as input images and output the correct 3D bounding box. Given many examples, such a supervised learning problem could in principle learn to predict reasonable 3D bounding boxes online because it has seen many examples of doorways paired with pose and size annotations. *A priori* object models are also an avenue for reducing geometric ambiguity; given a detailed model of the doorway, the estimation process need only estimate the pose of the object, therefore removing ambiguity in shape.

However, a significant practical hurdle to training state-of-the-art deep 3D bounding box estimators is collecting a sufficiently large dataset of 3D bounding box labels or *a priori* models. Unfortunately, in contrast to indirect 2D image plane annotations such as 2D bounding boxes or pixel-wise segmentation, 3D annotations are more difficult and expensive to obtain, as in most cases they require additional information or sophisticated interfaces [4, 185, 96]. Additionally, building detailed *a priori* geometric models such as CAD models is a labor intensive task requiring expert domain knowledge. Building accurate models from sensor data is possible, but can require specialized hardware [22] or careful procedures [150]. The onerous effort required to build 3D datasets for new objects hinders our ability to use deep 3D object estimation to provide object-level contextual information for autonomous navigation.

In Chapter 5, we introduce a novel, weakly supervised, deep learning framework for estimating 3D objects monocular 3D object estimation, which does not require expensive 3D annotations or detailed *a priori* information. We assume a parallel 2D object detection process, and "lift" the detections to 3D using our network. Our key insights are to approximate objects as ellipsoids, which allows us to estimate object properties using indirect image-space annotations on 2D images as a weak supervisory signal and to exploit low shape variance induced by semantic class.[5] By

[5]The nomenclature and categorization of deep learning approaches is an evolving topic. We refer to our method as weakly supervised because we do not rely on direct labels of the quantity of interest, but instead rely on indirect labels, such as 2D bounding box annotations.

approximating object geometry with ellipsoids, we can exploit differentiable geometric and algebraic relationships between ellipsoids and 2D annotations to enable a weakly supervised learning process and significantly lower the annotation burden for deeply learned methods. Assuming that the objects of interest have low size variation, we also impose a cost on intra-class shape variance to further mitigate geometric ambiguity. Given online instance segmentation and point-cloud information, our object estimates can be further optimized online. We present quantitative results on a simulated dataset with access to ground-truth annotations, and qualitative results on a real-world dataset without ground-truth annotations. Finally, we present an integration of the planning method proposed in Chapter 3 and our weakly supervised learning approach, demonstrating the suitability of the 3D object estimation pipeline for informing autonomous navigation.

1.4 Combining Semantics and Geometry

In this thesis, we posit that combining object-level semantic information and geometric information can tractably improve navigation and estimation. As we have discussed in this chapter, both navigation and object-level estimation algorithms can suffer from geometric ambiguity when navigating in novel environments. With the understanding that to use object-level representations to improve autonomous navigation we must also improve object-level estimation, we tackle both planning and estimation — employing approaches characterized by the theme of combining semantics and geometry. Rather than viewing semantics as a replacement for detailed geometry, we continue to rely on the strengths of geometric representations and seek to incorporate semantics to mitigate its weaknesses. In each case, we improve efficiency along some pertinent metric[6] by encoding the effects of semantic information into our geometric optimizations as cost functions and priors. We seek to exploit both dense geometry, which can provide precise information but is difficult in general to

[6]In Chapter 3 we consider the sample efficiency of a SBMP as well as the efficiency of the overall distance travelled. In Chapter 4 we improve the measurement efficiency of object-level SLAM. In Chapter 5 we improve the annotation efficiency of deep object estimation.

obtain at scale, and object-level semantic information, which is compact and imposes structure on the world, but can be relatively sparse.

Our approach to the synthesis of hybrid semantic and geometric representations has several important benefits. First, we show that semantics can provide powerful heuristic cues to inform geometric optimization when incomplete information induces ambiguity, and thus improve the efficiency of autonomous navigation and estimation. Second, because semantic abstractions are leveraged to bias geometric solutions instead of incorporated as hard constraints, additional geometric measurements can potentially overcome incorrect heuristics. Under our proposed framework, we guide the optimization of geometric solutions even when geometric information is incomplete, and can improve the solution as additional geometric measurements are obtained.

1.5 Statement of Contributions

Our contributions are as follows:

- A novel approach for improving planning efficiency in unknown environments by encoding promising navigation strategies in learned sampling distributions informed by partial occupancy and object-level maps (Chapter 3).
- Improvements to the measurement efficiency of object-level SLAM with ellipsoid landmarks under challenging camera motions by incorporating semantic shape priors and introducing a novel geometric texture measurement model (Chapter 4).
- A novel weakly-supervised deep learning framework for monocular 3D object estimation, which combines geometric properties such as stereo projective consistency with semantic properties such as intra-class consistency to avoid requiring expensive 3D annotations or detailed *a priori* geometric information (Chapter 5).

Portions of Chapter 2 were orginally presented in Liu [102]. The works presented in Chapter 3, Chapter 4, and Chapter 5 were originally presented in Liu and Stadler et al. [100], Ok and Liu et al. [120], and Liu et al. [101], respectively.

Chapter 2

Background

In this chapter, we review existing approaches to autonomous navigation and object-level estimation, with the objective of establishing context and identifying deficiencies that motivate our approaches. To this end, we assess both the computational and operational efficiency of existing online navigation methods, as well as the suitability of existing object-level estimation methods for supporting online applications. We are especially interested in algorithmic performance under limited *a priori* geometric information.

Beginning with Section 2.1, we discuss several approaches to solving the autonomous planning problem, focusing on methods that attempt to reduce computational complexity via representational choices (Section 2.1.1). We then discuss various methods of lowering the latency of the planning process using search heuristics (Section 2.1.2). Section 2.1.3 delves into approaches designed for navigation in uncertain environments. The section closes with methods specifically combining semantics and geometry to enable more efficient navigation in uncertain environments (Section 2.1.4).

Next, we discuss methods for object-level estimation, with a particular focus on methods for online estimation suitable for supporting efficient navigation. Section 2.2.1 describes several common object models. We describe various methods of estimating object representations, including regression and classical detection (Section 2.2.2), SLAM (Section 2.2.2), deep learning (Section 2.2.4), and hybrid methods (Section 2.2.5).

2.1 Autonomous Navigation

Robot navigation from one location in the world to another is a fundamental component of a wide variety of embodied robotic systems. In this thesis, we are primarily interested in the problem of moving from some starting location to some goal location defined in a known reference frame. Although goals specified in non-geometric terms are certainly a natural way for humans to communicate goals, the "point-A-to-point-B" planning problem is useful in many contexts where the exact coordinates of a goal can be specified and may serve in the future as a valuable building block for more sophisticated systems.[1]

In this section, we unpack various approaches to solving Equation 1.1 in the context of realistic real-world robotic systems. To deploy a planning system that centers on solving Equation 1.1 on realistic real-world robotic systems, the solutions must be found fast enough to enable real-time planning[2] under the computational constraints of the vehicle[3] in environments with potentially complex environmental geometry. We explore two major thrusts in solving Equation 1.1 quickly – the representation of the planning problem (Section 2.1.1), and the search heuristics used to find a solution (Section 2.1.2). Additionally, a major limitation in many approaches to solving the planning problem is the assumption of a known, deterministic mapping from a state to a binary classification of whether the state is in collision with the environment or not. Therefore, while the frontmatter of this section focuses largely on algorithms developed with assumed access to *a priori* geometric maps only, in latter portions we turn our attention to methods explicitly designed for navigation under uncer-

[1]Under the nomenclature of Anderson et al. [6], we focus on the PointGoal problem, rather than the ObjectGoal or AreaGoal objectives. "Go to the couch" is, for example, an ObjectGoal that has geometric implications but is not specified with exact coordinates, potentially because coordinates may not be known *a priori*.

[2]We will occasionally refer to this property as *op-tempo* planning, for planning at operational tempo.

[3]Although off-board computational infrastructure might enable plans to be computed using resources that do not have to be carried by the vehicle, such systems can struggle in environments and missions where communication is degraded.

tainty (Section 2.1.3). Finally, we focus on recent methods that consider semantics in addition to geometry to further improve planning outcomes (Section 2.1.4).

2.1.1 Speeding up Search via Representation

The chosen representation of a planning problem can have enormous practical and theoretical impact on the quality and tractability of autonomous navigation. In this section, we discuss three families of geometric representations: continuous, deterministic discretized graphs, and stochastic graphs.

Continuous Methods

Some methods solve Equation 3.1 by assuming a trajectory form, and iteratively optimizing an initial solution [139, 131] or solving a mixed integer linear program [152]. For example, both Ratliff et al. [131] and Richter et al. [139] assume a set number of waypoints and define trajectory properties between the waypoints. Ratliff et al. [131] allow the waypoints to move during the continuous optimization, while Richter et al. [139] update only the polynomials connecting the waypoints, or repair trajectory segments that come into collision with the environment by adding waypoints to further constrain the trajectory. Although the trajectories are discretized in this fashion, the parameters under optimization usually take on continuous values, enabling gradient-based optimization methods and in some cases constrained optimization frameworks. While these approaches are promising for agents with complex dynamics, they usually require initial trajectory estimates that can be difficult to obtain in complex environments without auxiliary planning processes. For example, Richter et al. [139] assume an initial trajectory provided by a rapidly exploring random tree, which we discuss in Section 2.1.1.

Deterministic Discretized Graph Methods

Discretized approaches to planning divide the state space of the robot into discrete intervals, thereby reducing the number of possible states considered by the robot

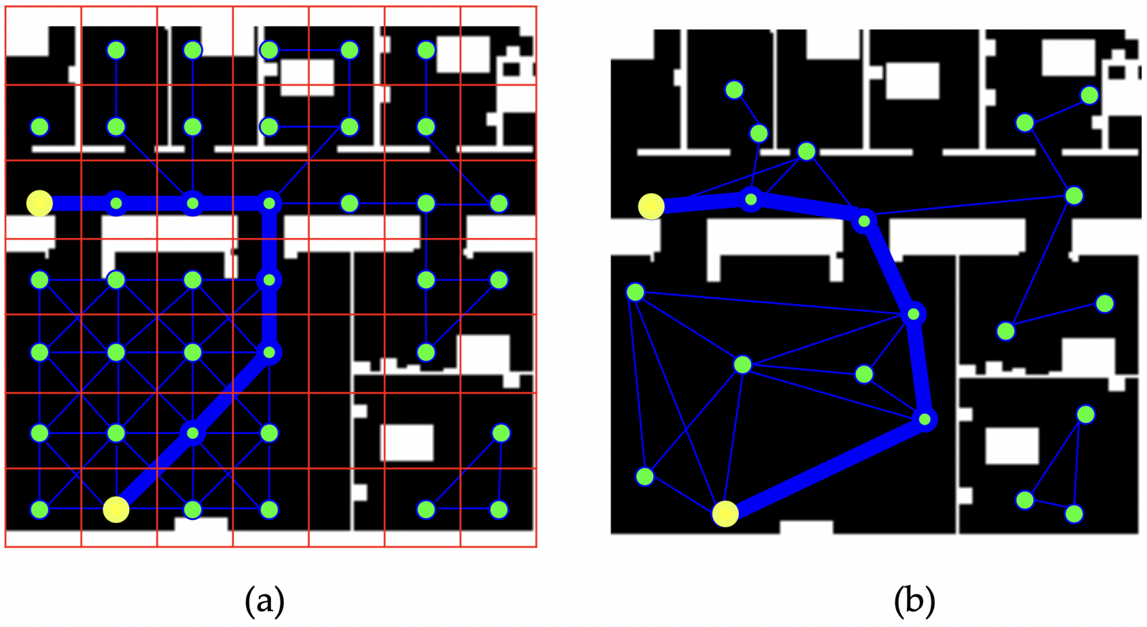


Figure 2-1: Illustration of a deterministic discretized graph vs. a stochastic graph. The vertices are represented by green circles and edges by blue lines. The start and goal are indicated with yellow vertices, where the ordering is irrelevant. The shortest valid path in each representation is indicated by bolded blue outlines. In the underlying occupancy map, black indicates free space and white indicates occupied space. (a) Deterministic discretized graph methods assume a deterministic (and often constant) discretization over the state space, illustrated by the red grid. Discretized graph methods require several implementation choices, including the discretization strategy and how states are connected for navigation purposes. In this example, the discretization is assumed to be constant and an action set comprising of the four cardinal directions as well as four diagonal directions is assumed. Each vertex is assumed to fall within the center of the discretization, and edges are assumed to be straight-line paths and removed if they are in collision with the environment. (b) Stochastic graph methods randomly sample vertices and typically use an edge connection strategy to add only edges to the graph that represent collision-free paths. In some cases, stochastic graphs can find solutions similar to discretized methods using a smaller number of nodes, although in this illustration both the discretized method and the stochastic method fail to find the shortest path between the two yellow vertices.

during planning from infinity to some finite, countable number of states. Consider, for example, a discretization of a robotic agent operating in two dimensions. With a 10×10 meter environment and 1 meter discretization, the state space will have 100 unique states, rather than the potentially infinite set of states considered by continuous methods. Many early approaches to planning therefore chose graph representations over discretized state spaces and applied graph search algorithms to find trajectories.

The state space discretization defines vertices V of a graph, and the edges E are defined by a set of valid actions u, where $f(x, u)$ maps from an initial state to the state after applying action u. Vertices have the same domain as the robot state. As in LaValle [95], we define the navigation graph G as follows[4]:

$$G = \{V, E\}, e_{i,j} \in E \iff \exists u \in U(x) \text{s.t.} v_j = f(v_i, u), \tag{2.1}$$

where $U(x)$ defines actions that can be taken from the state x. To manage the branching factor of the graph, the number of potential actions is limited, i.e., by allowing navigation between neighboring nodes using only the four cardinal directions. An example of a discretized planning approach is given in Figure 2-1a. After discretizing the environment to build a graph representation of the problem in Equation 1.1, graph search algorithms such as breath first search (BFS) or Dikjstra's [36] shortest-path algorithm may be applied.

State space discretization is the backbone of many classical planning algorithms, but such approaches can still suffer from the curses of scale and dimensionality, quickly becoming computationally infeasible. As the scale of the environment increases, the size of the state space increases rapidly. Doubling the extents of our hypothetical environment from 10×10 meters to 20×20 meters while holding the discretization constant results in a four-fold increase in the number of unique states; in this case the scale of the environment increases the state space quadratically. Adding a dimension to the consideration of the algorithm (consider a 3D environment of $10 \times 10 \times 10$ meters) results in a 10-fold increase in the number of unique states.

[4]LaValle [95] refers to the graph as defined as the *state transition graph*; we will refer the graph as the navigation graph.

Another practical difficulty in using discretized methods is that the discretization is often considered constant, and tuned to meet the requirements of the mission. For example, the smallest expected aperture that the robot is expected to be able to traverse through is a common requirement that drives planning and estimation discretization levels. For example, in Figure 2-1a, a shorter path could be found by increasing the resolution of the discretization so that there exists a valid connection through the left most doorway, but increasing the discretization level can be computationally expensive. In contrast, Larsson et al. [92] propose an information-theoretic method for generating useful multi-resolution discretized representations. Despite there being nothing in the formulation of general graph search that precludes multi-resolution discretization, such approaches have not yet been widely adopted.

Stochastic Graph Methods

Rather than pre-divide the state space into discrete intervals, stochastic methods reduce computational complexity by randomly sampling from the configuration space. Sampling-based motion planners (SBMPs) stochastically build graphs by sampling candidate vertices, therefore avoiding total enumeration of the state space and explicit construction of $\mathcal{X}_{free}$. SBMPs are especially suited for applications involving navigation over long length scales and robots with high-dimensional state spaces. In Chapter 3, we utilize a SBMP for navigation over long length-scales.

Two key functions drive the graph building process in SBMPs. The first is vertex generation (i.e., generating the set of V), and the second is edge selection (i.e., determining the set of E). In general, SBMPs build planning graphs such that all vertices are located in free space, and all edges represent sub-trajectories traversing between the two vertices they connect. Some, such as Probabilistic Roadmaps (PRMs) [81] first construct the graph and then search in two distinct stages. Others, especially tree-variants such as Rapidly-Exploring Random Trees (RRTs) [94] and Fast Marching Trees (FMTs) [72], interleave graph construction and graph search for any-time algorithms; feasible solutions are refined given additional computation time. Many SBMPs have asymptotically optimal variants; several of which (such as the RRT*) were presented in the seminal work of Karaman and Frazzoli [78], which introduced conditions on how vertices are added to the graph and the concept of "re-wiring" the graph by changing the connectivity between vertices to achieve optimality. A simple illustration of a SBMP is given in Figure 2-1b.

Stochastic graphs are powerful representations for enabling faster planning, but still suffer from sample complexity. Early implementations of SBMPs suggested the use of uniform sampling to ensure diverse coverage of the configuration space. However, uniform sampling can lead to planning difficulties in non-trivial environmental geometries. For example, to improve the solution found in Figure 2-1b, one approach might involve simply allowing the planner to sample more vertices until the shorter trajectory is found, but the construction of large graphs can be at odds with maintaining a low-latency online planner. Another more promising strategy to enable the planner to find the shortest trajectory is to bias the generation of vertices to enable useful graph connectivity. In this case, placing vertices in doorways may be a useful strategy to enable finding the shortest route quickly. Various methods of this strategy are discussed in the following section.

2.1.2 Speeding up Planning via Heuristics

Another important method of increasing search speed is to use heuristics to effectively deploy computational resources. A simple heuristic that reduces computation time is to approximate the planning problem by focusing on making locally reasonable solutions within some horizon [152, 153]. However, such methods can be prone to exacerbating myopic navigation behavior if they consider only local information when making decisions.[5] For instance, as Schouwenaars et al. [153] demonstrate, accounting for navigation within a limited time horizon can result in the vehicle being in a state that yields future collisions due to dynamics. The authors propose a method for ensuring safe navigation by including a future feasibility constraint, but do not focus

[5]The most dramatic instance of this planning paradigm is reactive planning, which makes decisions solely using the most recent sensor measurements. We defer discussion of methods combining short-range planners with longer range planners to Section 2.1.3

on improving total distance travelled in addition to feasibility.

In this section, we focus on heuristics that are used to find a solution to the entire planning problem from the start to the goal, even if the solution is sub-optimal or approximate. In particular, we focus on heuristics used in conjunction with SBMPs, given their potential for efficient planning in large scale planning as discussed in Section 2.1.1.

Geometric Heuristics

Rather than limiting the planning horizon, methods that use geometric heuristics exploit geometric properties of the planning problem to guide the search over the possible space of plans in an efficient manner. Such approaches realize that although there may be many feasible solutions in a planning graph, not all are relevant to obtaining a final solution, and avoiding the computational expense of considering those solutions can reduce overall computation time. For example, the Euclidean distance heuristic paired with the ubiquitous A^* algorithm is a canonical example of a geometric heuristic applied to path planning. In this case, the cost to go is estimated by the Euclidean distance from the expanded node to the goal; crucially, this heuristic is an *admissible* heuristic, meaning that it is guaranteed to be less than or equal to the true cost that will be incurred. Although the Euclidean distance is often a crude estimate of the remaining distance to be travelled, it can nevertheless drastically reduce search times, while being provably resolution-optimal.

Geometric heuristics are particularly well-deployed with SBMPs, not only to search the planning graphs for solutions, but to bias the building of the graph itself. In particular, given that a trajectory would eventually be found with a uniform sampling strategy if one exists, we discuss methods that attempt to find such solutions much sooner than the limit of infinite samples. Practitioners of SBMPs have long investigated how to improve solutions using non-uniform sampling. Early approaches to intelligent sampling targeted a scenario noticeably difficult for SBMPs: the narrow passageway. Although the iterative, random construction of planning graphs in SBMPs leads to appreciable computational gains for large problems, it also means that it can difficult to construct useful planning graphs in areas of the environment that require very specific kinds of trajectories to traverse. For the narrow passageway, vertices must be sampled so that connecting edges do not intersect with the walls of the passageway. In other words, the corresponding region of the passageway in the configuration space is low volume, leading to low probability of being sampled under uniform sampling strategies. To overcome such geometrically adversarial environment features, approaches such as Gaussian sampling [16] and the bridge test [69] were developed to sample vertices in or near environmental features. For example, Gaussian sampling is biased towards free space near occupied regions, thus intuitively increasing the probability that samples will be placed along narrow passageways.

Although early geometric heuristics provided large gains in planning efficiency under certain circumstances, they are not inherently adaptive, and choosing which heuristics to apply can be challenging in practice. The selection of planning heuristics largely falls to system designers with specific domain knowledge, capable of identifying heuristic strategies likely to enable more efficient planning. Indeed, Geraerts and Overmars [55] compare several different sampling strategies on a diverse set of planning problems, concluding that methods developed for narrow corridors should only be deployed in regions with narrow corridors.

Later, more modern methods were developed that focused on improving solutions faster once a feasible solution has been found. These methods largely exploit geometric bounds on optimization. For example, Batch Informed Trees (BIT*) [51] and Informed RRT* (IRRT*) [50] both use ellipsoid-based bounds to focus vertex generation to those that will improve solutions. Akgun and Stilman [5] suggest sampling near the current best solution for local improvements as well as actively managing the tree to not add vertices that do not improve the solution, to keep the number of vertices in the tree low (and therefore manage computational overhead). However, while the heuristics we have discussed for improving solutions once an initial feasible solution is found are powerful and quite general, they do not generally aid in finding feasible solutions faster. The time to the first feasible solution is equally important to trajectory quality for op-tempo autonomous navigation.

Learned Heuristics

In an effort to overcome the brittleness of hand-coded heuristics, some SBMP methods have turned to learned heuristics. We differentiate *adaptive workspace biasing*, a method that uses previous sampling outcomes to iteratively improve future planning performance, from *directed sampling*, which prescribes *a priori* distributions over the configuration space. However, we observe that adaptive workspace biasing can be seen as online learning of sampling distributions, just as learned directed sampling can be iteratively updated.

Adaptive workspace biasing methods aim to improve planning performance over time. Some focus on improving search during a single query [73, 91]. For example, Joshi and Panagiotis [73] propose to use various geometric heuristics over kernel vertices to choose which vertex to extend next. Lai et al. [91] model tree growth as a Multi-Arm Bandit (MAB) problem and update proposal distributions via a Bayesian framework to adapt online to the configuration space. Other methods use features of the environment to iteratively optimize sampling distributions modelled by Gibbs distributions [191] or Gaussian mixture models [24]. Given access to features extracted from new environments, these methods have shown some generalization success.

In recent years, there has also been increased interest in applying deep learning to heuristic discovery. For example, Ichter et al. [70] show how, given a dataset of optimal trajectories paired with maps, a neural network can be trained to learn a sampling distribution to inform a SBMP. The authors first train a conditional variational autoencoder (CVAE) conditioned on workspace information such as occupancy maps. The CVAE is then used to generate samples for a SBMP. By augmenting the samples drawn from the trained network with a uniform sampling strategy, the authors also discuss how asymptotic properties of the planners are preserved. In later work, Ichter et al. [71] suggest training a neural network to predict so-called *critical samples*, which are related to a graph-theoretic centrality metric. The Neural RRT* [179] trains a convolutional neural network (CNN) to predict pixel-wise sampling probabilities from a modified occupancy map input. Although these methods can be seen as learned methods of directed sampling as they are largely trained offline, some methods such as the one proposed by Ichter et al. [70] have shown compelling results from iterative training of the networks with data produced by the learned sampling distributions.

Learned heuristics have shown great promise in speeding up search using prior experience, rather than hand-selecting geometric heuristics. However, because they in general are largely based on geometric properties only, they rely on *a priori* known maps. The requirement for known geometry makes existing literature on learned heuristics difficult to apply to problems where the environment is inherently uncertain. In particular, Ichter et al. [70] suggest incorporating semantics as a promising avenue for future work. In Chapter 3, we present a method for learning a sampling distribution that considers both object-level semantic representations and dense geometry, both of which can be partially observed.

2.1.3 Navigation in Unknown Environments

Thus far, we have focused on methods of planning specifically designed and optimized to work in known environments. Much of the existing literature regarding planning representations and search heuristics focus on lowering planning latency. In practice, action-perception loops in known environments often involve estimating the pose of the vehicle and deploying some closed-loop controller to follow an extracted trajectory.[6] When uncertainty is considered, it is often uncertainty that stems from random noise in the sensing or actuation process. For example, various methods have been proposed to enable planning routes that consider state estimation accuracy [145, 128]. In this thesis, we are interested in improving autonomous navigation in environments that are unknown *a priori*.

Navigation in novel environments with finite geometric sensing induces geometric ambiguity. Consider, for example, an extreme case where the depth sensor of a robot can observe structure at a maximum of 5 meters away, but the nearest pertinent structure is 10 meters away from the robot. In this case there is little to no infor-

[6]Of course, in carefully controlled environments where actuation is extremely accurate and precise, open-loop execution may be possible.

mation to disambiguate which direction is most promising. The performance of the action-perception loop during navigation in novel environments is often even more critical in unknown environments because the world is incrementally revealed to the agent, potentially requiring the trajectory to be substantially updated. In the action step, the agent must choose a trajectory given the potentially incomplete information that it has. Upon executing some portion of the trajectory, new portions of the environment are revealed, which in turn requires re-planning. Trajectories that were once thought to be feasible may be revealed to pass through obstacles that were occluded or beyond the range of the sensor. Reducing re-planning latency is a first order solution to planning in unknown environments, enabling the agent to react quickly to new information as it is acquired.

Choosing better actions given limited perceptions is another avenue that can improve navigation. In this section, we discuss methods designed specifically for navigation in novel environments. The benefits of such approaches are that rather than simply reacting quickly to new information about the environment, the algorithms are in general designed to attempt to make reliable planning decisions in the face of uncertainty. We note that many of these approaches are built upon frameworks described in the previous section.

Multi-Scale Planning

A common system design in planning in unknown environments is to generate plans of varying levels of fidelity to different ranges. Liu et al. [103] propose combining a short and long-range planner, where the short-range planner plans trajectories to a frontier determined by various properties including nearness to the goal, and a long-range planner is used if the short-range planner fails to find a trajectory under the required conditions. The short-range planner uses only the most recent sensor measurement. Similarly, Ryll et al. [149] propose filtering motion primitives based on a sliding local body-centered map and instantaneous depth measurements. A global plan that assumes straight-line trajectories beyond the extents of the local map is used to further rank the local trajectory selection. These approaches use local dense geometry to plan dynamically feasible trajectories, deferring detailed decisions about navigation in ambiguous portions of the map for which dense geometry is not available. One benefit of such approaches is their modularity; they can in principle be combined with more sophisticated methods as well. However, because these methods largely defer decisions beyond some range, they can also suffer from myopic navigation behaviors.

Environment Estimation

One approach to improving planning in novel environments is to estimate the properties of the portions of the environment for which dense geometry is unknown. Such methods seek to mitigate geometric ambiguity by predicting the structure of unobserved portions of the environment. In recent years, various methods have been proposed to leverage datasets of partial maps to predict environmental geometry beyond the frontiers of observed space. Deep learning has emerged as a popular choice for occupancy prediction, given that 2D occupancy maps of constant discretization are similar in domain to image-space measurements (i.e., greyscale images) and therefore lend themselves particularly well to integration into existing network architectures designed for images. Katyal et al. [79] perform analysis of several different generative neural network models for occupancy map prediction, and Shrestha et al. [157] propose map completion using a variational auto-encoder network.

Although generative models are intuitive models for occupancy prediction as they attempt to model distributions over environments, rather than single predictions, it can be unclear how to leverage the implicit distribution at run time. Elhafsi et al. [44] show that occupancy predictions from a non-generative neural network can be used to bias dynamics constrained planning. The authors train a neural network to predict occupancy values in unknown regions of the map, and use the predictions at runtime to bias trajectories away from regions with high predicted probability. Notably, the authors discuss the benefits of using the prediction as a soft penalty on the trajectory optimization, rather than choosing a hard threshold for occupancy.

Successful methods for map prediction tend to do so either do so with relatively small portions of the map missing [79, 157], or to short horizons [44]. Such limitations are likely due to the inherent complexity in modelling map distributions.

Utility Estimation

An alternative approach to improving planning under environmental uncertainty is to instead estimate the utility of future actions or regions of the state space. In this section, we consider both estimating properties of actions as well as directly estimating the next action to take. Utility estimation approaches differ from environmental estimation in that they do not attempt to explicitly model all possible environment configurations[7], but instead do so implicitly. Utility estimation techniques mitigate geometric ambiguity by indicating which actions or regions are more likely to lead to the goal, essentially attempting to resolve the issue of many trajectories appearing equally feasible due to lack of information.

Previous work has shown that estimating useful properties over a set of potential actions can improve planning outcomes in unknown environments. For example, rather than attempting to predict the exact geometry of the unobserved portions of the world, Richter and co-authors [140, 137, 138, 141] enable the high speed navigation of an autonomous RC car by learning to predict the collision probabilities of a set of motion primitives. Stein et al. [164] show that rather than simply predicting collision probabilities, learning can be deployed to directly predict the costs of a POMDP planning formulation. The authors compactly abstract the navigation problem to the high-level problem of choosing the next best frontier, and train the network to predict costs such as the expected cost-to-go to the goal from a frontier as well as the cost that would be incurred if the frontier is incorrectly chosen.

Other methods instead focus on learning an intermediate representation encoding the next best action. Directly predicting an action allows for some methods to navigate in unknown environments without maintaining a map representation, i.e., so-called *map-less navigation.* Such methods map directly from sensor measurements

[7]One potential way to conceptualize action estimation approaches is as the result of marginalizing out the distribution over environments from some joint distribution between environments and actions.

to actions. Tai et al. [170] employ reinforcement learning to predict the optimal action from a vector of laser range measurements, the previous vehicle velocity, and the goal relative to the robot. Loquercio et al. [105] propose learning steering angles in addition to the probability of collision from images to enable reactive navigation of a quadrotor. Kaufmann et al. [80] enable map-less drone racing by learning to predict a goal in image-coordinates and scaled target speed. Bansal et al. [11] learn to predict intermediate waypoints and then optimize trajectories via an optimal LQR controller. While these approaches have the potential to be low latency, they will struggle in large-scale environments with complex geometry. Without using map representations as a historical artifact of past actions, map-less planners can in principle be particularly susceptible to infinite loops where the agent chooses a series of actions leading to the same outcome repeatedly.[8]

A common theme in action utility estimation for navigation in unknown environments is the use of clever abstractions such as using motion primitives [138] or using frontiers as sub-goals [164] to limit the dimension of the required prediction. However, such abstractions can potentially be difficult to extend to new problems, and may be brittle. Consider using frontiers as sub-goals — in noisy maps there can be many spurious discontinuities in the map that appear to be frontiers, and 3D frontiers are more difficult to robustly identify than 2D frontiers. To overcome the potential brittleness of limiting to an action set that may not encompass all necessary maneuvers, as well as the potential representation limits of waypoint prediction, a promising approach is to estimate intermediate properties of the environment to enable the selection of trajectories, which can be seen as a hybrid of the two classes of approaches discussed thus far. For example, Zeng et al. [190] propose an end-to-end learning method that predicts cost volumes over the state space. The predicted cost volumes are then used to score candidate trajectories. Rather than explicitly enumerating all possible ac-

[8]Imagine, for example, a robot choosing to follow a hallway that will lead to a dead-end. Once reaching the dead end, if the reactive controller directs the robot back to the initial state, the agent may again recieve the same sensor measurements that once again lead to the agent traversing the same hallway. Methods that maintain fused representations over time are more robust to this failure model; even if a history of all past actions is not maintained, the map will reflect that the hallway is a dead-end, helping to avoid a similar decision from being made again.

tions, in Chapter 3 we use a deep neural network to predict areas of the environment useful for navigation and select trajectories.

2.1.4 Semantics and Geometry in Uncertain Environments

Semantic representations have demonstrated compelling abilities to improve navigation outcomes by providing contextual information that is difficult to obtain from geometry alone. Various semantic representations have been proposed for inclusion in navigation algorithms for a range of purposes, including to provide additional context when geometric information is missing.

Using Hierarchical Semantic Representations

The use of hierarchical semantic representation has been explored in the context of robotic navigation as early as the 1970s. For example, Kuipers and Levitt [88] describe a spatial semantic hierarchy comprised of four levels and present several planning algorithms under the proposed framework. The TOUR model [87], for example, stores experiential data to build a spatial model that can be queried for plans.

More modern methods exploit the the development of increasingly sophisticated perception and mapping. For instance, the Deep Spatial Affordance Hierarchy (DASH) [129], includes a perceptual layer, a peripersonal layer, a topological layer, and a semantic layer. The perceptual and peripersonal layer capture dense geometric (i.e., occupancy) and semantic (i.e., objects and landmarks) with respect to the robotic agent, while the topological layer captures the high level structure of the environment via a set of poses, some of which can be labelled as placeholders for unknown space. Although the authors do not provide navigation experiments, they observe that their representation is specifically designed with navigation in mind. Ravichandran et al. [132] propose a method that uses graph neural networks paired with reinforcement learning to to learn a policy from 3D Dynamic Scene Graphs (DSG) [144]. The DSG used by the authors features layers that include objects and agents, places and structure, rooms, and buildings; the DSG is condensed to an “observation graph structure”, from which a graph neural network is used to learn a feature representation, which is in turn used to train a reinforcement learning algorithm via actor-critic methods.

Both the DSG and DASH representations relate geometric information with the semantic, and further explicitly relate information between different layers of semantic geometric information. For example, DASH uses a deep generative model, which enables both bottom up inference (i.e., estimating whether a given occupancy map representation is a doorway, corridor, etc) and top down inference (i.e., generating candidate occupancy maps given a partial map and label that the location is a hallway). Such a methodology is similar to the map prediction methods discussed in Section 2.1.3, except that explicit semantics are used to improve prediction. DSG representations model relationships via edge connections between entities at different layers in the hierarchical model. Although promising, complex hierarchical semantic models are generally not lightweight models to build online.

Using Dense Overhead Representations

Rather than relying on complex hierarchical semantic models, other methods use learning to estimate intermediate planning properties to enable sub-goal selection using top-down representations. Everett et al. [45] estimated the cost-to-go over a discrete set of states. The input to the system is a RGB-D camera measurement, which when paired with a semantic segmentation approach enables the construction of a 2D overhead semantic map. An image-to-image translation network, Deep Cost to Go, is trained to convert partial overhead semantic maps to cost-to-go representations using ground-truth data. Although the number of training environments is limited, the authors discuss how to generate many different pairs of cost to go maps and partial maps by generating diverse masks over the annotated overhead semantic maps. At run time, the Deep Cost to Go network is used to choose a frontier to explore. Georgakis et al. [54] propose a method for navigating to an object of unknown location in novel maps. The authors use a deep neural network to predict occupancy maps from partial occupancy maps. The predicted occupancy maps are combined with the observed image-space semantic segmentation projected into an overhead semantic map, and passed to a second neural network that predicts the probability of different semantic values in unobserved portions of the map. Using an upper confidence bound, the algorithm then chooses the best intermediate subgoal to find the object, balancing between exploration and exploitation. A deep reinforcement learning model is then used to navigate to the sub-goal.

Using Object Level Representations

Dense representations built by semantic segmentation and sophisticated hierarchical semantic models are generally more expensive to build than object maps. Previous work has shown the object-level semantic context can be used to improve navigation. Sünderhauf [169] assumes an *a priori* known environment modelled by a graph with locations as well as static objects (such as sofas) and trains a graph neural network to predict a distribution over goals to find moveable objects (i.e., a remote). In unknown environments, Vasilopoulos et al. [174] present a method for a provably safe reactive planner that can not only incorporate object estimates for collision avoidance, but also use cues from tracked humans to effect navigation (i.e., using hand gestures to stop robot navigation). Cosgun and Christensen [31] build semantic representations that include truncated planes, door signs, and humans. The robot is tasked with following a person, but a naive strategy without contextual context can cause the robot to impede the person's ability to open the door. The authors describe a system whereby the robot monitors the human's proximity to doors (inferred from door-signs), and responds to a human-pointing gesture to not block a doorway. The so-called door-blocking scenario is an excellent example scenario demonstrating how semantic representations can enable seemingly subtle shifts in strategy that lead to more sophisticated navigation. However, although hand-coded behaviors triggered by semantic context are a practical solution in simple contexts, the list of desired behaviors presented is limited in scope. The interaction between semantics and geometry can grow increasingly complex, requiring a larger and larger database of rules to be specified.

Notably, the approaches discussed thus far in this subsection focus only on improving metrics such as overall distance travelled, but do not explicitly consider utilizing semantic context to focus computation. As discussed in Sections 2.1.2 and 2.1.2, there is a large literature of geometry-only approaches that use geometric context to improve planning speed by identifying planning states useful for finding a solution.

Existing approaches to using semantics for motion planning to focus computation largely focus on per-object learning, which can be brittle. Previously described methods to enable sampling in narrow passageways are hand-designed heuristics to exploit specific types of geometric patterns such as hallways. For example, some methods model sampling distributions as multinomial Dirchelet distributions [10] or Gaussian mixture models [9] over so-called *semantic fields*, where both geometric (where) and extra-geometric (what) information should determine the best navigation strategy. The results are focused on learning the effect of single semantic entities such as doorways or traffic circles, and therefore either do not involve dense geometry [9] or assume fully known geometry and semantic maps [10]. Unlike DSG and DASH, they do not explicitly model the interaction between geometry and semantics. Furthermore, the utility of object relationships can be a complex function that may change conditioned on the ultimate objective. For example, the usefulness of a door with an exit sign is determined by whether the goal is outdoors.

A key insight in this thesis is that *both* semantics and geometry should be used to improve the computational efficiency of autonomous navigation. In Chapter 3, we present an algorithm that enables more efficient navigation in unknown environments by exploiting both semantics and geometry. We choose object-level representations over denser and hierarchical representations as our choice of semantic information for their compactness and ability to be built online on a SWaP vehicle. To this end, we rely on light-weight object-level representations that can be projected to discrete maps of a similar form to discrete occupancy maps.

2.2 Object-Level Estimation

While there are many types of extra-geometric information, we focus in this work on object-level representations as they have the potential to be compact and semantically meaningful for the navigation process. The main goal of object-level estimation is to infer or predict useful geometric and semantic properties of objects in the environment. Useful properties often include the semantic class of the object as well as its geometric extents.

The advent of deep learning has led to a proliferation of methods of so-called 2D "semantic extractors" such as 2D bounding box detection and semantic segmentation [184]. 2D bounding box detection algorithms (such as YOLO [135, 133, 134], SSD [104], and RCNN variants [56, 136]) predict axis-aligned rectangles on the image plane that tightly surround the objects of interest from query images (usually greyscale or RGB images). In general, object detection is more difficult than object classification, as the algorithm must not only estimate *what* an object is but also *where* it is. The measurements generated by modern object detectors therefore provide both geometric and extra-geometric information. Similarly, semantic segmentation methods such as U-Net [142] label the class of every pixel in an image, while instance segmentation methods such as Mask R-CNN [61] provide segmentation for individual detected objects. The density of the annotation can potentially require a more time consuming annotation process.

Although the availability of object-detection in particular has naturally led to a renewed interest in utilizing semantic object measurements for a variety of robotic tasks, estimating the 3D geometry of objects from 2D image-space measurements remains a challenge. As seen in Figure 1-2, an object estimate that appears to be consistent with a 2D bounding box detection may be very incorrect. One of the key technical challenges of lifting image-space detections to world coordinates for the purposes of autonomous navigation is geometric ambiguity, similar to the issues faced by monocular vision-based SLAM. To estimate 3D object geometry from images object-level estimation algorithms must concurrently perform *pose estimation* and *geometric reconstruction.* The quality of the estimate depends on the quality of measurements available, and efficient autonomous navigation often does not provide ideal measurements for estimation processes.

In this section, we first discuss several common geometric models used in object-level estimation (Section 2.2.1), with particular focus on the benefits of approximate geometric models. We then describe three key methods of estimating object-level geometry: online regression (Section 2.2.2), simultaneous localization and mapping (Section 2.2.3), and deep learning (Section 2.2.4). Finally, we present hybrid methods that use a combination of different estimation approaches (Section 2.2.5).

2.2.1 Object Representations

Key to any estimation process is determining the model to be optimized. There are a plethora of options for object representations, ranging from highly detailed to extremely approximate. We briefly describe several common object-level representations that will be utilized in later sections on optimization techniques over object-level representations.

Detailed Geometric Models

Point-based representation are a mainstay of vision-based mapping techniques. Point-cloud representations are defined by a set of unique points, where each point in the set has a 3D position and corresponds to the surface of the surrounding geometry. Meshes include a set of points (interpreted as vertices) combined with edges connecting nearby points such that finite planes can be defined between points. Commonly used in vision-based simultaneous localization and mapping (vSLAM) as landmark representations, meshes [59] and point-clouds [114] can describe arbitrarily complex surface geometry because the surface approximation becomes more refined as more points are added. Another popular representation are truncated signed distance functions (TSDFs). Proposed by Curless and Levoy [33], TSDFs are voxel-based volumetric representations, where the value encoded by each voxel represents the signed distance to the nearest surface. Unlike mesh models, TSDFs are continuous representations parameterized the discrete voxel grid; the surface of an object encoded by a TSDF can therefore be extracted from solving for points in space such that the signed distance is zero. TSDFs have shown great promise, especially paired with a depth sensor for dense mapping [115].

Detailed geometric models are intuitive object representations and a common paradigm for how humans record object models. Many object-level datasets provide object models in the form of point-clouds or meshes. For example, Calli et al. [22] present the Yale-CMU-Berkeley (YCB) dataset, which includes mesh models of household objects geared towards advancing robotic manipulation. Other databases of objects include ShapeNet [25] and the 3D Warehouse [2].

Although expressive, detailed geometric models of real-world objects can be difficult in practice to obtain, and may not be necessary for providing semantic context for navigation. For instance, the mesh model generation process for the YCB dataset [22] required that objects be placed on a turnable and observed by specialized scanning hardware. Additionally, the computational overhead required to estimate low-level detailed geometry may not always be useful for autonomous navigation. While detailed mesh models of household goods are likely necessary for robotic manipulation, we posit that semantic class and approximate geometry may be sufficient for providing first-order semantic context to guide navigation.

Pose-Only Models

While meshes and point-clouds encode fine detailed geometric details of objects, other methods estimate only components of the pose of objects. For example, some methods only estimate the position of objects, approximating objects as point features [38]. Other methods estimate the full 6D pose of the objects [186, 173]. Pose-only object models avoid the difficulty of reconstruction and focus on accurately estimating the location of the object in world coordinates.

Point-based models can enable the computation of more sophisticated estimation approaches such as multi-hypothesis approaches [17, 38], but we observe that the size

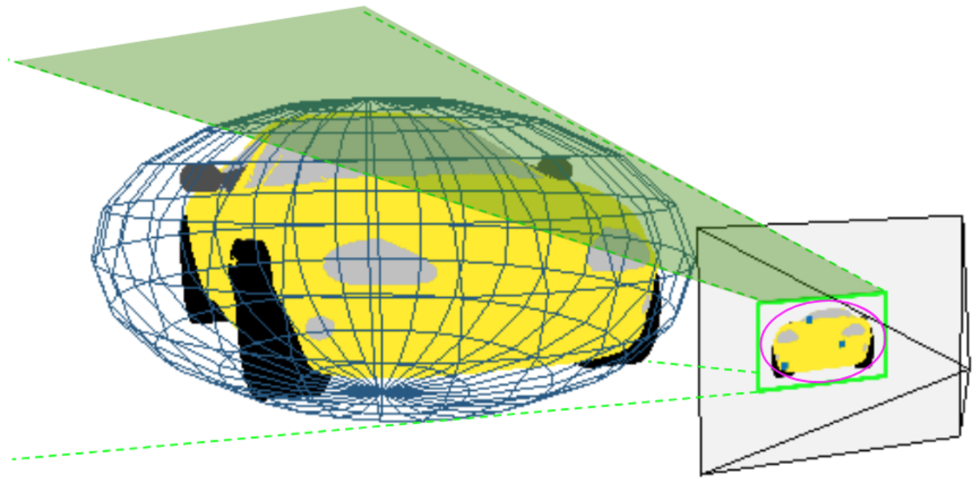


Figure 2-2: Illustration of dual ellipsoid measurement model. Ellipsoids are a special class of quadrics that are especially useful for object approximate as they are finite (unlike other classes of quadrics, such as hyperboloids and infinite cylinders). In this figure, we illustrate an ellipsoid approximation of a care by the blue mesh surrounding the yellow car. The dual ellipsoid is implicitly defined by the infinite set of 3D planes that are tangent to the ellipsoid, and the projection of the ellipsoid onto the image plane is a conic (shown in pink). The axis-aligned bounding box can be extracted from the conic by solving a system of equations. Intuitively, each bounding box edge can be interpreted as projecting into a plane that is tangent to the ellipsoid (visualized here by the semi-transparent plane projecting to the top edge of the image-space bounding box). The dual ellipsoid formulation is especially attractive because it provides a closed-form function that projects ellipsoid estimates to 2D image-plane bounding boxes (green box).

and orientation of objects can also be semantically meaningful for navigation. For example, one could imagine the orientation of a doorway indicating the direction of a hallway, and the size of a doorway indicating the potential for the doorway to be a main entrance or exit to a building.

Approximate Geometric Models

In terms of model expressiveness, approximate geometric models such as cubes, ellipsoids, etc, lie between detailed geometric models and pose-only models. Unlike meshes and point-clouds, approximate geometric models abstract object geometry to simple volumetric primitives defined by several parameters. For example, cuboid representations are a popular primitive model [187, 53, 28, 29, 108, 173] as they can capture the size of objects in addition to orientation. Superquadric representations allow for a diverse range of primitive shapes including objects that are qualitatively cube-like [161, 176]. Quadrics, a special case of superquadrics, still offer a range of primitives such as cones, cylinders, planes, and ellipsoids. The recently proposed QuadricSLAM algorithm [146] demonstrates how the dual form of the ellipsoid representation, which is implicitly defined by the infinite set of 3D planes tangent to the ellipsoid, can be used to provide an intuitive measurement function relating 2D image space bounding box detections to 3D ellipsoid estimates. An illustration of the dual ellipsoid model is shown in Figure 2-2.

Approximate geometric models are an attractive choice for object-level representations for autonomous navigation because they can offer a compromise in terms of semantically useful geometric information such as rotation and size, while also avoiding the computation of properties which may not be useful for the purposes of navigation.[9] Additionally, there is compelling evidence that such models can be composed heterogeneously with other primitive types to describe world geometry [67, 188], homogeneously to better represent object geometry [123, 124], or hierarchically to increase estimation robustness [121]. In Chapters 4 and 5 we will leverage the ellipsoid representation as an approximate geometric model for object-level estimation.

2.2.2 General Object Estimation

In this section, we discuss several general methods of object estimation including single view and multi-view regression. We classify regression methods as those that estimate the pose and or shape of objects from either a single or multiple views of an object by solving an optimization problem to minimize some cost function. We also briefly discuss 3D object detection methods.

Given *a priori* geometric models, iterative closest point (ICP) methods can be deployed. A method of point-cloud registration, ICP enables estimating the position

[9]This sentiment echos the observation of Rosen et al. [143] that semantics are contextual and the properties of import vary given the objective of the robot.

and rotation of a known model to agree with a measured pointcloud [14]. Given parametric models of the object, single and multi-view regression methods optimize object parameters to fit observed data. Implicit shape models such as quadrics and super-quadrics are particularly popular due to their continuous implicit representations, which can be used to formulate a regression problem. Since the seminal work of Solina and Bajcsy [161], most existing approaches [177, 43] to modelling objects as superquadrics have focused on directly regressing superquadric parameters to fit data via online gradient descent. Vezzani et al. [176] propose a system to estimate the parameters of a superquadric online from a segmented point-cloud measurement to enable robot manipulation. In a later work, Vezzani et al. [177] demonstrate that first categorizing the shape of an object can speed up inference times. The authors note that estimation quality depends on the viewing angle of the object.

One downside to single-view regression is geometric ambiguity. Multi-view methods can provide better estimates by fusing observations from different sensor poses with respect to the object. Methods such as structure from motion estimate world structure by projecting object estimates into camera measurements, and minimizing the disagreement between the induced measurement and the observed measurement. Crocco et al. [32] introduce a structure from motion method for estimating objects as 3D ellipsoid and affine camera calibrations from a set of 2D bounding box measurements by first fitting 2D image-space conics to the bounding boxes and exploiting the dual conic form. The work was then extended for perspective cameras [146]. The authors of EllipseRCNN [40] train a neural network to detect objects in images. Rather than using a 2D bounding box measurement, the method models objects as 2D ellipses, which can then be synthesized over multiple views to obtain a 3D ellipsoid estimate. Later work explores more sophisticated uncertainty modelling in the context of ellipse detection [39].

Beyond approximate geometric models, detailed geometric models can also be reconstructed given a set of 3D laser scans [148] or RGB images [19]. Similar to general structure and motion paradigms, Brown and Lowe [19] propose to associate RGB views of an object using SIFT features, and formulate object reconstruction as optimizing the pose of individual cameras and points of the object. Ruhnke et al. [148] filter out background points from a series of 3D laser scans to obtain partial object pointclouds. To merge two candidate partial pointclouds, several randomly sampled viewpoints from each pointcloud are used project the pointclouds into depth images. Corner points are detected and local pixels used to generate feature vectors. Given the features, the algorithm searches for correspondences. After further verification, the models can be joined. The process enables the reconstruction of pointcloud models of objects. Finding appropriately discriminating feature points is crucial to both methods, and thus they are likely to struggle for extremely symmetric objects in the case of depth images, or repeating patterns in the case of RGB images.

Some early 3D object detection methods from RGB images extended early approaches to 2D detection methods, such as template matching and Hough Forests. Hinterstoisser et al. [63] propose multi-modal templates over both RGB and depth information to detect and estimate discrete object poses by comparing to features from reference images. Wang and Posner [178] train a linear SVM to perform classification using a sliding window method over pointcloud data converted to a voxel representation. Fidler et al. [46] use a deformable cuboid model with six faces and performed template matching in the image-space. Doumanoglou et al. [41] utilize detailed textured 3D models of objects to enable an approach that builds Hough Forests using patch features learned from an auto-encoder. The 6D pose is determined using Hough voting.

2.2.3 SLAM

Unlike single-measurement online regression methods, simultaneous localization and mapping (SLAM) methods use a series of measurements to estimate both the pose of the robot as well as the pose of *landmarks*, with the implicit goal of enabling online estimation. SLAM methods were historically developed to enable autonomous agents to concurrently estimate the pose of J landmarks $\boldsymbol{l}$ and also use K measurements $\boldsymbol{y}$ of the landmarks to estimate the pose of the robot state $\boldsymbol{x}$ over time. The state $\boldsymbol{x}$ is often assumed to be of the special Euclidean group SE(3) such that $\boldsymbol{x} \in \mathrm{SE}(3)$, although other representations are certainly possible. As we discuss in this section, the domains of landmarks $\boldsymbol{l}$ and measurements $\boldsymbol{y}$ vary between approaches. We assume the existence of some deterministic functions $f(x)$ and $h(x)$ that define the *motion model* and *measurement model* respectively. In stochastic frameworks, these models are assumed to be corrupted by additive noise, leading to the following expressions:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + w \tag{2.2}$$

$$\boldsymbol{y}_k = h(\boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}) + v \tag{2.3}$$

where $\boldsymbol{u}_t$ is the control input to the system at time t. The additive noise terms w and v are drawn from the motion and measurement noise models, respectively, and are usually taken to be normally distributed such that $w \sim \mathcal{N}(0, \mathbf{W}), v \sim \mathcal{N}(0, \mathbf{V})$. Note that under the deterministic motion model and measurement models, the state $\boldsymbol{x}_{t+1}$ depends only on the previous state $\boldsymbol{x}_t$ and the control input $\boldsymbol{u}_t$, and the measurement $\boldsymbol{y}_k$ depends only on the state of the robot at the time of measurement $\boldsymbol{x}_{i_k}$ and the state of the landmark $\boldsymbol{l}_{j_k}$. It is common for modern visual SLAM algorithms to be formulated as dependent on measurements only, but we include the motion model with control inputs here for completeness. We can express the joint probability over the robot states, landmark states, and measurements as

$$p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:K}, \boldsymbol{l}_{0:J}) = p(\boldsymbol{x}_0) \prod_{i=1}^{T} p(\boldsymbol{x}_i | \boldsymbol{x}_{i-1}; \boldsymbol{u}_i) \prod_{k=1}^{K} p(\boldsymbol{y}_k | \boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}), \tag{2.4}$$

assuming a uniform prior over landmark positions and that the control inputs are given parameters. The problem of associating measurements to landmarks is also usually assumed to be solved in a separate process.

Good choices for feature-based sparse landmarks[10] are those that are re-observable, distinguishable from other landmarks, and have useful measurement functions that

[10]It is important to note that while in the development of this section we have focused on *sparse* features as landmark examples, dense map representations also exist. For instance, modern visual SLAM methods have explored dense map representations such as DTAM [116] and KinectFusion [115]. However, sparse landmark representations most naturally develop into the modern object-level SLAM methods we are ultimately interested in.

allow for the state of the vehicle and landmarks to be observed. Over the years, various types of landmarks have been used, from radar reflectors artificially added to the environment [37] to point-based landmarks [47, 114, 85] extracted using image features (i.e., SIFT [106], ORB [147], etc) detected in images.

Although early backend[11] implementations of SLAM largely relied on Kalman filtering [160, 77], or particle filtering [111, 110], in recent years smoothing techniques have been popularized. Introduced by Dellaert and Kaess [34], Square-Root Smoothing and Mapping ($\sqrt{\text{SAM}}$) began a line of one of the most well-known modern smoothing methods. The goal of the smoothing problem is to find the *maximum a posteriori* (MAP) estimate of the trajectory and landmark positions.[12] Instead of marginalizing to solve only for the most recent robot and landmark poses, $\sqrt{\text{SAM}}$ solves for the entire history of robot and landmark poses by optimizing the negative log-likelihood of the joint probability from Equation 2.4:

$$\begin{aligned} \boldsymbol{x}^*_{0:T}, \boldsymbol{l}^*_{0:J} &= \underset{\boldsymbol{x}_{0:T}, \boldsymbol{l}_{0:J}}{\arg\max}\, p(\boldsymbol{x}_{0:T}, \boldsymbol{l}_{0:J} | \boldsymbol{y}_{0:K}) \\ &= \underset{\boldsymbol{x}_{0:T}, \boldsymbol{l}_{0:J}}{\arg\min} -\log(p(\boldsymbol{x}_{0:T}, \boldsymbol{y}_{0:K}, \boldsymbol{l}_{0:J})). \end{aligned} \tag{2.5}$$

Given the static world assumption, only the robot poses are indexed with time. Substituting Equations 2.2, 2.3 into Equation 2.5 and applying the Gaussian noise model assumption, we can express Equation 2.4 as a nonlinear least squares problem:

$$\boldsymbol{x}^*_{0:T}, \boldsymbol{l}^*_{0:J} = \underset{\boldsymbol{x}_{0:T}, \boldsymbol{l}_{0:J}}{\arg\min} \sum_{i=1}^{M} \| f(\boldsymbol{x}_{i-1}, \boldsymbol{u}_i) - \boldsymbol{x}_i \|^2_\Lambda + \sum_{k=1}^{K} \| h(\boldsymbol{x}_{i_k}, \boldsymbol{l}_{j_k}) - \boldsymbol{y}_k \|^2_\Sigma, \tag{2.6}$$

where $\| \|^2_\Sigma$ is the Mahalanobis norm that directly scales the modeling error inversely proportional to the square root of the covariance term Σ. Dellaert and Kaess [34] show that the linearized form of Equation 2.6 can be efficiently solved using sparse linear algebra tools, such as Cholesky factorization. Kaess et al. [76] later introduced

[11]The process by which the probabilistic model is optimized is often referred to as the *backend* of a SLAM system, while the generation of the individual terms is referred to as the *frontend*.

[12]As discussed by Cadena et al. [21], under certain conditions the MAP estimation problem is reduced to a problem of maximum likelihood estimation (MLE).

iSAM, an incremental approach to lower computational latency.

Object-level SLAM is in many ways a natural extension to existing SLAM methods, and may be equally aptly described as *SLAM with object landmarks*, where instead of single point entities the landmarks encode the parameters and properties of objects. The object-landmark paradigm contrasts to approaches that seek to *extract* object-level representations from more general world representations.[13] The evolution from point landmarks to object level landmarks presents many exciting opportunities for expanding robotic performance. In their survey of important historical topics and future directions in SLAM research, Cadena et al. [21] speculate that the development of object-level representations will be important in the "future of SLAM". In particular, they observe the utility of estimating properties beyond position. Additionally, the higher-level abstraction presents opportunities to exploit semantic structure such as scale [49] and inter-object relationships [188]. We are particularly interested in the use of object-level maps to aid in autonomous navigation, and consider various representations choices to this end.

Pose-Only Object-Level SLAM

A common choice of geometric representation is to only optimize the *pose* of landmarks, which can be achieved either by representing objects as point landmarks or assuming access to *a priori* geometric models. Point-landmarks are often seen in sophisticated multi-hypothesis SLAM methods such as those proposed by Bowman et al. [17] and Doherty et al. [38]. Such methods are powerful in that they can overcome the brittleness that results from incorrect data associations or uncertain class labels. However, as discussed previously, estimating objects as point landmarks can remove potentially useful information for autonomous navigation. SLAM++ [150] is one of the earliest methods of object-level SLAM and is an example of a pose-only method that does not assume objects are points. SLAM++ assumes a database of known *a priori* dense 3D geometric object models, allowing for a RGB-D camera to compare

[13]For example, Pillai and Leonard [127] demonstrate that a "SLAM-aware" object recognition system could exploit a point-landmark based map along with vehicle pose estimates to enable multi-view object recognition, but do not explicitly incorporate object-landmarks into the SLAM optimization.

depth measurements against complete geometric models. SLAM++ utilizes objects as a compact representation encoding the geometric patterns of the surrounding environment. Instead of having to track against partial models typical of early-stage mapping, SLAM++ leverages prior information about the objects to predict information not yet densely mapped — without requiring an entire *a priori* map. A later example of object-level SLAM, Deep-SLAM++, exploits class-specific deep neural networks to predict class specific binary occupancy grids that can then in turn be converted to pointclouds to enable point set registration factors. In both instances, significant geometric pre-processing is required. For SLAM++, the object database is created using KinectFusion and then cleaned up in a manual post-processing step. The mapping is interactive, and the authors note that the object is circled during the mapping process and put in an environment without occlusions. Deep-SLAM++ requires a network pre-trained on a large dataset of detailed geometric models.

Object-Level SLAM with Detailed Geometric Representations

TSDFs are another representational choice for object-level SLAM. McCormac et al. [109] present Fusion++, which builds a single TSDF per object (reconstruction) while concurrently estimating the object pose (pose estimation). The scale of each TSDF voxel representation varies with the object, allowing for object models with different resolutions. Each object in Fusion++ takes the average over observed class labels to provide a class-wise distribution. The scene background is used to estimate camera-to-camera constraints (i.e., vehicle motion) and the objects are used to estimate object-to-camera constraints (i.e., map-based localization). TSDFs are generally quite computationally expensive, although the notion of object-ness allows such methods to maintain a single TSDF per object, instead of a single consolidated TSDF over the entire environment. Recent work by Sharma et al. [156] demonstrates that the compositional nature can aid in estimation and rendering, especially when exploiting a GPU.

Object-Level SLAM with Approximate Geometric Representations

Approximate geometric models present a compelling representational option, as they provide a compromise between expressiveness of detailed geometric models and the computational benefits of pose-only point models. CubeSLAM [187] models object landmarks as cubes. Determining the projected 2D image-space bounding box induced by a 3D bounding box involves taking the maximum and minimum of the eight 3D bounding box corners projected onto the image plane, requiring special attention while optimizing.[14] Recently, dual ellipsoid measurements have become a representation of interest in the SLAM community due to their intuitive and differentiable (in some regimes) relationship to bounding box measurements. Nicholson et al. [118] extend the work of Rubino et al. [146] and developed the use of the dual ellipsoid formulation in their proposed algorithm, QuadricSLAM, which uses bounding box measurements in a SLAM framework, estimating both the camera pose and object properties.

As we will show in this thesis, utilizing approximate geometric models in SLAM frameworks can result in geometrically ambiguous estimation under efficient navigation motions, which often are often characterized by straight-line, low-baseline motions. Jointly estimating both pose and shape can be ambiguous unless given many diverse viewpoints[15]; methods that require *a priori* detailed geometric models avoid significant ambiguity by helping to constrain the solution space by virtue of having a known surface, or refraining from estimating the size. In the original QuadricSLAM experiments, the carefully constructed orbiting trajectories and offline batch computation helped to overcome the additional ambiguity induced by unknown approximate geometric models. In the case of CubeSLAM, Yang and Scherer explicitly acknowl-

[14]Numerical gradient calculations are one method of handling the maximum and minimum operators. Alternatively, an approach similar to the maxpool operator for deep neural networks could be considered, where the gradient propagates only through the vertex that induces the bounding box edge.

[15]Consider, for example, approaching a large building from a distance, in a straight-line trajectory. Although bounding box measurements may help constrain the position estimate in the horizontal and vertical directions with respect to the image place, the position of the building along the direction pointing into the image plane is ambiguous. The ambiguity can largely be reduced by *orbiting* the building to provide many diverse views to further constrain the solution.

edge that many 3D bounding boxes project into the same 2D image space bounding box and therefore generate 3D bounding box proposals by sampling vanishing points. The proposals are then scored using geometric heuristics such as angle alignment of detected line segments and penalizing objects with large size skew. Others have suggested augmenting the ellipsoid model with semantic key-points [154], but semantic keypoint detectors are usually computationally expensive.

In Chapter 4, we discuss improvements to object-level SLAM with dual ellipsoid landmarks to improve the performance under challenging vehicle motions typical of efficient autonomous navigation. We exploit additional geometric measurements extracted from a series of RGB images and impose a shape prior over the object.

2.2.4 Deep Learning Methods

Deep learning methods for 3D object estimation exploit datasets during an offline training stage to train deep neural networks.[16] Again, let $\Phi(\boldsymbol{y}_I, \boldsymbol{\tau})$ be a neural network parameterized by $\boldsymbol{\tau}$ that takes as input some measurement $\boldsymbol{y}_I$ and outputs the parameters $\boldsymbol{O}$ of an object estimate. Given N training labels $\boldsymbol{y}_L$, the offline training stage involves optimizing the parameters $\boldsymbol{\tau}$ to minimize the cost function $\boldsymbol{\mathcal{F}}$ (often referred to as a *loss*) that measures the agreement between the predictions and labels. We reproduce Equation 1.3 for readability:

$$\boldsymbol{\tau}^* = \arg\min_{\boldsymbol{\tau}} \sum_{i=1}^{N} \boldsymbol{\mathcal{F}}(\Phi(\boldsymbol{y}_{I_i}, \boldsymbol{\tau}), \boldsymbol{y}_{L_i}), \tag{2.7}$$

It is important to observe that $\boldsymbol{\mathcal{F}}$ can also involve optimization objectives to enable object *detection*, in addition to estimating the 3D properties of objects. One benefit of learning methods is that they can potentially enable single-view estimation, i.e., predict object properties with a single sensor measurement, rather than requiring a series of measurements collected over time. This is by accomplished querying the

[16]For brevity, we defer a comprehensive overview of deep neural networks and convolutional neural networks to Goodfellow et al. [58], which provides an excellent foundation. In this work, we consider neural networks to be general models and instead choose to focus on the high level optimization methodologies used to train the networks.

trained network with a measurement $\boldsymbol{y}_I$ i.e.,:

$$\hat{\boldsymbol{O}}^* = \Phi(\boldsymbol{y}_I, \boldsymbol{\tau}^*) \tag{2.8}$$

For the purposes of autonomous navigation, obtaining the semantic context of the environment sooner is intuitively useful. The potential upside in image to estimate latency is gained by the offline training stage that can require potentially difficult to obtain data to feed the offline training phase.

Learning methods are often categorized based on the type of label $\boldsymbol{y}_{L_i}$ required. Although many terms exist, we observe that the field is a diverse and prolific area of research in recent years and the chosen terminology is still being developed. For the purposes of our discussion, we choose to organize the following sections according to the type of annotation or *a priori* information required. In particular, we consider methods which use a subset of the following types of information for training:

- **Direct Annotations**: Annotations that label exactly the quantities of interest, i.e., pose and shape of a primitive object model. An example annotation would be the parameters of a 3D bounding box for a 3D bounding box estimator. Direct annotations provide *directly supervision* to the learning problem; *full supervision* is possible when direct annotations are available for all quantities of interest.

- **Indirect Annotations**: Annotations that indirectly capture geometric properties of the object. For example, annotations in 2D pixel coordinates such as class segmentation or image-space keypoints can provide information about the pose and shape of objects, although they require intermediate calculations. Indirect annotations provide *weak supervision* as they provide signals of the quantities of interest, rather than the quantities themselves.

- ***A Priori* Geometric Models**: Geometric model of objects, either per object, or in general database form. We focus on methods that utilize mesh models of known scale and known object 3D bounding box sizes as *a priori* models.

We are primarily interested in methods that consume RGB images at inference time. As noted by Arnold et al. [7], many deep learning methods which operate on pointcloud data are computationally expensive due to the practical consequences of 3D convolution operations and input size. While methods for object estimation without annotations or prior models of any type do exist, they usually involve involving computations over over LiDAR data [148] or pointclouds [183]. Additionally, the recent success of pseudo-LiDAR methods [180, 130, 181] that convert RGB images into the dataform of LiDAR measurements are a potential route for avoiding expensive sensors at inference time. Such methods in general do not avoid computational issues unless they use networks that consume LiDAR data in the form of a bird's eye view (BEV) measurements, which in turn imposes potentially undesirable 2.5 assumptions. It is also important to note that 3D object estimation and 3D object detection are closely related. Many 3D object detection frameworks perform detection (i.e., finding objects) and estimation (i.e., estimating the size of the 3D bounding box). While some methods for 3D object detection are end-to-end, some are multi-stage methods that separate the detection of objects and the estimation of object parameters to different stages of the algorithm[173].[17] In Chapter 5 we assume a parallel process for 2D object detection.

Direct Annotations

Early approaches to 3D bounding box detection formulated object estimation as a fully supervised learning problem, and thus relied on datasets of images labelled with object sizes, orientations and shape. Building on previous success in 2D object detection, Mono3D [28] focuses on 3D object proposal generation. The algorithm takes into consideration semantic segmentation, shape, context, and object location to generate proposals. Ultimately, the 3D bounding boxes are scored by a detection algorithm using a modified Fast R-CNN and a multi-task loss over classification, orientation and offsets for the bounding boxes. Chen et al. [29] present an approach that combines

[17]Unlike Kim and Hwang [83], we consider end-to-end methods as those which do not involve further computation such estimating 3D pose from 2D correspondences.

different modalities of information from several different views including a bird's eye view and a front view of lidar data, in addition to RGB images. ROI-1OD [108] is an end-to-end learning approach that fuses the features of a RGB image and a depth image estimated by a neural network. From the fused feature representation, for each 2D detection the ROI is "lifted" to a 3D bounding box detection. In both Chen et al. [29] and ROI-10D, the detection is scored by comparing the eight 3D bounding box corners to the ground-truth 3D bounding box corners.

Although 3D bounding boxes are by far the most popular approximate geometric model for estimating object pose and shape, recent works have begun to extend supervised learning methods to other primitive model types given labels. For instance, [159] propose learning the parameters of superquadrics from range images.

In general, supervised learning presents practical challenges attempting to extended to novel object classes due to the onerous burden that 3D annotation presents, especially compared to 2D image-space annotation. In contrast to 2D labelling problems such as bounding boxes or pixel-wise segmentation, which are annotated purely on the image plane, annotating the 3D object parameters from 2D image data often requires interfacing with additional information beyond the image itself, such as dense 3D pointclouds [53], detailed geometric models [185], or using pre-trained 3D object detection neural networks [96]. Although modern object detectors such as Redmon et al. [135], He et al. [61] have benefited immensely from large open-sourced datasets with 2D annotations over diverse classes of objects such as [99], such datasets are much more difficult to obtain for 3D object estimation tasks. Existing large-scale datasets are largely focused on the autonomous driving domain, such as the KITTI dataset [53] and the Waymo Open Dataset [168]), or the household domain such as the Google Objectron Dataset [4]. The relative lack of diverse 3D datasets and the cost of obtaining 3D annotations pose practical limitations to extending supervised approaches to arbitrary classes. In Chapter 5 we present a deep learning method for 3D object estimation that does not require 3D annotations.

A Priori Geometric Models

Rather than assuming a labelled dataset of 3D bounding box annotations per object, some methods assume access to geometric models of the objects to learn latent representations of objects that can potentially enable unsupervised learning methods for deep object estimation. State-of-the-art methods [13, 122] pair large object datasets with differentiable rendering.

Beker et al. [13] propose a self-supervised learning method that relies on a learned representation that can be used to render objects. First, an object latent space is learned from a large repository of 3D CAD models of cars using an auto-encoder structure that outputs both a detailed shape and texture for the object. Given the shape, texture, object pose, and object dimensions, a differential renderer generates the 2D projection of the object. Because both the renderer and the decoder network are fully differentiable, the 3D properties of a 2D detection can be determined via analysis-by-synthesis. In particular, a variety of losses are constructed, including RGB appearance, bounding box comparisons, and depth map comparisons.

LatentFusion [122] further exploits large datasets of *a priori* geometric models to generalize to novel objects. Park et al. train a neural network to predict a latent object that can be transformed using rigid body 3D transformations and from which RGB images and depth images can be differentiably rendered. To train the network, 12 reference RGB images, segmentations, and poses are sampled from around a given object from the dataset, in addition to 12 targets. The network learns to use the reference frames and poses to predict latent objects that can be used to render the measurements matching the target frames. To estimate the pose of an arbitrary object without requiring an *a priori* model, the network is again queried using reference information. Because the rendering pipeline is differentiable, pose of the object can be iteratively optimized to minimize error between the rendered measurements and observed measurements.

Despite the expressiveness of the methods discussed, they both require potentially difficult data to collect, i.e., a dataset of detailed CAD models for a single class [13] or reference frames with known object poses [122].

Indirect and Direct Annotations

Indirect annotations have the potential to be much easier to annotate if they are performed in the image-space. The proliferation of datasets and annotation tools for 2D bounding box and 2D image segmentation speaks volumes of the relative ease of obtaining indirect annotations.

The algorithm proposed by Mousavian et al. [113] is an example of requiring both direct and indirect annotations. The proposed approach requires labels only of the size and orientation of the object for training. After the network is trained to predict the size and orientation of the 3D bounding box, the authors propose using a 2D bounding box measurement of the object to lift the estimate into 3D using geometric constraints. Therefore, while some direct labels are required, properties such as position are estimated by the indirect annotation of 2D bounding boxes. Specifically, by assuming that the 2D bounding box should tightly enclose the 3D bounding box, a corner of a 3D bounding box should induce one of the 2D bounding box edges when projected into the image space. Although for arbitrary 3D bounding boxes there are thousands of candidate correspondences of 3D bounding box corners to 2D bounding box edges to consider, given the assumption that the object is upright and object roll is minimal the authors consider only 256 correspondences.

Direct Annotations and *A Priori* Geometric Models

Pairing direct annotations with *a priori* geometric models can enable more expressive learning. In ROI-10D, the authors discuss how the 3D bounding box estimator can be augmented with a shape space learned from commercially available models of cars. The learned shape space allows ROI-10D to additionally predict a detailed geometric shape of the object. Similarly, 3D-RCNN [89] uses a database of detailed geometric models to determine a shape basis via principle component analysis (PCA). The algorithm supervises directly on pose and shape if annotations are available. Among other terms, the loss function for 3D-RCNN includes a render-and-compare cost, which compares a projected 2D segmentation or depth image generated by the 3D models to the ground-truth measurements.

The combination of direct annotations and *a priori* geometric models can also enable methods that estimate the pose of objects only, assuming known object models or dimensions. Xiang et al. [186] propose PoseCNN, an algorithm for learning to detect objects and estimate their 6D poses. The authors assume ground-truth position annotations, but do not use ground-truth orientations directly. Instead, given *a priori* mesh models of the objects, they propose a novel loss function to regress the rotation of the object by comparing the points on the mesh model transformed by the pose estimate to the points on the mesh model transformed by the true pose. The ground-truth rotation is still required to acquire the ground-truth object mesh in world coordinates, but access to the detailed object model enables the authors to propose a novel method for orientation regression.

Indirect Annotations and *A Priori* Geometric Models

Given detailed *a priori* geometric models, some methods have shown promising results by relaxing the direct annotation requirement to indirect annotations, such as 2D keypoints. Keypoint-based methods combine indirect 2D-pixel point annotations with *a priori* geometric models to enable deep 3D object estimation. Keypoint-based methods use *a priori* geometric models to predict where specific points on the object models project into image-coordinates. The type of keypoint predicted varies from general to object-specific. Tremblay et al. [173] present Deep Object Pose Estimation (DOPE). Using a photo-realistic simulator, they project the ground-truth 3D bounding box corners of rendered objects into image pixel coordinates. They then train a neural network to predict the projected 2D pixel coordinates of the bounding box corners and solve for the 6D pose of a 3D bounding box of known size by solving a Perspective-n-Point (PnP) problem.[18] Pavlakos et al. [126] use a dataset

[18]A case could be made for considering DOPE a method that combines direct annotations and priori geometric models, given that the authors leverage large-scale photo-realistic simulators enabled by detailed geometric models and known ground-truth in simulation is what enables the 3D bounding box keypoint annotations. We choose to categorize the method as using indirect labels to highlight

of annotated object-specific keypoints in image-coordinates to train a neural network to predict their pixel-coordinates. At inference time, a deformable shape model is optimized so that the projected keypoint locations agree with the predicted keypoint location using a weighted loss function that attempts to model the noise in the keypoint prediction. Both approaches require *a priori* geometric knowledge, i.e., size information in the case of DOPE and a detailed deformable shape model in the case of Pavlakos et al. [126].

Direct Annotations, Indirect Annotations, and *A Priori* Geometric Models

Unsurprisingly, using both direct and indirect annotations in addition to *a priori* geometric models is also a candidate approach if all three types of information are obtainable. In SSD-6D, Kehl et al. [82] use detailed models of specific objects to render different "viewpoints" of objects. Rather than estimating a continuous rotation value, the network is then tasked with learning to score a discrete set of possible viewpoints of the object in a candidate 2D bounding box image. As the size of the object is known, bounding box ratios are used to determine the position of the object. To verify the final solution, the algorithm again renders the object and scores based on image-space gradient properties compared to the observed image. The *a priori* geometric model additionally enables further online fitting by considering the rendered contour of the object and rendered depth image. In the case of SSD-6D, *a priori* geometric models provide direct annotations by enabling the generation of synthetic training data, and enable the use of indirect labels such as object contours and depths.

As discussed in this section, various combinations of direct annotations, indirect annotations, and *a priori* geometric models have been proposed to enable deep 3D object estimation. Direct annotations for 3D properties are onerous to obtain as they usually require data that is more difficult to interact with for annotation purposes. *A priori* geometric models are often non-trivial to obtain for arbitrary classes. In Chapter 5, we propose a method for training a deep neural network to predict 3D

the use of 2D image-space annotations, which could potentially come from a human annotator.

object properties from a monocular image using indirect annotations only.

2.2.5 Hybrid Methods

Hybrid methods use deep 3D object estimation in conjunction with multiple estimation techniques. One promising direction for hybrid methods is to learn representations or measurement functions that are then incorporated into larger estimation frameworks. For instance, Sucar et al. [167] show how a latent representation can be learned using an auto-encoder architecture, whose objective is to reconstruct a voxel occupancy mapping of an object. After the auto-encoder has been trained, the resulting latent space is used as the representation of the landmark. The authors show how depth measurements can be used to update the learned representation using differentiable rendering and the trained decoder. The resulting measurement function is incorporated into a SLAM framework called NodeSLAM.

Another intuitive method for using multiple object estimation techniques together is exploiting one method to generate priors and another to continue online estimation. For example, in PoseCNN [186], the initial neural network pose estimate of the object is used to seed an iterative closest point optimization. The authors show large improvements in estimation qualitative after additional online optimization. Deng et al. [35] propose PoseRBPF, which utilizes a Rao-Blackwellized Particle Filter where object states are encoded by a learned representation for 6D object pose estimation. Importantly, they demonstrate that by adding particles sampled near PoseCNN estimates in addition to their proposed method, object estimation performance can exceed both PoseCNN and PoseRBPF for some objects in the YCB Video Dataset. In Chapter 5, we show how our deep learning approach to object estimation can be further optimized online given additional sensor measurements.

2.3 Summary

In this chapter, we have reviewed existing approaches to autonomous navigation and object-level estimation. We discussed various methods of solving the autonomous planning problem, from representational choices to defining search heuristics. We then described planning methods developed specifically for navigation in uncertain and unknown environments, closing with methods that seek to combine semantics and geometry. We then focused on object-level estimation, first discussing common geometric representations for objects. Finally, we explored various methods for optimizing object-level representations, including regression and classical detection, SLAM, deep learning, and hybrid methods, with emphasis on SLAM and deep learning approaches.

In the following chapters, we develop methods that combine geometric and semantic information to improve autonomous navigation (Chapter 3), object-level SLAM (Chapter 4, and deep 3D object estimation (Chapter 5).

Chapter 3

Learned Sampling Distributions for Efficient Planning

3.1 Introduction

In this chapter, we develop a method for improving navigation in unknown environments using both object level semantics and dense geometry. As we discussed in the introduction, algorithms that rely on dense geometric representations to generate motion plans often result in myopic behavior when deployed in novel, unknown environments. Although object-level semantic information is an intuitive modality to provide important contextual cues, it is not immediately obvious how to incorporate it into planning formulations. For example, we have discussed in Section 2.1.4 how some methods intelligently factor the environment into frontiers (i.e., boundaries between known and unknown space), but while doors are frontiers between rooms and hallways, not all objects should be navigation waypoints — exit signs are objects that provide information about nearby doors, and windows indicate the extents of buildings. Furthermore, the utility of object relationships can be a complex function that may change conditioned on the ultimate objective; the usefulness of a door with an exit sign is determined by whether the goal is outdoors. Other methods use rich hierarchical semantic representations that are dense and potentially difficult to quickly build online.

Our key insight is to combine the computational power of randomized motion planners with higher-level semantic information via a learned sampling distribution, enabling intelligent navigation in structured, unknown environments. We predict a sampling distribution from local dense geometric representations that track unobserved space, explicit object-level contextual cues both within and beyond the range of dense geometry, and information about the goal. The ability to predict this distribution enables SBMPs to not only find plans quickly, but in certain environments also reason about which plans are most likely to reach the goal despite incomplete geometric information. Rather than pre-specifying navigation rules and heuristics over our representations, we show that a Convolutional Neural Network (CNN) can be leveraged to synthesize multimodal map information into a proper sampling distribution, and that learning a sampling distribution over geometric and semantic information improves navigation results in unknown environments. We first train a network general sampling distribution, then train a second network to modify the general sampling distribution given contextual information about the goal. Additionally, as ground-truth labels of sampling distributions are extremely difficult to obtain, our network is weakly supervised on expert trajectories.

Our method combines ideas from Chapter 2 to tackle both the sample efficiency and operational efficiency of autonomous navigation in unknown environments using SBMPs. We avoid the use of potentially brittle action abstractions such as frontiers and instead learn the utility of different areas of the workspace (Section 2.1.3), by encoding promising navigation strategies in a sampling distribution for use with a SBMP. Furthermore, our approach extends the use of learned distributions to focus computation from domains requiring *a priori* geometric environment representations (Section 2.1.2) to partially observed hybrid geometric and semantic environment representations (Section 2.1.4). We propose to use the learned distribution not only for focusing computation, but to bias solutions to pass through regions predicted to be more likely to lie on the optimal path by setting planning graph edge weights using predicted probabilities.

In the following section, we formalize the planning problem, describe our proposed neural network architecture and training procedure, and detail how our learned distribution is used online (Section 3.2). In Section 3.3, we evaluate the performance of our learned sampling distribution in novel environments for a simulated holonomic robot, and show that a PRM using our learned distribution is not only more likely to find trajectories in partially-mapped environments, but can also result in lower overall distances travelled on average. Finally, we demonstrate the utility of the learned sampling measure on a real-world RC car platform.

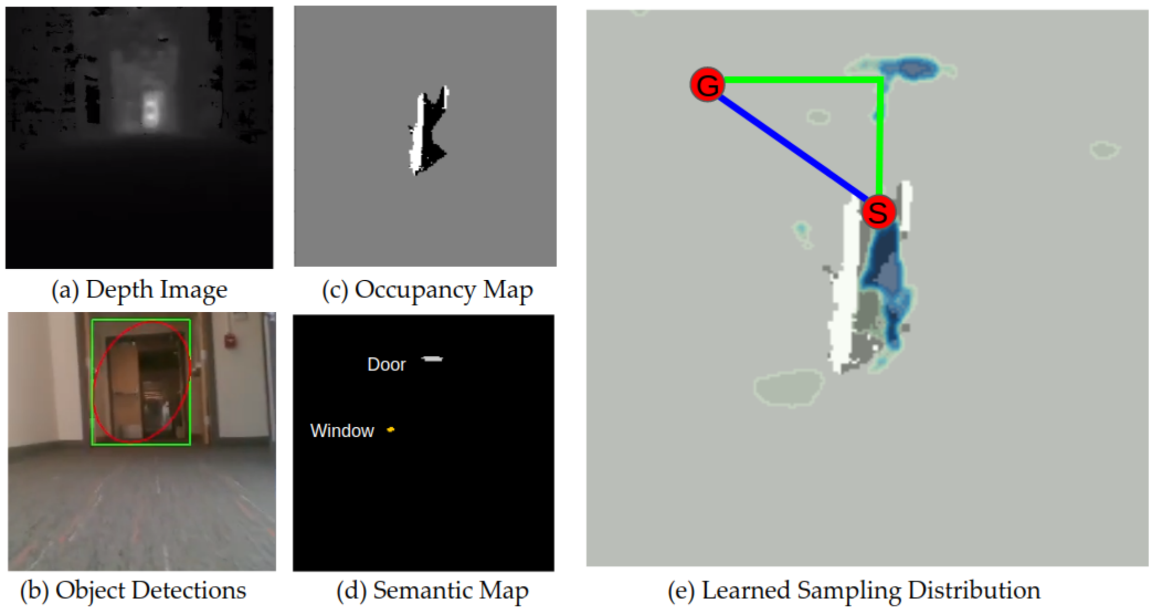


Figure 3-1: In this work, we combine geometric (c) and object-level (d) representations built from depth images (a) and object detections (b) to predict sampling distributions for sampling-based motion planners (e). Myopically assuming that unknown space is always traversable results in trajectories such as the blue path. Our learned sampling distribution can be used to leverage contextual cues such as the existence of a door to exit the room and to generate paths similar to the illustrative green path.

3.2 Learned Sampling Distributions

In this section, we first review the definition of a planning problem, then re-formulate the problem in the context of learning a maximum-likelihood sampling distribution informed by multimodal information. Finally, we describe how to generate training data and the structure of the model.

3.2.1 Optimal Planning in Unknown Environments

Recall the planning problem introduced in Chapter 1, repeated here for readability:

$$\begin{aligned} &\lambda^* = \arg\min_{\lambda \in \Lambda} c(\lambda) \\ \text{s.t.} \quad &\lambda^*(t) \in \mathcal{X}_{free} \; \forall t \in [0,1] \\ &\lambda^*(0) = x_s, \lambda^*(1) = x_g \end{aligned} \tag{3.1}$$

where $\mathcal{X} \in \mathbb{R}^d$ is the state of the robot, $\mathcal{X}_{free}$ is the set of unoccupied states, and $x_s, x_g \in \mathcal{X}$ are the start and goal states respectively. A path is denoted λ, the set of all paths is denoted Λ, and the cost of a path given as $c(\lambda)$. Equation 3.1 seeks to find a collision free path from the start to the goal that also minimizes the length of the trajectory.

Sampling-based motion planners solve this optimization problem by extracting random graphs [51] to approximate a solution to Equation 3.1. Graphs are generated by sampling states $x \in \mathcal{X}$ from a sampling distribution $P(x;\Gamma)$, where Γ are optional additional inputs to the distribution, then constructing a navigation graph G from valid states, where $G = \{V, E\}$, $V = \{v_0, v_1, ...v_n : v \sim P(x;\Gamma)\}$, $E = \{e_0, e_1, ...e_m\}$, and $V, E \in \mathcal{X}_{free}$. For notational convenience, we will also interchangeably express an edge as e_{ij} to represent the directed edge from v_i to v_j. In this representation, the n vertices of the graph correspond to the sampled states, and the m edges represent feasible traversals from one state to another[1]. A valid trajectory $\mathcal{T}$ is a connected, acyclic sub-graph of G_t. The optimization in Equation 3.1 can then be formulated as:

$$\begin{aligned} &\mathcal{T}^* = \arg\min_{\mathcal{T}} \boldsymbol{C}(\mathcal{T}) \\ \text{s.t.} \quad &\mathcal{T}^* = \{V^*, E^*\}, b(e_0) = x_s, t(e_m) = x_g, \\ &v \neq v' \;\; \forall v, v' \in V^* \\ &t(e_i) = b(e_{i+1}) \; \forall i \in \{1...m-1\} \\ &V^* \in V, E^* \in E, \end{aligned} \tag{3.2}$$

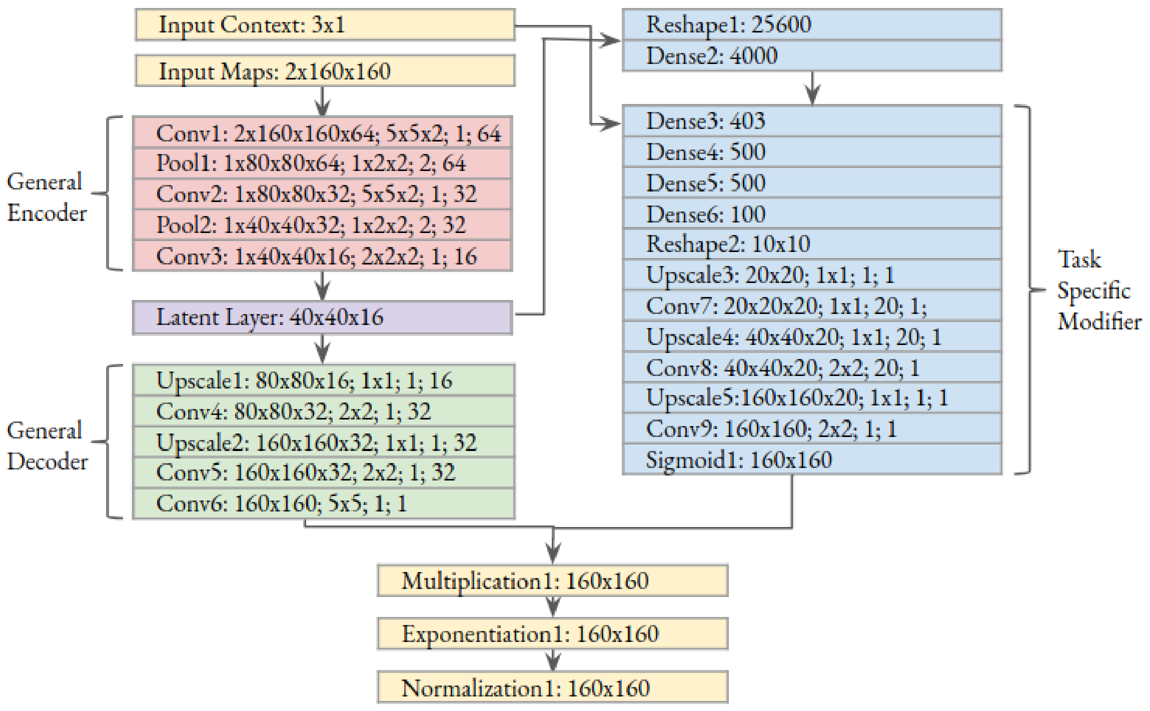
[1]We observe that the vertex connection strategy varies between different algorithms, but neglect the notation here for readability.

where $\boldsymbol{C}(\mathcal{T}) \in \mathbb{R}$ is a scalar cost function that evaluates the cost of a sub-graph, $t(e)$ indicates the terminal node of edge e, $b(e)$ indicates the start node of edge e, and $V, E \in \mathcal{X}_{free}$.

Intuitively, a good choice of $P(x; \Gamma)$ is one more likely to sample Gs that contain $\mathcal{T}$s that are close to the solution to Equation 3.2. A common approach is to leverage a fully known geometric map and or the geometric location of the goal (M_g^* and x_g) as conditioning parameters, $\Gamma = \{M_g^*, x_g\}$ [189, 20, 70]. However, this choice of parameterization is often limiting in real-world environments, where sensor limitations mean that geometric models are almost certainly inaccurate and incomplete. Assuming maps are complete is particularly detrimental when traditional methods rely on complete geometric information to eliminate low-cost but ultimately infeasible paths by searching only in the space of valid trajectories (i.e. $\{\mathcal{T} \mid \forall e \in \mathcal{T}, e \in \mathcal{X}_{free}\}$).

3.2.2 Multimodal Information for Navigation

In this work, we propose a novel approach to navigation in unknown environments, which learns a sampling strategy from both geometric and non-geometric navigational cues. Specifically, we would like to take advantage of sparse, metrically aligned object-level maps, which can provide semantic information both within and beyond the partial dense geometric map, to inform planning. By learning a predictive sampling distribution inferred from information about unknown space, low-level geometry, and explicit object-level contextual cues such as doorway and exit signs, we can empower SBMPs to reason about the pertinent structure of environments that have only been partially observed and therefore find better plans. For example, instead of having to densely map the walls of a room in order to exit it, the presence of a door is a clear semantic cue for a navigation strategy that has high probability of success (see Figure 5-1). Formally, we let $\Gamma = \{\mathcal{M}, \Upsilon\}$, where $\mathcal{M}$ denotes a *hybrid* form of map representation that incorporates low-level geometric information with semantic, object-level information (both of which may be incomplete). Υ denotes planning context parameters, such as the goal location and whether the goal is indoors or outdoor. We solve Equation 3.2 over the graph sampled from $P(x; \mathcal{M}, \Upsilon)$.



Input Context: 3x1
Input Maps: 2x160x160
General Encoder
Conv1: 2x160x160x64; 5x5x2; 1; 64
Pool1: 1x80x80x64; 1x2x2; 2; 64
Conv2: 1x80x80x32; 5x5x2; 1; 32
Pool2: 1x40x40x32; 1x2x2; 2; 32
Conv3: 1x40x40x16; 2x2x2; 1; 16
Latent Layer: 40x40x16
General Decoder
Upscale1: 80x80x16; 1x1; 1; 16
Conv4: 80x80x32; 2x2; 1; 32
Upscale2: 160x160x32; 1x1; 1; 32
Conv5: 160x160x32; 2x2; 1; 32
Conv6: 160x160; 5x5; 1; 1
Reshape1: 25600
Dense2: 4000
Dense3: 403
Dense4: 500
Dense5: 500
Dense6: 100
Reshape2: 10x10
Upscale3: 20x20; 1x1; 1; 1
Conv7: 20x20x20; 1x1; 20; 1;
Upscale4: 40x40x20; 1x1; 20; 1
Conv8: 40x40x20; 2x2; 20; 1
Upscale5:160x160x20; 1x1; 1; 1
Conv9: 160x160; 2x2; 1; 1
Sigmoid1: 160x160
Task Specific Modifier
Multiplication1: 160x160
Exponentiation1: 160x160
Normalization1: 160x160

Figure 3-2: Neural network used to generate learned sampling distributions. The network uses geometric and semantic information to learn general and task specific sampling distributions that empower a robot to reason about navigational modes in unknown space. Layer data indicates, in order, layer size, convolution size, stride size, and number of filters. Dropouts are not pictured but used for regularization. The context information is a vector with the first two values proportional the relative location of the goal, and a scaled indicator variable indicating whether the goal is inside or outside.

However, even with an optimized sampling distribution, we observe that simply being more likely to place samples along the optimal path does not encourage more likely trajectories to be chosen if the objective function of the graph search is not also informed by the information in the hybrid map. It is easy to see that in the limit of infinite samples, an objective function that utilizes distance only would revert back to the myopic behavior of SBMPs using uniform sampling when operating in unknown environments. To mitigate this, we propose using the probability function not only to define the implicit planning graph, but *also* as a cost function over graph edges:

$$\boldsymbol{C}(\mathcal{T}) \propto -\log(\mathrm{P}((\mathcal{T})). \tag{3.3}$$

3.2.3 Learning Sampling Distributions

In general, determining the form of $P(x; \mathcal{M}, \Upsilon)$ analytically is intractable. Instead, we would like to approximate the distribution with some mapping ϕ that depends on the hybrid map representation and the contextual information:

$$P(x; \mathcal{M}_i, \Upsilon_i) \approx \phi(x; \alpha, \mathcal{M}_i, \Upsilon_i). \tag{3.4}$$

where α denotes the parameters of the predictive sampling distribution model. A naive approach to the optimization of ϕ is to assume a dataset of optimal sampling distributions $P^*(x; \mathcal{M}_i, \Upsilon_i)$, then attempt to minimize some distance metric between $P^*(x; \mathcal{M}_i, \Upsilon_i)$ and $\phi(x; \alpha, \mathcal{M}_i, \Upsilon_i)$. In practice, specifying $P^*(x; \mathcal{M}_i, \Upsilon_i)$ is extremely difficult, especially because such a distribution must account for the existence of multiple optimal trajectories that may exist in a complex environment.

While it is difficult to obtain example ground-truth sampling distributions, obtaining a representative dataset $\mathcal{D}$ of q example trajectories along with partial, hybrid environmental information and context information such that $\mathcal{D} = \{(\mathcal{T}_0, \mathcal{M}_0, \Upsilon_0), (\mathcal{T}_1, \mathcal{M}_1, \Upsilon_1), ..., (\mathcal{T}_q, \mathcal{M}_q, \Upsilon_q)\}$ is a far more tractable endeavor. The dataset pairs complete trajectories through incomplete maps, as seen in Figure 3-6. Therefore, we propose learning a mapping for the sampling distribution by maximizing the probability that an optimal trajectory $\mathcal{T}_i$ will be sampled given $\mathcal{M}_i$ and Υ_i, over the training dataset:

$$\arg\min_{\alpha} \sum_{i}^{q} -\log P(\mathcal{T}_i; \mathcal{M}_i, \Upsilon_i), \tag{3.5}$$

where the maximum likelihood optimization has been converted into a negative log-likelihood minimization. The dataset $\mathcal{D}$ may be obtained offline in simulation, or in a gradual online fashion running any traditional planner that eventually finds optimal plans.

However, learning a joint distribution over V and E is extremely difficult due to the exponential state space of all possible sampled graphs. We therefore approximate Equation 3.5 as the joint distribution over nodes only:

$$\arg\min_{\alpha} \sum_{i}^{q} -\log P(V_i; \mathcal{M}_i, \Upsilon_i). \tag{3.6}$$

Equation 3.6 abstracts trajectories into collections of nodes, rather than edges, and is similar to the approach taken in [70]. For tractability, we make a final approximation that the vertices $V_i = v_{i_0}...v_{i_j}$ in the sampled trajectory are conditionally independent:

$$\arg\min_{\alpha} \sum_{i}^{q} -\log(\prod_{j=0}^{|\mathcal{T}_i|} P(v_{i_j}; \mathcal{M}_i, \Upsilon_i)). \tag{3.7}$$

Finally, we substitute in our learned approximation ϕ for P:

$$\arg\min_{\alpha} \sum_{i}^{q} (\sum_{j=0}^{|\mathcal{T}_i|} -\log\phi(v_{i_j}; \alpha, \mathcal{M}_i, \Upsilon_i)). \tag{3.8}$$

In practice, we optimize a scaled version of Equation 3.8 for numerical stability.

3.2.4 Neural Network Model Structure and Optimization

In this work, a convolutional neural network serves as the base form of model ϕ, and the optimization over α in Equation 3.8 is the optimization of the network weights. CNNs are particularly attractive due to their demonstrated successes in feature extraction, especially for multi-modal data inputs [117]. Nevertheless, how to condition multi-modal information for deep learning algorithms is still an area of open research. To effectively use both geometric and semantic information we must determine an appropriate representation for combining our geometric and object-level information.

We choose to represent object-level information as a discretized overhead map, with the same scale and origin as the occupancy map. To encode semantic information, the value at a discrete location is determined by the object class. There are several benefits to this choice. First, we can represent an arbitrary number of objects with a fixed-size representation.[2] Second, by translating object-level information into a representation very similar to occupancy information, we can reasonably exploit the modeling power of stacked convolutional layers to form our hybrid representation, i.e., $\mathcal{M} = [M_g; M_s]$ such that $\mathcal{M} \in \mathbb{R}^{k \times k \times 2}$, where $M_g \in \mathbb{R}^{k \times k}$ is a local pixel-wise map with occupancy information, $M_s \in \mathbb{R}^{k \times k}$ is a discretized local semantic information map. Practically, we can still maintain the sparsity of an online object estimation algorithm, projecting the list of known objects to a 2D discrete representation when needed.

The two maps M_g and M_s are concatenated together depth-wise before being passed to an encoding CNN ϕ^{enc}, whose output is then decoded via ϕ^{dec} to form a general un-normalized distribution ϕ^{gen}:

$$\phi^{gen}(x_d; \alpha_{\{dec,enc\}}, \mathcal{M}) = \phi^{dec}(\phi^{enc}(x_d; \alpha_{enc}, \mathcal{M}); \alpha_{dec}), \tag{3.9}$$

where $k \in \mathbb{Z}_{>0}$, $x_d \in \mathcal{X}_d$, $\mathcal{X}_d$ is a discretized form of $\mathcal{X}$, and $\alpha_{dec}, \alpha_{enc}$ denote the network weights for their respective networks. To incorporate contextual information with map-level information, we concatenate the output of the encoding CNN with Υ, and pass the new matrix through a second decoding ϕ^{cs} that modifies the general distribution from Equation 3.9 to form an unnormalized context-specific distribution

[2]This benefit is also exploited by Driess et al. [42] who use an overhead depth image to predict plan feasibility when there are an arbitrary number of objects in a table-top setting.

ϕ^P, i.e.,

$$\phi^P(x_d; \alpha_{\{dec,enc,cs\}}, \mathcal{M}, \Upsilon) = \phi^{gen}(x_d; \alpha_{\{dec,enc\}}, \mathcal{M}) \circ \phi^{cs}(\phi^{enc}(x_d; \alpha_{enc}, \mathcal{M}); \Upsilon, \alpha_{cs}), \tag{3.10}$$

where the context variable includes the relative location of the goal and a flag indicating whether the goal is indoors or outdoors, such that $\Upsilon \in \mathbb{R}^{d+1}$. To enforce a proper probability distribution P, we pass the raw output of the network through a softmax layer to obtain the final normalized distribution:

$$\phi(x_d; \alpha_{\{dec,enc,cs\}}, \mathcal{M}, \Upsilon) = \text{softmax}(\phi^P(x_d; \alpha_{\{dec,enc,cs\}}, \mathcal{M}, \Upsilon)). \tag{3.11}$$

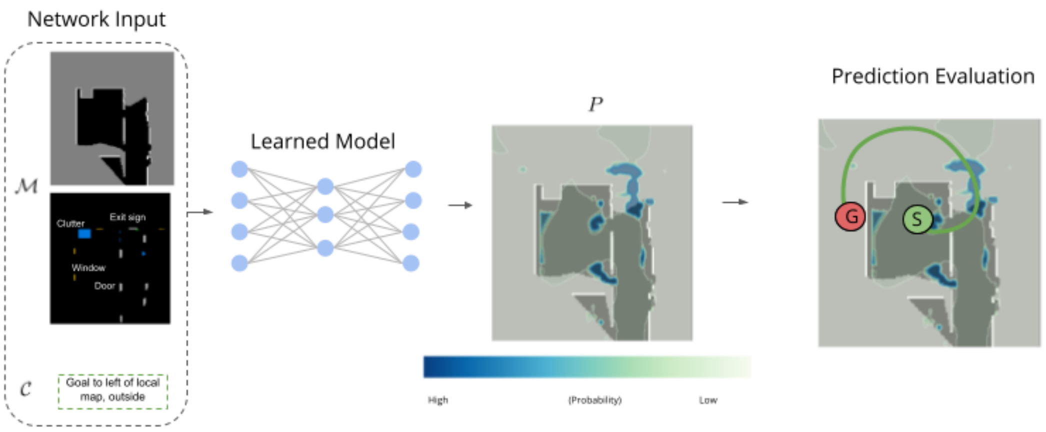


Figure 3-3: High-level overview of the inputs, outputs, and loss function evaluation of the network. Partial occupancy and object maps, where a learned model predicts a probability distribution (see Figures 3-2 and 3-4 for additional details on the form of the model). The probability that the labelled, resolution optimal, trajectory would be sampled under the predicted distribution is then calculated using Equation 3.8.

We propose a two-stage optimization that learns two types of navigational modes separately. In the first stage, we optimize α_{enc} and α_{dec} and learn a task-independent sampling distribution that highlights general navigation modes of the environment. In the second stage, we freeze the network weights associated with general navigation modes to avoid task-specific overfitting, and learn a task-specific modifier of the original distribution that takes into account the context associated with a specific task (i.e., optimize α_{cs}). See Figure 3-2 for network details, and Figure 3-5 for a qualitative example of how the general distribution and task-specific distribution differ. The network optimizes Equation 3.11 via back-propagation and stochastic gradient descent (SGD) with mini-batches.

3.2.5 Online Planning in Unknown Environments

To use the learned distribution online, we first optimize Equation 3.8 offline by applying SGD to the structure in Equation 3.11. During each timestep of online use, we generate incomplete local geometric and semantic maps, then run the feed-forward model to recover a normalized probability distribution over the current, partially known map. Grid locations outside of the local sliding window are set to the minimum probability of the predicted window, and the sampling distribution over the entire map is again re-normalized. In practice, we sample from the discretized state for efficiency, but observe that uniformly sub-sampling within a discretized state could extend this approach to more complex systems[3]. Finally, we use a probabilistic roadmap (PRM) to generate a trajectory from the current robot position to the goal, but modify the planner to sample graph vertices from the learned distribution and to score graph edges using the objective defined in Equation 3.3.

3.3 Experiments

In this section, we describe the experimental procedure used to benchmark our learned sampling distribution against a uniform sampling distribution in simulated floorplan environments, and report aggregate metrics. Finally, we show qualitative results on data collected by a real-world vehicle.

[3]It is important to note that although the state is discretized, we avoid the extremely limited action set the discretized methods discussed in Section 2.1.1 use in practice.

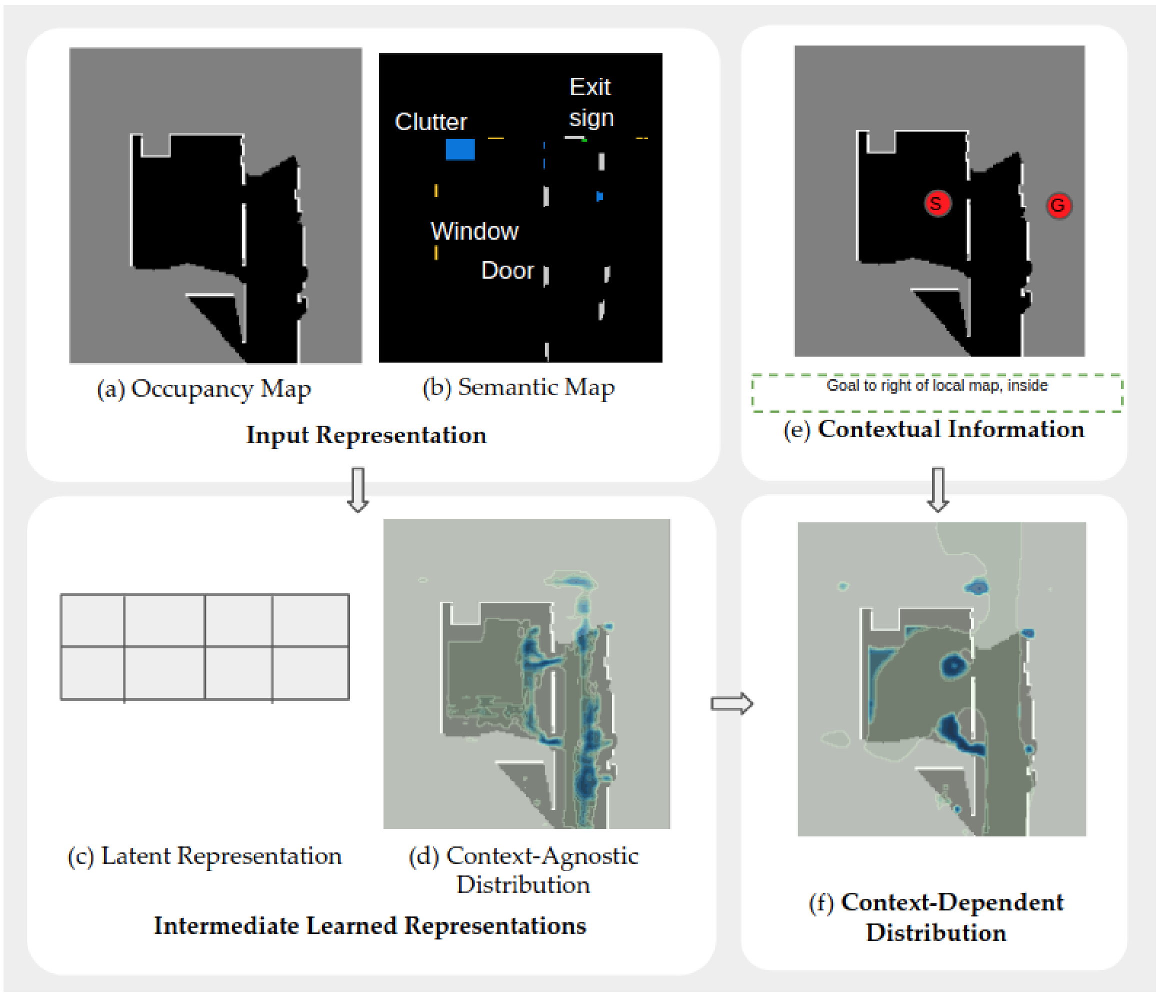

Clutter
Exit sign
Window
Door
S
G
Goal to right of local map, inside
(a) Occupancy Map
(b) Semantic Map
Input Representation
(e) Contextual Information
(c) Latent Representation
(d) Context-Agnostic Distribution
Intermediate Learned Representations
(f) Context-Dependent Distribution

Figure 3-4: Overview of two stage network training process.. Learned distributions are plotted as filled contour plots (blue is high probability, grey-green is low probability) overlaid on occupancy maps. In the first half of the network, local occupancy (a) and semantic (b) maps are passed through a CNN that learns a latent representation (c) and a context-agnostic distribution (d). A second network takes the latent layer (c) and contextual information (e) to learn a context-specific modifier, which is multiplied against the general distribution to obtain a context-dependent distribution (f). Without contextual information, the distribution learns only general navigation heuristics.

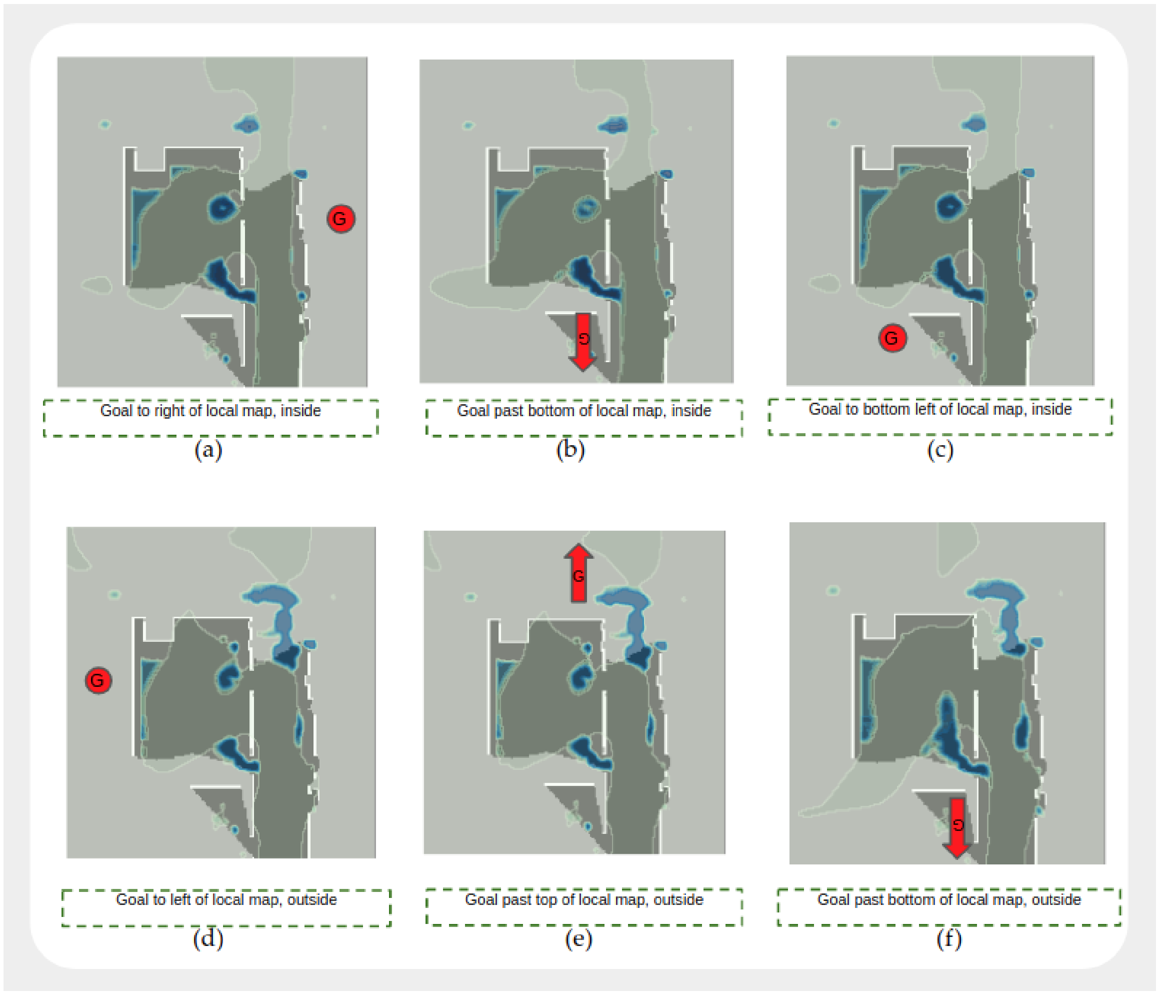

Goal to right of local map, inside
(a)
Goal past bottom of local map, inside
(b)
Goal to bottom left of local map, inside
(c)
Goal to left of local map, outside
(d)
Goal past top of local map, outside
(e)
Goal past bottom of local map, outside
(f)
G

Figure 3-5: Qualitative examples of the effect of different contextual inputs to the second, context-dependent distribution. Goal indicators for qualitative purposes only; goals indicated by red arrows are beyond the local map. After adding contextual information about the goal, the network learns to encode different navigation strategies. For example, when the goal is to the right and inside, the exit sign seen in Figure 3-4b has low probability (a), but when the goal is to the left and outside, the exit sign has higher probability, biasing the planner to exit the building (d). The learned distribution is able to place probability near the exit sign, despite not having densely mapped the region.

Dataset	*II*	*II*	*II*	*II*	*IO*	*IO*	*IO*	*IO*
N	100	500	1000	5000	100	500	1000	5000
$\bar{C}$	0.84	0.95	0.92	0.90	0.95	1.11	1.01	0.99
$\lvert\lvert D \rvert\rvert$	1500	1162	1016	783	1500	550	705	595
$\lvert\lvert D_m \rvert\rvert$	642	819	787	669	140	132	229	215
R^2 Score	0.81	0.87	0.77	0.82	0.67	0.34	0.63	0.59
$\overline{C_l/C^*}$	1.21 ±0.01	1.24 ±0.01	1.31 ±0.02	1.42 ±0.03	1.22 ±0.03	1.73 ±0.16	1.75 ±0.09	2.41 ±0.09
$\overline{C_b/C^*}$	1.40 ±0.02	1.32 ±0.01	1.38 ±0.02	1.51 ±0.03	1.24 ±0.02	1.51 ±0.06	1.69 ±0.07	2.35 ±0.08

Table 3.1: A comparison of plan costs between the learned sampling distribution and baseline planners when both planners succeed, broken out by dataset and planner iterations per query (N), which is correlated to the number of samples drawn per step. $||D||$ and $||D_m||$ are the total number of trials run and the number of trials where both planners succeeded. $||D||$ varies over N due to the time involved in running longer trials. For each mutually successful trial, given the length of a resolution-optimal trajectory (assuming an *a priori* map, with soft costs) C^*, the total distance travelled by the learned planner C_l, and the total distance travelled by the baseline planner C_b, we quantify performance via several metrics. $\bar{C}$ and R^2 score are the slope and fit score of a linear regression (i.e., of C_b vs C_l) with zero intercept. $\overline{C_l/C^*}$ and $\overline{C_b/C^*}$ are the mean and standard error of the mean of the trajectory cost divided by the resolution-optimal trajectory cost for learned and baseline respectively. In the *Inside-Inside* dataset, the learned sampling distribution finds lower cost plans than the baseline, indicating that using the learned sampling distribution with our chosen SBMP results in better navigation outcomes.

3.3.1 Experimental Setup

We first demonstrated our approach in simulation using real-world floorplan data from MIT [182], which included floorplans from 13 buildings, split into 10 train buildings and 3 test buildings. The data were used to generate discretized occupancy and semantic maps, where door locations were provided by the dataset; plausible windows and exit signs were manually annotated. We assumed a holonomic vehicle with a planar depth sensor similar to an Intel RealSense sensor with an 85.2 degree field of view and 5 meter range and a RGB vision system with a 69.4 degree field of view capable of observing objects from 10 meters away, with no sensor or pose estimate noise.

3.3.2 Dataset Generation

We generated 1500 navigation tasks in known maps, categorized by indoor or outdoor start and goal locations, omitting outdoor only tasks due to the lack of semantic information outdoors in our environments. We simulated a robot completing each navigation task using a modified open source implementation of the PRM [166] and a Euclidean objective function without *a priori* map information.

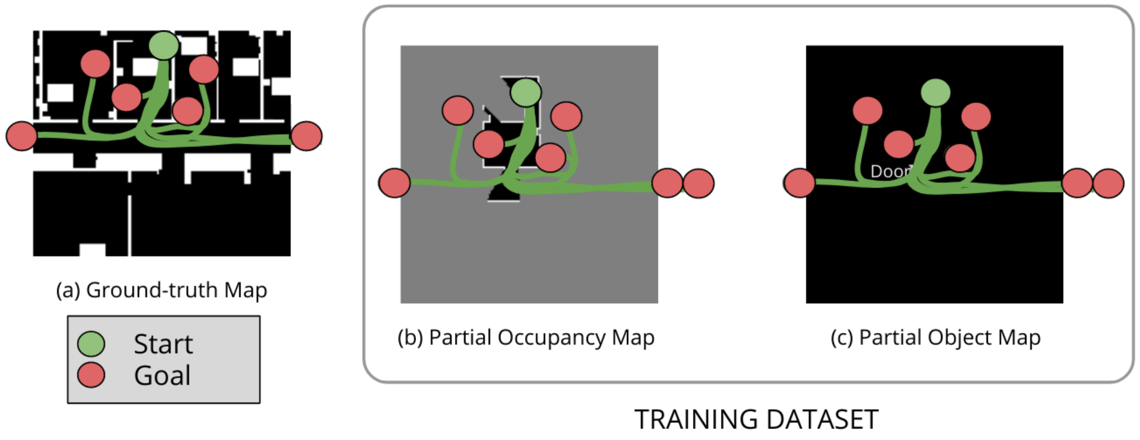


Figure 3-6: Example illustration of dataset generation. The robot navigates an unknown environment. (a) At regular intervals, several goals are sampled and optimal trajectories computed the using ground-truth map. The trajectories and goals are then paired with the (b) partial occupancy map and (c) partial object map at the time of the query. The procedure allows for the generation of a training dataset that pairs incomplete partial maps with expert trajectories. Paths are hand-drawn for illustration purposes only.

At each timestep, the robot generated and executed a plan, uncovering geometric and semantic information. We assumed that both maps were well approximated by raycasting in known maps according to the sensor characteristics. To generate diverse expert trajectories, at random intervals during the task we randomly selected a new goal location and solved for a dense, resolution optimal path to the goal location using a graph-based search over the ground-truth occupancy map with soft costs. In practice, we found it important to simulate the types of maps generated by sub-optimal navigation, as the robot may make sub-optimal decisions at planning time. Figure 3-6 provides a high-level illustration. In practice, expert trajectories may be generated in many ways. We have assumed that these paths approximate optimal sub-graphs and can be represented as a set of edges and vertices. We then extracted body-centered local maps of 160x160 pixels (i.e., M_g, M_s) and labelled them with the points in the example trajectory. This dataset was then used to train the model in Equation 3.8. To examine the utility of our approach under different contexts, we created two evaluation datasets, the first where the robot begins indoors, and is told that the goal is also inside (*Inside-Inside*, or *II*), and another where the robot begins indoors, and is informed that the goal is outdoors (*Inside-Outside*, or *IO*). An evaluation dataset where the robot begins outdoors is not included, as our floorplans lack outdoor semantic information.

3.3.3 Simulation Evaluation Results

To evaluate our approach, we simulated a robot completing online navigation tasks in unknown environments using a modified version of the PRM [166] with our learned sampling distribution and cost function. We compared the PRM using our learned sampling distribution (*PRM + LSD*) to a baseline approach using a uniform sampling distribution (*PRM + Unf*) on a test set composed of random start and goal locations over three unseen floorplans. To estimate the cost in Equation 3.3, each edge was weighted using an approximate integral cost. To simulate different types of initial partial maps, initial yaws were uniformly sampled at the beginning of each trial; the initial yaws varied between experiments but were consistent between the two approaches for each single trial. *PRM + LSD* used an integral approximation of the negative log probability of the learned sampling distribution along each edge as a search objective. *PRM + Unf* used a Euclidean search objective. We varied a value roughly proportional to the number of samples used to build the graph the respective planners were allowed to draw to generate a roadmap, and simulated trials for start and goal locations for which feasible trajectories existed. We allowed the simulated robot to follow a trajectory for up to 20 planning timesteps, with re-planning triggered if occupancy information is acquired indicating a collision. Given that both planners rely on a stochastic planner, under sample constraints it is possible for both planners to fail to build a graph that is sufficiently useful to find a feasible trajectory. If no trajectory was found at a given timestep, or the simulation of a trial exceeded 10,000 timesteps, the entire trial was marked as a failure.

We generate several figures of qualitative results to demonstrate our algorithm per-

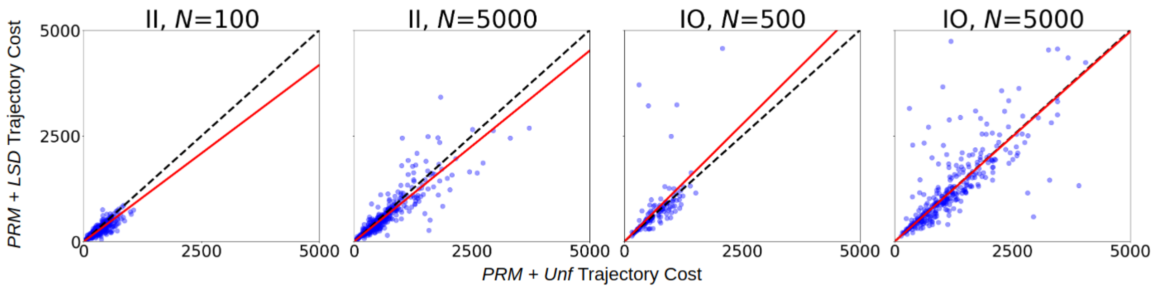


Figure 3-7: Selected scatter plots of distance travelled in simulation units for *PRM + LSD* vs. *PRM + Unf* (blue circles) and the slope calculated by linear regression (red line) for various test conditions, where N is as in Table 3.1. The dotted line is plotted as a reference for equal cost.

formance. Figure 3-5 shows a qualitative result for the distribution trained without context, as well as the distribution after multiplication with various context-specific modifiers. The results demonstrate the utility of adding the semantic context of the goal. In the example shown, whether the exit sign is an area of high probability is related to goal being located inside or outside. In Figure 3-9, we show several timesteps in a single trial, where the baseline planner greedily explores many rooms, while the learned planner places probability in the hallways and exit doors and navigates directly to the goal. Figures 3-10 and 3-11 show example final trajectories of both approaches in the II and IO environments respectively. In cases such as 3-11-(b), a suboptimal learned distribution causes *PRM + LSD* to suffer from failure modes similar to those of *PRM + Unf*, which leads to inefficient trajectories.

Our simulation results indicate that our approach to combining geometry and extra-geometric representations into a heuristic prior for sampling helps to focus both computation and navigation to promising areas even when geometric information is incomplete, resulting in greater sample efficiency and in some test sets, more efficient navigation. For similar sample counts, *PRM + LSD* was more likely to find feasible plans than *PRM + Unf*, as shown in Figure 3-8. For example, for the $N = 500$ case, *PRM + Unf* had a success rate of 25%, while *PRM + LSD* had a success rate of 69%. The largest gains in sample efficiency were observed in the *IO* test set. However, although *PRM + LSD* enabled more successful planning in both scenarios, the maximum percent of successful trials for the *IO* test set was significantly lower

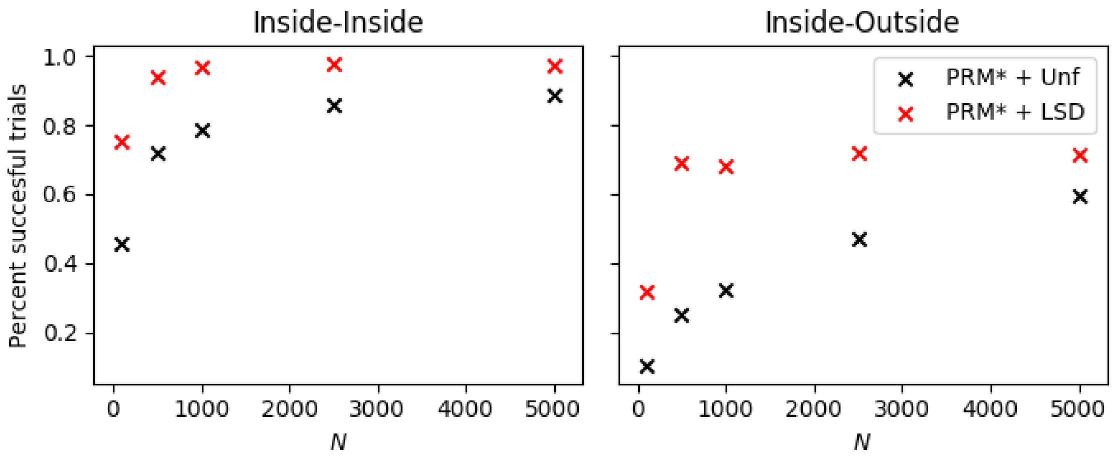


Figure 3-8: A comparison of plan success rates between the learned (*PRM + LSD*, shown in red) and baseline (*PRM + Unf*, shown in black) planners, where N is as in Table 3.1. The learned sampling distribution finds plans more frequently than the baseline, demonstrating that our learned sampling measure empowers PRM to find plans more quickly. We observe that the limitations placed on the timesteps per trial may inhibit the convergence of the harder *IO* test set. A small percentage ($< 2\%$) of trials were marked as failures due to the vehicle coming into collision with the environment; these trials were removed when calculating success statistics.

than for the *II* test set. The differences in performance may be because goals outside the buildings are biased to be further away than goals inside the house and thus involve navigating over longer length-scales. It is worth observing that the feed-forward distribution prediction step of the neural network poses additional computational overhead as compared to *PRM + Unf*, but in general, the minimum number of samples required to be likely to find a plan increases as the environment size increases, inducing a trade-off between sample efficiency and network complexity. Additionally, our real-world experiments indicate that even without a GPU our approach is able to run online.

We also demonstrated that our approach enables efficient navigation in certain domains. A comparison of plan costs is shown in Table 3.1; Figure 3-7 includes scatter plots of trajectory costs for a selection of scenarios. For the *II* test set, we demonstrated between a 5% and 16% decrease in plan cost (determined by linear regression) when using the learned sampling distribution. In the *IO* test set, the overlap of the standard error of the mean (SEM) estimates in the bottom two rows of

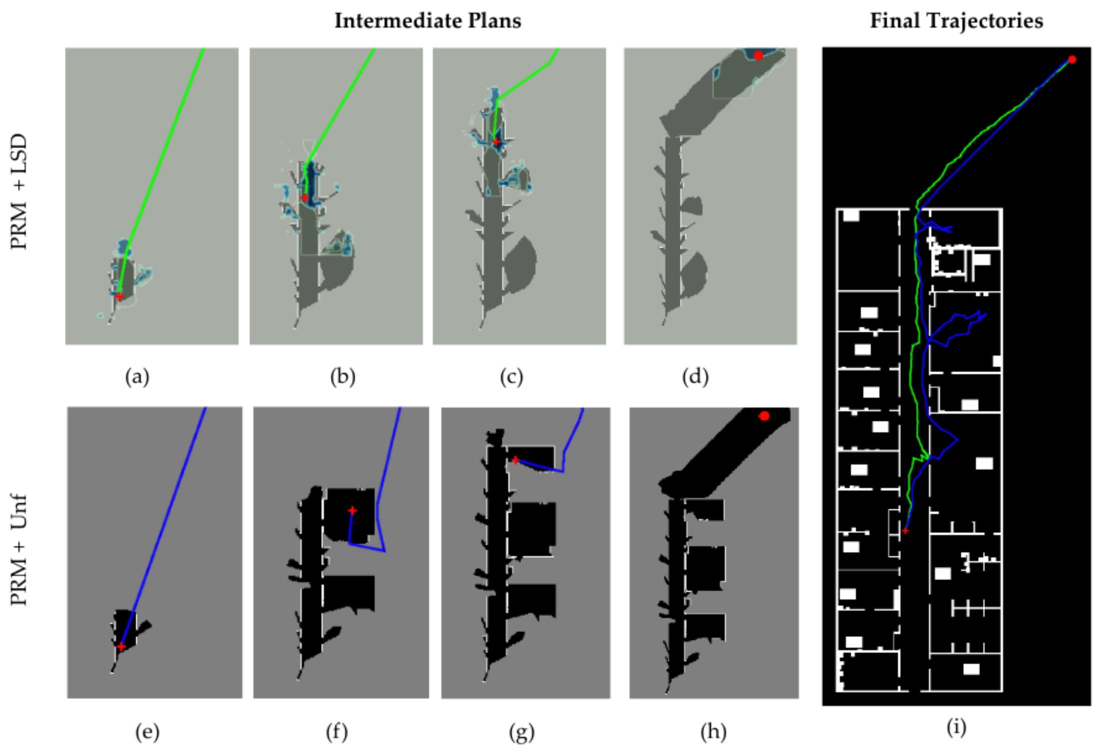


Figure 3-9: Example intermediate and final trajectories of the baseline and our method. In the left image, we show the sampling distribution overlaid on the robot's most recent occupancy map (darker blue is higher probability). The goal is set to the top right corner of the map, and the learned sampling distribution is also given the context that the goal is outdoors. Unlike the baseline (e)-(h), which uses only a Euclidean distance metric and greedily explores more rooms in the hopes of reaching the goal, the learned distribution largely encourages the planner to follow the hallway (a)-(d). Without the contextual information implicit in the learned sampling distribution, the baseline travels a much longer distance to reach the goal (i).

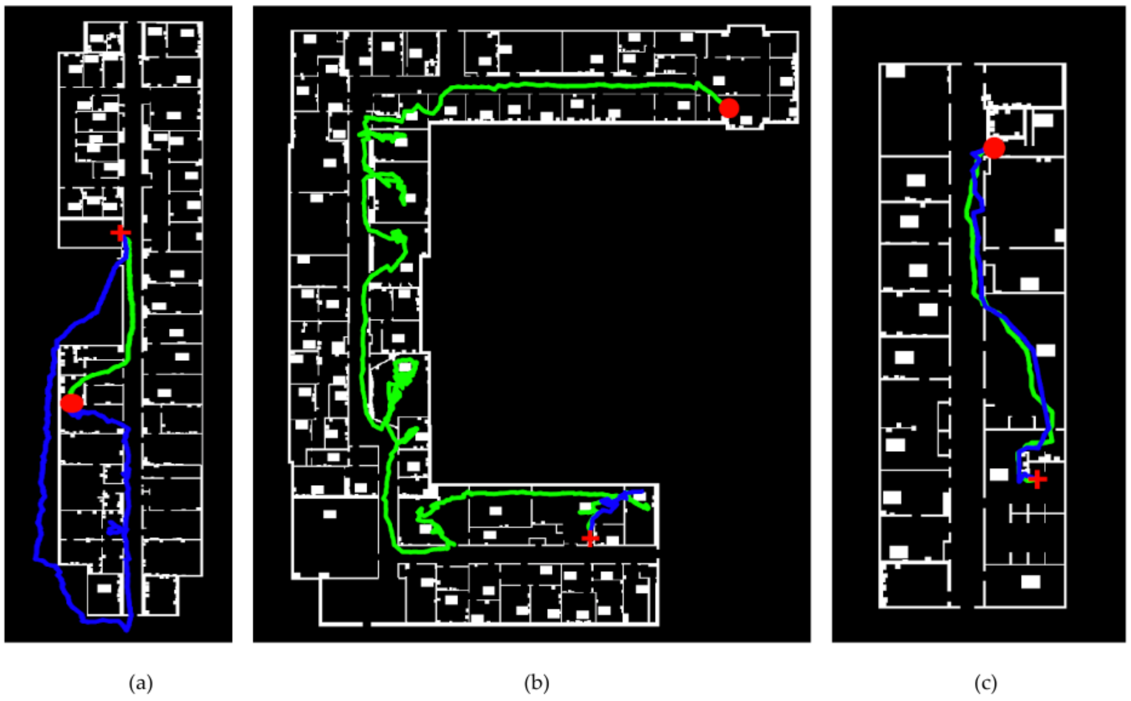


Figure 3-10: Comparative examples of planning performance, where the robot begins inside and is given the contextual information that the goal is also indoors (II). We show the final trajectories of the learned (green) and baseline (blue) planners traversing from start (red cross) to goal (red circle). In (a), the baseline planner mistakenly exits the building although the goal is indoors, but the learned planner largely remains in hallways and reaches the goal. In (b), the baseline planner fails to find a trajectory with the allotted sample count, while the learned planner mistakenly visits many rooms. Such a case is not included in the aggregate metrics as the baseline planner failed. In (c), both the learned and baseline planners mistakenly enter rooms prior to reaching the goal.

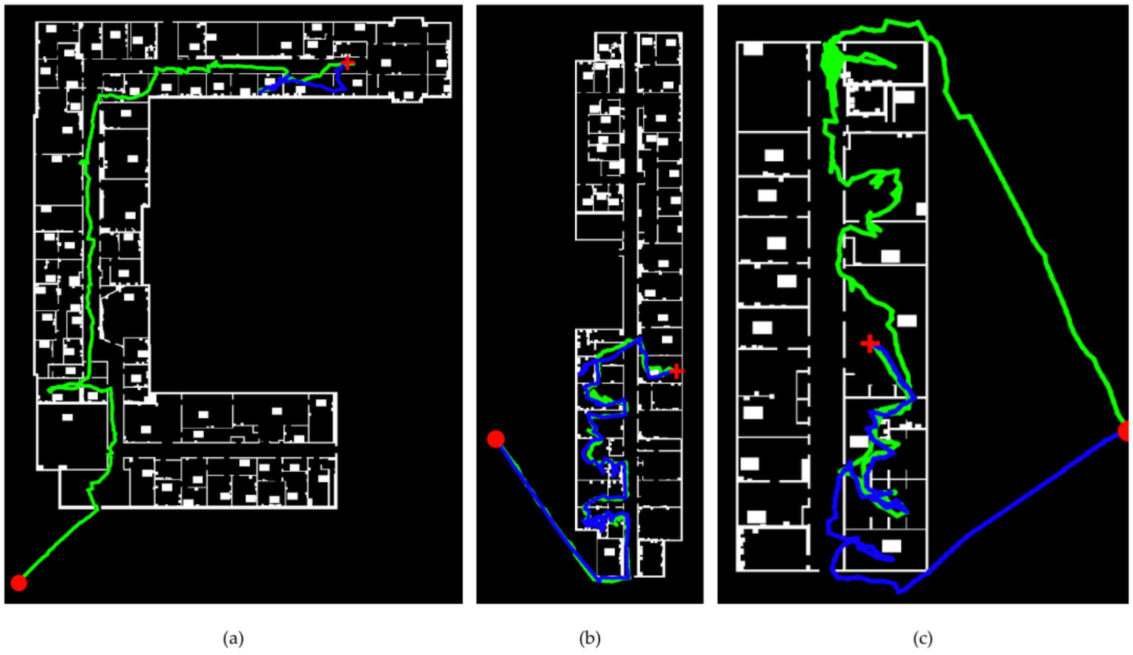


Figure 3-11: Comparative examples of planning performance for Inside-Outside trials (IO), where the robot begins inside and is given the contextual information that the goal is outdoors. We show the final trajectories of the learned (green) and baseline (blue) planners traversing from start (red cross) to goal (red circle). In (a), the baseline planner fails to find a plan somewhat early in the trial, while the learned planner successfully navigates the robot out the building. In (b), both the baseline and the learned planners take inefficient routes but successfully reach the outdoor goal. In (c), the baseline planner finds a shorter path to the goal than the learned approach. We observe that while trial (c) is included in the aggregate metrics and is in favor of the baseline planner, trial (a) is used only to calculate success rates.

Table 3.1 suggests no statistically significant differences in plan cost between the two approaches, although the *IO-500* results are nearly overlapping. In general, due to the large differences between the success rates between *PRM + LSD* and *PRM + Unf* in the *IO* domain, the aggregate cost metrics are likely less meaningful in the *IO* trials. For example, in Figure 3-11-(a) we show an experimental trial where the learned planner finds a near optimal plan and the baseline fails. Although this trial clearly demonstrates the ability of the learned planner to find low cost trajectories, it is not a mutually successful trial and therefore is not included in the aggregate statistics. Overall, we demonstrate that our learned sampling distribution significantly improves the success rate of PRM, and decreases plan costs over the *II* test set.

3.3.4 Real-World Evaluation Results

To evaluate the performance of a SBMP paired with the learned sampling distribution on real-world data, we integrated our approach on a real world robot. We qualitatively compare the planning performance of *PRM + LSD* and *PRM + Unf* on the same dataset. The dataset was collected by a 1/10th scale racecar platform, carrying the Intel RealSense T265 and D435i modules (which provide state estimates and RGB-D images, respectively) and an Intel Nuc i7. Dense geometric maps were generated using a standard geometric estimation and mapping stack [65]. An object detector based on SSD-Mobilenet [68, 104, 3] was fine-tuned to detect doors and windows with data from the OpenImages dataset [90], and an exit sign detector was written using HSV filtering. Discrete 2D object-level maps were generated by a rough conversion [4] of the output of an object mapping system [120] described in Chapter 4.

Our experiments show that our proposed neural network architecture is lightweight enough to run on a CPU to enable real-time navigation on SwAP vehicles. We observe that unlike in the simulation environment, the object-level maps built online are noisier and less accurate. We first compared *PRM + LSD* and *PRM + Unf* on the same input data to compare planning performance with identical inputs. Figure

[4]The rough conversion sometimes led to inconsistencies in the 2D estimates of objects, especially in orientation. We describe a more accurate conversion via orthographic projection in Chapter 5.

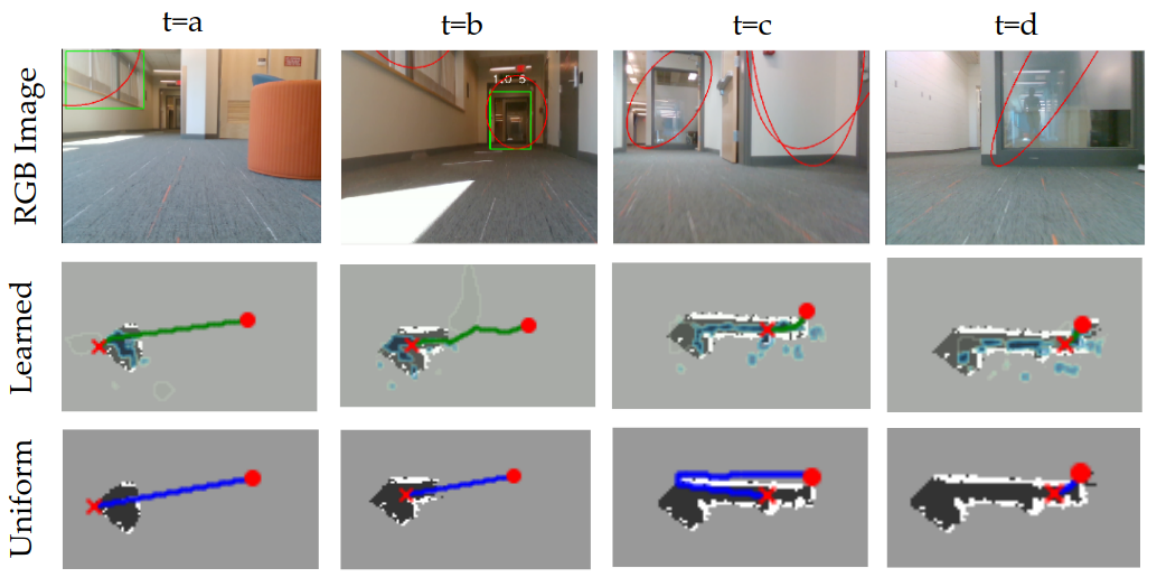


Figure 3-12: Sensor data and plans at similar points on the same real-world dataset. RGB images show the object detections (green) and the estimated volume of objects projected into the image plane (red). *PRM + LSD* generally plans to go down the middle of the hallway, while *PRM + Unf* greedily attempts to go through an unseen wall. In this case, the learned distribution empowers the PRM to plan outside of the known geometric map by generally guiding the agent down the middle of the hallway, despite the noisy occupancy and object-level maps. All images are approximately aligned

3-12 shows qualitative examples of trajectories and RGB images at various points during navigation, comparing the planning performance *PRM + LSD* and *PRM + Unf*. The learned method was more likely to plan trajectories down the middle of the hallway, rather than attempting to greedily plan through the wall. We then tested our approach for in the loop planning. Although the network was only ever trained in environments with open doors, the when in-the-loop planning test environment featured a closed door early in the trajectory. Because our proposed approach synthesizes the extra-geometric information into a sampling distribution with non-zero probability at all locations, *PRM + LSD* is still able to plan a collision-free trajectory to the goal given sufficient samples, as seen in Figure 3-13.

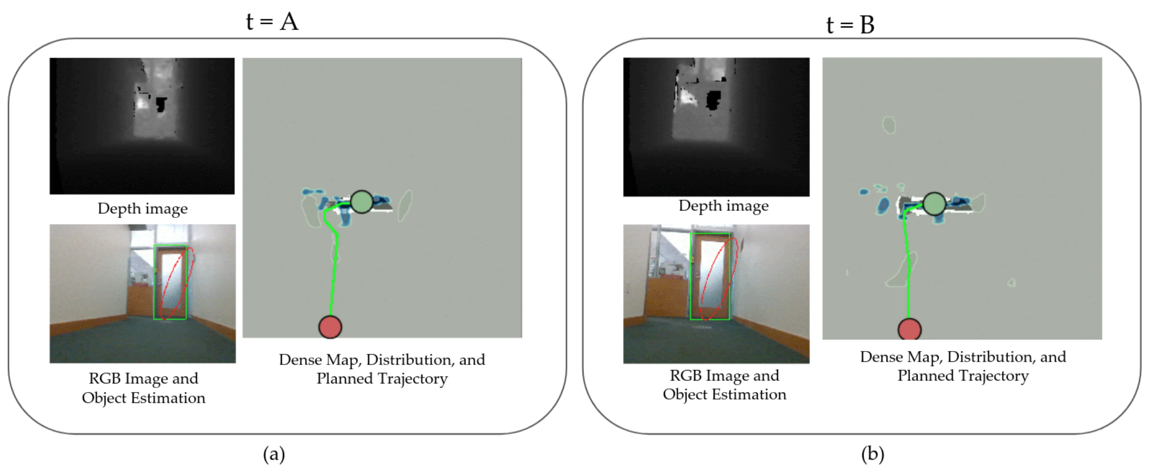

t = A
Depth image
RGB Image and Object Estimation
Dense Map, Distribution, and Planned Trajectory
(a)
t = B
Depth image
RGB Image and Object Estimation
Dense Map, Distribution, and Planned Trajectory
(b)

Figure 3-13: Qualitative demonstration of faulty semantic information. Sensor images as well as planning artifacts are shown for two timesteps in real-world, in the loop planning. Before the closed door has been observed in the dense occupancy map, high probability regions are present near the detected door. However, as further occupancy information is collected, planner adjusts to the hard constraint of non-collision, despite the fact that the learned distribution has never seen a closed door.

3.4 Limitations

As shown in Table 3.1 and Figure 3-8, we observed significant differences between the performance of our approach on the trials where the robot's goal was inside versus outside. In particular, the learned planner proved to be significantly more sample efficient compared to the baseline planner (i.e., with a greater margin of improvement in the IO trials vs the IO trials), but both the baseline and our planner were less sample efficient in the Inside-Outside trials than the Inside-Inside trials. Additionally, for the Inside-Outside trials, our planner performed about as well as the baseline planner. This suggests that while our learned sampling distribution has captured useful properties for finding *feasible* plans, the task of choosing near-optimal trajectories to navigate from inside the building to the outside is a harder task than navigating from one point inside a building to another point inside a building and that greater structure in planning may be needed. For instance, it is key that the planner exit a door with an exit sign. However, if an exit sign has not yet been observed, our proposed strategy does not explicitly use an exploration phase to search for such a signal.

There is also evidence that the use of the learned sampling distribution as a cost function may require additional structure. For example, we occasionally observed waffling behavior as seen in Figure 3-10c, where the robot backtracked several times, indicating that a history of trajectories may be helpful. Additionally, later analysis presented by Stadler [163] suggests that using a Euclidean search metric with the graph built using our learned sampling distribution did not significantly degrade performance. However, using the learned sampling distribution to weight graph edges results in a clear reduction in the frequency of necessary re-planning. Essentially, using the cost function to weight graph edges resulted in trajectories that could be followed for longer without being rendered invalid, indicating that the cost function is more useful locally. We suggest potential routes for improvements in Chapter 6.

3.5 Chapter Summary

We have presented a novel method for extending sampling-based motion planners into unknown environments by learning a sampling distribution. We demonstrated that our method results in significantly higher success rates compared to a uniform sampling strategy and can lower traversal costs in some domains. We have shown that learning a sampling distribution using object-level semantic information and geometric maps can enable long-horizon navigation in unknown environments, outperforming baseline, uniform sampling strategies.

Notably, our approach uses geometric and semantic information to learn heuristics to bias the both the construction and search of a planning graph. The distribution is specifically formulated so that there is strictly positive probability at each discretized location, and edges are still pruned using geometric evidence. Even if a state is incorrectly predicted to have high likelihood of lying on the optimal trajectory, the sampled vertex will be rejected if the location is occupied. Therefore, as the limit of complete geometric information is approached, the resulting trajectories will comply with the observed detailed geometric information. When geometric information is sparse, the learned distribution can be leveraged to apply structure to the problem to mitigate geometric ambiguity.

However, our proposed approach relies on the ability to build metric and semantic object-level maps online. In particular, we require maps of objects with class labels as well as geometric object properties that can be projected into a representation similar to occupancy maps. For our simulation experiments, we approximate the construction of such maps using raycasting techniques, but for online deployment as in Section 3.3.4 we require a lightweight online process. The next two chapters of this thesis are therefore devoted to improving object-level estimation to support hybrid semantic and geometric planning.

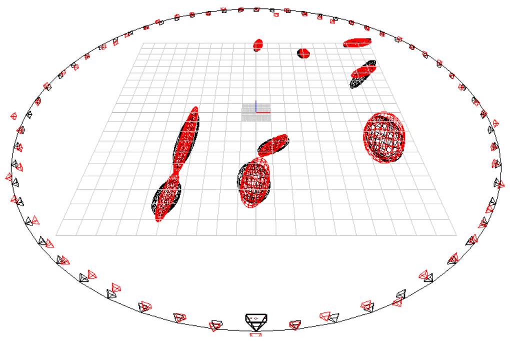
Chapter 4

Object Level-SLAM for Autonomous Navigation

4.1 Introduction

In this chapter, we improve object-level SLAM for the purposes of autonomous navigation. Although some state of the art methods have assumed the existence of tightly coupled hierarchical semantic representations (Section 2.1.4) to inform navigation, in the previous chapter we presented an approach that requires only light-weight object representations in world coordinates. As discussed in Chapter 2, approximate geometric models are a promising compromise between requiring detailed *a priori* models and overly simplistic point-based object representations (Section 2.2.1). Object-level SLAM approaches that estimate approximations of objects generally do so by fitting bounding box measurements from an object detector to generic low-dimensional parametric models, ranging from cuboids to ellipsoids (Section 2.2.3.). The recently popularized ellipsoid model is particularly attractive due to a differentiable relationship between the object estimate and bounding box measurement.

A key challenge in estimating geometric object properties online to support autonomous navigation is that common types of vehicle maneuvers such as straight line motions may not generate viewpoints diverse enough to accurately estimate the objects. Therefore, to support our planning approach we specifically require an object


(a) Orbit Path

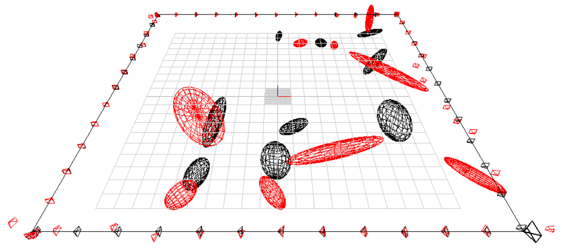

(b) Forward Path

Figure 4-1: Comparison of final ellipsoid estimates generated by an orbiting camera path vs. a forward path. Ground-truth estimates (black) and final estimates (red) of ellipsoids and cameras inferred with an online approach using a bounding box measurement model, similar to the model proposed by Nicholson et al. [118]. As seen in (a), estimating the parameters of the ellipsoids using diverse viewpoints of an orbiting vehicle path resulted in a small error in shape and position for both methods. However, when using measurements from a forward-moving vehicle path, where only limiting views were available as exemplified in (b), estimation performance severely degrades. Details of both the experiment and simulation environment are provided in Section 4.4.

level mapping approach that can estimate objects under challenging camera motions. Previous approaches [146, 118] have shown promising results in building lightweight approximations of objects offline by fitting bounding box measurements. One benefit of offline batch optimization is that the algorithm can utilize all views to estimate the ellipsoid. However, early estimates in online versions of this approach do not have access to all views of the object that will be collected over time. Additionally, under the constraints of realistic autonomous navigation, the robot may never collect a large set of diverse views. As shown in Figure 4-1, given diverse views generated by an orbiting camera path, an ellipsoid-based SLAM system that consumes only bounding box measurements can perform quite reasonably under certain conditions of the trajectory. However, when the camera trajectory is instead composed of straight-line, forward facing trajectories which are more common in efficient autonomous navigation, estimation performance degrades.

Our key observation is that RGB images contain additional information useful for 3D object estimation, beyond just 2D bounding box measurements. For instance, object texture can be used to triangulate points on the surface of the object. Furthermore, the object class as given by the bounding box detector provides important semantic context for the optimization. In order to better constrain an object-based SLAM system that lacks diversity in viewpoints, we show how to use these two additional sources of information: texture on objects that can be used to infer the distance to the objects and semantic knowledge of shapes of objects that can mitigate the scale unobservability problem in monocular cameras. While similar to recent work [67] which uses surface normals from RGB-D cameras to further constrain quadrics, we focus on adding only the information available in a monocular camera.

We propose robust object-based SLAM for high-speed autonomous navigation (ROSHAN), where we represent semantically-meaningful objects *volumetrically* as ellipsoids, and infer the parameters of the ellipsoids online using three sources of information: bounding box detections, texture, and semantic shape. We introduce our factor-graph based SLAM [34] formulation, that contrary to modern offline methods [146, 118] do not assume known data associations or batch optimization. We additionally propose a single measurement initialization scheme that can be useful on a fast-moving vehicle. Finally, we demonstrate the advantages of ROSHAN in simulation using 50 randomly generated maps of ellipsoids, where we outperform the baseline, reducing the median error on the shape estimates by 83% and the median error on the position estimates by 72% when compared to the baseline in a forward-moving camera sequence. In addition, we present promising results running ROSHAN real-time on simulated and real autonomous high-speed flight sequences.

4.2 Problem Overview

In ROSHAN, we represent objects as ellipsoids and infer their parameters using object detections, object texture, and semantic knowledge in a SLAM framework. In this section, we first discuss the strengths of our landmark representation then formulate our object-based SLAM problem.

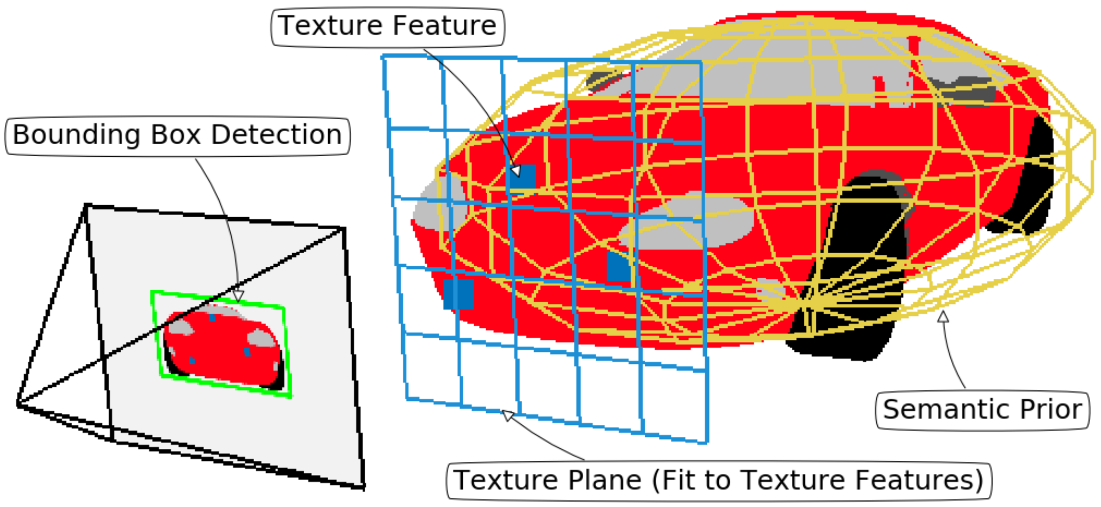


Figure 4-2: In ROSHAN, we combine bounding box detections (green), texture planes (blue), and semantic knowledge of the shape of objects (yellow) to achieve an ellipsoid-based object SLAM system robust to undesirable camera motions.

4.2.1 Ellipsoids as Object Representation

We choose the ellipsoid representation, a specific form of quadric representation as the low-dimensional parametric form of our objects. Similar to Rubino et al. [146], we minimally parametrize the ellipsoid with 9 independent parameters that represent the orientation $\boldsymbol{R} \in \mathrm{SO}(3)$, position $\boldsymbol{t} \in \mathbb{R}^3$, and shape $\boldsymbol{d} \in \mathbb{R}^3$ of the ellipsoid. While there are two forms of ellipsoids, the primal form $\boldsymbol{Q}$ and the dual-form $\boldsymbol{Q}^* = \operatorname{adjoint}(\boldsymbol{Q})$, in this chapter we are primarily interested in the dual-form. As discussed by Hartley and Zisserman [60], the dual matrix $\boldsymbol{Q}^*$ can be transformed by some transformation matrix $\boldsymbol{Z}$ via the relationship

$$\boldsymbol{Q}^* = \boldsymbol{Z}\breve{\boldsymbol{Q}}^*\boldsymbol{Z}^T. \tag{4.1}$$

We take $\boldsymbol{Z}$ to be a homogeneous transformation matrix and as in Rubino et al. [146] we define $\breve{\boldsymbol{Q}}^*$ as a scaled dual ellipsoid at the origin:

$$\boldsymbol{Z} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0_3}^T & 1 \end{bmatrix}, \breve{\boldsymbol{Q}}^* = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{0_3} \\ \boldsymbol{0_3}^T & -1 \end{bmatrix}, \tag{4.2}$$

and where $\boldsymbol{D} \in \mathbb{R}^{3\times3}$ is a positive diagonal shape matrix with the diagonal entries formed with regularized squared shape parameters. In this chapter, we define $\boldsymbol{D}_{i,i} = \boldsymbol{d}_i^2 + \gamma$, where $\gamma \in \mathbb{R}$ is a regularization constant enforcing a minimum shape. We constrain our representation under the parameterizations of Equations 4.1 and 4.2, yielding $\boldsymbol{Q}^* \in \mathbb{E}^{4\times4}$, where $\mathbb{E}^{4\times4}$ represents the subset of all 4×4 symmetric matrices defined by

$$\boldsymbol{Q}^* = \begin{bmatrix} \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T - \boldsymbol{t}\boldsymbol{t}^T & -\boldsymbol{t} \\ -\boldsymbol{t}^T & -1 \end{bmatrix}. \tag{4.3}$$

While an ellipsoid is only a rough approximation of an object in 3D, a strong advantage of the ellipsoid representation is that its entire parametrization can be constrained using only bounding box measurements from camera images. This property of ellipsoids comes from the dual-form where all homogeneous planes $\boldsymbol{\pi}_k \in \mathbb{R}^4$ tangent to the dual-form of an ellipsoid ${\boldsymbol{Q}^*}_j$ must obey

$$\boldsymbol{\pi}_k^T {\boldsymbol{Q}^*}_j \boldsymbol{\pi}_k = 0. \tag{4.4}$$

This system of equations, when solved as a function of the vehicle pose $\boldsymbol{x}_{t_k} \in \mathrm{SE}(3)$ and the observed ellipsoid ${\boldsymbol{Q}^*}_{j_k}$ as illustrated in section 4.3.1, forms a closed-form differentiable bounding box measurement model

$$\hat{\boldsymbol{B}}_k = h_{bb}({\boldsymbol{Q}^*}_{j_k}, \boldsymbol{x}_{t_k}; \boldsymbol{K}), \tag{4.5}$$

where $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ in the camera intrinsic matrix and $\hat{\boldsymbol{B}}_k \in \mathbb{R}^4$ is the predicted bounding box measurement. While the family of quadrics all share the same smooth measurement model, we specifically limit our landmarks to ellipsoids to further constrain the landmarks without losing the ability to approximate objects for the purpose of collision-checking.

Coupled with computationally inexpensive object detectors [133, 68], the above closed-form measurement model allows for the use of readily available bounding box detections as the only source of measurements to fully constrain vehicle poses and approximate object volumes. This property makes the ellipsoid representation attractive for graph-based SLAM [34] formulations, and is similar to the property of point-based landmarks in feature-based SLAM [119] that associated feature detections in camera images can be the only source of information to constrain the entire system.

4.2.2 SLAM Formulation

We would like to solve for all ellipsoidal approximations of objects $\boldsymbol{\mathcal{Q}} = \{\boldsymbol{Q^*}_j\}_{j=0}^{J}$ with J objects of interest, and T poses of the vehicle $\boldsymbol{\mathcal{X}} = \{\boldsymbol{x}_t\}_{t=0}^{T}$, where $\boldsymbol{x}_t \in \mathrm{SE}(3)$. We are given T images $\boldsymbol{\mathcal{I}} = \{\boldsymbol{I}_t\}_{t=0}^{T}$ with $\boldsymbol{I}_t : \Omega \in \mathbb{N}^2 \to \mathbb{R}$, where Ω is the image pixel domain. Using an object detector, we extract K bounding box measurements of objects $\boldsymbol{\mathcal{B}} = \{\boldsymbol{B}_k \in \Omega^2\}_{k=0}^{K}$ along with the semantic class labels $\boldsymbol{\mathcal{C}} = \{c_k \in \mathbb{N}\}_{k=0}^{K}$, where each bounding box is parametrized by two pixel locations representing the opposite corners of the bounding box. We extract features [147] from the texture of the objects in images, and fit a homogeneous plane $\boldsymbol{\pi}_d^t \in \mathbb{R}^4$ to the triangulated locations of the features of each object; these D planes $\boldsymbol{\Pi}^t = \{\boldsymbol{\pi}_d^t\}_{d=0}^{D}$ that we call *texture planes*, e.g. the blue plane in Fig. 5-1, represent measurements of the distance between the cameras and the camera-facing sides of objects. Assuming a uniform prior on the measurements and independence assumptions between all measurements, we write our object-level SLAM problem as

$$\begin{aligned} P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Q}} | \boldsymbol{\mathcal{B}}, \boldsymbol{\Pi}^t, \boldsymbol{\mathcal{I}}; \boldsymbol{\mathcal{C}}) \propto & \underbrace{\prod_{k=0}^{K} P(\boldsymbol{B}_k | \boldsymbol{Q^*}_{j_k}, \boldsymbol{x}_{t_k})}_{\text{Bounding Box (4.3.2)}} \times \\ & \underbrace{\prod_{d=0}^{D} P(\boldsymbol{\pi}_d^t | \boldsymbol{Q^*}_{j_d}, \boldsymbol{x}_{t_d})}_{\text{Texture (4.3.3)}} \underbrace{\prod_{j=0}^{J} P(\boldsymbol{Q^*}_j; c_j)}_{\text{Semantic Prior (4.3.4)}} \underbrace{\prod_{t=0}^{T} P(\boldsymbol{x}_t | \boldsymbol{I}_{0:t})}_{\text{Pose Prior}}, \end{aligned} \tag{4.6}$$

where we assume that the data association problem has been pre-solved (implementation details discussed in section 4.5.1), i.e., that the associated indices j_k and j_d for objects and t_k and t_d for poses are known for each of the measurements $\boldsymbol{B}_k$ and $\boldsymbol{\pi}_d^t$, and that the class labels c_j for ellipsoids are deduced from labels c_k of bounding boxes.

We can then obtain optimal estimates of vehicle poses $\boldsymbol{\mathcal{X}}^*$ and objects $\boldsymbol{\mathcal{Q}}^*$ by maximizing the posterior probability

$$\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{Q}}^* = \underset{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Q}}}{\arg\max}\, P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Q}} | \boldsymbol{\mathcal{B}}, \boldsymbol{\Pi}^t, \boldsymbol{\mathcal{I}}; \boldsymbol{\mathcal{C}}). \tag{4.7}$$

In the following sections, we discuss the details of the bounding box measurement model (4.3.2), texture plane measurement model (4.3.3) and the semantic prior on the ellipsoids (4.3.4) to demonstrate how multiple sources of information can be combined to constrain ellipsoidal approximations of objects. However, in this work we assume an external vision-based[1] localization system f_{pose} [165] produces pose estimates $\boldsymbol{x}_t = f_{pose}(\boldsymbol{I}_{0:t})$ to be consumed by our system and incorporate the MAP estimates along with a heuristic covariance as priors on our vehicle poses. In the next section, we describe a limitation in the state-of-the-art bounding box measurement model [118], and suggest an improved bounding box measurement model.

4.3 Robust Object-based SLAM for High-speed Autonomous Navigation

In ROSHAN, we combine bounding box measurements, texture plane measurements, and semantic shape priors in an online optimization framework to realize an object-level SLAM system that is more robust under undesirable vehicle motions. Before introducing the two additional sources of information, texture and semantic knowledge, we first revisit the state-of-the-art bounding box measurement model [118].

4.3.1 Geometric Bounding Box Measurement Model

The projection of a dual-form of a quadric on a camera plane is called a dual-conic $\boldsymbol{C}^* \in \mathbb{R}^{3\times3}$, and has a similar property that all tangent lines must obey

$$\boldsymbol{l}_h^T \boldsymbol{C}^* \boldsymbol{l}_h = 0, \tag{4.8}$$

where $\boldsymbol{l}_h \in \mathbb{R}^3$ is a homogeneous form of a line. Since a dual-form of a quadric can be projected to a dual-conic [118] by

$$\boldsymbol{C}^* = \boldsymbol{K}[\boldsymbol{R}_t|\boldsymbol{t}_t]\boldsymbol{Q^*}_j[\boldsymbol{R}_t|\boldsymbol{t}_t]^T\boldsymbol{K}^T, \tag{4.9}$$

where $\boldsymbol{K} \in \mathbb{R}^{3\times 3}$ is the camera intrinsic matrix, $\boldsymbol{R}_t \in \mathrm{SO}(3)$ is the rotation, and $\boldsymbol{t}_t \in \mathbb{R}^3$ is the translational portion of the pose $\boldsymbol{x}_t \in \mathrm{SE}(3)$, we can solve Eq. 4.8 in closed-form for the bounding box edges $\boldsymbol{l}_u = [1, 0, -u]$ and $\boldsymbol{l}_v = [0, 1, -v]$, i.e.,

$$\begin{aligned}\hat{u}_{min}, \hat{u}_{max} &= \frac{1}{\boldsymbol{C}^*_{3,3}}[\boldsymbol{C}^*_{1,3} \pm \sqrt{{\boldsymbol{C}^*_{1,3}}^2 - \boldsymbol{C}^*_{1,1}\boldsymbol{C}^*_{3,3}}],\\ \hat{v}_{min}, \hat{v}_{max} &= \frac{1}{\boldsymbol{C}^*_{3,3}}[\boldsymbol{C}^*_{2,3} \pm \sqrt{{\boldsymbol{C}^*_{2,3}}^2 - \boldsymbol{C}^*_{2,2}\boldsymbol{C}^*_{3,3}}],\end{aligned} \tag{4.10}$$

to form the closed-form measurement model in Eq. 4.5, where the predicted bounding box is a collection of these edge locations, i.e., $\hat{\boldsymbol{B}}_k = [\hat{u}_{min}, \hat{u}_{max}, \hat{v}_{min}, \hat{v}_{max}]_k$.

4.3.2 ROSHAN Bounding Box Measurement Model

The assumption that each bounding box edge measurement $\boldsymbol{l}_h$ projects to a plane $\boldsymbol{\pi}_h$ tangent to the object of interest is broken in the case of partial, occluded, or truncated detections. A naive approach to using bounding boxes as measurements might simply keep all measurements, and hope that enough additional measurements will be made to mitigate the erroneous measurements. Nicholson et al. [118] present a truncated measurement model that ignores the portion of the measurement error that is outside of the image boundaries. When an object is well estimated, the truncated geometric model does reduce false measurement errors on edges that do not constrain the object by recognizing that the measured bounding box edge is the best observation the object detector can make. However, the truncated measurement model underestimates error in cases where the instantaneous bounding box estimate of the object position in the image plane is poor and a measured bounding box edge *is* in fact a constraining edge. For example, if the true object projects entirely into the image, but the instantaneous estimate of that object in the image plane is an

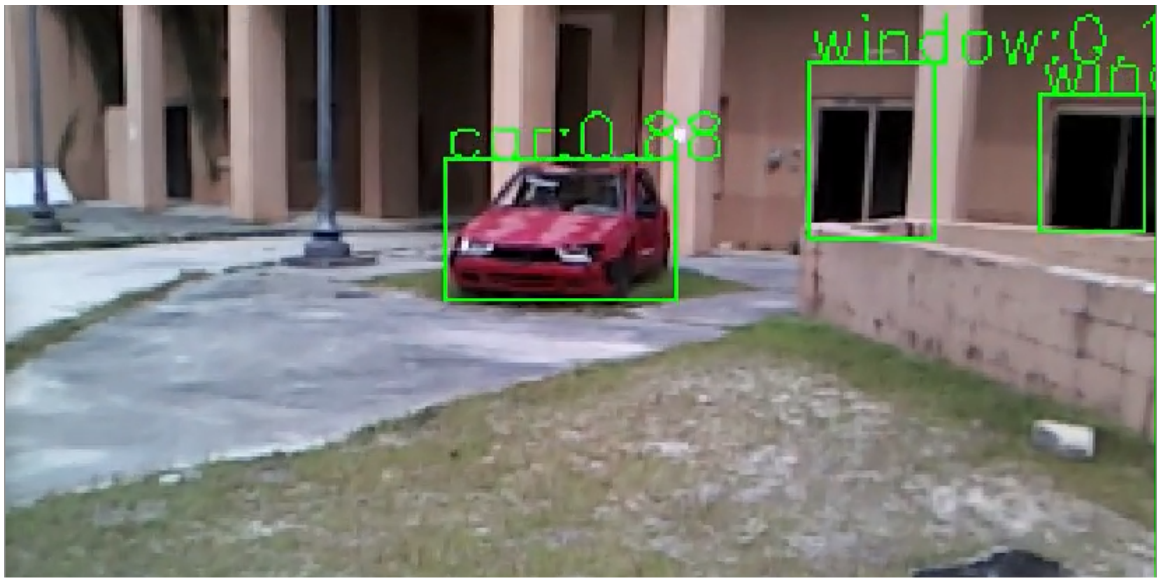


Figure 4-3: An example of bounding box detections containing different types of non-constraining edges. The right edge of the rightmost window is a non-constraining edge at an image boundary, and the right edge of the leftmost window is a non-constraining edge formed by an occlusion between two objects: a pillar and a window. The car in the middle is tightly detected, as expected in the nominal case. Figure generated by Nicholas Villanueva.

overestimate that extends off the image, the truncated model will underestimate the error.

In ROSHAN, we first classify a bounding box edge as constraining (tangent to the object) or non-constraining (not tangent to the object) based on the proximity to the closest image boundary, before adding the edge as a constraint on the detected object. As shown in Fig. 4-3, we observe that a non-constraining edge can be formed both near the image boundaries and the occlusion boundaries between objects. However, as is the case of the truncated measurement model [118], we focus on identifying only the non-constraining edges near the image boundaries and leave potential ways to identify occlusions between objects, such as using depth discontinuities from learned depth images [57], as future work. Once a bounding box edge is classified as non-constraining based on the distance to the closest image boundary, instead of applying a truncated measurement model, we simply discard the edge, realizing that it is not an actual constraint.

Note that the closed-form measurement model in Eq. 5.10 has imaginary solutions when the term under the square root is negative. Geometrically, the imaginary solutions represent a camera being inside or axis-aligned with an observed ellipsoid, which may happen when the estimates of the ellipsoid parameters move during the optimization. In the case of this degeneracy, we set the measurement error to be high to discourage the iterative optimizer from stepping towards the degenerate solution; an alternative way would be to add an explicit cost such as the inverse barrier cost [23].

4.3.3 Texture Plane Measurement Model

While bounding box measurements from diverse viewpoints can fully constrain an ellipsoid, given any *single* viewpoint, there are parameters of an ellipsoid that a bounding box measurement simply cannot observe. This is similar to the case in feature-based monocular SLAM where in any single image, a 2D landmark detection can only constrain the bearing of the landmark, but not the depth [114]. Similarly, a bounding box detection, which is a set of 4 orthogonal planar constraints induced by each of the bounding box edges, cannot fully constrain an ellipsoid inside a cuboid, i.e., fully constrain the volume of the ellipsoid, without two additional orthogonal planes for the missing faces of the cuboid.

However, there is a fifth measurable plane that is parallel to the camera image plane and fit to the high-gradient texture on the object. This plane that we refer to as the *texture plane* can be measured using triangulated feature points on the surface of the object, i.e., detected inside the bounding box, with co-observations in two or more cameras. Assuming that the triangulated feature points are all observations of the same tangent plane $\hat{\boldsymbol{\pi}^t}_d = [0, 0, 1, -\hat{z}]$, we can utilize the plane exactly the same way that bounding box planes are used to constrain an ellipsoid, i.e., solve the system of equations

$$[0, 0, 1, -\hat{z}]^T \left([\boldsymbol{R}_t | \boldsymbol{t}_t] \boldsymbol{Q^*}_j [\boldsymbol{R}_t | \boldsymbol{t}_t]^T \right) [0, 0, 1, -\hat{z}] = 0, \tag{4.11}$$

and obtain the predicted measurement of the texture plane $\hat{z}$ as a differentiable closed-form solution, i.e.,

$$\hat{\boldsymbol{\pi}}^{t}{}_{d} = h_{tp}(\boldsymbol{Q}^{*}{}_{j_d}, \boldsymbol{x}_{t_d}; \boldsymbol{K}). \tag{4.12}$$

This additional texture plane helps better constrain our SLAM system, when the vehicle motion is not orbital and diverse viewpoints of objects cannot be guaranteed.

4.3.4 Semantic Shape Prior

While the texture plane introduces a fifth plane to constrain an ellipsoid, for any single viewpoint there is one more orthogonal plane needed to mimic a 3D bounding box constraint. In the absence of this plane or a different view, the scale of the ellipsoid is ambiguous and the ellipsoid may be arbitrarily long on the other side of the texture plane.

To mitigate this problem of scale unobservability, we introduce *semantic* priors on the ellipsoids where we assume a semantically-informed Gaussian priors on the shape $\boldsymbol{d} \in \mathbb{R}^3$ and uniform priors on the position and the orientation of ellipsoids. While the semantic priors could be learned from large data sets as done in [52], we observe that many objects of interest are relatively consistent in size to allow a model-free specification using standard sizes. In this work, we create a function h_{shape} using publicly available data on the metric shape of objects to approximate the mean $\mu_{c_j} \in \mathbb{R}^3$ based on the class label $c_j \in \mathbb{N}$, i.e., $\mu_{c_j} = h_{shape}(c_j)$, and specify a diagonal covariance matrix $\Sigma_{c_j} \in \mathbb{R}^{3\times3}$ per object class to reflect the degree of consistency in the shape of objects. In our real-world and simulated flight experiments, we use the dimensions of a Toyota Camry as a reasonable mean of the prior over objects recognized as cars, with the largest covariance on the length of the car to account for longer size cars.

4.3.5 Single Image Initialization

Similar to undelayed-initialization methods for point-based landmarks such as inverse depth initialization [30], we can also initialize ellipsoids using a single bounding box measurement without having to do the delayed initialization in Nicholson et al. [118], allowing ROSHAN to quickly perceive and avoid obstacles during high-speed flight. To realize a fast initialization scheme for the full 9 parameters of an ellipsoid, we make three reasonable assumptions. First, we assume that the position of the ellipsoid is somewhere along the camera ray that passes through the center of the bounding box [17]; the depth along this ray is estimated to be at an experimentally chosen reasonable object distance similar in spirit to the average scene depth initialization in [47]. Second, while there are single-image object orientation estimators [151], we assume the initial orientation to be the identity for simplicity. Lastly, we assume the shape of the ellipsoid to be at the mean of the semantic shape prior.

Given these assumptions, we initialize an ellipsoid with the first detection, trading off the accuracy in our initial estimates for faster perception. In ROSHAN, the inaccuracy in the initial estimates is mitigated by the faster converging bounding box model discussed in the previous sections.

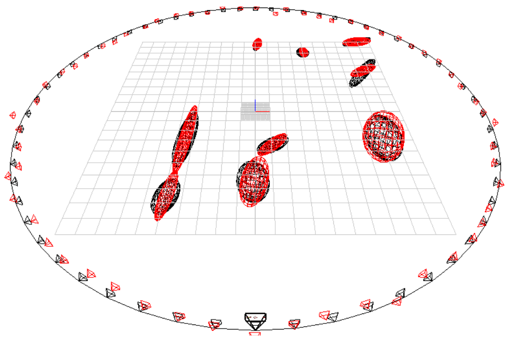
4.3.6 Online Optimization

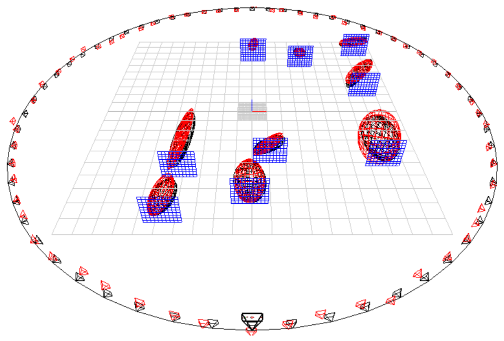
Assuming Gaussian measurement and process models, we can write Eq. 4.6 as a nonlinear least-squares problem [34]:

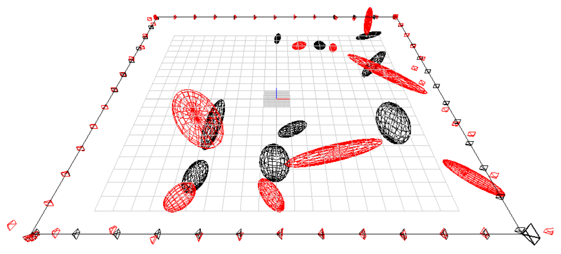
$$\begin{aligned}\boldsymbol{\mathcal{X}}^*, \boldsymbol{\mathcal{Q}}^* &= \operatorname*{arg\,min}_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Q}}} \; -\log P(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Q}} | \boldsymbol{\mathcal{B}}, \boldsymbol{\Pi}^t, \boldsymbol{\mathcal{I}}; \boldsymbol{\mathcal{C}}) \\ &= \operatorname*{arg\,min}_{\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{Q}}} \Big\{ \sum_{t=0}^{T} \| f_{pose}(\boldsymbol{I}_{0:t}) - \boldsymbol{x}_t \|^2_{\Sigma_{o_t}} + \sum_{k=0}^{K} \| h_{bb}(\boldsymbol{Q^*}_{j_k}, \boldsymbol{x}_{t_k}; \boldsymbol{K}) - \boldsymbol{B}_k \|^2_{\Sigma_{b_k}} + \\ &\sum_{d=0}^{D} \| h_{tp}(\boldsymbol{Q^*}_{j_d}, \boldsymbol{x}_{t_d}; \boldsymbol{K}) - \boldsymbol{\pi}^t_d \|^2_{\Sigma_{t_d}} + \sum_{j=0}^{J} \| h_{shape}(c_j) - d(\boldsymbol{Q^*}_j) \|^2_{\Sigma_{c_j}} \Big\},\end{aligned} \tag{4.13}$$

where $\|\cdot\|^2_\Sigma$ is the Mahalanobis norm that directly scales the measurement error inversely proportional to the square root of the covariance term Σ. The covariance on the pose prior $\Sigma_{x_t} \in \mathbb{R}^{6\times 6}$, which is computationally expensive to obtain from the external source, is set to a heuristically chosen value, the diagonal covariance on the

[1]In later work, Chen et al. [27] propose to improve our method by considering the average depth of points triangulated on the object, in addition to their own novel initialization approach.


(a) Baseline (Orbit Path)


(b) ROSHAN (Orbit Path)


(c) Baseline (Forward Path)

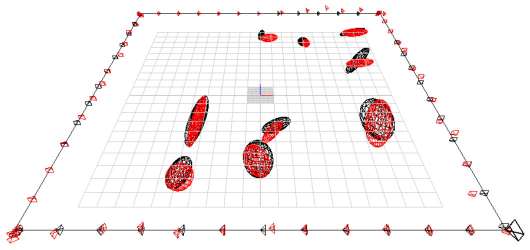

(d) ROSHAN (Forward Path)

Figure 4-4: Final estimates (red) of ellipsoids and cameras inferred using the baseline (bounding boxes only) and ROSHAN in a randomly generated map of ellipsoids. Shown in (a) and (b), estimating the parameters of the ellipsoids using diverse viewpoints of an orbiting vehicle path resulted in a small error in shape and position for both methods. However, when using measurements from a forward-moving vehicle path, where only limiting views were available, ROSHAN outperformed the baseline by a larger margin showing the strength of our approach under undesirable but common vehicle motions. Sub-figures (a) and (c) are reproduced from 4-1 for viewing convenience.

bounding box measurements $\Sigma_{b_k} \in \mathbb{R}^{4\times4}$ is also set to an experimentally chosen noise value, the variance on the texture plane $\Sigma_{t_d} \in \mathbb{R}$ is the empirical variance in the depth of the triangulated points for real-world experiments, and the covariance on the prior $\Sigma_{p_j} \in \mathbb{R}^{6\times6}$ is specified as described in section 4.3.4.

We periodically linearize the problem in Eq. 4.13, and optimize in real-time for the cameras and the objects using Levenberg-Marquardt [112] algorithm. In the next section, we present experimental results on simulated and real flight sequences using an online optimization scheme, which can be more susceptible to poor solutions compared to offline batch methods, to demonstrate the advantages of ROSHAN.

Table 4.1: Median error in estimated ellipsoids for ROSHAN and the baseline in 50 randomly simulated maps of ellipsoids.

	Orbit Path			Forward Path		
	shape	pos.	orient.	shape	pos.	orient.
Baseline	0.26	0.11	26.81	1.16	1.66	43.65
ROSHAN	**0.17**	**0.10**	**17.97**	**0.20**	**0.47**	**30.93**

4.4 Experimental Results in Simulation

We tested ROSHAN in an OpenGL simulation, where all objects are exactly ellipsoids, so that the ground-truth parameters of the ellipsoids can be used to evaluate the estimation accuracy of ROSHAN and a baseline in terms of shape, position, and orientation. In the following experiments, we considered the baseline to only use the bounding box measurements as done in [118], but kept our online optimization framework with improvements on shape regularization and the bounding box measurement model to obtain a baseline meaningful for comparison. Rather than using triangulated texture points on the object, the simulator provides a fifth plane measurement to ROSHAN at every timestep. The simulator also provides ground-truth data association for each measurement.

We compared ROSHAN against the baseline in 50 randomly generated maps in two sequences with diverse (Orbit) and non-diverse (Forward) paths with Gaussian noises added to the bounding boxes and initial estimates for poses and ellipsoids. There are many potential methods for determining distances between ellipsoids as all parameters are deeply coupled; in this work we compare shape, translation, and rotation parameters of the ellipsoids directly, and attempt to account for symmetries in ellipsoids by bounding the rotation error. The development of additional comparison metrics may also lead to further insights in future work. Summarized in Table 4.1, we observed that all systems performed similarly well when given diverse viewpoints (Orbit). However, as illustrated in Fig. 4-4, when given degenerate viewpoints typical of forward-moving vehicle motions (Forward), ROSHAN outperformed the baseline with a 83% reduction in the error in the shape estimates (meters) based on the median error across average error per map, and 72% error reduction in the position estimates (meters); there was a smaller improvement of 29% on the orientation estimates (degrees) but neither method performed particularly well.

To further analyze the effect of degenerate viewpoints on the systems, we randomly sampled 20 ellipsoids, and for each ellipsoid, estimated its parameters using randomly sampled views from a Gaussian clipped to fixed ranges of viewpoints (yaw) around the ellipsoid. Shown in Fig. 4-5, for the baseline method, more views from a greater viewpoint range was required to reduce the error in both the shape and the position. However, for ROSHAN, the error in the shape estimate was small even with a single view due to the usage of shape information, and the error in position was also relatively small even with less views and viewpoint ranges, indicating a more robust system under challenging motions.

These results show that ROSHAN is able to estimate the parameters of the objects more efficiently than the baseline by leveraging both geometric measurements (bounding box edges and the plane induced by texture tracking) and extra-geometric, semantic information (class shape priors). Specifically, given the same camera motions, ROSHAN is able to more accurately estimate object parameters, indicating better suitability for supporting efficient autonomous navigation.

4.5 Experimental Results on Flight Sequences

To demonstrate the advantages of ROSHAN, we evaluated the performance of the algorithm both in simulation and on real-world data collected in an urban environment using a flight stack developed by the MIT/Draper team for the DARPA Fast Lightweight Autonomy (FLA) program. The photo-realistic simulation environment is a mock city rendered via the Unity Game Engine [74]; for the simulation experiments we used ground-truth poses and added Gaussian noise to the bounding boxes at run-time. On the real flight data, the pose estimates were provided by an external SAMWISE VIO algorithm [165], which consumed monocular images and measurements from an IMU. For object detection on the real flight, we used the Mobilenet-SSD network [68, 104] running at roughly 8 Hz. In both flight segments, the vehicles

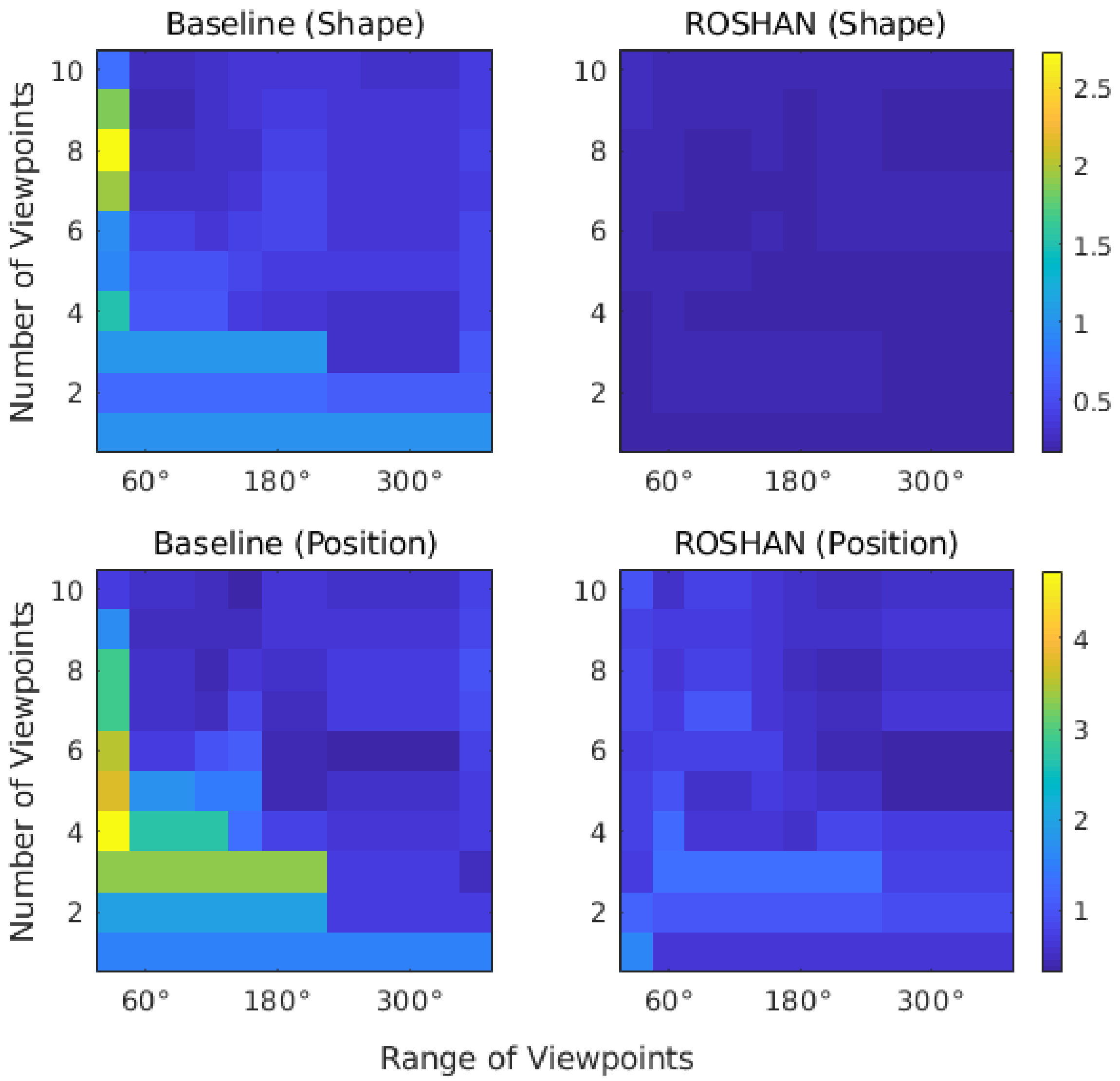


Figure 4-5: Median shape error (top row) and position error (bottom row) for baseline (left column) and ROSHAN (right column) computed using different number of viewpoints (y-axis) randomly sampled from varying allowed ranges of yaw (x-axis). For the baseline method, more views from a greater viewpoint range was required to reduce the error (meters) in both shape and position. However, for ROSHAN, both errors were small even with less views from limiting viewpoints due to combining multiple sources of information.

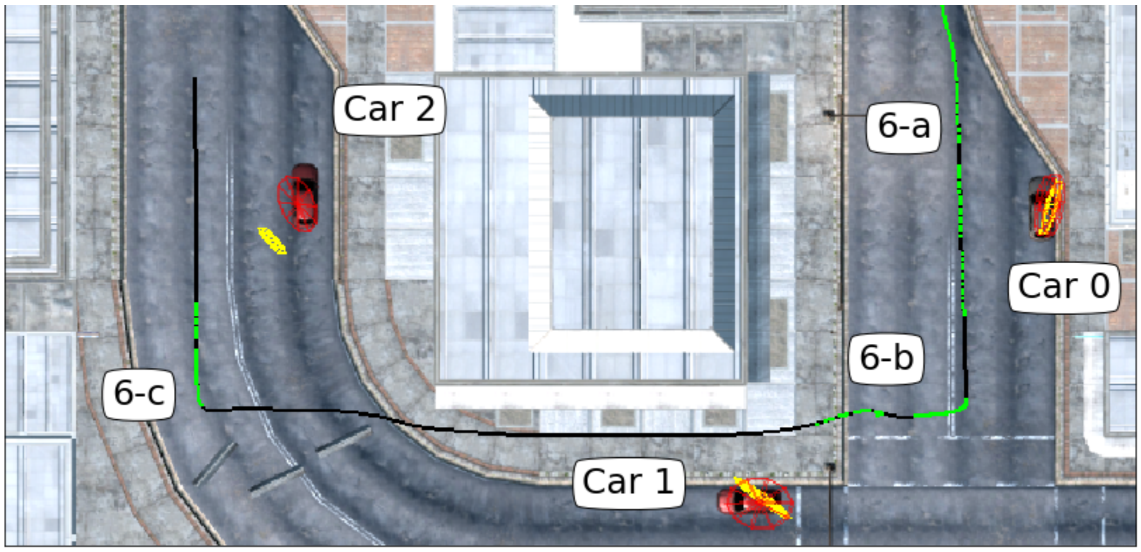
observed three cars, and did not explicitly orbit the cars.

4.5.1 Implementation details

Each valid bounding box detection was associated to an existing ellipsoid, or triggered a new landmark creation. Given a new bounding box detection, we filtered out ellipsoids using the distance between the measured centroid and predicted centroid, and the best match was chosen using a correlation score between the image hue and saturation histograms within a detection and those of previous detections; if no match was found, we initialized a new landmark. To exploit texture information, we extracted ORB features [147] from the bounding box patches, and used Lucas-Kanade [107] to track the features, both implementations utilizing the open-source OpenCV library [18]. While a more sophisticated sparse SLAM system [114] could be used instead, in this work we used a minimal technique, where a simple triangulation was performed between two detections of the object; the texture plane was then fit to the mean depth of the points. We observed that our assumption that all triangulated points lie on the same tangent plane can be broken here if an object is oblong and sufficiently rotated; we carefully chose which planes to add to the graph using metrics such as the variance of the triangulated points and the length of the baseline. This differs from the experiments in Section 4.4 in that not every bounding box measurement is accompanied by a fifth plane measurement. As in the OpenGL simulation experiments, the baseline had improvements in ROSHAN but did not incorporate the texture plane or the semantic shape prior. As the system was run online, measurements were sometimes dropped from the measurement buffer; we present here representative results from both methods.

4.5.2 Results on Simulated and Real Flight

We tested ROSHAN on data collected both in simulation and in the real-world, qualitatively demonstrating the online capabilities of our approach. For visualization purposes, in each experiment the objects were overlaid on an overhead image in



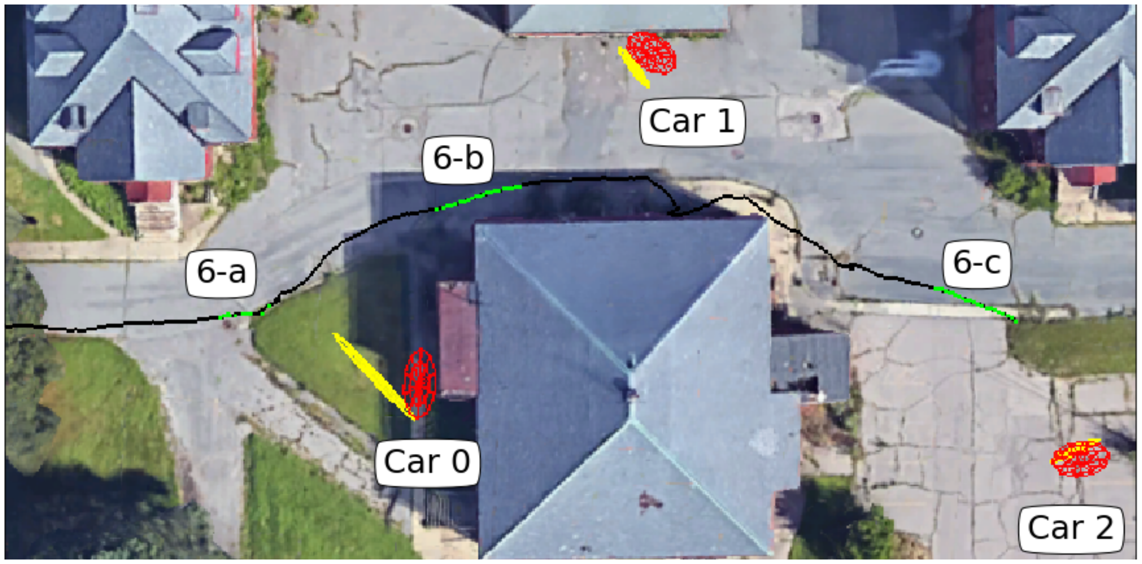


Figure 4-6: Ellipsoids estimated by ROSHAN (red) and the baseline (yellow) drawn as orthographic projections along with the raw trajectory (black) and ROSHAN estimated poses with valid object detections (green). In the photo-realistic simulation (top), ROSHAN had a lower average position error of 0.84m, compared to the 1.54m of the baseline. In the real-world experiment (bottom), the origin of the projected estimates were hand-aligned in 2D to a metrically scaled overhead GPS image for qualitative analysis only.

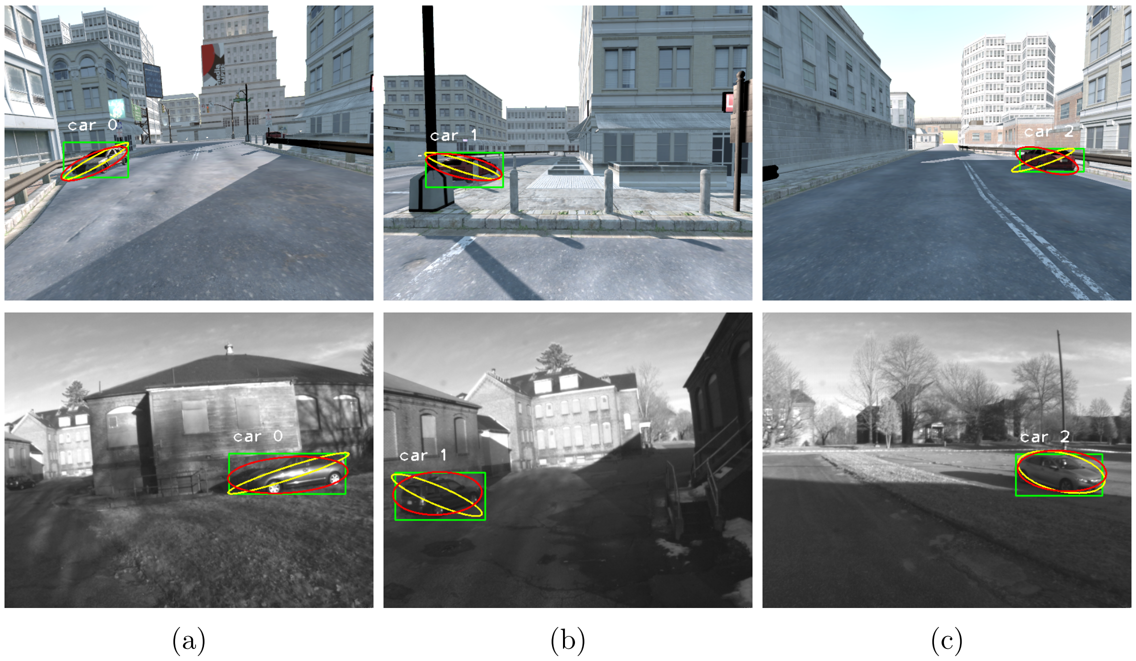


Figure 4-7: Comparison of the projected ellipsoid estimates of ROSHAN (red) and the baseline (yellow) implementations onto the images at similar points in the raw test trajectory for the photo-realistic simulation (top) and the real-world flight (bottom). The noisy bounding boxes (green) correspond to the baseline run. Projected conics estimated with ROSHAN better approximated the outline of the cars.

Fig. 4-6. Figure 4-7 shows that given similar measurements[2], our approach generates estimates that better fit the object in question on the image plane. Without semantic shape information, the baseline often optimized to low-volume ellipsoids that still satisfied the bounding box constraints. By adding the semantic shape information, we were able to avoid solutions of unreasonable volumes. These results complement those obtained in the texture-less OpenGL simulation, showing that by combining additional geometric information with extra-geometric, class specific priors, ROSHAN is capable of obtaining better estimates under difficult navigation behaviors.

[2]As the system was run online, measurements were sometimes stochastically dropped; we present here representative results from both methods.

4.6 Limitations

There are several limitations to the work presented in this chapter that may be of particular interest for future exploration. First, while we do not hand-label data associations as done in the experiments for QuadricSLAM [118], the objects mapped in our experiments are relatively well spaced apart and of a single class. Therefore, our simple data association methods were aggressively tuned to try to limit erroneous associations, but may struggle in more cluttered scenes where disambiguation is more difficult (such as when many objects occlude each other). Some methods we have explored to make this process more robust include learning deep bounding box descriptor spaces [155] and applying traditional patch tracking methods to ensure stable object tracks in the image-space before estimating objects as ellipsoids [121], but further work in improving object-level data association will lead to increasingly robust object-SLAM systems. Second, we rely on hand-specified object shape priors, which while suitable for many objects may be more difficult to obtain for unique objects such as buildings.

4.7 Chapter Summary

We have presented ROSHAN — an ellipsoid-landmark based object-level SLAM system that improves estimation quality in the case of vehicle trajectories characterized by those typical of efficient autonomous navigation by incorporating additional geometric and semantic information. These improvements are achieved by the introduction of a texture plane factor, which constrains the depth of the landmark by exploiting texture information, and a prior on object shape that enables fast object initialization, useful for high-speed vehicle motions. We have shown in an OpenGL simulation featuring forward-motion that using these extra sources of information reduced the median errors on shape and position by 83% and 72% respectively, compared to the baseline. Similar improvements were also observed in a photo-realistic Unity simulation environment, and qualitative results were obtained on a real-world dataset, where ROSHAN estimated the shape and the position of cars reasonably well. Our approach allows for accurate estimates even under challenging camera motions, and contrary to modern offline methods [146, 118] runs online and can initialize landmarks from a single detection.

In the continued theme of combining geometric and semantics, our approach formulates a new geometric constraint in addition to introducing a class-based shape prior. Formulating the extra-geometric information as a shape prior rather than a hard constraint has two key benefits: first, it biases the geometric optimization to a reasonable estimate which enables fast landmark initialization. Second, it allows the shape to be improved as additional geometric measurements are acquired.

Although object-level SLAM is a powerful and general framework, the approach presented in this chapter is designed with the expectation of multiple measurements being fused together before reliable estimates are obtained. In the next chapter, we explore an alternative avenue for 3D object estimation from a single sensor measurement, seeking to lower the latency between detection and estimate.

Chapter 5

Learning Deep Object Estimation from Indirect Annotations

5.1 Introduction

In the previous chapter, we discussed a vision-based simultaneous localization and mapping approach that fuses multiple object-level measurements to build an object-level map. While fusing a number of measurements can benefit the estimation accuracy and aid with observability issues, the inference process introduces latency between image registration and estimation. Although methods for single view object regression exist, they still require an online optimization process (Section 2.2.2). In this chapter, we leverage deep learning to enable low-latency object-level estimation from RGB sensors.

As we explored in Chapter 2, deep learning has gained popularity as a method to exploit offline datasets to reduce latency and enable single-shot 3D object estimation, but generally requires direct annotations or *a priori* geometric labels that are typically more difficult to obtain than 2D image-space annotations (Section 2.2.4). Unlike 2D image-space annotations, such as 2D bounding boxes or pixel-wise segmentation, 3D annotations often require additional infrastructure (Section 2.2.4). This requirement presents a practical difficulty when attempting to use deep object estimation to provide context for autonomous navigation. Therefore, while deep learning approaches have the potential to enable object prediction from a single measurement by leveraging an offline learning phase, the information required for that learning process raises the barrier to entry.

In this work, we introduce a novel framework for learned volumetric monocular estimation (VoluMon) capable of estimating the position, orientation, and size of objects, while requiring only indirect, image-space annotations on 2D images at training time instead of detailed geometric models or 3D labels. By approximating object geometry with ellipsoids, we can exploit differentiable geometric and algebraic relationships between ellipsoids and 2D annotations to enable a weakly supervised learning process (i.e., supervised via indirect annotations only) and significantly lower the annotation burden for deeply learned methods. VoluMon trains a deep neural network to predict the 3D size and 6D pose of objects using annotated bounding boxes, instance segmentations paired with depth images, or both. VoluMon also encourages objects of the same class to be similar sizes by penalizing intra-class shape variance. The core of our approach requires only a bounding box detection and single RGB image at inference time; obtaining an estimate of the pose and size of an object is simply a feed-forward pass through the network, rather than an iterative optimization process. However, given additional segmented depth information at run-time, the ellipsoid approximation also allows for object pose estimates from VoluMon to be further refined without requiring *a priori* geometric models. An overview of our approach is given in Figure 5-1.

We demonstrate the advantages of our approach by exploring several variants of VoluMon depending on the available sensors (i.e., RGB vs. RGB-D) and annotations (bounding boxes and or instance segmentations) at training time and run time on subset of the Falling Things Dataset [172]. We show that using only a monocular sensor at inference time, VoluMon performs similarly to a naive point-cloud based online ellipsoid estimation approach while requiring less than 1% of the time.[1] Given segmented depth information to further refine pose estimates at inference time, for some metrics we approach or surpass the performance of a deeply learned 6D object

[1]Detailed timing results can be found in Table 5.1.

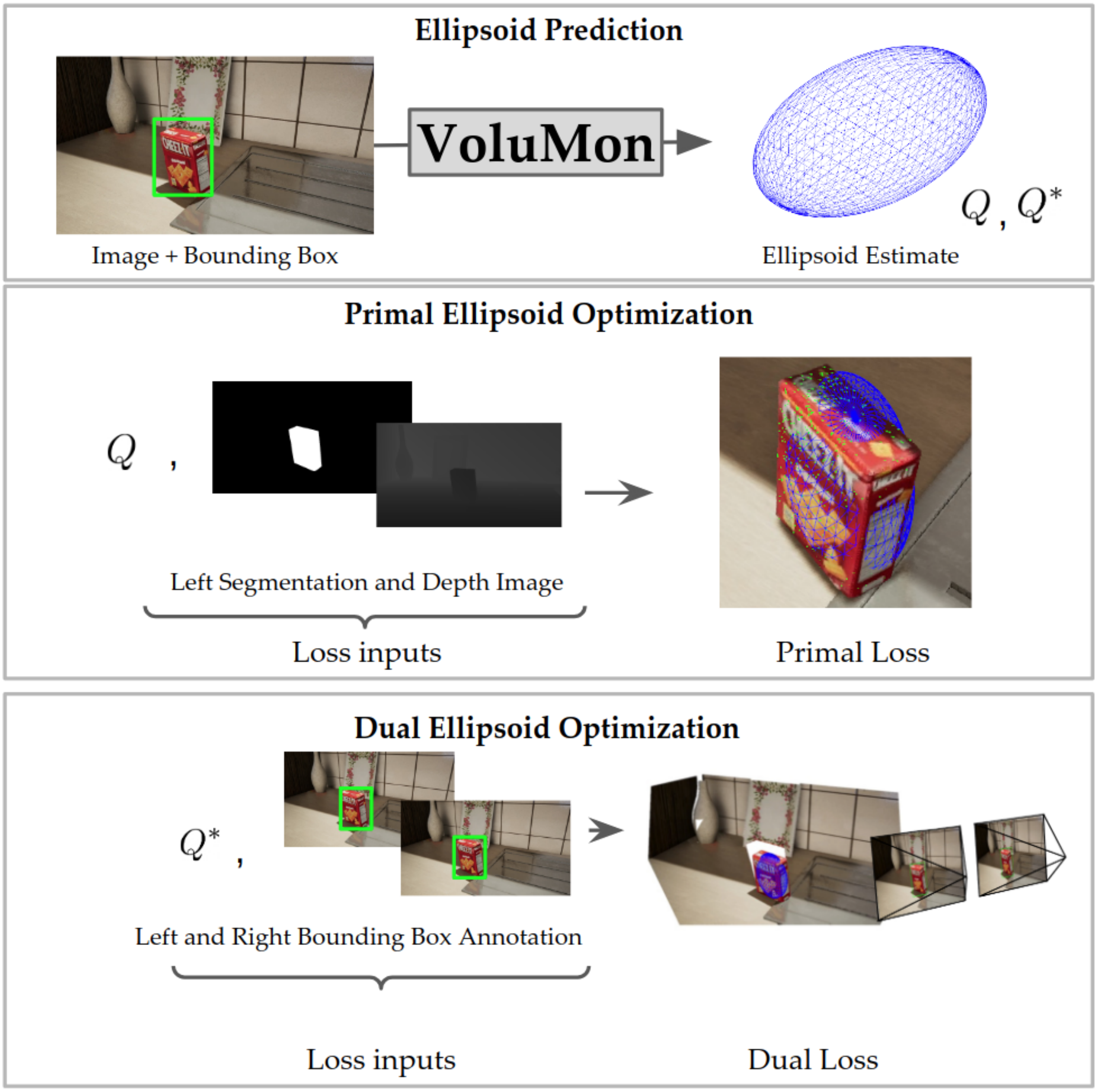


Figure 5-1: (Top row) VoluMon predicts the 3D object pose and shape from a monocular image and bounding box detections (green box) by approximating objects as ellipsoids (blue mesh). We weakly supervise learning by exploiting two different forms of 3D ellipsoid representations. (Middle row) The primal form of the ellipsoid provides a differentiable algebraic metric to fit to 3D points (green points) extracted from an instance segmentation annotation and depth image. (Bottom row) The dual form of the ellipsoid provides a differentiable geometric metric for VoluMon to compare annotated bounding boxes from a stereo pair with the projected bounding box from the ellipsoid estimate. Without requiring an *a priori* model, VoluMon can also further refine the estimated ellipsoid parameters online given a pointcloud.

pose algorithm that assumes groundtruth size information.

As the ultimate goal of our work is to lower the annotation and *a priori* geom-

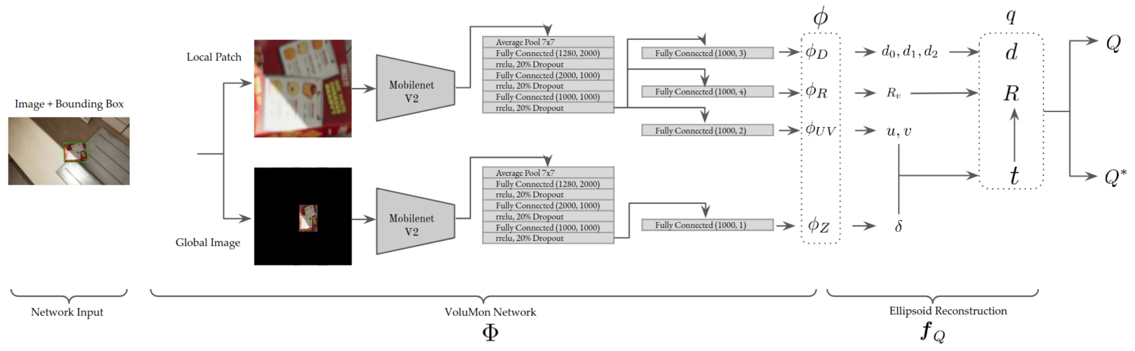


Figure 5-2: VoluMon system diagram. Given an image and bounding box, VoluMon passes the resized contents of the bounding box to the local patch sub-network, and the resized full image with contents outside of the bounding box set to zero in all channels to the global image sub-network. Each image and bounding box pair yields one ellipsoid estimate. Both sub-networks utilize MobilenetV2 as a convolutional feature extractor that feeds into a series of fully connected layers. To improve generalization performance, VoluMon predicts the image coordinates of the projected object centroid, the allocentric rotation, and shape parameters from the contents in the region of interest only, while depth of the centroid of the object is predicted using features from the global image. The raw output of the network $\boldsymbol{\phi}$ are reconstructed to q to constrain predictions to reasonable ellipsoids. The subscript notation i is dropped for readability.

etry barrier for low-latency deep object estimation, we conduct several additional experiments. First, we present promising qualitative results on a real-world dataset of objects without annotated ground-truth pose or size, using only bounding box annotations from a stereo pair. Second, we extend our base approach to perform end-to-end detection and 3D objectprediction of seven objects and show preliminary results that suggest our method can ultimately be made even faster. Finally, we integrate VoluMon with the planning approach proposed in Chapter 3, and show an example of how contextual cues provided by VoluMon can inform autonomous navigation.

5.2 Preliminaries

Similar to the approach described in the previous chapter, VoluMon approximates 3D objects with ellipsoidal volumes parameterized by the orientation $\boldsymbol{R} \in \mathrm{SO}(3)$ and position $\boldsymbol{t} \in \mathbb{R}^3$ with respect to the sensor viewing the object, as well as the size $\boldsymbol{d} \in \mathbb{R}^3$ of the major axes of the ellipsoid[2]. In particular, we consider two mathematically convenient forms of the ellipsoidal representation: the dual and the primal.

As discussed in Chapter 4, the dual form of the ellipsoid defines the infinite set of planes $\boldsymbol{\pi}$ tangent to the surface of the ellipsoid, i.e., $\boldsymbol{\pi}^T\boldsymbol{Q}^*\boldsymbol{\pi} = 0$, where

$$\boldsymbol{Q}^* = \begin{bmatrix} \boldsymbol{R}\boldsymbol{D}\boldsymbol{R}^T - \boldsymbol{t}\boldsymbol{t}^T & -\boldsymbol{t} \\ -\boldsymbol{t}^T & -1 \end{bmatrix}. \tag{5.1}$$

In contrast, the primal ellipsoid in 3D can be expressed as the set of all homogeneous points satisfying the implicit algebraic relationship $x^T\boldsymbol{Q}x = 0$. A naive approach to obtaining the primal ellipsoid might be to exploit the fact that $\boldsymbol{Q}$ is equal to the inverse of $\boldsymbol{Q}^*$ if $\boldsymbol{Q}^*$ is invertible (up to scale), as given by Hartley and Zisserman [60]. However, the inversion of an arbitrary matrix can be difficult to integrate into conventional optimization algorithms.[3] Instead, we derive a simple closed-form solution for the primal ellipsoid, much as we did for the dual ellipsoid in Section 4.2.1.

A point quadric can be transformed by a transformation matrix $\boldsymbol{Z}$ as follows [60]:

$$\boldsymbol{Q} = \boldsymbol{Z}^{-T}\breve{\boldsymbol{Q}}\boldsymbol{Z}^{-1}. \tag{5.2}$$

We again take $\boldsymbol{Z}$ to be defined as in Equation 4.2 and for convenience we define $\breve{\boldsymbol{Q}}$ to be a scaled primal ellipsoid at the origin. As $\boldsymbol{Z}$ is a homogeneous transformation matrix, the inverse takes the form [93]:

$$\boldsymbol{Z}^{-1} = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0_3}^T & 1 \end{bmatrix}. \tag{5.3}$$

[2]Note that we have used $\boldsymbol{d}$ to denote the minor axes of the ellipsoid in previous chapter.

[3]It is worth observing that most previous literature that model objects as ellipsoids use either the primal or dual form in isolation. In Chapter 4, we induce a plane measurement from a set of surface points, using only the dual form of the ellipsoid. Liao et al. [97] use a pointcloud from a depth measurement to estimate an initial ellipsoid, but ultimately represent the measurement via bounding planes in the optimization framework. Hosseinzadeh et al. [66] model points, planes, and quadrics in a consolidated SLAM framework, with point-plane, plane-plane, quadric-plane constraints, but not point-quadric constraints.

To find the representation of a dual ellipsoid with primal size parameters $\boldsymbol{D}$, rotation $\boldsymbol{R}$, and translation $\boldsymbol{t}$ we apply Equation 5.2:

$$\boldsymbol{Q}' = \begin{bmatrix} \boldsymbol{R} & \mathbf{0_3} \\ -\boldsymbol{t}^T\boldsymbol{R} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{D}^{-1} & \mathbf{0_3} \\ \mathbf{0_3^T} & -1 \end{bmatrix} \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \mathbf{0_3^T} & 1 \end{bmatrix}, \tag{5.4}$$

yielding

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{A} & -(\boldsymbol{A})\boldsymbol{t} \\ -\boldsymbol{t}^T(\boldsymbol{A}) & -1 + \boldsymbol{t}^T(\boldsymbol{A})\boldsymbol{t} \end{bmatrix}, \tag{5.5}$$

where $\boldsymbol{A} = \boldsymbol{R}\boldsymbol{D}^{-1}\boldsymbol{R}^T$, and $\boldsymbol{D} = (\text{diag}(\frac{\boldsymbol{d}}{2}))^2$.

A strong advantage of approximating objects as ellipsoids, as opposed to other geometric models such as 3D bounding boxes, is that both the dual and primal forms provide differentiable relationships between the object and quantities obtained from image-space annotations paired with sensor measurements, suggesting their suitability use in deep learning frameworks that are optimized via back-propagation. The dual ellipsoid form allows for the closed-form calculation of a projected image axis-aligned 2D bounding box induced by an ellipsoid estimate, therefore providing a differentiable geometric metric with which to compare bounding box detections of an object. The primal form of the ellipsoid provides a differentiable algebraic metric to measure how well an observed surface point of an object, which can be obtained from an instance segmentation and depth image, agrees with an ellipsoid estimate. VoluMon trains a deep neural network (described in Section 5.3) to predict the parameters $q_i = (\boldsymbol{d}_i, \boldsymbol{R}_i, \boldsymbol{t}_i)$ of an ellipsoid approximation using differentiable measurement functions derived from the ellipsoid representation (described in Section 5.4).

5.3 Model Overview

VoluMon trains a model (shown in Fig. 5-2) to predict ellipsoid parameters from images $\boldsymbol{I}_t : \Omega \in \mathbb{N}^2 \to \mathbb{R}$, where Ω is the image pixel domain, and bounding boxes $\boldsymbol{B}$, which are characterized by the pixel locations of the four corners. Let Φ be a neural network parameterized by $\boldsymbol{\beta}$ that takes as input a set of images $\boldsymbol{\mathcal{I}} = \{\boldsymbol{I}_i\}_{i=0}^K$ and bounding boxes $\boldsymbol{\mathcal{B}} = \{\boldsymbol{B}_i \in \Omega^2\}_{i=0}^{K}$ around the K objects of interest. The network outputs free parameters $\boldsymbol{\phi}_i = [\boldsymbol{\phi}_{i,D}, \boldsymbol{\phi}_{i,R}, \boldsymbol{\phi}_{i,UV}, \boldsymbol{\phi}_{i,Z}] \in \mathbb{R}^{10}$ per object estimate, where $\boldsymbol{\phi}_{i,D} \in \mathbb{R}^3$, $\boldsymbol{\phi}_{i,R} \in \mathbb{R}^4$, $\boldsymbol{\phi}_{i,UV} \in \mathbb{R}^2$, and $\boldsymbol{\phi}_{i,Z} \in \mathbb{R}^1$ are used to reconstruct respectively the size, rotation, centroid projection, and depth of the object in the camera frame. To ensure the prediction of valid ellipsoids, we formulate an additional function $\boldsymbol{f}$ that maps the outputs of the model Φ to reasonable $\boldsymbol{q}$, yielding the relationship

$$\boldsymbol{q} = \boldsymbol{f}(\Phi(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{B}}; \boldsymbol{\beta}), \boldsymbol{\mathcal{B}}), \tag{5.6}$$

where $\boldsymbol{q} = \{q_i\}_{i=0}^{K}$, and $\boldsymbol{f}$ takes as input $\boldsymbol{\mathcal{B}}$ to constrain the projection of the centroid estimate to lie within the detected bounding box in the image frame.

Rather than predict the rotation, shape, and translation of an object from a single set of shared features, which could be difficult to generalize to arbitrary object locations, VoluMon splits the prediction of object properties between two decoupled sub-networks. The local patch sub-network predicts object properties that are independent of where in the image the bounding box is located and outputs $\boldsymbol{\phi}_{i,D}, \boldsymbol{\phi}_{i,R}, \boldsymbol{\phi}_{i,UV}$ per object, while the global image sub-network helps to estimate global pose properties and outputs $\boldsymbol{\phi}_{i,Z}$ per object. Finally, the mapping from $\boldsymbol{\phi}$ to q developed in the following sections defines the function $\boldsymbol{f}$ required by Equation 5.6.

Global Image Sub-Network

The global image sub-network receives the RGB image resized to 224x224 pixels,[4] with all channels outside the observed bounding box set to zero with some padding, and predicts a bounded real number as the "disparity" of the centroid, δ_i. The possible centroid depth is obtained by constraining the raw outputs such that $\delta_i = \alpha_\delta \text{sigmoid}(\boldsymbol{\phi}_{i,Z})$, and letting $t_{z,i} = bf/\delta_i$, where f, b are set to roughly the focal length and baseline of the stereo camera, and α_δ is the maximum allowed disparity. This parameterization seeks to abstract prediction of the centroid depth from camera parameters.

[4]We resize to use MobileNet as our feature extractor, and scale the image non-uniformly.

Local Patch Sub-Network

The local prediction network receives the image in the bounding box, resized to 224×224 pixels. Rather than allowing for arbitrary object location, VoluMon constrains the projected centroid to lie within the 2D bounding box. To ensure reasonable $\boldsymbol{t}$ and $\boldsymbol{d}$, the raw outputs of the network are constrained such that $[d_{i,0}, d_{i,1}, d_{i,2}] = \exp(\boldsymbol{\phi}_{i,D})$, and $[u_i, v_i] = \text{sigmoid}(\boldsymbol{\phi}_{i,UV})$, where exp and sigmoid are applied element-wise. We then formulate the per-object translation and shape estimates as

$$\begin{aligned} \boldsymbol{t}_i &= [((u_i w_i + u_{min,i}) - \bar{u}_i) t_{z,i}/f, \\ &\quad ((v_i h_i + v_{min,i}) - \bar{v}_i) t_{z,i}/f, t_{z,i}] \\ \boldsymbol{d}_i &= \alpha_y [d_{i,0}, d_{i,0} + d_{i,1}, d_{i,0} + d_{i,1} + d_{i,2}] + \epsilon_s, \end{aligned} \tag{5.7}$$

where u_i, v_i are the projected centroid coordinates with respect to the image bounding box edges. Additionally, $w_i, h_i, u_{min,i}, v_{min,i}, \bar{u}_i, \bar{v}_i$ are the bounding box width, height, the two coordinates of the lower left corner of the bounding box, and the image center, respectively. α_y is a size scaling parameter and ϵ_s enforces a minimum shape. In an effort to reduce the optimization space due to the potentially ambiguous relationship between rotation and shape, Equation 5.7 also constrains the representation of $\boldsymbol{d}$ to learn a shape where each axis is of increasing size.

From the local patch, we also predict the rotation $\boldsymbol{R}_v$ with respect to the object, i.e., the allocentric rotation. We constrain the raw output of the network such that $[r_{i,0}, r_{i,1}, r_{i,2}, r_{i,3}] = \frac{\boldsymbol{\phi}_{i,R}}{||\boldsymbol{\phi}_{i,R}||_2}$, where the left hand vector is interpreted as $\boldsymbol{R}_v$ in quaternion form. Unlike the egocentric (i.e., camera-centric) rotation of the object, previous works in deeply learned 3D bounding box detection [158, 89] have shown that objects with similar allocentric (i.e., object-centric) rotations have similar visual appearances in the local patch. To recover the egocentric rotation $\boldsymbol{R}_i$ we use a similar transform as described in [89] using $\boldsymbol{t}$ and $\boldsymbol{R}_v$.

5.4 Learning Ellipsoid Prediction from Image-Space Annotations

In this work, rather than assuming a dataset of 3D size and 6D pose annotations, we rely instead on 2D image-space annotations on RGB and depth images, which we posit are easier in practice to obtain. We assume in this work images are obtained from a calibrated stereo pair and that $\boldsymbol{P}_L, \boldsymbol{P}_R \in \mathbb{R}^{3x4}$ are known and constant projection matrices from world coordinates to the image plane (i.e., includes both the intrinsics and extrinsics) for the left and right cameras. We futher assume a dataset of measurements of K labelled objects by $\mathcal{G} = \{\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{B}}_L, \boldsymbol{\mathcal{B}}_R, \boldsymbol{\mathcal{S}}\}$. $\boldsymbol{\mathcal{B}}_L$ and $\boldsymbol{\mathcal{B}}_R$ are the set of bounding box observations from the left images $\boldsymbol{\mathcal{I}}$ and right image, respectively. The set of pixel-wise segmentations $\boldsymbol{\mathcal{S}} = \{\boldsymbol{S}_k\}_{k=0}^{K}$ is composed of individual segmentations $\boldsymbol{S}_k : \Omega \in \mathbb{N}^2 \rightarrow \{0, 1\}$ that denote whether a pixel in the left image is associated with the object in question.

From $\mathcal{G}$, 3D points expected to lie on the surface of the object can also be extracted, provided accurate depth images. Given a stereo dataset, we assume the existence of a pre-processing step that calculates a depth image from a left and right image and uses $\boldsymbol{P}_L$ and $\boldsymbol{S}$ to return the set of J_k 3D points $\boldsymbol{X}_k = \{x_{k,i}\}_{j=0}^{J_k}$ corresponding to the pixels annotated to be upon the object k. The respective sets of points for the objects are then aggregated in $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_k\}_{k=0}^{K}$. We observe that $\boldsymbol{\mathcal{X}}$ could also in practice come from an arbitrary depth sensor aligned with the RGB images.

To optimize the parameters of Equation 5.6, VoluMon leverages two loss functions given different types of image-space annotation: a loss $\mathcal{L}_D$ based on the dual form given two bounding box measurements of the object from a stereo pair (described in Section 5.4.1), and a loss $\mathcal{L}_P$ based on the primal form given observed points from the surface of the object from a segmented depth image (described in Section 5.4.2). Intuitively, VoluMon attempts to learn to predict ellipsoid parameters that are consistent with observed measurements. The overall loss function is

$$\begin{aligned}\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \, & \alpha_P \boldsymbol{\mathcal{L}}_P(\boldsymbol{f}(\Phi(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{B}}_L; \boldsymbol{\beta}), \boldsymbol{\mathcal{B}}_L), \boldsymbol{\mathcal{X}}) \\ & + \alpha_D \boldsymbol{\mathcal{L}}_D(\boldsymbol{f}(\Phi(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{B}}_L; \boldsymbol{\beta}), \boldsymbol{\mathcal{B}}_L), \boldsymbol{\mathcal{B}}_L, \boldsymbol{\mathcal{B}}_R) \\ & + \alpha_S \boldsymbol{\mathcal{L}}_S(\boldsymbol{f}(\Phi(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{B}}_L; \boldsymbol{\beta}), \boldsymbol{\mathcal{B}}_L))\end{aligned} \tag{5.8}$$

where $\alpha_P, \alpha_D, \alpha_S \geq 0$ are all hand-tuned weighting terms. Equation 5.8 also includes a regularization term $\boldsymbol{\mathcal{L}}_S$ that encourages shape predictions of an object to be similar (described in Section 5.4.3).

5.4.1 Bounding Boxes and Dual Ellipsoid Optimization

To develop a loss function that allows ellipsoid prediction to be learned from bounding boxes, we work with the dual ellipsoid representation. 2D bounding box detections from many state-of-the-art object detection pipelines can be interpreted as measurements of axis-aligned bounding planes for objects approximated as 3D ellipsoids, where each edge of a bounding box detection projects into a plane in 3D space which constrains the ellipsoid. As in Chapter 4, to solve for the expected bounding box measurements given some $\boldsymbol{Q}^*$ and camera projection matrix $\boldsymbol{P}$, we first project the dual ellipsoid to a dual-conic $\boldsymbol{C}$ on the image plane:

$$\boldsymbol{C}^* = \boldsymbol{P}\boldsymbol{Q}^*\boldsymbol{P}^T. \tag{5.9}$$

Solving for axis aligned bounding boxes that satisfy the implicit dual conic function in Equation 5.9 yields

$$\begin{aligned}B_{umin}, B_{umax} &= \frac{1}{\boldsymbol{C}^*_{3,3}}[\boldsymbol{C}^*_{1,3} \pm \sqrt{{\boldsymbol{C}^*_{1,3}}^2 - \boldsymbol{C}^*_{1,1}\boldsymbol{C}^*_{3,3}}\,], \\ B_{vmin}, B_{vmax} &= \frac{1}{\boldsymbol{C}^*_{3,3}}[\boldsymbol{C}^*_{2,3} \pm \sqrt{{\boldsymbol{C}^*_{2,3}}^2 - \boldsymbol{C}^*_{2,2}\boldsymbol{C}^*_{3,3}}\,].\end{aligned} \tag{5.10}$$

Using Equations 5.9 and 5.10, we can form a closed-form, differentiable measurement model $\hat{\boldsymbol{B}} = \boldsymbol{h}(q, \boldsymbol{P})$ that maps quadric parameters to an expected 2D bounding box measurement[5]. For any given ground-truth bounding box $\boldsymbol{B}$ and predicted bounding box $\hat{\boldsymbol{B}}$, we define the projection error $e_b(\boldsymbol{B}, \hat{\boldsymbol{B}})$ as the sum of squared differences between the bounding box centroids and dimensions.

A single bounding box measurement is insufficient to fully constrain all parameters of the ellipsoid representation. In the previous chapter, we explored partially resolving the measurement ambiguity by triangulating the quantity of interest from multiple views. In this work, we propose using stereo data at training time to impose projective consistency, as visualized in the bottom panel of Figure 5-1. Let $\boldsymbol{Q}^*$ be defined with respect to the left camera. The final loss function for using bounding boxes from a stereo pair to estimate object parameters is then

$$\begin{aligned}&\mathcal{L}_D(\boldsymbol{q}, \boldsymbol{\mathcal{B}}_L, \boldsymbol{\mathcal{B}}_R) = \\ &\frac{1}{K}\sum_{k=0}^{K}[e_b(\boldsymbol{B}_{L,k}, \boldsymbol{h}(q_k, \boldsymbol{P}_L)) + e_b(\boldsymbol{B}_{R,k}, \boldsymbol{h}(q_k, \boldsymbol{P}_R))],\end{aligned} \tag{5.11}$$

where we have used the known and constant stereo projection matrices $\boldsymbol{P}_L, \boldsymbol{P}_R$. Although the bounding box label in the right image is used at training time to calculate the loss for back-propagation, it is not required at inference time.

5.4.2 3D Points and Primal Ellipsoid Optimization

To specify a loss function for ellipsoid prediction from segmented depth images, we turn to the primal ellipsoid representation. An algebraic error metric on an observed surface point x can be obtained from implicit algebraic definition of a primal ellipsoid. In particular, $x\boldsymbol{Q}x^T$ evaluates to strictly less than zero if x is inside the ellipsoid, strictly greater than zero if x is outside the ellipsoid, and to zero if and only if the point x lies on the surface of the ellipsoid. An algebraic error metric follows directly:

$$e_s(x, q) = e_s(x, \boldsymbol{d}, \boldsymbol{R}, \boldsymbol{t}) = \sqrt{\boldsymbol{d}_0\boldsymbol{d}_1\boldsymbol{d}_2}(x\boldsymbol{Q}x^T)^2. \tag{5.12}$$

[5]We have reproduced Equation 5.10.

Similar to other approaches using superquadrics or quadrics [161, 98], an additional term involving the product of the axes lengths is added to mitigate the bias of fitting to larger primitive sizes.

Taking the average of Equation 5.12 per object over the entire dataset is then:

$$\mathcal{L}_P(\boldsymbol{q},\boldsymbol{\mathcal{X}}) = \sum_{k=0}^{K}\sum_{j=0}^{J_k} \frac{e_s(x_{k,j}, q_k)}{J_k K}. \tag{5.13}$$

The middle panel of Figure 5-1 visualizes an example of an ellipsoid estimate and points extracted from a pixel-wise segmentation paired with a depth image. Although auxiliary information of segmentation and depth image are required during training, they are not required for a feed-forward pass of the network.

5.4.3 Intra-Class Size Consistency Loss

While stereo triangulation provides up to eight bounding edges, the quality of the triangulation can vary with the stereo baseline and size of the bounding box detection. Additionally, although $\mathcal{L}_P$ relies on depths extracted from a stereo pair, severely self-occluded views (such as seeing only the front surface of a box) can introduce significant shape ambiguity. Therefore, to impose additional structure to the optimization, we introduce an intra-class size consistency loss by penalizing the shape variance with constant offset ϵ_v:

$$\begin{aligned}\mathcal{L}_S(\boldsymbol{q}) = \operatorname{var}(\{\boldsymbol{d}_{i,0}\}_{i=0}^{k}) + \operatorname{var}(\{\boldsymbol{d}_{i,1}\}_{i=0}^{k}) \\ + \operatorname{var}(\{\boldsymbol{d}_{i,2}\}_{i=0}^{k}) + \epsilon_v,\end{aligned} \tag{5.14}$$

where the sets of $\boldsymbol{d}$ can be obtained from q. The size consistency loss can be useful for object classes that are expected to have similar dimensional characteristics, such as mass-produced household objects.

5.4.4 Network Training and Post Inference Refinement

VoluMon optimizes network parameters to minimize Equation 5.8 via backpropagation using the PyTorch implementation of Adam [84]. In practice, we do not calculate losses over all K objects in the dataset, but optimize over minibatches. At runtime, we assume that an off-the-shelf object detector such as SSD [104] or YOLO [135] can be used to provide an initial detection, and prediction is simply a feedfoward pass through the network.

While VoluMon implicitly learns a regression of object parameters from measurements, ellipsoid parameters may also be regressed directly. Given a segmented depth image providing a measured pointcloud $\boldsymbol{X}_k$ and bounding box $\boldsymbol{B}_k$ at run-time, the ellipsoid estimate provided by the network can also be used as an initial estimate for further online optimization. Let $\boldsymbol{\phi}_k$ again be free parameters for a given object. We apply Equation 5.13 to directly update $\boldsymbol{\phi}_k$ via gradient descent methods, i.e.,

$$\boldsymbol{\phi}_k^* = \arg\min_{\boldsymbol{\phi}_k} \mathcal{L}_P(\boldsymbol{f}(\boldsymbol{\phi}_k, \boldsymbol{B}_k), \boldsymbol{X}_k). \tag{5.15}$$

Although initial estimates for $\boldsymbol{\phi}_k$ may be obtained from $\boldsymbol{X}_k$, we will show in Section 5.5 that direct online regression can be both slow and inaccurate compared to a learned model. However, using VoluMon's learned model to provide an initial estimate for the regression (similar in spirit to Xiang et al. [186]) enables faster and more accurate estimates on some metrics. In this work, we choose to update only the pose components $\boldsymbol{\phi}_{k,R}, \boldsymbol{\phi}_{k,UV}, \boldsymbol{\phi}_{k,Z}$ when using an initial estimate from VoluMon, keeping $\boldsymbol{\phi}_{k,D}$ fixed. The ellipsoid parameters can be reconstructed as $q_k^* = \boldsymbol{f}(\boldsymbol{\phi}_k^*, \boldsymbol{B}_k)$.

5.5 Simulation Experiments

To evaluate the performance of our proposed approach, we investigated the performance of several variants of VoluMon compared to two baseline methods on simulated data with ground-truth annotations.

5.5.1 Methods

We consider four variants of VoluMon in our experiments. In addition to training on both the bounding box and segmentation annotations (where $\alpha_D = 1, \alpha_P = 1$,

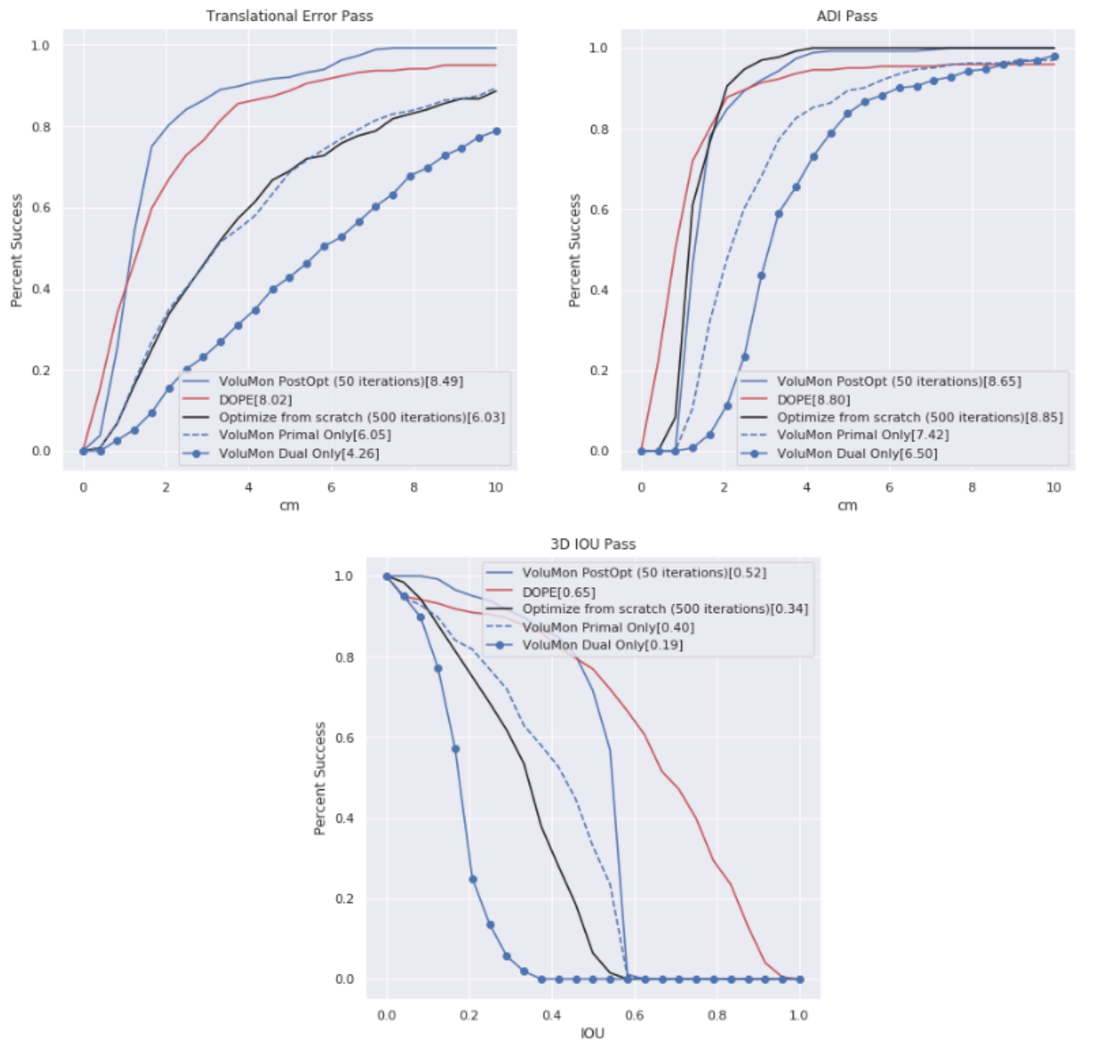


Figure 5-3: The performance of various variants of VoluMon as well as the two baselines are shown for translation error (lower better), ADI (lower better), and 3D bounding box IOU (higher better). Trained on only 2D annotations, *VoluMon Primal Only* (blue dashed) performs similarly to an online optimization approach (black) with respect to translational error without requiring online optimization. After online optimization (solid blue) VoluMon approaches or surpasses the performance of *DOPE* [173] (red), a deep 6D pose estimation approach, with respect to translational error and ADI. All ellipsoid-based methods underperform *DOPE* in the 3D IOU metric, which is not unexpected, given that *DOPE* is a 6D optimization assuming known object dimensions, while the other methods estimate both pose and dimension.

denoted *VoluMon Both*), we evaluate the performance of the network using bounding box annotations only (where $\alpha_D = 1, \alpha_P = 0$, denoted *VoluMon Dual Only*), the network trained using segmented depth images only (where $\alpha_D = 0, \alpha_P = 1$, denoted

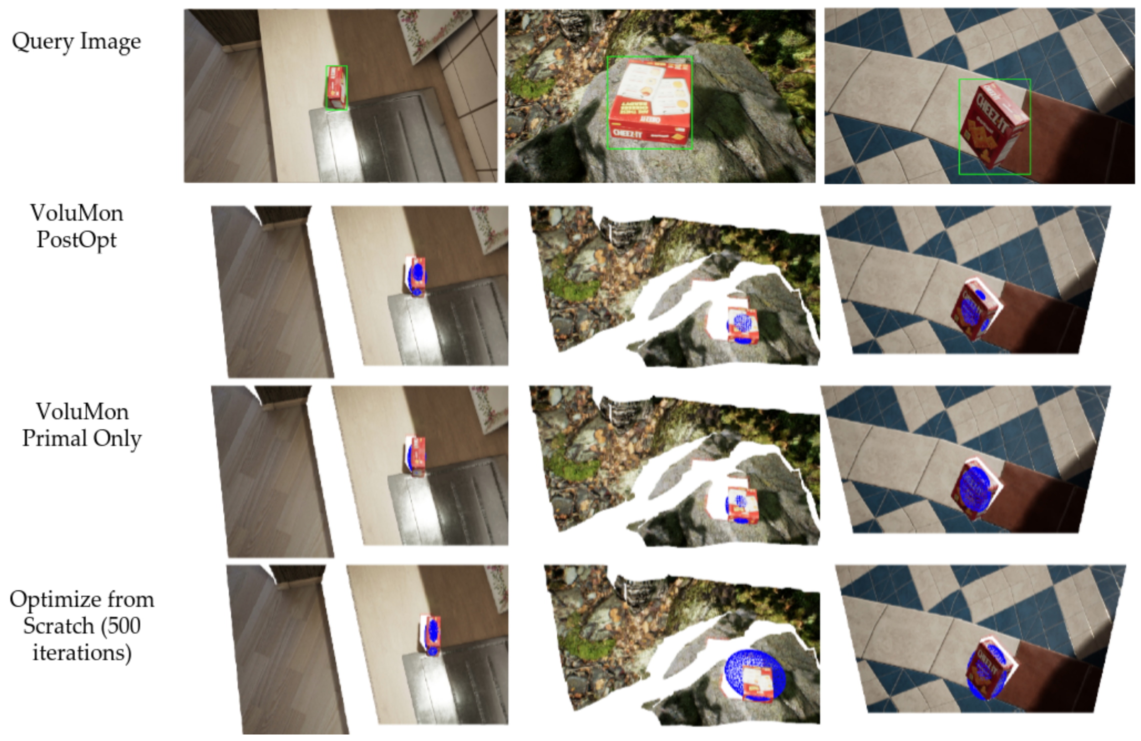


Figure 5-4: Qualitative results for selected methods. Ellipsoid estimates visualized as blue meshes. Although *VoluMon PostOpt* requires less steps of online optimization compared to *Optimize from scratch (100 iterations)*, the initial pose estimates provided by *VoluMon Primal Only* enable *VoluMon PostOpt* to generate more accurate estimates of the ellipsoids. Potentially due to shape ambiguities, it can be difficult to estimate the extents of the object; by keeping the size estimate from *VoluMon Primal Only*, *VoluMon PostOpt* benefits from observations over a dataset to estimate size at run-time.

VoluMon Primal Only). For training and evaluation purposes, we use ground-truth bounding box and segmentation annotations as an input to our approach. To evaluate the performance of VoluMon assuming noisy bounding boxes at run-time, we test a variant of *VoluMon Primal* denoted *VoluMon Noisy Primal* where the test time input data is obtained from MaskR-CNN trained on the same train test spilt. To condition the network for noisy bounding boxes at run-time, we add random noise to ground-truth bounding box input to the network during training, and keep the original bounding box annotation for loss calculation.[6] To ensure only a single detection per

[6]Adding random noise to the ground-truth bounding box annotations on the fly means that the network has seen noisy, imperfect bounding boxes at training time. Using the ground-truth bounding box annotation to calculate the loss ensures that the best possible label is used to fit the estimated bounding box projections.

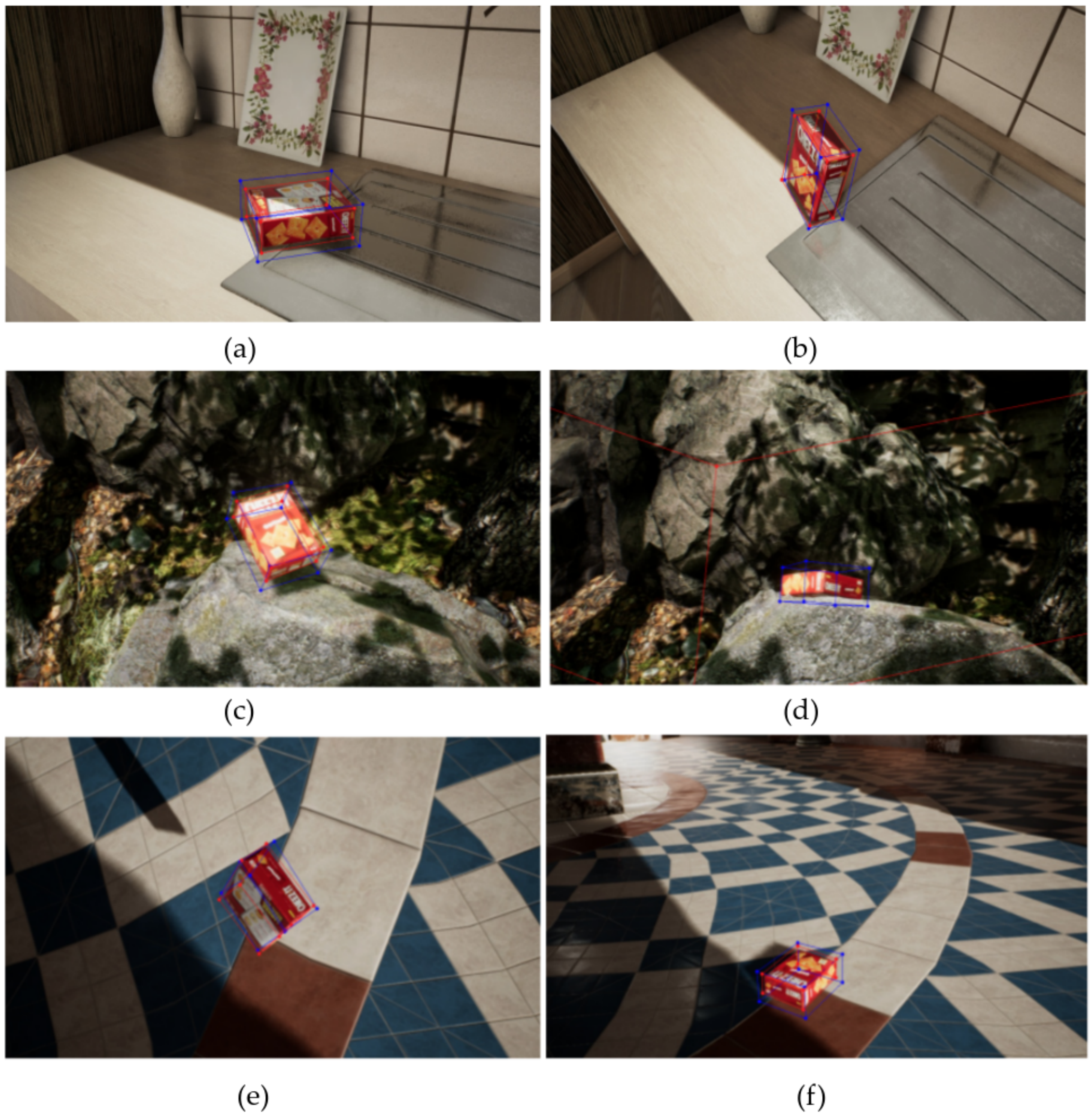


Figure 5-5: Ellipsoid estimates from *VoluMon Primal Only* visualized as blue bounding boxes, and the estimates from DOPE visualized as red bounding boxes using the ground-truth object size. VoluMon constrains the projected centroid of the object to fall within the 2D object bounding box, helping to avoid some failure cases of DOPE, e.g., top row, right column. Additionally, while VoluMon tends to slightly overestimate the size of the object, many of the estimates appear qualitatively reasonable.

image, we hand-tune for the minimum probability to accept a detection from the object detector on the test set and keep the highest probability detection. We leave further study of the interaction between object detector and VoluMon for future work. We also test a variant of VoluMon (denoted *VoluMon PostOpt*) where the pose estimate of the *VoluMon Primal Only* network is further refined for 50 optimization steps as described in Section 5.4.4.

Additionally, we consider a regression only technique (denoted *Optimization from scratch*), which optimizes Equation 5.15 without learning, but for both shape and pose parameters. Rather than use VoluMon to set an initial estimate, we use the average depth of 100 points sampled from the observed object points as the initial depth, and assume the projected centroid of the object is in the center of the bounding box detection. The extents of the segmented pointcloud with respect to the frame of reference of the camera set the initial shape parameters, and the initial rotation estimate is set to identity. If a valid initial size estimate cannot be obtained, potentially due to lack of diversity in the points measured or a violation of miniminum size constraints, an initial size estimate is approximately set to a sphere of 3 cm in diameter (the minimum object size). We consider two variants of the regression only technique with different numbers of optimization iterations, i.e., *Optimize from scratch (500 iterations)* and *Optimize from scratch (50).* Both *VoluMon PostOpt* and *Optimize from scratch* approaches sample 300 points once at run-time for the primal loss calculation and use a learning rate of 0.05. Finally, we compare our approach to a deep 6D object pose learning method [173] (*DOPE*) that requires the training data to be annotated with object dimensions and projected 3D bounding box vertices. *DOPE* does not require depth images at training time and the results are not further refined online.

All variants of VoluMon as well as the pure online optimization approach are implemented in Python using PyTorch [125]. For VoluMon, we set $\alpha_S = 100, \alpha_y = 10.0, \epsilon_s = 3.0, \alpha_\delta = 150, \epsilon_v = 0.1$. We train with a batch size of 50 and a learning rate of 0.00001 for 14000 epochs, resetting the optimizer state halfway through training. We use open-source code released by Tremblay et al. [173] to train *DOPE* for 240 epochs, reducing the batch size at train time to 10 to optimize on lower cost GPUs.

5.5.2 Dataset and Metrics

We test on a subset of the Falling Things Dataset [172], which provides groundtruth bounding boxes, object pose, and geometry. The Falling Things Dataset is attractive due to access to ground-truth annotations from simulation, while still being photo-realistic. The dataset features a wide variety of household objects from the YCB dataset [22]. Although for these experiments we consider only a single object, in later experiments we expand to a wider variety of objects. In comparison to datasets for autonomous cars, the Falling Things Dataset features objects of arbitrary rotation (rather than assuming objects have only yaw).

We focus on the cracker box object data shown in Figure 5-4, which features three different environment categories (kitchen, kite, and temple) with four variants in each yielding a total of twelve different environments with one hundred data points per camera. Each training sample includes the left RGB image, ground-truth depth image, groundtruth 2D bounding boxes for the left and right images, and segmentation images for all images. Each image contains a single object instance. For VoluMon, we filter out data points with bounding boxes extending past the image boundaries, and withhold one environment from each category `kitchen_0, kite_0, temple_0` for the test set, resulting in 1066 datapoints in *Cracker Train* and 264 datapoints in *Cracker Test*. For DOPE, we allow the network to train on all available data and test on *Cracker Test*. While training VoluMon, we use built-in PyTorch functions for random data augmentation, including adding random color jitter to the hue and saturation and random erasing. In practice, we randomly sample 4 points from the extracted surface points every epoch to calculate the primal loss.

We report translational prediction performance by comparing the true centroid of the object to the predicted centroid. Additionally, given true mesh models, we compare the distance between the mesh vertices $\{m_0 ... m_M\}$ and points on the surface of the prediction $\{p_0 ... p_N\}$ [64], i.e.,

$$ADI = \frac{1}{N} \sum_{j=0}^{N} \min_{i \in M} ||m_j - p_i||, \tag{5.16}$$

	Estimated Parameters	Train Time Requirements	Inference Time Requirements	Timing Mean/STD (ms)	Translation AUC	ADI AUC	3D IOU	Translation AUC (NF)	ADI AUC (NF)	3D IOU (NF)
VoluMon PostOpt (50 Iterations)		Left bounding box Left RGB image Left segmentation Depth image	Left bounding box Left segmentation Left RGB image Depth image	15/9 for network 341/3 for optimization + detection time (variable)	**8.49**	8.65	0.52			
VoluMon Dual Only		Left bounding box Right bounding box Left RGB image			4.26	6.50	0.19			
VoluMon Primal Only		Left bounding box Left RGB image Left segmentation Depth image			6.05	7.42	0.40			
VoluMon Both		Left bounding box Right bounding box Left RGB image Left segmentation Depth image			5.80	7.60	0.31			
VoluMon Both Noisy	$\boldsymbol{d}, \boldsymbol{R}, \boldsymbol{t}$	Left bounding box Right bounding box Left RGB image Left segmentation Depth image	Left bounding box Left RGB Image	15/9 for network + detection time (variable)	4.96	7.05	0.28	4.94	7.02	0.27
DOPE	$\boldsymbol{R}, \boldsymbol{t}$	Projected cuboid corners Bounding box size RGB image	RGB image	220/12	8.02	8.80	**0.65**	**6.71**	**7.37**	**0.55**
Optimize from scratch (500 iterations)	$\boldsymbol{d}, \boldsymbol{R}, \boldsymbol{t}$	None	Left bounding box Left segmentation Left RGB image Depth image	3346/17 + detection time (variable)	6.03	**8.85**	0.34			
Optimize from scratch (50 iterations)				341/3 + detection time (variable)	6.55	8.03	0.29			

Table 5.1: Performance metrics over the various methods, with best performance bolded and the second best underlined. (NF) indicates when aggregate metrics consider instances when objects are not detected as failure to pass, where for VoluMon we impose a minimum detection probability from the auxiliary object detector. Timing results are collected using a GTX 1070Ti. We expect all variants of VoluMon to have similar run time for a single feed-forward pass of the network, and for the same number of optimization steps to have similar run times, reporting accordingly. *VoluMon* and *Optimize from scratch* timing results do not include detection and segmentation; the object detector used to generate test time detections for *VoluMon Both Noisy* requires roughly 200 ms per image, but the speed of the object detector can vary depending on the architecture. Timing results for *DOPE* include detection and PnP solving.

where p_i are interpolated points from the surface of the prediction for the ellipsoid-based methods, and true mesh model transformed using the estimated parameters for the supervised method. Although this metric is generally used to compare two known mesh models [186, 64], we apply it to measure surface fitting performance. Because *DOPE* is both a detection and estimation algorithm, we report pass rates where missed detections are removed entirely from the total number of samples considered, and pass rates where the missed detections are penalized (denoted NF). We also report the 3D IOU with groundtruth 3D bounding boxes.

5.5.3 Simulation Experimental Results

We show the pass rate metrics for selected variants of each algorithm in Figure 5-3, and report area under the curve (AUC) and timing results in Table 5.1. Qualitative samples are depicted in Figures 5-4 and 5-5. The AUC metric is calculated using a naive rectangular integral approximation. On simulated data, *VoluMon Primal Only* and *VoluMon Both* outperform *VoluMon Dual* with respect to AUC for translation and ADI. This matches our intuition that while using only two bounding box measurements may struggle to overcome shape and rotation ambiguity for certain objects, as reflected in the relatively poor ADI performance. However, it is important to note that *VoluMon Dual* requires only bounding box annotations from a stereo pair for training and a single camera at inference time, and yet achieves about 67% the ADI AUC of the highest performing method in the table. We additionally observe that training on both the primal and the dual objectives does not yield noticeable performance benefits when using groundtruth depth images, motivating using *Primal Only* in the post optimization experiments.

Our results also show that a single forward pass through the network of *VoluMon Primal Only* achieves a higher 3D IOU AUC score as compared to *Optimize from scratch (500 iterations)*, despite taking less than 1% of the time. Although all network-only VoluMon variants underperform the pure optimization methods in terms of ADI, because the ADI metric is the average over the minimum distance from every true model point to the approximated ellipsoid, it can fail to capture certain types of shape estimation errors as suggested by the 3D IOU results. As seen in Fig. 5-3, the translation performance of *VoluMon Primal Only* and *Optimize from scratch (500)* are also relatively similar. After applying the same number of further post optimization to the estimates from *VoluMon Primal*, *VoluMon PostOpt* outperforms *Optimize from scratch (500 iterations)* on translation AUC by over 40% and on 3D IOU AUC by over 50%, indicating that the estimates from VoluMon can be useful initial estimates for downstream algorithms.

Additionally, assuming that pointcloud data is available, *VoluMon PostOpt* outperforms *DOPE* with respect to translation AUC, and approaches the ADI performance with an ADI AUC that is 98% of *DOPE*'s. We observe that although one of the contributions of Tremblay et al. [173] is to use large quantities of simulation data, the performance on our much smaller datasets still enables an approximately 60%

pass rate for both the 2 centimeter threshold for translation and ADI. By leveraging the ellipsoid representation and object consistency properties, VoluMon does not require 3D annotations or *a priori* geometric models, and can still be further refined online if additional computation time and segmented depth information is available.

5.6 MIT Desk Experiments

To test the performance of VoluMon on real-world objects, we collected additional real-world datasets for four objects (a mug, a toy bus, an orange, and a bowl) using a ZED Mini stereo sensor. An initial set of annotations was generated using Mask R-CNN pre-trained on COCO, with inaccurate annotations removed in a manual post-processing step. The networks were trained using bounding box annotations only (i.e., *VoluMon Dual*) on approximately 500 images per object, with a batch size of 25, and a learning rate of 0.0001 for the mug, orange, and bus, and 0.00001 for the bowl. We allow a maximum centroid disparity of 300, a minimum size of 3 cm and set a size scaling factor of 10 cm. As seen in Figure 5-6, without requiring detailed 3D models or 3D annotations, for some objects VoluMon produces qualitatively reasonable pose and size approximations on withheld evaluation data. We observed greater optimization instability on real data, and report results from the stable regimes of training. We also found that in practice, using the bounding box annotations only on real data generally outperformed other VoluMon variants, and enabled the most reasonable performance for the majority of objects investigated. This may be due to the noisier nature of the stereo pointcloud, seen in Figure 5-7c.

The qualitative performance on real-world data demonstrates several key benefits of our approach. Most importantly, VoluMon can learn qualitatively reasonable estimates of 3D object properties given no 3D annotations. Additionally, because only 2D annotations are required, a 2D object detector also trained on 2D annotations can be used to generate candidate labels. Although we may expect annotations from a network such as Mask R-CNN to be less accurate than those generated by a manual annotation pipeline, our results indicate reasonable qualitative performance. Finally,

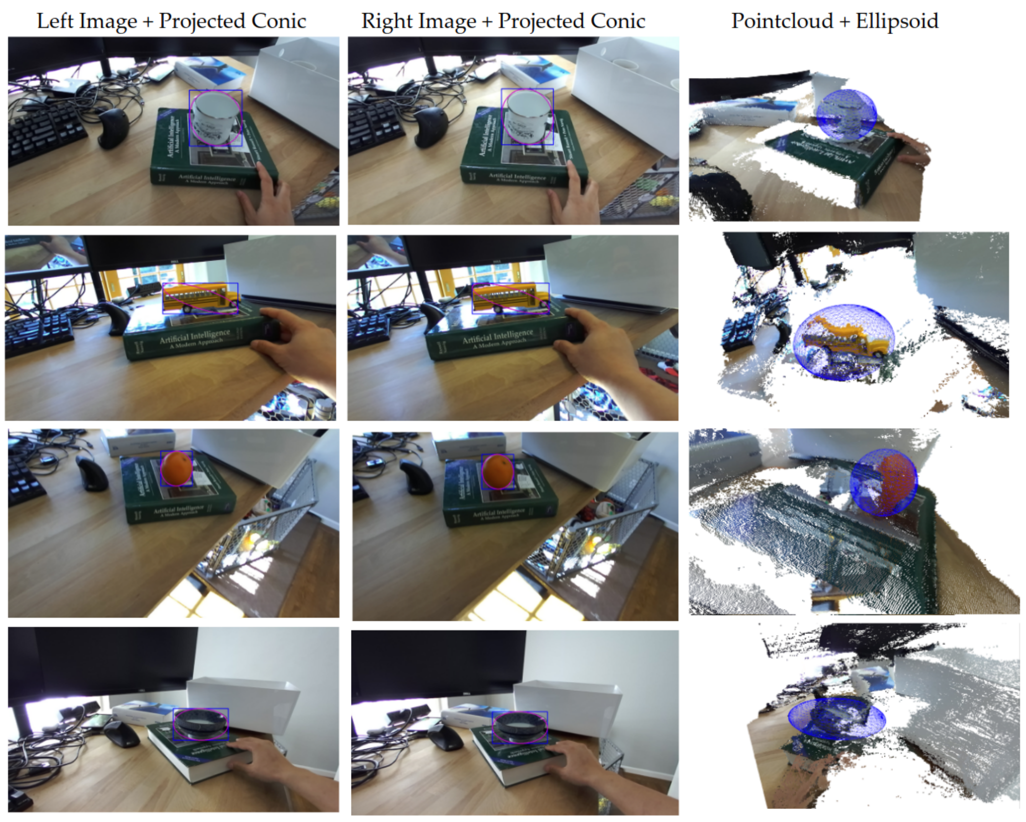


Figure 5-6: Qualitative results from the MIT Desk Dataset. The first and second columns show projected conics (pink) and bounding boxes (blue) for the left and right images respectively; the second column visualizes the primal ellipsoid (blue mesh). While in this case bounding boxes from both the left and right sensor are used at train time, VoluMon predicts object geometry and pose using only the left image and bounding box input at run-time.

by using bounding boxes from a stereo pair, VoluMon can optimize multi-view projective consistency without requiring a separate process to estimate the pose of the sensor during data collection.

5.7 Experiments in End-to-end Prediction

Although the system design of parallel object detection and estimation networks has the benefit of the ability to substitute different bounding box detection algorithms, running two separate neural networks may not be computationally feasible on all systems. Additionally, the system as outlined in Section 5.3 does not consider class probabilities of detections, and instead assumes a single network is trained per object

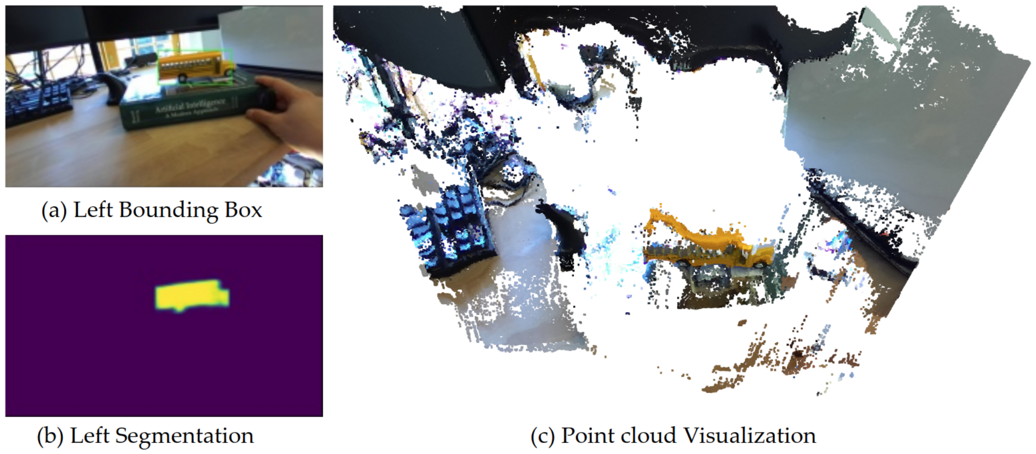


Figure 5-7: To collect image space annotations on the real-world datasets, we leverage a pretrained network to annotate bounding boxes (a) and object segmentations (b) to use with the estimated pointcloud (c), reducing by-hand annotation burden.

and that the classes of detections are accurate. In this section, we briefly discuss preliminary results in consolidating detection and object estimation for an end-to-end ellipsoid multi-class estimation pipeline.

5.7.1 End-to-end Detection and 3D Object Estimation

To enable end-to-end detection and ellipsoid estimation, we propose integrating our weakly supervised ellipsoidal prediction framework with Faster-R-CNN [136], a popular object detection framework that was previously used in parallel to provide detections. We take the RoI feature vector used to predict class probabilities and 2D bounding box regression parameters to predict VoluMon free parameters. We implement an architecture similar to the one visualized in Figure 5-2 except rather than using MobileNet for feature extraction over the entire image, the RoI features from Faster-R-CNN are directly passed to both the global and local branches. We remove the average pooling layer and change the first fully connected layer to have an input size of 1024. For each RoI, the newly added ellipsoid parameter prediction branch outputs free parameters for ellipsoid prediction, denoted $\boldsymbol{\phi} \in \mathbb{R}^{N \times 10}$, where N is the number of possible classes and each row of the matrix represents a class-specific parameter estimate and is denoted $\boldsymbol{\phi}_i$. A high-level diagram is given in Figure 5-8.

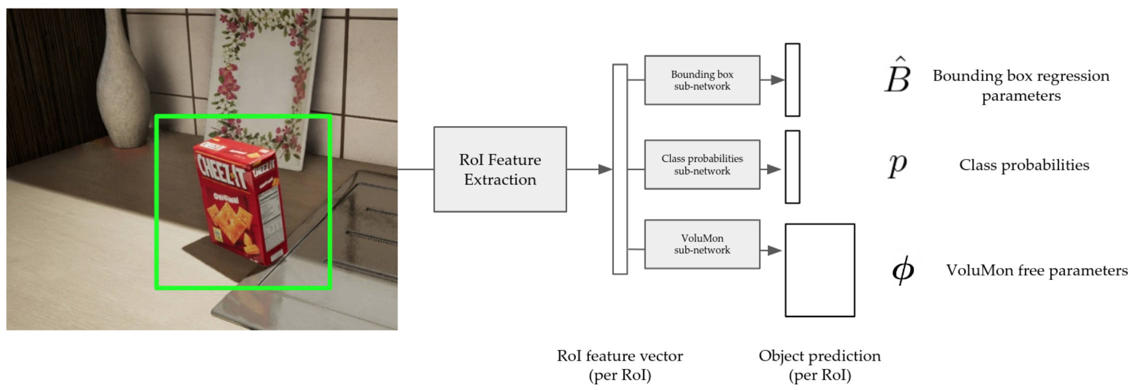

RoI Feature Extraction
Bounding box sub-network
Class probabilities sub-network
VoluMon sub-network
$\hat{B}$ Bounding box regression parameters
p Class probabilities
ϕ VoluMon free parameters
RoI feature vector (per RoI)
Object prediction (per RoI)

Figure 5-8: High-level diagram for end-to-end object detection and prediction network. VoluMon Multi-Class R-CNN is an augmentation of Faster R-CNN [136]. From a given RoI feature vector, Faster R-CNN estimates the parameters for regressing an estimated bounding box $\hat{\boldsymbol{B}}$ and the class probabilities $\boldsymbol{p}$. VoluMon Multi-Class R-CNN appends an additional sub-network to predict a matrix of free parameters from which ellipsoid estimates can be re-constructed. This extension allows for the network to concurrently detect objects and predict their 2D image-space bounding boxes, 3D pose and size, and class probabilities.

The benefits of combining the systems is twofold. First, the overall computational load can be potentially be reduced as the object detection and ellipsoid estimation share a feature extractor. Second, because the network is optimizing both the class probabilities as well as the object properties, we can investigate the utility of using evolving estimates of class labels during ellipsoid estimation.[7] Provided a method of obtaining a single estimate of the free parameters $\hat{\boldsymbol{\phi}}$, we can then project the expected free parameters into our valid space of R, D, t using the mapping function $\boldsymbol{f}$:

$$\boldsymbol{R}, \boldsymbol{D}, \boldsymbol{t} = \boldsymbol{f}(\hat{\boldsymbol{\phi}}, \bar{\boldsymbol{\mathcal{B}}}), \tag{5.17}$$

where $\bar{\boldsymbol{\mathcal{B}}}$ is now the estimated bounding box, rather than the ground-truth label.

We consider two methods of obtaining an estimate of the free parameters. First, we can consider simply indexing the column of $\boldsymbol{\phi}$ corresponding to the ground-truth label $\boldsymbol{\phi}_{gt}$, i.e., $\hat{\boldsymbol{\phi}} = \boldsymbol{\phi}_{gt}$, where $\hat{\boldsymbol{\phi}}$. As an alternative method, we propose calculating

[7]It is important to distinguish here the difference between label uncertainty in the dataset versus of the estimator. *VoluMon R-CNN Multi-Class* still requires a ground-truth dataset with correct labels, but utilizes its own *estimated* class probabilities during optimization.

the expectation over free parameter values as

$$\hat{\boldsymbol{\phi}} = \mathbb{E}[\boldsymbol{\phi}] = \sum_{1}^{N} p_i \boldsymbol{\phi}_i, \tag{5.18}$$

where $\hat{\boldsymbol{\phi}} \in \mathbb{R}^{10}$, and p_i is the *predicted* probability that the RoI is of class i. We observe that the described expectation is over the free parameter space, rather than the constrained pose and shape space. While Equation 5.18 is simple to implement, it is not a proper method of rotation averaging. Instead, the expected values of free parameters are projected onto the space of valid rotations by Equation 5.17. We leave further analysis of expectation calculation to future work. In the following experiments, we will refer to the model *VoluMon R-CNN Multi-Class Expectation* as the method that substitutes 5.17 into 5.8, and *VoluMon R-CNN Multi-Class* as the method where the column of $\boldsymbol{\phi}$ is selected based on the ground truth annotation during the training stage, similar to the bounding box regression loss function.

5.7.2 Training and Testing

The network was trained end-to-end, by adding the VoluMon loss terms to the existing Faster-R-CNN loss terms which include a classification loss, a bounding box regularization loss, and a loss indicating whether the RoI contains an object. In all variants, loss terms were calculated only for those bounding box detections that have been associated to ground-truth bounding box annotations, with the exception of the intra-class variance loss, which was calculated over all object predictions and summed over all the classes. After applying non-maximal suppression using the 2D detections, each object detection returned a tuple containing a 2D bounding box, a 3D ellipsoid estimate, a class and the probability of the object being of the class. The ellipsoid estimates were then reconstructed from to the column of $\boldsymbol{\phi}$ used corresponding to the class of the prediction.

To evaluate the proposed extension, we again assumed a dataset of bounding box annotations on the left and right images of a stereo pair. We included seven objects:

a cracker box, a mustard bottle, a jello box, a banana, a tuna can, a soup can, and a sugar box; we trained on 7688 training examples in total. The network was implemented by modifying the open-source implementation of Faster-RCNN with a MobileNet backbone provided by PyTorch [125]. We trained using only the dual loss (scaled by a factor of 0.00001) and intra-class shape variance (scaled by a factor of 10.0). The network was trained using stochastic gradient descent for 750 epochs total with a training parameter reset at 350 epochs, a learning rate of 0.001, momentum of 0.9, and a burn in period of 5 epochs where only the original Faster-R-CNN loss terms were used. We used a batch size of 32, enabled by spreading computation over 4 GPUs. In practice, we also observed greater sensitivity to the parameters governing minimum object size and object size scaling when learning to estimate the sizes for several object classes. Fitting performance improved when the minimum size and size scaling parameter were set so that the initial network estimates were smaller than most of the objects in the dataset. The minimum shape was therefore accordingly set to $\epsilon_s = 1$ cm and $\alpha_y = 10$.

5.7.3 Experimental Results

Our results showed relatively little difference between *VoluMon R-CNN Multi-Class Expectation* and *VoluMon R-CNN Multi-Class* in terms of our metrics of interest, but the integration of VoluMon into R-CNN leads to computational benefits. The translation, ADI, and 3D IOU metrics are shown in Figure 5-10 and qualitative examples on the withheld test data are shown in Figure 5-9. Although the integrated detection and estimation versions of VoluMon output many object detection and estimate pairs after non-maximal suppression, we exploited the fact that each image had only one object of interest and took the highest probability object detected in the image to compare to the single ground-truth object. The AUC for the version that did not take the expectation over the free parameters was higher for translation and ADI compared to the method which used the expectation. This may be because the detection task was relatively simple; each image had only one un-occluded instance of the object. Using the *VoluMon R-CNN Multi-Class Expectation* for classes with

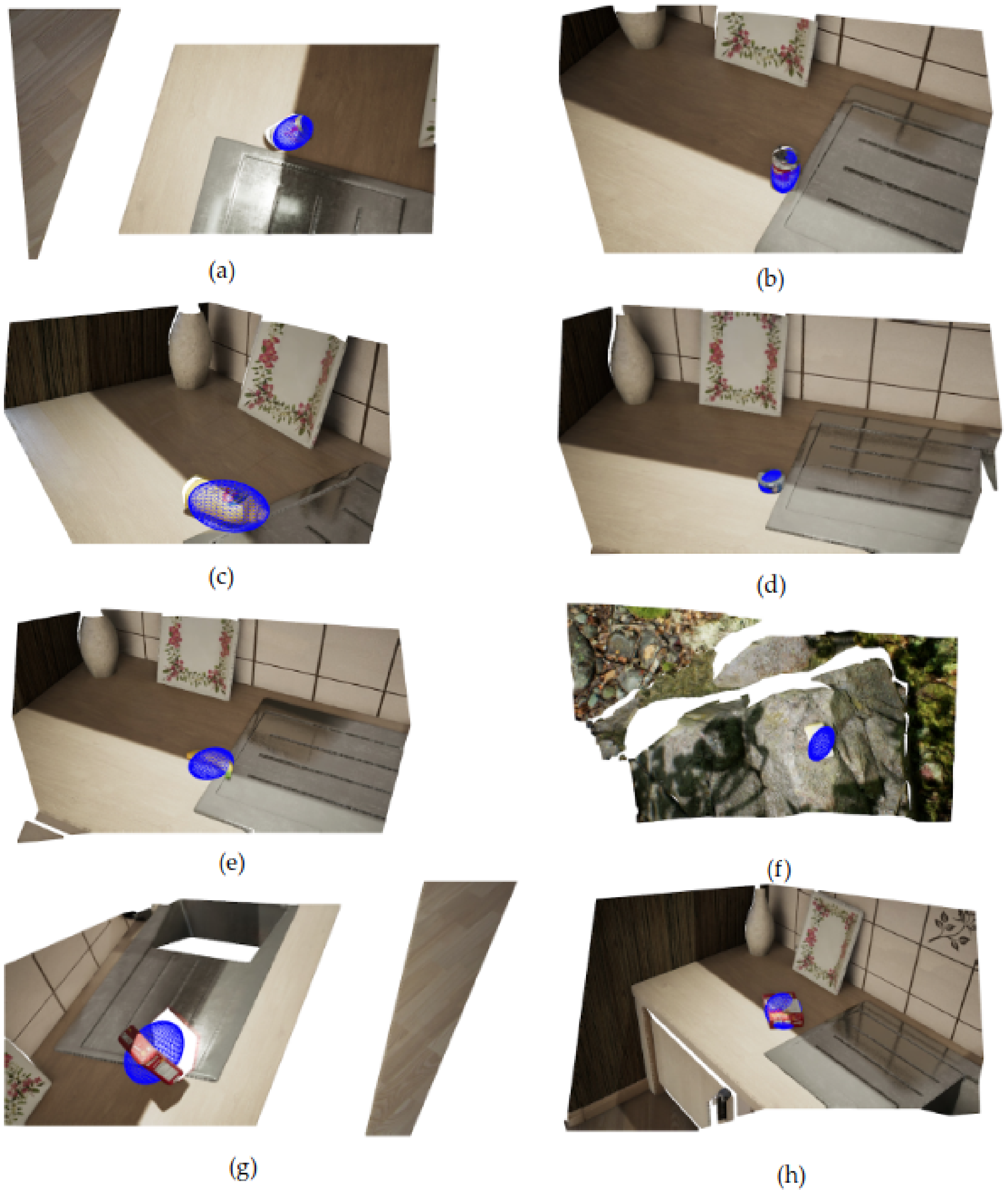


Figure 5-9: Qualitative results using *VoluMon R-CNN Multi-Class* for detection and estimation. We trained and tested VoluMon R-CNN variants on a mustard bottle (a, c), soup can (b), tuna can (d), banana (e), sugar box (f), cracker box (g, h), and jello box (not pictured). The shape diversity in this dataset was larger than the cracker box only dataset used to train VoluMon. We observe in particular that the translation and shape estimates look largely reasonable, but the estimates for rotation are less accurate. This may be due to the lack of use of the primal, or from taking the expectation over free parameters instead of in the rotation space.

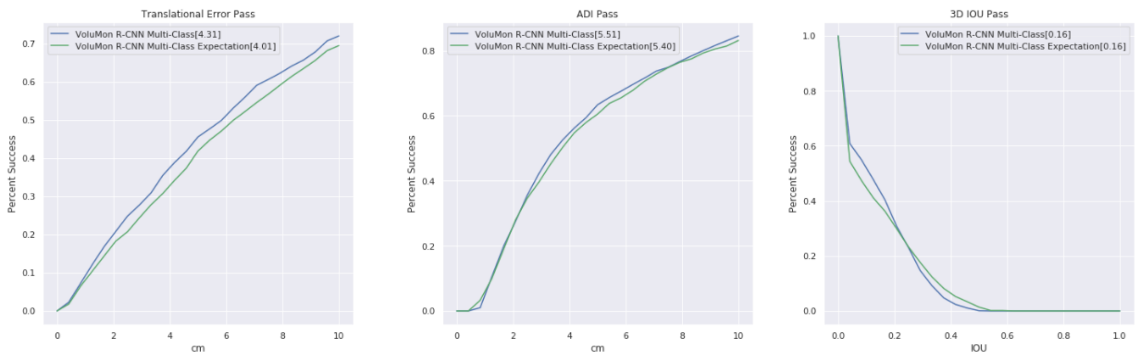


Figure 5-10: The performance of VoluMon Multi-Class variants are shown for translation error (lower better), ADI (lower better), and 3D bounding box IOU (higher better). Both methods are trained on only 2D annotations and perform similarly over all metrics pictured.

greater visual class-ambiguity (i.e., different types of cans) or occlusions may outperform the baseline version, but we leave this for future work. In terms of timing, to perform detection and estimation for seven objects, a single query to our VoluMon R-CNN model took an average of 84 ms, with a standard deviation of 20ms. Although slower than a single query through VoluMon (which takes an average of 15 ms), the proposed end-to-end approach both detects and estimates the 3D parameters of 7 object classes, while VoluMon requires a parallel object detection process.

5.8 Object Estimation for Planning

One of the main motivations in this thesis for the development of deep 3D object estimation with a lesser annotation burden is to make online object-level estimation more accessible for use in downstream processes such as the planning approach described in Chapter 3. In this section, we investigate the suitability of VoluMon estimates to inform online planning. We first describe how we train VoluMon to predict the pose and size of doorways in a simulated office environment (Section 5.8.1). We then describe how estimates from VoluMon can be used to create and update simple 3D object-level maps (Section 5.8.2). Finally, we describe a system that uses dense local geometric maps in addition to object-level maps generated using VoluMon estimates to predict a sampling distribution, and show that our proposed method results in more reasonable trajectories than a baseline method that considers geometry only (Section 5.8.3).

5.8.1 Data Collection and Training

We trained the *Dual-Only* variant of VoluMon to predict the pose and size of the doorways in the simulated office environment. Notably, the primal loss is not valid in the doorway detection application as the doorway is defined by negative space rather than measurable surface points. The simulator was built using the Unity rendering engine [74], using an office environment prefab [1] scaled to enable comfortable navigation. To collect the training dataset, a stereo pair with a 2.0 meter baseline was manually guided to observe doors from many diverse views, and ground-truth associated bounding box detections provided by the simulator. Door detections with bounding boxes less than 2000 square pixels, or less than 50 pixels from the edge of the image boundary were filtered out of the dataset. The *VoluMon Dual* variant was trained on 1178 doorway annotations. We observe that due to simulation artifacts, highly occluded doorways were still detected as in Figure 5-11 both at training time and test time. Instead of using the pseudo-disparity formulation to project the free parameters to translation, the translation parameterization was changed to an absolute parameterization, allowing for translation up to 50 meters. We set the shape parameters as $\epsilon_s = 0.1m$, $\alpha_y = 0.5$. The network was trained with a learning rate 0.00001 and batch size of 10. Qualitative visualizations of performance on the training data are shown in Figures 5-12 and 5-13.

5.8.2 Building Maps with VoluMon

To generate a map of objects, we implemented a very simple mapping approach. Qualitative examples of various points in a single mapping run are shown in Figure 5-14. Unlike the subset of the Falling Things Dataset considered, or the MIT Desk Dataset, there may be several of the same object of interest in the same RGB image (i.e., if two doors are in view at the same time, as in Figure 5-11c). To associate 3D

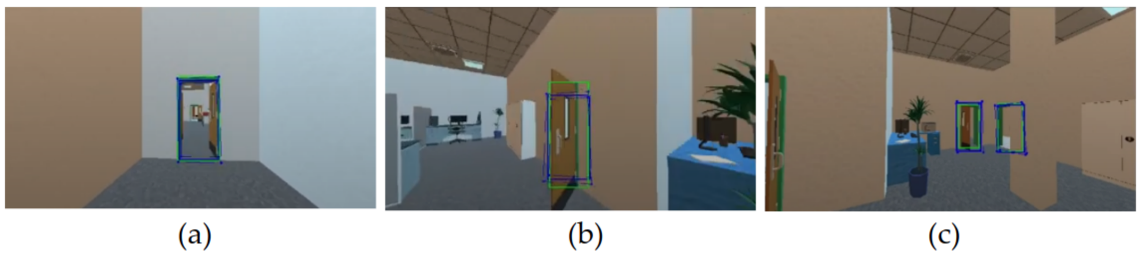


Figure 5-11: Examples of VoluMon estimates in office simulation under autonomous navigation. Using the ground-truth detections from the simulator (green) and RGB images, VoluMon predicts 3D ellipsoids visualized as 3D bounding boxes (blue). The system is monocular only. The system can also handle two bounding box detections as in (c) by building two separate bounding box and RGB queries.

object detections, the mapping system performed a simple centroid-based distance check to greedily associate measurements to existing objects in the database, or decide to instantiate a new entry. New 3D object measurements overwrote previous estimates in the database. Although we tested in the same environment the training data was collected in, images were generated by a vehicle navigating autonomously at test time, rather than a camera being manually moved. A qualitative example of the map built from VoluMon estimates is shown in Figure 5-14.

5.8.3 Integration Experiment

We integrated the VoluMon-generated maps with the planning approach described in Chapter 3 to generate high-level global trajectories. 3D ellipsoids were projected to conics onto a horizontal 2D plane using an orthographic projection, and the conics were then discretized using the same discretization as the occupancy map[8]. Dense local maps were generated from the depth image[9] using a 3D sliding local window approach before being converted to 2D occupancy maps, and the high level trajectory found by LSD was used to guide a motion-primitive based planning method. Both the geometric mapping method and motion primitive selection process were tuned versions of the algorithms presented by Ryll et al. [149]. To predict sampling distri-

[8]We observe that this projection is a more proper projection than implemented in the real-world experiments described in Chapter 3.

[9]For the purposes of obstacle avoidance, a depth image at a lower resolution than the RGB image proved suitable.

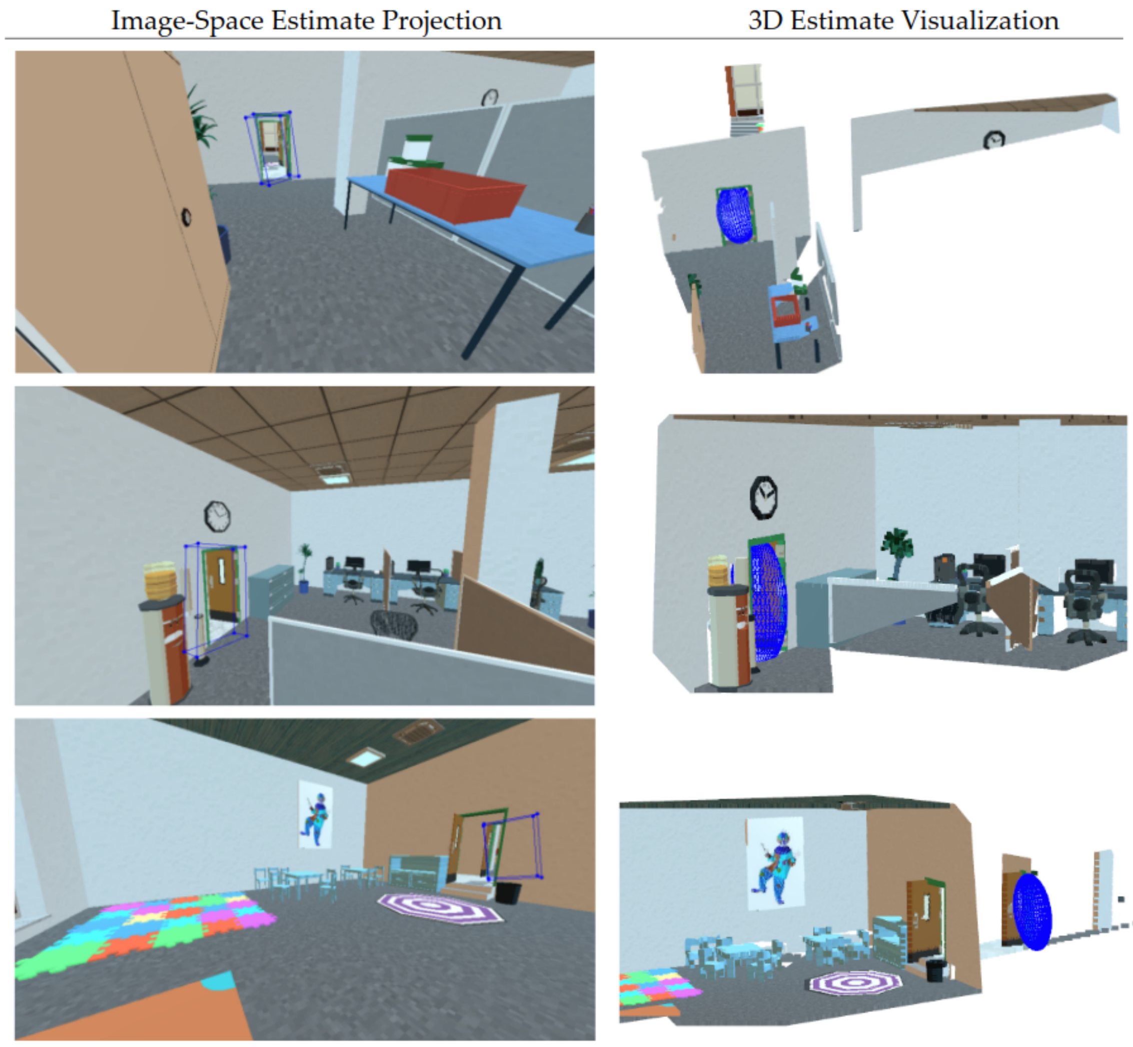
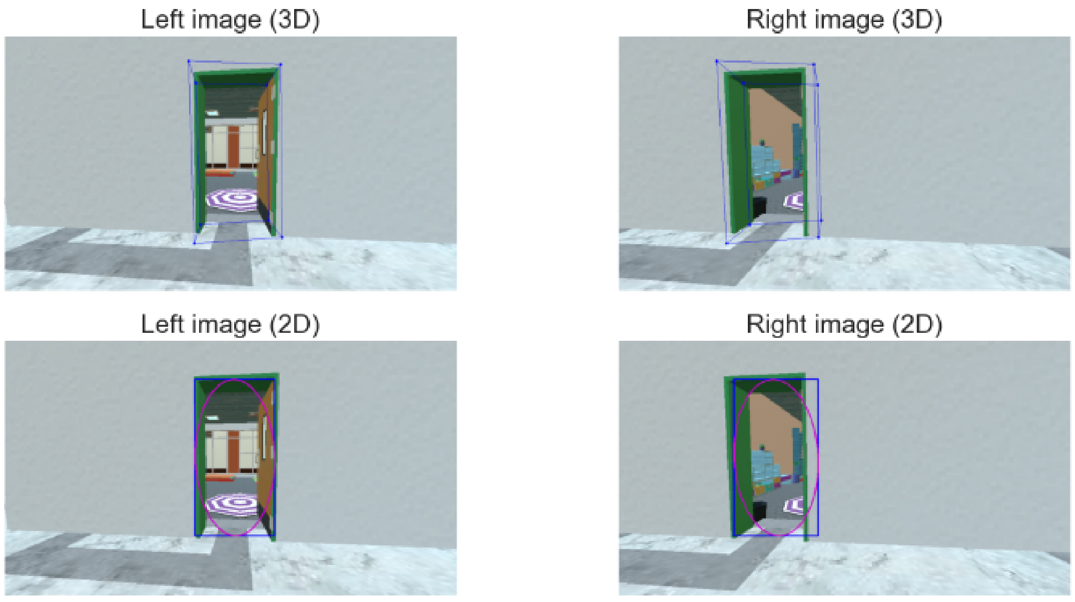
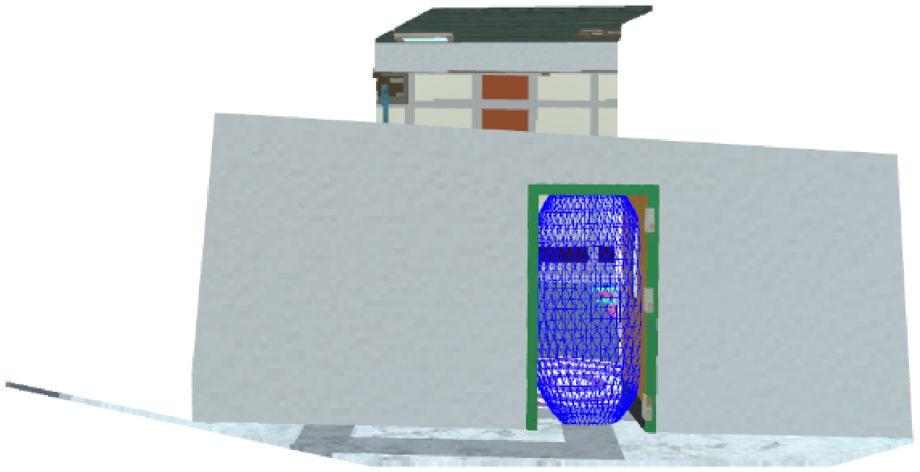


Figure 5-12: Qualitative examples of performance on the office training dataset. Given a 2D bounding box and RGB image, VoluMon predicts 3D ellipsoids, visualized as a 3D bounding box (left column) and as a mesh in the pointcloud (right column). The examples show different viewing angles and distances to the doors; we observe that due to the bounding boxes coming from simulation, even highly occluded doors as in the bottom corner can be estimated.



(a)



(b)

Figure 5-13: 2D and 3D visualization of VoluMon prediction on office training data. For a single datapoint in the training dataset, we show (a, top row) the 2D projection of the ellipsoid as a 3D bounding box, (a, bottom row) the projected conic in magenta and induced 2D bounding box projection in blue, and (b) the ellipsoid plotted as a blue mesh in the pointcloud. Although a doorway is defined by negative space, *VoluMon Dual-Only* is able to estimate a qualitatively reasonable volume.

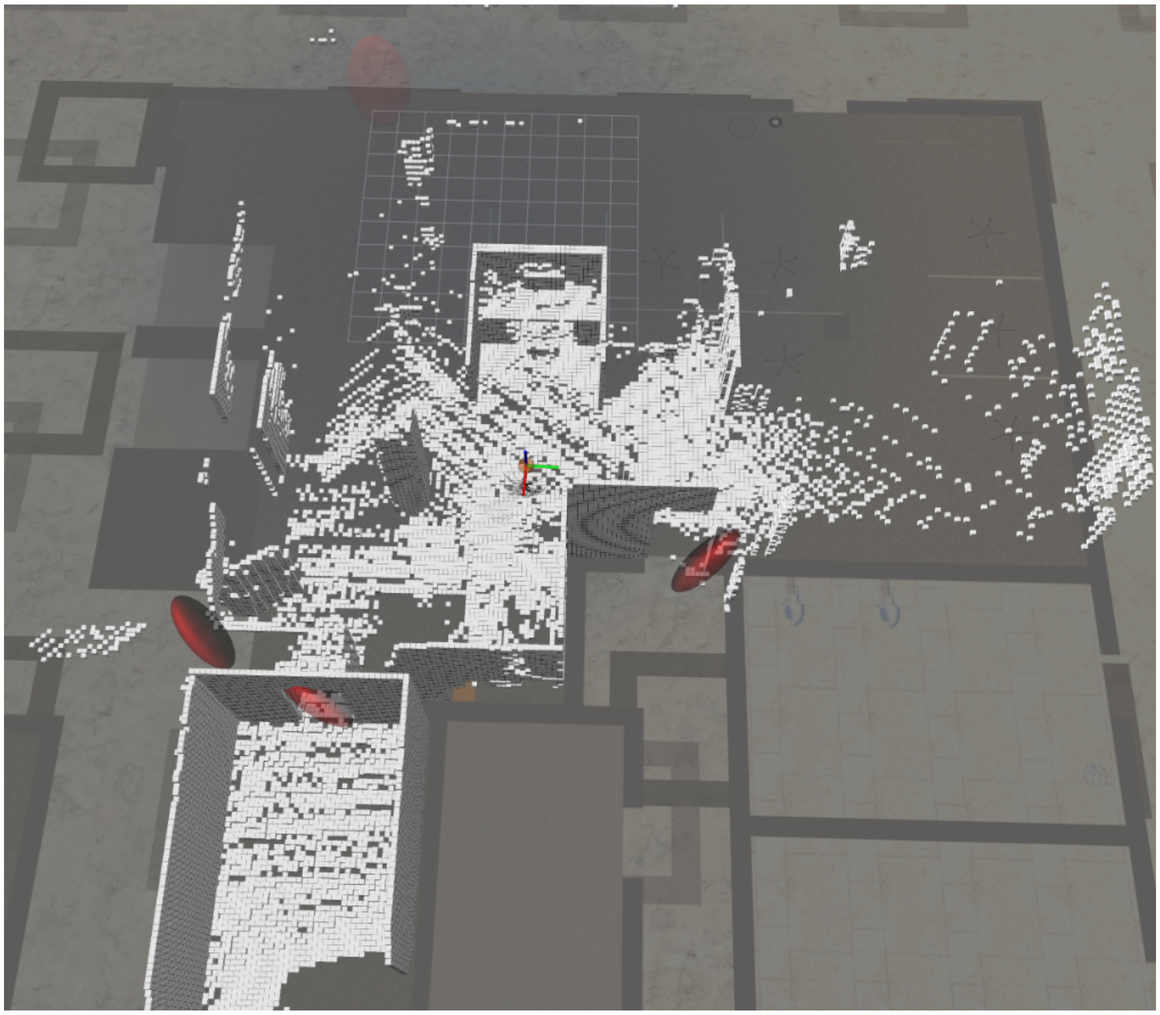


Figure 5-14: Qualitative example of simple mapping using VoluMon estimates. VoluMon estimates plotted as red ellipsoids, and the sliding window 3D occupancy map shown with white voxels. A 2D projection of the floorplan is visualized on the ground plane. VoluMon predicts 3D doorway estimates, even in regions where the dense mapping has little to no information.

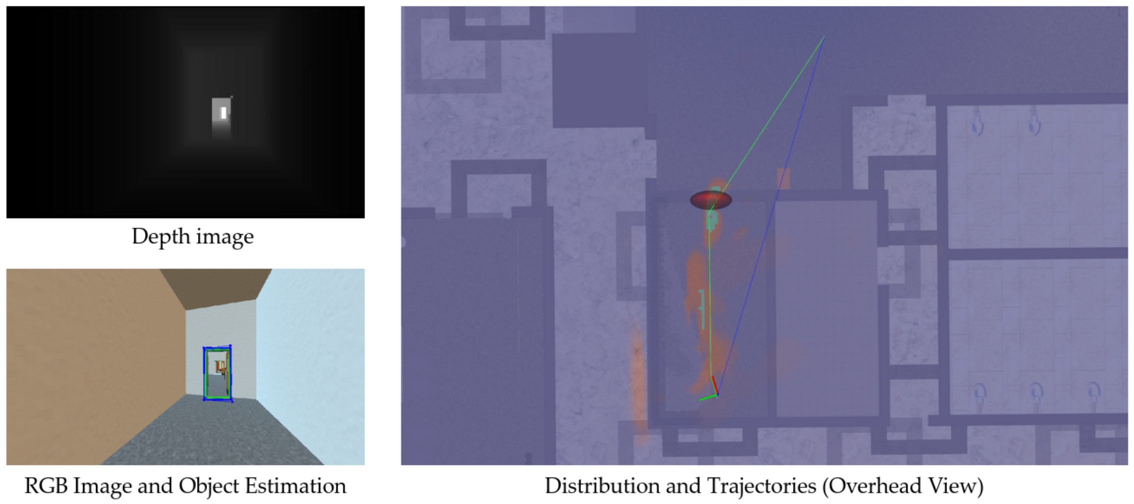

Depth image
RGB Image and Object Estimation
Distribution and Trajectories (Overhead View)

Figure 5-15: Overhead image of VoluMon and LSD integration with exemplar simulated measurements. In the simulated experiments, the depth image (top left) and RGB image (bottom left) are used to build a dense geometric occupancy map and object-level maps respectively. VoluMon lifts the doorway detection (pictured in green in the RGB image) to 3D, populating the red ellipsoid in the overhead image (right). LSD plans a trajectory (green) that passes through the doorway by using the learned sampling distribution which is shown overlaid on an overhead projection of the environment (teal is high probability, orange is medium probability, and purple is low probability). In contrast, the blue trajectory shows how a PRM using a uniform sampling distribution greedily plans through a corner of the room that has not been densely mapped yet.

butions, we used the same network that was used to generate results in Chapter 3, even though the office simulation is not a floorplan from the MIT building system and therefore likely subtly different. We compared our planner to a PRM using a uniform sampling distribution and Euclidean search metric.

As we show qualitatively in Figures 5-15 and 5-16, object-level estimates provided by VoluMon can be used to inform more efficient planning by providing context to planning. As seen in Figure 5-15, given an RGB image and 2D detection of a door, VoluMon estimates the doorway as a 3D ellipsoid, which the learned sampling distribution predicts to be likely to lie on the optimal trajectory to the goal. As shown in Figure 5-16, the naive planner planned through an unobserved wall, while our proposed planner used the contextual information provided by VoluMon to bias trajectories through the doorway. Importantly, VoluMon provided useful 3D object estimation from a monocular sensor despite being trained only on 2D bounding boxes from a stereo pair.[10]

5.9 Limitations

The experiments presented in this chapter have several limitations. First, while we have presented evidence of the suitability of our approach when there is more than a single object in the camera image (Figure 5-11c), we have relied on un-occluded objects. In principle, if 2D bounding boxes can still be properly annotated the dual loss may still be used. However, under heavily occluded examples the weighted primal loss will be biased to underestimating the size of the object, as it will attempt to fit a subset of the points that would be observed if the object were not occluded. Second, like most learning algorithms, hyper-parameter tuning still plays a significant role in performance. In particular, although the network is not given a ground-truth size

[10]Notably, this experiment demonstrated the potential benefits of incorporating semantics as a prior rather than a constraint. The object-level maps built in the office simulation environment had only doorway information, in contrast to the types of semantic maps the sampling distribution prediction network was trained on in Chapter 3. Furthermore, unlike the maps the learned sampling distribution network was trained on, the test environment was not a building from the MIT campus. Despite the novelty in semantic map configurations and dense geometry, in this example our approach still found a more reasonable plan than a planner that considered geometry only.

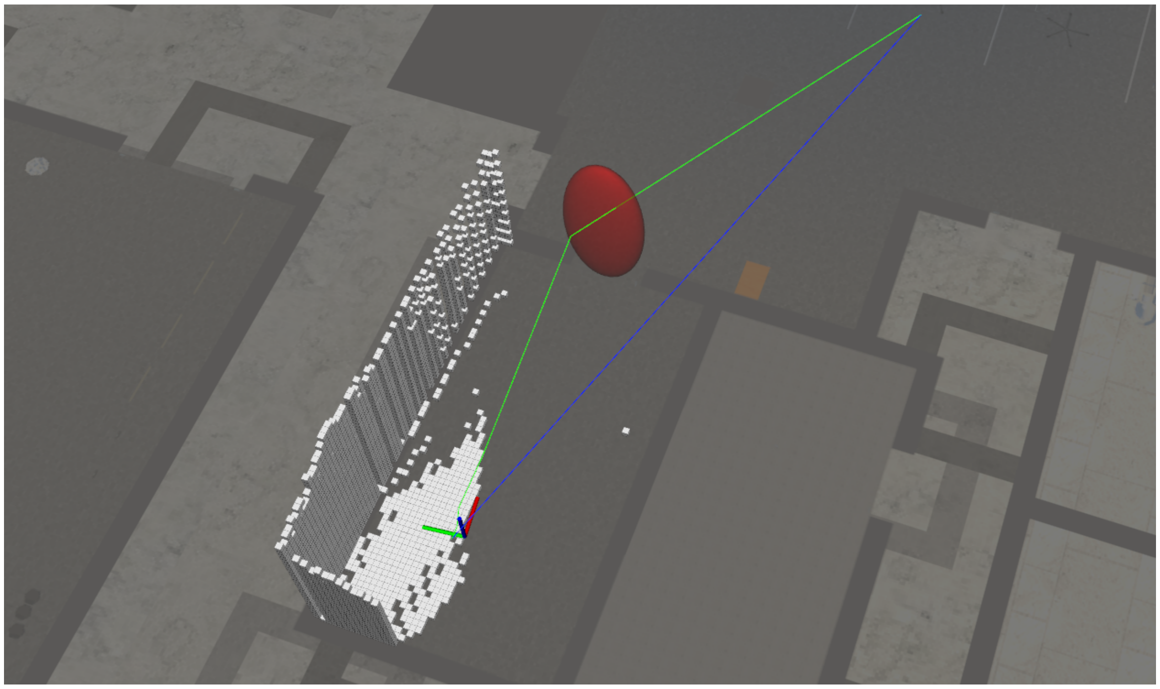


Figure 5-16: Qualitative 3D view of a single planning query comparing our approach to a naive approach. In this figure, the blue trajectory is generated with a PRM using a uniform sampling distribution and Euclidean search metric, which does not consider semantic information in finding the shortest path global plan and therefore plans through an as of yet unobserved corner of the room. In contrast, our method, which is shown by the green trajectory, uses VoluMon to estimate the 3D parameters of the doorway to inform our learned sampling distributions approach. LSD plans through the doorway, rather than naively and optimistically planning through the corner of the room.

for objects, the minimum object size, object size scaling parameter, and scaling of the intra-class variance cost are hand-set. Third, we have considered objects where the dimension is consistent between train and test time. While we expect the intra-class variance penalty to prove useful in practice when there is size diversity, we leave further exploration to future work. Finally, our representation is tested and trained with data from the same sensor. Although care has been taken to parameterize the objects using camera parameters where possible, further investigation is required to enable successfully training and testing on cameras with different characteristics.

5.10 Chapter Summary

We have presented VoluMon, a novel method for weakly supervised monocular object estimation. VoluMon leverages the ellipsoid representation to train a neural network to predict the parameters of a bounding ellipsoidal volume for an object. It does so by using both the primal and dual forms of the ellipsoid to enable weak supervision based on image space annotations. Additionally, intra-class size is regularized to further constrain the optimization. We have shown promising quantitative and qualitative results on synthetic data.

We also demonstrated the utility of VoluMon to provide object-level context to LSD. Although this result is qualitative, we believe it provides compelling evidence that the object-level estimates can be used. Although the general performance of the integration could be improved by re-training the LSD network in floorplans more similar to the simulated office environment, the results also suggest that navigation behaviors conditioned on object-level abstractions can be powerful tools for generalization. Additionally, because our planning and estimation systems are not tightly coupled, as further improvements are made to the subsystems they can be incorporated with relative ease.

Chapter 6

Conclusions and Future Work

In this thesis, we presented a collection of novel approaches aimed at improving the efficiency of autonomous navigation in novel environments. To mitigate the effects of geometric ambiguity, we introduced object-level abstractions to complement dense geometric representations. We made contributions along two main thrusts: navigation and estimation. Our contributions were made with an appreciation that the object estimation process can inform autonomous navigation, but that efficient navigation presents challenging measurement constraints on the object estimation processes. In each case, we combined both geometric and extra-geometric information, building cost functions and priors that can guide optimization when the lack of detailed geometry results in ambiguity.

In Chapter 3, we presented a novel method for planning in unknown environments by leveraging both traditional dense geometric representations and object-level maps. Using expert trajectories, we trained a deep neural network to predict a sampling distribution from a hybrid representation containing a partial occupancy map as well as a partial object-level map. The partial occupancy map was built online, tracking occupied, unoccupied, and unobserved space. The object-level map was also assumed to be built online, and provided the pose, size, and class of a list of objects built from object detections. Our learned sampling distribution exploits both dense geometry and extra-geometric representations to guide the graph building of a SBMP (in our case a PRM) to be more sample efficient than the canonical uniform sampling distribution. Our proposed approach found more efficient trajectories in terms of total distance travelled in some environments in a simulated environment.

In Chapter 4, we improved object-level SLAM with ellipsoid representations to better support the camera motions induced by efficient autonomous navigation. Our key observation was to formulate additional measurements of the ellipsoid representation from RGB images. To augment the bounding box measurements proposed by the state of the art approach, we formulated an additional geometric 3D plane measurement inferred from texture on the object. To further constrain estimates under limited views, we also incorporated a shape prior based on semantic class, effectively leveraging the extra-geometric information to impose a geometric prior. We demonstrated in simulation that our approach enabled more accurate estimates compared to a baseline approach that did not use texture or semantic information. Finally, we showed qualitative examples of performance on both real and simulated data collected from a quadrotor.

In Chapter 5, we proposed a novel weakly-supervised learning approach for 3D object estimation that predicts object properties from a single RGB image. We proposed representing objects as ellipsoids and exploiting two mathematical forms of the ellipsoid representation to estimate the position, rotation, and size of objects using indirect image-space annotations, such as 2D bounding boxes from a stereo pair or 2D instance segmentation paired with a depth image. To further mitigate geometric ambiguity we introduced a cost function that penalized intra-class shape variance, using the intuition that for some objects, the size of the objects varies very little. Using both geometric and semantic information, we showed our approach enabled much faster prediction than an online regression method, and after further online refinement performed similarly to a deep 6D pose estimation technique that assumes known object sizes. We also showed promising qualitative results on a real-world dataset without access to ground-truth, and on an extension to our approach that enables end-to-end detection and estimation over several object classes. Finally, we demonstrated via a qualitative experiment that the estimates from our weakly supervised deep learning method could inform the planning method outlined in Chapter 3.

Together, our contributions demonstrate that combining semantics information and geometric information can tractably improve navigation and estimation.

6.1 Future Work: Efficient Autonomous Navigation

In Chapter 3, we observed that while our learned sampling distribution improved computational efficiency statistics compared to a uniform sampling strategy in all scenarios considered, it did not notably improve overall distance travelled statistics for the trials where the robot begins indoors and the goal is outdoors. One of the key difficulties in the indoor to outdoor navigation task is a relatively sparse semantic signal that may not always be present; only doors that lead outdoors can be expected to have an exit sign near them. In such a case, our intuition is that *semantic search* is more useful than naively extending geometric coverage. Additionally, although Figure 3-5 clearly demonstrates that the learned distribution can predict that regions near exit signs are important for navigation when the goal is outdoors. But, when the goal is in the opposite direction of the high probability regions it may still be difficult for the edge weighting to overcome the geometric bias of the location of the goal. This phenomenon is likely especially true when a relevant decision a small proportion of the total trajectory. This was a phenomenon we observed in several environments, including the integration presented in Section 5.8.

These observations bring us to a larger point about the *staging* of navigation decisions. One compelling way to enable such a capability is to introduce hierarchical structure, i.e., by choosing higher-level actions such as sub-goals. Oftentimes, hierarchical abstractions are used to lower planning latency in large problems. For example, Stadler et al. [162] use two levels of map discretization to improve planning speeds. However, in Section 2.1.4 we presented several methods using geometry and semantics for sub-goal prediction to improve overall distances travelled. One could envision extending our work to learn sampling distributions at several different discretization levels, or using the hybrid geometric-semantic maps to predict sub-goals for our approach. Continuing to use learned sampling distributions in such systems could potentially continue to provide computational focus. A key challenge in pursuing this extension is defining the relationships between the different levels of abstraction, as well as maintaining computational tractability for real-time navigation.

6.2 Future Work: Object-Level Estimation

In Chapters 4 and 5, we presented two methods for enabling object-level estimation, with different benefits. The approach in Chapter 4 synthesizes many measurements over time to estimate object geometry, while the approach described in Chapter 5 estimates object geometry from a measurement from a single point in time. While the lower latency can be useful for navigation, representations fused over time are often more robust and stable. An intriguing future direction for object-level estimation is *task-driven estimation*, tailored for navigation. If the learning problem presented in Chapter 3 could be further factored to provide distributions over which objects are pertinent to the estimation task at hand, or object sensitivities on navigation estimated, different estimation techniques could be brought to bear. For example, it may be useful to estimate more precisely the geometry of a doorway that the robot intends to travel through, rather than doorways that are not of immediate importance.

The methods we have proposed have also focused primarily on estimating the approximate geometric model of objects. However, object texture and appearance can also have semantic import that can effect navigation. Numbered doorway signs are an excellent example of semantic information that cannot as easily be reasoned over in our frameworks (as opposed to exit signs, for example). Estimating the object texture of approximate geometric models may require the use of more sophisticated models such as general quadrics or superquadrics, rather than only ellipsoids.

6.3 Future Work: Combining Semantics and Geometry

In this thesis, we have shown how semantics can provide context when detailed geometric information has not been acquired yet. In this section, we outline potential opportunities including re-incorporating geometric information and considering how object classes effect the future state of the world.

First, it would be useful to consider how further geometric information could be re-introduced to approaches that have begun to exploit semantic information. Combining unsupervised depth prediction [57] with VoluMon may enable discarding the requirement for depth images even at training time. The reliable range of depth images on commodity RGB-D depth sensors can be limited, which in part motivated the development of object-level SLAM approaches using bounding box detections. However, we have shown in this thesis that both coarse semantic priors (i.e., intra-class shape variance, class shape priors) can improve early estimation until higher-fidelty geometric information can be obtained. Following this trend, future approaches to object-level SLAM with ellipsoid representations may consider using an approach such as the one presented in Chapter 5 to provide initial estimates, the approach presented Chapter in 4 to incorporate lightweight bounding boxes online, and depth measurements to further improve object estimates when the objects are in range of dense geometric sensors.

Additionally, in this thesis we have considered the world to be a static environment. Dynamic environments introduce computational complexity as the hypothesis space increases quite quickly even when simply introducing velocity states. However, as several of the approaches described in Section 2.1.4 demonstrate, dynamics such as human motion can also imply future states of the world *beyond* simply the state of the human (i.e., the future state of doors, etc). Key areas of investigation will be efficiently estimating velocity, inferring the implications of semantic context on navigation, and incorporating future states of a dynamic world in planning. Object-level abstractions are beginning to prove useful in dynamic world modelling [62], just as they did in reconstruction. Capturing the effects of dynamic semantic objects on navigation may also be well served by learned sampling distributions, given a more sophisticated planning approach that considers temporal structure.

Bibliography

[1] Polygon office building: 3D urban: Unity asset store. URL `https://assetstore.unity.com/packages/3d/environments/urban\/polygon-office-building-82282`.

[2] 3D warehouse. URL `https://3dwarehouse.sketchup.com/`.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[4] A. Ahmadyan, L. Zhang, J. Wei, A. Ablavatski, and M. Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *arXiv preprint arXiv:2012.09988*, 2020.

[5] B. Akgun and M. Stilman. Sampling heuristics for optimal motion planning in high dimensions. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2011.

[6] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[7] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis. A survey on 3D object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[8] N. Atanasov, B. Sankaran, J. L. Ny, G. J. Pappas, and K. Daniilidis. Nonmyopic view planning for active object detection. *arXiv preprint arXiv:1309.5401*, 2013.

[9] I. Baldwin and P. Newman. Non-parametric learning for natural plan generation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2010.

[10] I. Baldwin and P. Newman. Teaching a randomized planner to plan with semantic fields. *Towards Autonomous Robotic Systems Conference*, 2010.

[11] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin. Combining optimal control and learning for visual navigation in novel environments. In *Proceedings of the Conference on Robot Learning*, 2020.

[12] J. E. Banta, L. Wong, C. Dumont, and M. A. Abidi. A next-best-view system for autonomous 3-D object reconstruction. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2000.

[13] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. Monocular differentiable rendering for self-supervised 3D object detection. *arXiv preprint arXiv:2009.14524*, 2020.

[14] P. J. Besl and N. D. McKay. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, 1992.

[15] M. F. Bonner and R. A. Epstein. Coding of navigational affordances in the human visual system. *Proceedings of the National Academy of Sciences*, 2017.

[16] V. Boor, M. H. Overmars, and A. F. Van Der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.

[17] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas. Probabilistic data association for semantic SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] M. Brown and D. G. Lowe. Unsupervised 3D object recognition and reconstruction in unordered datasets. In *Fifth International Conference on 3-D Digital Imaging and Modeling*, 2005.

[20] B. Burns and O. Brock. Toward optimal configuration space sampling. In *Proceedings of Robotics: Science and Systems*, 2005.

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 2016.

[22] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar. Yale-CMU-Berkeley dataset for robotic manipulation research. *The International Journal of Robotics Research*, 2017.

[23] C. W. Carroll. The created response surface technique for optimizing nonlinear, restrained systems. *Operations Research*, 1961.

[24] C. Chamzas, A. Shrivastava, and L. E. Kavraki. Using local experiences for global motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[25] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. Shapenet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[26] W. G. Chase. Spatial representations of taxi drivers. In *The Acquisition of Symbolic Skills*. 1983.

[27] S. Chen, S. Song, J. Zhao, T. Feng, C. Ye, L. Xiong, and D. Li. Robust dual quadric initialization for forward-translating camera movements. *IEEE Robotics and Automation Letters*, 2021.

[28] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3D object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[29] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[30] J. Civera, A. J. Davison, and J. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 2008.

[31] A. Cosgun and H. I. Christensen. Context-aware robot navigation using interactively built semantic maps. *Paladyn, Journal of Behavioral Robotics*, 2018.

[32] M. Crocco, C. Rubino, and A. Del Bue. Structure from motion with objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[33] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques*, 1996.

[34] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 2006.

[35] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox. PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking. *IEEE Transactions on Robotics*, 2021.

[36] E. W. Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1959.

[37] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 2001.

[38] K. Doherty, D. Fourie, and J. Leonard. Multimodal semantic SLAM with probabilistic data association. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[39] W. Dong and V. Isler. Ellipse regression with predicted uncertainties for accurate multi-view 3D object estimation. *arXiv preprint arXiv:2101.05212*, 2020.

[40] W. Dong, P. Roy, C. Peng, and V. Isler. Ellipse R-CNN: Learning to infer elliptical object from clustering and occlusion. *IEEE Transactions on Image Processing*, 2021.

[41] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim. Recovering 6D object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[42] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint. Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.

[43] K. Duncan, S. Sarkar, R. Alqasemi, and R. Dubey. Multi-scale superquadric fitting for efficient shape and pose recovery of unknown objects. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[44] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone. Map-predictive motion planning in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.

[45] M. Everett, J. Miller, and J. P. How. Planning beyond the sensing horizon using a learned context. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2019.

[46] S. Fidler, S. Dickinson, and R. Urtasun. 3D object detection and viewpoint estimation with a deformable 3D cuboid model. *Proceedings of Advances in Neural Information Processing Systems*, 2012.

[47] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.

[48] K. M. Frey, T. J. Steiner, and J. P. How. Efficient constellation-based map-merging for semantic SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[49] D. Frost, V. Prisacariu, and D. Murray. Recovering stable scale in monocular SLAM using object-supplemented bundle adjustment. *IEEE Transactions on Robotics*, 2018.

[50] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2014.

[51] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015.

[52] P. Gay, C. Rubino, V. Bansal, and A. Del Bue. Probabilistic structure from motion with objects (PSfMO). In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[53] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[54] G. Georgakis, B. Bucher, K. Schmeckpeper, S. Singh, and K. Daniilidis. Learning to map for active semantic goal navigation. *arXiv preprint arXiv:2106.15648*, 2021.

[55] R. Geraerts and M. H. Overmars. Sampling techniques for probabilistic roadmap planners. In *Conference on Intelligent Autonomous Systems*, 2004.

[56] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[57] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[58] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT Press, 2016.

[59] W. N. Greene and N. Roy. Flame: Fast lightweight mesh estimation using variational smoothing on delaunay graphs. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[60] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, 2004.

[61] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[62] M. Henein, J. Zhang, R. Mahony, and V. Ila. Dynamic SLAM: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129. IEEE, 2020.

[63] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[64] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab. Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In *Proceedings of the Asian Conference on Computer Vision*. Springer, 2012.

[65] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013.

[66] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid. Structure aware SLAM using quadrics and planes. In *Proceedings of the Asian Conference on Computer Vision*, 2018.

[67] M. Hosseinzadeh, Y. Latif, T. Pham, N. Suenderhauf, and I. Reid. Towards semantic SLAM: Points, planes and objects. *arXiv preprint arXiv:1804.09111*, 2018.

[68] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[69] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[70] B. Ichter, J. Harrison, and M. Pavone. Learning sampling distributions for robot motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[71] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust. Learned critical probabilistic roadmaps for robotic motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.

[72] L. Janson and M. Pavone. Fast marching trees: A fast marching sampling-based method for optimal motion planning in many dimensions. *Proceedings of the International Symposium on Robotics Research*, 2013.

[73] S. S. Joshi and T. Panagiotis. Non-parametric informed exploration for sampling-based motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[74] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

[75] L. P. Kaelbling and T. Lozano-Pérez. Hierarchical planning in the now. In *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[76] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 2008.

[77] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME—Journal of Basic Engineering*.

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

[79] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager. Uncertainty-aware occupancy map prediction using generative networks for robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[80] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In *Proceedings of the Conference on Robot Learning*, 2018.

[81] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996.

[82] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[83] S. Kim and Y. Hwang. A survey on deep learning based methods and datasets for monocular 3D object detection. *Electronics*, 2021.

[84] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*, 2015.

[85] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[86] D. Koller, K. Daniilidis, T. Thorhallson, and H.-H. Nagel. Model-based object tracking in traffic scenes. In *Proceedings of the European Conference on Computer Vision*, 1992.

[87] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 1978.

[88] B. J. Kuipers and T. S. Levitt. Navigation and mapping in large scale space. *AI Magazine*, 1988.

[89] A. Kundu, Y. Li, and J. M. Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[90] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and V. Ferrari. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale. *International Journal of Computer Vision*, 2020.

[91] T. Lai, P. Morere, F. Ramos, and G. Francis. Bayesian local sampling-based planning. *IEEE Robotics and Automation Letters*, 2020.

[92] D. T. Larsson, D. Maity, and P. Tsiotras. Information-theoretic abstractions for resource-constrained agents via mixed-integer linear programming. *arXiv preprint arXiv:2102.10015*, 2021.

[93] S. LaValle. Virtual reality. 2016.

[94] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[95] S. M. LaValle. *Planning algorithms.* Cambridge University Press, 2006.

[96] J. Lee, S. Walsh, A. Harakeh, and S. L. Waslander. Leveraging pre-trained 3D object detection models for fast ground truth generation. In *Proceedings of the IEEE International Conference on Intelligent Transportation Systems.*

[97] Z. Liao, W. Wang, X. Qi, and X. Zhang. RGB-D object SLAM using quadrics for indoor environments. *Sensors*, 2020.

[98] Z. Liao, W. Wang, X. Qi, X. Zhang, L. Xue, J. Jiao, and R. Wei. Object-oriented SLAM using quadrics and symmetry properties for indoor environments. *arXiv preprint arXiv:2004.05303*, 2020.

[99] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, 2014.

[100] K. Liu, M. Stadler, and N. Roy. Learned sampling distributions for efficient planning in hybrid geometric and object-level representations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.

[101] K. Liu, K. Ok, and N. Roy. VoluMon: Weakly-supervised volumetric monocular estimation with ellipsoid representations. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*. IEEE, 2021.

[102] K. Y. Liu. *Learning Gaussisan noise models from high-dimensional sensor data with deep neural networks.* PhD thesis, Massachusetts Institute of Technology, 2018.

[103] S. Liu, M. Watterson, S. Tang, and V. Kumar. High speed navigation for quadrotors with limited onboard sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016.

[104] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *Proceedings of the European Conference on Computer Vision*, 2016.

[105] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza. DroNET: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 2018.

[106] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2004.

[107] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 1981.

[108] F. Manhardt, W. Kehl, and A. Gaidon. ROI-10D: Monocular lifting of 2D detection to 6D pose and metric shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[109] J. McCormac, R. Clark, M. Bloesch, A. Davison, and S. Leutenegger. Fusion++: Volumetric object-level SLAM. In *Proceedings of the International Conference on 3D Vision*, 2018.

[110] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2002.

[111] M. Montemerlo, S. Thrun, D. Koller, B. Wegbreit, et al. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2003.

[112] J. J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. In *Numerical Analysis*. Springer, 1978.

[113] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3D bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[114] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015.

[115] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[116] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[117] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng. Multimodal deep learning. In *Proceedings of the International Conference on Machine Learning*, 2011.

[118] L. Nicholson, M. Milford, and N. Sünderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robotics and Automation Letters*, 2018.

[119] K. Ok, D. Gamage, T. Drummond, F. Dellaert, and N. Roy. Monocular image space tracking on a computationally limited MAV. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2015.

[120] K. Ok, K. Liu, K. Frey, J. P. How, and N. Roy. Robust object-based SLAM for high-speed autonomous navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[121] K. Ok, K. Liu, and N. Roy. Hierarchical object map estimation for efficient and robust navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2021.

[122] K. Park, A. Mousavian, Y. Xiang, and D. Fox. LatentFusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[123] D. Paschalidou, A. O. Ulusoy, and A. Geiger. Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[124] D. Paschalidou, L. V. Gool, and A. Geiger. Learning unsupervised hierarchical part decomposition of 3D objects from a single RGB image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[125] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information*

Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-\style-high-performance-deep-learning-library.pdf`.

[126] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-DOF object pose from semantic keypoints. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[127] S. Pillai and J. Leonard. Monocular SLAM supported object recognition. *Proceedings of Robotics: Science and Systems*, 2015.

[128] S. Prentice and N. Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *Robotics Research*. 2010.

[129] A. Pronobis, F. Riccio, and R. P. Rao. Deep spatial affordance hierarchy: Spatial knowledge representation for planning in large-scale environments. In *ICAPS Workshop on Planning and Robotics*, 2017.

[130] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. End-to-end pseudo-lidar for image-based 3D object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[131] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[132] Z. Ravichandran, L. Peng, N. Hughes, J. D. Griffith, and L. Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3D scene graphs using graph neural networks. *arXiv preprint arXiv:2108.01176*, 2021.

[133] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[134] J. Redmon and A. Farhadi. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[135] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[136] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Proceedings of Advances in Neural Information Processing Systems*, 2015.

[137] C. Richter and N. Roy. Learning to plan for visibility in navigation of unknown environments. In *Proceedings of the International Symposium on Experimental Robotics*, 2016.

[138] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. In *Proceedings of Robotics: Science and Systems*, 2017.

[139] C. Richter, A. Bry, and N. Roy. Polynomial trajectory planning for quadrotor flight. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013.

[140] C. Richter, J. Ware, and N. Roy. High-speed autonomous navigation of unknown environments using learned probabilities of collision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2014.

[141] C. Richter, W. Vega-Brown, and N. Roy. Bayesian learning for safe high-speed navigation in unknown environments. In *Proceedings of the International Symposium on Robotics Research*, 2015.

[142] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[143] D. M. Rosen, K. J. Doherty, A. Terán Espinoza, and J. J. Leonard. Advances in inference and representation for simultaneous localization and mapping. *Annual Review of Control, Robotics, and Autonomous Systems*, 2021.

[144] A. Rosinol, A. Gupta, M. Abate, J. Shi, and L. Carlone. 3D dynamic scene graphs: Actionable spatial perception with places, objects, and humans. *Proceedings of Robotics: Science and Systems*, 2020.

[145] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Proceedings of Advances in Neural Information Processing Systems*, 1999.

[146] C. Rubino, M. Crocco, and A. Del Bue. 3D object localisation from multi-view image detections. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[147] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the IEEE International Conference on Computer Vision*, 2011.

[148] M. Ruhnke, B. Steder, G. Grisetti, and W. Burgard. Unsupervised learning of 3D object models from partial views. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[149] M. Ryll, J. Ware, J. Carter, and N. Roy. Efficient trajectory planning for high speed flight in unknown environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[150] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[151] A. Saxena, J. Driemeyer, and A. Y. Ng. Learning 3-D object orientation from images. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2009.

[152] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference*, 2001.

[153] T. Schouwenaars, É. Féron, and J. How. Safe receding horizon path planning for autonomous vehicles. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, 2002.

[154] M. Shan, Q. Feng, and N. Atanasov. OrcVIO: Object residual constrained Visual-Inertial Odometry. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2020.

[155] Y. Shaoul, K. Liu, K. Ok, and N. Roy. Online descriptor enhancement via self-labelling triplets for visual data association. *arXiv preprint arXiv:2011.10471*, 2020.

[156] A. Sharma, W. Dong, and M. Kaess. Compositional scalable object SLAM. *arXiv preprint arXiv:2011.02658*, 2020.

[157] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan. Learned map prediction for enhanced mobile robot exploration. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2019.

[158] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling monocular 3D object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[159] J. Šircelj, T. Oblak, K. Grm, U. Petković, A. Jaklič, P. Peer, V. Štruc, and F. Solina. Segmentation and recovery of superquadric models using convolutional neural networks. *arXiv preprint arXiv:2001.10504*, 2020.

[160] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous Robot Vehicles*. Springer, 1990.

[161] F. Solina and R. Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.

[162] M. Stadler, K. Liu, and N. Roy. Online high-level model estimation for efficient hierarchical robot navigation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2021.

[163] M. K. Stadler. *Learned Functions for Perceptually Informed Robot Navigation*. PhD thesis, Massachusetts Institute of Technology, 2020.

[164] G. J. Stein, C. Bradley, and N. Roy. Learning over subgoals for efficient navigation of structured, unknown environments. In *Proceedings of the Conference on Robot Learning*, 2018.

[165] T. J. Steiner, R. D. Truax, and K. Frey. A vision-aided inertial navigation system for agile high-speed flight in unmapped environments. In *Proceedings of the IEEE Aerospace Conference*, 2017.

[166] I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 2012. `http://ompl.kavrakilab.org`.

[167] E. Sucar, K. Wada, and A. Davison. NodeSLAM: Neural object descriptors for multi-view shape reconstruction. *Proceedings of the International Conference on 3D Vision*, 2020.

[168] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[169] N. Sünderhauf. Where are the keys?–learning object-centric navigation policies on semantic maps with graph convolutional networks. *arXiv preprint arXiv:1909.07376*, 2019.

[170] L. Tai, G. Paolo, and M. Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2017.

[171] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How. Search and rescue under the forest canopy using multiple uavs. *The International Journal of Robotics Research*, 2020.

[172] J. Tremblay, T. To, and S. Birchfield. Falling things: A synthetic dataset for 3D object detection and pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Workshop).*

[173] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *Proceedings of the Conference on Robot Learning*, 2018.

[174] V. Vasilopoulos, G. Pavlakos, S. L. Bowman, J. D. Caporale, K. Daniilidis, G. J. Pappas, and D. E. Koditschek. Reactive semantic planning in unexplored semantic environments using deep perceptual feedback. *IEEE Robotics and Automation Letters*, 2020.

[175] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian. Volumetric next-best-view planning for 3D object reconstruction with positioning error. *International Journal of Advanced Robotic Systems*, 2014.

[176] G. Vezzani, U. Pattacini, and L. Natale. A grasping approach based on superquadric models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2017.

[177] G. Vezzani, U. Pattacini, G. Pasquale, and L. Natale. Improving superquadric modeling and grasping with prior on object shapes. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[178] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Proceedings of Robotics: Science and Systems*, 2015.

[179] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng. Neural RRT*: Learning-based optimal path planning. *IEEE Transactions on Automation Science and Engineering*, 2020.

[180] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[181] X. Weng and K. Kitani. Monocular 3D object detection with pseudo-lidar point cloud. In *Proceedings of the IEEE International Conference on Computer Vision (Workshop)*, 2019.

[182] E. Whiting, J. Battat, and S. Teller. Topology of urban environments. In *Computer-Aided Architectural Design Futures*. 2007.

[183] Y. Wu, T. Marks, A. Cherian, S. Chen, C. Feng, G. Wang, and A. Sullivan. Unsupervised joint 3D object model learning and 6D pose estimation for depth-based instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (Workshop)*, pages 0–0, 2019.

[184] L. Xia, J. Cui, R. Shen, X. Xu, Y. Gao, and X. Li. A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots. *International Journal of Advanced Robotic Systems*, 2020.

[185] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond PASCAL: A benchmark for 3D object detection in the wild. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 2014.

[186] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *Proceedings of Robotics: Science and Systems*, 2018.

[187] S. Yang and S. Scherer. CubeSLAM: Monocular 3-D object SLAM. *IEEE Transactions on Robotics*, 2019.

[188] S. Yang and S. Scherer. Monocular object and plane SLAM in structured environments. *IEEE Robotics and Automation Letters*, 2019.

[189] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[190] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[191] M. Zucker, J. Kuffner, and J. A. Bagnell. Adaptive workspace biasing for sampling based planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2008.