On-line, incremental visual scene understanding for an indoor navigating robot

by

Grace S. Tsai

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (Electrical Engineering: Systems) in the University of Michigan 2014

To everyone who has supported me over the years

Acknowledgments

Foremost, I would like to express my appreciation to my advisor, Professor Benjamin Kuipers, for his guidances throughout my PhD study. I especially appreciate his efforts of helping me became an independent researcher. Without him, I would not have been able to make as much progress as I have achieved in my research, and in my professional development. Working with him is one of the most valuable and joyful experiences I had in life.

I would like to thank my other committee members: Jeffery Fessler, Vineet Kamat, Gregory Wakefield for your collective efforts that helped me complete my dissertation. I would also like to express my appreciation to Silvio Savarese for his in-depth suggestions to my work before and during my thesis proposal.

Thanks to all of the members (past, present, and unofficial) in the Intelligent Robotics Lab at the University of Michigan for their supports, inspirations, and discussions on my work: Jingen Lui, Changhai Xu, Collin Johnson, Jong-Jin Park, Zhen Zeng, Ron Gaynier, Chia-Wui Luo, Bangxin Hu, and Paul Foster. My thanks also go to Andrew Richardson, Lauren Hinkle, Chen Feng, Yingze Bao, Yu-Wei Chao, and Min Sun, for providing various help and suggestions for my work.

Last but not the least, I would like to thank my lovely family for their understanding and encouragements throughout my study. They are always very excited to learn my achievements and are always extremely supportive when I was struggling. I would particularly like to thank my partner, Simon Huang, for his remarkable patience and endless support, and for being with me during all my ups and downs for all these years.

Grant Acknowledgments

This thesis work has taken place in the Intelligent Robotics Lab in the Computer Science and Engineering Division of the University of Michigan. Research of the Intelligent Robotics lab is supported in part by grants from the National Science Foundation (CPS-0931474, IIS-1111494, and IIS-1252987), and from the TEMA-Toyota Technical Center.

Table of Contents

Dedication **ii**

Acknowledgments **iv**

List of Figures **viii**

List of Tables **x**

Abstract **xi**

1 Introduction **1**
- 1.1 Problem Statement 1
- 1.2 Approach Overview 2
- 1.3 Scope of the Thesis 6
- 1.4 Significance 7
- 1.5 Thesis Overview 8

2 Literature Review **10**
- 2.1 Representation 10
- 2.2 Algorithms 15
  - 2.2.1 Single-Image Approaches 15
  - 2.2.2 Multiple-Image Approaches 17
- 2.3 Summary 19

3 Technical Preliminaries **21**
- 3.1 Coordinate Systems 21
  - 3.1.1 Image Space 21
  - 3.1.2 Camera Coordinate 22
  - 3.1.3 3D Coordinate 22
  - 3.1.4 Ground-Plane map 23
- 3.2 Recursive State Estimators 23
  - 3.2.1 The Bayesian Filter 24
  - 3.2.2 The Extended Kalman Filter 25

4 Datasets **26**
4.1 Michigan Indoor Corridor 2011 Video Dataset 27
4.2 Michigan Indoor Corridor 2012 Video Dataset 28
4.3 Michigan Indoor Corridor 2014 Video Dataset 29

5 On-line Scene Understanding **31**
5.1 Hypotheses Generation 33
5.2 3D Construction from Single Frame 34
5.3 Point Feature Tracking 35
5.4 Pose Estimation 36
5.5 Hypotheses Testing using Bayesian Filtering 38
5.6 Evaluation 39
5.7 Summary 47

6 Incremental Scene Understanding **48**
6.1 Planar Semantic Model (PSM) 49
6.1.1 Image Projection of PSM 51
6.1.2 3D Construction of PSM from Single Image 52
6.2 Incremental Hypotheses Generation 54
6.3 Refining PSM Model 56
6.4 Incremental Hypotheses Testing using Bayesian Filtering 58
6.5 Evaluation 62
6.6 Summary 67

7 Handling Perceptual Clutter **68**
7.1 Ground Plane Extraction 70
7.2 Robot Pose Estimation 72
7.3 Features Extraction 73
7.4 Refining PSM Hypotheses 75
7.5 Incremental PSM Hypotheses Generation 75
7.6 Hypotheses Testing with Partial Explanation 76
7.6.1 Explaining vertical-plane features 82
7.6.2 Explaining point-set features 83
7.7 Evaluation 83
7.7.1 Repeatability of Feature Extraction 83
7.7.2 Evaluation of the Framework 83
7.7.3 Analyzing the Likelihood Function 89
7.8 Summary 94

8 Focusing Attention on Features that Matter **95**
8.1 Compute Informativeness of a Point Feature 97
8.2 Select Informative Point Features 99
8.3 Evaluation 101

8.4 Summary 107

9 Application — From Structure to Opportunities for Action 108

9.1 Action Opportunity Star 110

9.1.1 Extracting Opportunities from PSM 112

9.1.2 Extracting AOS from Opportunities 115

9.2 Evaluation 116

9.3 Summary 123

10 Conclusion and Future Work 124

10.1 Summary 124

10.2 Future Work 126

10.2.1 Extension of Proposed Method 126

10.2.2 Future for Scene Understanding for Indoor Navigating Robots . . 129

Bibliography 131

List of Figures

2.1 Related work on scene understanding 11

4.1 The Michigan Indoor Corridor 2011 Video Dataset 27
4.2 The Michigan Indoor Corridor 2012 Video Dataset 28
4.3 The Michigan Indoor Corridor 2014 Video Dataset 29

5.1 Proposed on-line scene understanding method 32
5.2 Examples of simple hypothesized model 33
5.3 Example of hypotheses testing with Bayesian filtering 41
5.4 Qualitative evaluation of the on-line scene understanding method 43
5.5 3D planar model constructed by our method 45
5.6 3D model constructed by other multiple-image methods 46

6.1 On-line generate-and-test framework 50
6.2 Planar Semantic Model (PSM) 53
6.3 Types of child hypotheses 55
6.4 Examples of the hypotheses generated by our incremental process 60
6.5 Examples of the on-line generate-and-test framework 61
6.6 Results of on-line generate-and-test framework in different environments . 64
6.7 Examples when failing to identify the best hypothesis 65
6.8 Bayesian filtering results of Dataset L 66

7.1 On-line generate-and-test framework with RGB-D image stream 69
7.2 Ground plane extraction from RGB-D image 71
7.3 Feature extraction from RGB-D image 74
7.4 Example of multiple valid hypotheses 78
7.5 Simplicity 80
7.6 Importance of partial explanation 81
7.7 Extracted features in multiple runs 84
7.8 Simplicity used in our experiments 85
7.9 Qualitative evaluation of the our framework in cluttered environments . . 87
7.10 Example of interpreting clutter region 88
7.11 Final hypotheses from INTERSECTION 93

8.1 Framework with attention focusing method 96
8.2 Informative regions 98
8.3 Qualitative evaluation of the attention focusing method 104
8.4 Example when baseline framework is more informative 106

9.1 Two-layer representation 109
9.2 Visualization of different opportunity types 112
9.3 Examples of the AOS in different locations 113
9.4 Matching opportunities to form AOS 116
9.5 Examples of the two-layer representation on a long video 118
9.6 Examples of the two-layer representation in a cross intersection 119
9.7 Examples of the two-layer representation at T intersections 120
9.8 Example of incorrect PSM but correct AOS 121

List of Tables

5.1 Computational time analysis 40
5.2 Quantitative analysis of the on-line scene understanding method 42

6.1 Quantitative evaluation of the on-line generate-and-test framework 65

7.1 Example for the three-way trade-off among the likelihood factors 79
7.2 Quantitative evaluation of our framework in cluttered environments . . . 89
7.3 List of experiments that are used to analyze the likelihood 90
7.4 Results for analyzing likelihood function 91

8.1 Quantitative comparison with and without attention focusing 103

Abstract

An indoor navigating robot must perceive its local environment in order to act. The robot must construct a model that captures critical navigation information from the stream of visual data that it acquires while traveling within the environment. Visual processing must be done on-line and efficiently to keep up with the robot's need.

This thesis contributes both representations and algorithms toward solving the problem of modeling the local environment for an indoor navigating robot. Two representations, Planar Semantic Model (PSM) and Action Opportunity Star (AOS), are proposed to capture important navigation information of the local indoor environment. PSM models the geometric structure of the indoor environment in terms of ground plane and walls, and captures rich relationships among the wall segments. AOS is an abstracted representation that reasons about the navigation opportunities at a given pose. Both representations are capable of capturing incomplete knowledge where representations of unknown regions can be incrementally built as observations become available. An on-line generate-and-test framework is presented to construct the PSM from a stream of visual data. The framework includes two key elements, an incremental process of generating structural hypotheses and an on-line hypothesis testing mechanism using a Bayesian filter.

Our framework is evaluated in three phases. First, we evaluate the effectiveness of the on-line hypothesis testing mechanism with an initially generated set of hypotheses in simple empty environments. We demonstrate that our method outperforms state-of-the-art methods on geometric reasoning both in terms of accuracy and applicability to a navigating robot. Second, we evaluate the incremental hypothesis generating process and demonstrate the expressive power of our proposed representations. At this phase, we also demonstrate an attention focusing method to efficiently discriminate among the active hypothesized models. Finally, we demonstrate a general metric to test the hypotheses with partial explanations in cluttered environments.

Chapter 1

Introduction

1.1 Problem Statement

A robot needs to perceive its local environment in order to act. Visual sensor has become popular because it captures a lot of information at low cost. When using vision as a primary sensor for a robot, the input to visual perception is a temporally contiguous stream of images, not simply a single image or a set of images. The output of visual perception must be a representation of the local environment that is useful for the robot to make plans. We call the process of constructing a representation of the local environment from sensory input *scene understanding*. For a robot, scene understanding must be an on-line and efficient process as opposed to a batch process so that the robot's representation can be updated as observations become available.

A representation is *semantically meaningful* if the representation specifies the robot's action opportunities. Depending on the purpose of the robot, different robots may have different meanings in terms of semantically meaningful representation. For example, a semantically meaningful representation for a navigating robot must specify where there is free-space for the robot to travel and where there are obstacles, while for a manipulating robot, a semantically meaningful representation must provide information implying where are the graspable points of the objects in the local environment.

Besides being semantically meaningful, the representation for the local environment must be capable of expressing *incomplete knowledge*. There are two types of incomplete knowledge for a robot. First, the representation must specify where there is missing information due to lack of observations. With incomplete knowledge representation, the robot can incrementally complete its understanding of the environment as observations become available. This is an essential part of on-line visual processing because visual sensors usually have a limited field of view. Second, the representation needs to specify regions that cannot be explained by existing models. As new models become available, the robot can increase its level of representation of the local environment with more varieties of action opportunities. Thus, with the ability to express incomplete knowledge, the robot can make plans to act, even if the local environment are not fully observed or explained.

For a robot to make real-time decisions, visual scene understanding needs to be an on-line and efficient process. An on-line process maintains an interpretation of the local environment and uses the current sensory data to update that interpretation. In addition, each update must be efficient to minimize the delay for the decision making. The ideal efficiency for a robot is to reach the frame-rate of the sensory input.

In summary, in this thesis, we identify three criteria for solving visual scene understanding problems:

- The robot must build a **semantically meaningful representation** of the local environment that implies the robot's action opportunities.
- Knowledge representations of the local environment must be **capable of representing incomplete knowledge** to specify where there is missing information due to lack of observations, and to specify regions that cannot be explained by existing models.
- Scene understanding must be done **on-line and efficiently** to keep up with the robot's needs.

In this thesis, we focus on a specific type of robot, an indoor navigating robot. The goal of this thesis is to propose a knowledge representation for the indoor environment that is useful for the navigating robot to make plans. In addition, this thesis demonstrates an on-line framework that efficiently constructs such a knowledge representation primarily from a stream of visual data.

1.2 Approach Overview

A robot should interpret the physical world with the foundational concepts of *space* and *objects*. From the visual sensor input stream $Z = Z_{0:t} = \{z_0, z_1, ..., z_t\}$ that the robot received from time step 0 to t, the process of perceiving the world is an incremental process, starting from modeling the space to understanding objects. Inspired by the Object Semantic Hierarchy (OSH) [81], this process can be decomposed into multiple levels of interpretations.

The first level of interpretation is to model the sensor input in terms of space, the geometric structure of the environment. The portions of the environment that can not be explained by the geometric structure are "clutter". Clutter consists of piles of objects, and is an intermediate level between the knowledge of space and the full knowledge of the environment. The later levels are then dedicated to modeling objects from clutter.

Level 0: Noisy World

Without any knowledge of the environment, the sensor input z_t is simply an array of high-dimensional pixel-level data,

$$z_t = \epsilon_0 \tag{1.1}$$

where z_t is the snapshot of the sensor input stream Z at time t, and ϵ_0 is a random variable that represents the sensor input as noise.

Level 1: Geometric Structure of Indoor Environment

For an indoor navigating robot, the majority of the observed image z_t is explained by a model M of the 3D geometric structure of the indoor environment,

$$z_t = G_1(M, \mathbf{x}_t) + \epsilon_1. \tag{1.2}$$

In order for a robot to navigate efficiently within the environment, M is usually a coarse-grained representation to capture the semantic structure of the building (e.g. walls and ground plane). $\mathbf{x}_t$ is the observing pose of the robot. Both M and $\mathbf{x}_t$ are defined with respect to the global reference frame. [1] G_1 is a function that predicts the 2D image projection of the structural model M at pose $\mathbf{x}_t$. ϵ_1 is the residual of the sensor input that is not explained by the structural model M and $\mathbf{x}_t$.

At this level, the robot explains much of its observations by M so the unexplained portion of the image decreases, $\|\epsilon_1\| \ll \|\epsilon_0\|$. The size of the residuals $\|\epsilon_1\|$ depends on

[1] It is common to define the global reference frame according to the initial pose of the robot $\mathbf{x}_0$. However, as the robot explores the environment, it might discover a more convenient global reference frame.

the granularity of the model M. If M is a point cloud (a collection of samples of the 3D environment), then $\|\epsilon_1\|$ is nearly 0 because the whole scene can be fully expressed by a point cloud. This is what SLAM (simultaneous localization and mapping) methods do, with laser or vision. Given observations z_t, SLAM methods construct a point cloud and localize $\mathbf{x}_t$ the robot within it to minimize the error ϵ_1 between the prediction and observation. If M is more concise and more abstract (coarse-grained model), such as a planar model, $\|\epsilon_1\|$ may not be 0 when parts of the environment cannot be modeled by planes.

Level 2: Clutter

"Clutter" consists of regions of the indoor environment that are not explainable by the structural model M. If M is the only model that the robot has, clutter is simply a bunch of "stuff" in the environment that is not interpretable by the robot. With the knowledge of clutter, the interpretation of a image snapshot z_t becomes,

$$z_t = G_2(M, C_1, ..., C_k, \mathbf{x}_t) + \epsilon_2 \tag{1.3}$$

where G_2 is a function that predicts the 2D image projection of M and a set of clutter regions $C_1, ..., C_k$, at pose $\mathbf{x}_t$. ϵ_2 is the residual noise after interpreting z_t with the structural model M and clutter, and thus, $\|\epsilon_2\| \ll \|\epsilon_1\|$. At this level, ϵ_2 is simply sensor noise.

Clutter does not have a fixed meaning because it depends on the structural model M. As discussed in Level 1, if M is a point cloud, then there is no clutter in the representation. If M is a coarse-grained model, then clutter may exist. The physical locations where clutter is present also depend on the model M.

Without the knowledge of objects, clutter is simply a collection of unstructured 3D points that occupy the space. Clutter is represented by a set of 3D regions, where each region is represented by a 3D point cloud in the global frame of reference. These regions can be determined by clustering the 3D points based on their distances in the 3D environment. The 2D image projection of a clutter region is a 2D blob. In fact, clutter is a type of incomplete knowledge of the indoor environment because it specifies regions of the environment that are occupied without specifying the actual actions that the robot can perform at these regions. To fully interpret these regions, the robot needs the concept of objects.

Level 3: Objects

A clutter region is an object or a pile of objects. In indoor environments, there tend to be multiple instances of object categories that could be explained by the same object model (e.g. chairs, doors, and pedestrians). The robot could build a model for each object instance, or a single general model for an object category with the information in the clutter regions. [2]

When clutter can be well clustered into regions $C_1, ..., C_k$, each clutter region becomes a smaller interpretation problem. If the robot has a set of object models, portions of each clutter region C_i can be explained by object models,

$$C_i = \langle\{\langle O_j, \mathbf{y}_t^j\rangle | j = 1, 2, ..., n_i\}, C'_i\rangle \tag{1.4}$$

where n_i is the number of objects within clutter region C_i, and O_j is the model of object j. Object models are more structured and semantically meaningful than 3D point clouds. The object pose $\mathbf{y}_t^j$ is the transformation between the reference frame of object O_j and the global reference frame. If $\mathbf{y}_t^j$ is constant for the whole time, O_j is a *static object*, an object that remains static with respect to the structural model M. If $\mathbf{y}_t^j$ varies through time, O_j is a *dynamic object*, an object that is moving relative to the structural model M. C'_i is the residual of the cluster region that cannot be explained by the robot with the current object models. Thus, C'_i is the remaining piece of incomplete knowledge represented by a 3D point cloud.

With some knowledge of objects, the sensor input z_t can be explained by,

$$z_t = G_3(M, \mathbf{x}_t, \{\langle O_j, \mathbf{y}_t^j\rangle | j = 1, 2, ..., N\}, C'_1, ...C'_k) + \epsilon_3 \tag{1.5}$$

where $N = n_1 + n_2 + ... + n_k$ is the total number of objects modeled in the clutter regions. G_3 is a function that predicts the 2D image projection of the 3D environment consisting of the structural model M, a set of objects $\{O_j | j = 1, 2, ..., N\}$, and remaining clutter regions $C'_1, ...C'_k$, and ϵ_3 is the residual noise.

At this level of interpretation, the robot has gained more understanding of the environment by explaining portions of clutter by a set of concise and structured object models. However, the process of extracting objects from the clutter may filter out some

[2] In computer vision, a common approach is to build an object classifier through supervised learning with a large amount of training examples of objects [43]. There are also methods that attempt to discover object instances through unsupervised learning algorithms [75].

3D points as noise, and thus, the residual noise may slightly increase, $\|\epsilon_3\| \geq \|\epsilon_2\|$. There are two sources for the noise to increase. The first source is the sensor noise from the image observation around the objects. The second source is the error caused by the simplicity of the object models. In order for the robot to act efficiently with the objects, object models are designed to be concise, coarse-grained abstractions in which small details of the objects are not captured by the 3D models. Ignoring small details causes error between the modeled environment and image observations.

Level n: More Objects

At any point, the robot may obtain new object models from the unexplained regions to increase its level of knowledge of the environment. If the robot has object models to explain the entire clutter, then it has the full knowledge of the local indoor environment. In other words, the sensor input z_t can be explained by the structural model M and a set of objects,

$$z_t = G_n(M, \mathbf{x}_t, \{\langle O_j, \mathbf{y}_t^j \rangle | j = 1, 2, ..., N\}) + \epsilon_n \tag{1.6}$$

where ϵ_n is the residual noise. At this level, ϵ_n only consists of noise from the physical sensor and the noise caused by abstracting the environment with concise models. In principle, once ϵ_n becomes IID Gaussian noise $\epsilon_n \sim \mathcal{N}(0, \sigma^2)$, the robot reaches full interpretation of the environment.

1.3 Scope of the Thesis

Section 1.2 describes a general approach for visual scene understanding with different levels of interpretations. This thesis focuses on understanding the *space* of the robot's local environment. This thesis proposes methods for building the structural model M of the environment (Level 0 to 2) and demonstrates the connection with existing object modeling research (Level 3). Specifically, this thesis targets the following:

(a) Formulate a concise, useful, and semantically meaningful representation of the 3D structural model M of the local indoor environment.

(b) Develop a generalized efficient on-line framework to construct M.

(c) Propose algorithms for factoring the description of the environment into M and regions C_i that are not explained by M — clutter.

1.4 Significance

Visual scene understanding from a robotic vision perspective shares concerns with the computer vision perspective, but has significant differences. In computer vision, the common approach to scene understanding is to collect a database of images with human annotated scene semantics, and then use statistical machine learning methods to infer annotations for given input images. The output of visual scene understanding is based on the humans semantics, not the robot's. In robotic vision, the goal is to have the robot build its own interpretation of the local environment from its own experience of moving within it. In other words, the robot cannot rely on additional knowledge sources, like human annotations; instead it must use the stream of visual data it collected within the environment to build its own understanding using geometric properties.

In robotic vision, the input to visual scene understanding is a temporally fine-grained stream of images collected by the robot, instead of a single image or an atemporal image collection. The expected output is a temporally coherent visual scene understanding result, which cannot be achieved by processing each image frame independently. On the one hand, since a large database of visual information related to the robot's semantics is not available as in computer vision, the robot has to incrementally build its visual understanding as observations become available. On the other hand, this thesis leverages the fact that these visual observations are temporally coherent by using geometric and probabilistic inferences to incrementally accumulate its understanding of the local environment.

This thesis proposes a general approach of visual scene understanding from the robot's point of view. The robot incrementally improves its level of understanding as more models become available. While this thesis focuses on understanding the structure of space, the output of the space interpretation helps the robot to focus its attention on understanding objects more efficiently. Many computer vision works have addressed the problem of understanding objects, but real-time computations of these methods are difficult to achieve due to the large search space of object locations. By focusing only on clutter regions when reasoning about objects, these computer vision algorithms can be more efficient. With knowledge of space and objects, the robot can make useful plans to act in the indoor environment.

A full spatial understanding of an indoor environment requires both mapping in large-scale space [8] and scene understanding of small-scale space (this thesis). Scene understanding interprets the local environment, a small portion of the actual environment that is perceivable by a robot within small amount of motions. Mapping in large-scale space takes scene understanding results as inputs to the mapping system, and construct a map that covers a much larger environment. For example, interpreting a corridor and an intersection are considered as scene understanding in small-scale space, while reasoning about a whole floor of a building is considered as mapping in large-scale space. The scene understanding representations and algorithm proposed in this thesis can be efficiently used as inputs to large-scale mapping.

1.5 Thesis Overview

The remainder of this thesis is organized as follows. Chapter 2 discusses related work on visual scene understanding for an indoor navigating robot, our relations to these related works, and our contribution to this problem. Chapter 3 introduces the coordinate systems and filtering techniques that are used throughout this thesis. Chapter 4 presents datasets that we collected to evaluate this thesis. We then introduce our proposed method, an on-line generate-and-test framework, for scene understanding to efficiently construct a concise and useful model of the indoor environment from a stream of visual data. The proposed method is developed and introduced in three stages, each stage corresponds to a chapter.

Chapter 5 introduces the general idea of on-line scene understanding in a simple empty [3] three-wall environment, where each wall is connected to its adjacent walls. In this stage, we start from an initial set of hypothesized models $\{M\}$ of the local environment generated from the first frame of the image stream, and demonstrate a mechanism to test the hypotheses from the stream of visual inputs using a Bayesian filter. (Results presented in this chapter are published in [74].)

Chapter 6 generalizes the simple on-line scene understanding method to empty indoor environments with more complex structures, such as intersections. We propose a representation, the Planar Semantic Model (PSM), to represent such a complex environment. In addition, since the full structure of the local environment may not be observed by the robot all at once, we introduce an incremental process of generating structural hypotheses to describe the same environment with more details. (Results presented in

[3]An empty environment means that the entire environment can be represented by the structural model M.

this chapter are published in [71].)

Chapter 7 further generalizes the on-line generate-and-test framework to handle cluttered indoor environments. The core of this chapter is to demonstrate a testing mechanism that allows a hypothesis to provide a partial explanation of the observations. While Chapter 5 and 6 demonstrate the framework using a stream of monocular images, this Chapter demonstrate the framework using a stream of visual data captured by a depth camera. Thus, the general on-line generate-and-test framework proposed in this thesis is applicable to different types of visual sensors. (Results presented in this chapter are published in [73].)

Chapter 8 and 9 present extensions and applications to the proposed method and the proposed representation, PSM. Chapter 8 presents an attention focusing method to select observations that are informative for testing the hypotheses, and demonstrates that this bias of search towards informative features helps the Bayesian filter to converge to a single hypothesis more efficiently, without loss of accuracy. Chapter 9 moves one step forward from modeling the geometric structure with PSM to reason about the action opportunities of an indoor navigating robot. (Results of these two chapters are published in [72, 70].) Finally, chapter 10 concludes this thesis and suggests future research directions.

Chapter 2

Literature Review

This chapter discusses related work on visual scene understanding for indoor navigating robots, the relation of this thesis to these related works, and our contribution to the problem. Figure 2.1 illustrates the field of visual scene understanding for indoor navigating robots. We analyze these works from two aspects. First, we discuss the representations that these works used to describe the environment (Section 2.1). Second, we analyze these works based on their algorithms (Section 2.2). We summarize these related works and illustrate our contribution to visual scene understanding for indoor navigating robots (Section 2.3).

2.1 Representation

Various representations have been proposed to describe the geometric structure of 3D indoor environments. The relations among these representations are illustrated in the boxes of Figure 2.1. In this section, we discuss these representations in the order of their granularities.

Point Cloud

The most fine-grained representation for geometric scene understanding is a dense 3D point cloud. A 3D point cloud is a set of 3D points in the same 3D coordinate frame. A dense point cloud captures all the details of the modeled environment. Since a dense point cloud requires a lot of memory and computational resources to process, methods have been proposed to efficiently compress a point cloud [38] or to represent a point

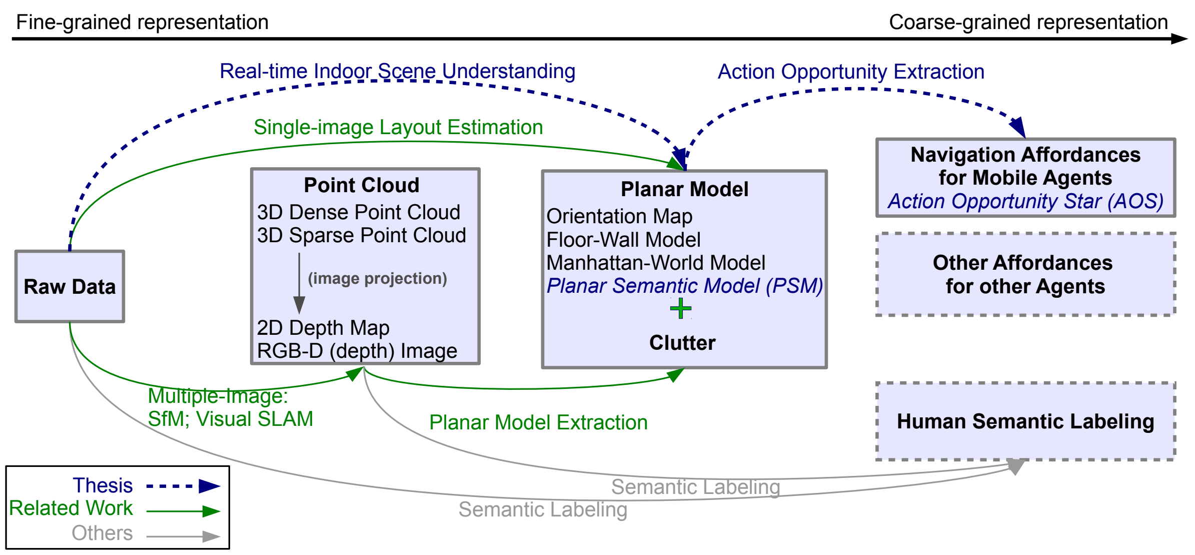

Fine-grained representation
Coarse-grained representation
Real-time Indoor Scene Understanding
Action Opportunity Extraction
Single-image Layout Estimation
Raw Data
Point Cloud
3D Dense Point Cloud
3D Sparse Point Cloud
(image projection)
2D Depth Map
RGB-D (depth) Image
Planar Model
Orientation Map
Floor-Wall Model
Manhattan-World Model
Planar Semantic Model (PSM)
+
Clutter
Navigation Affordances for Mobile Agents
Action Opportunity Star (AOS)
Other Affordances for other Agents
Human Semantic Labeling
Multiple-Image: SfM; Visual SLAM
Planar Model Extraction
Semantic Labeling
Semantic Labeling
Thesis
Related Work
Others

Figure 2.1: Related work on scene understanding. This figure illustrates how existing works and this thesis contribute to visual scene understanding for indoor navigating robots. (Best viewed in color.) Each box is a class of representations that are used in scene understanding. These boxes are ordered based on their granularities. As discussed in Chapter 1, a useful representation for a robot needs to capture critical information related to its goal and its action capabilities. For an indoor navigating robot, a useful representation captures free-space of the local environment and potential navigable directions around the robot. In this thesis, we propose a planar representation, Planar Semantic Model (PSM), to describe the geometric structure of the local environment, and propose an abstraction, Action Opportunity Star (AOS), to represent potential navigable directions from the robot's location. Other robots may have other goals or other action capabilities, and thus, require a different representation (Other Affordances for other Agents). The goal of visual perception for indoor navigating robots is to take the stream of raw visual data (Raw Data) and construct a scene interpretation that supports the robot to make navigation plans (Navigation Affordances for Mobile Agents). Each arrow is a class of methods that extract a model that has a higher granularity from a representation with lower granularity. Green arrows illustrate existing methods that contribute to visual scene understanding for indoor navigating robots. Blue dashed arrows illustrate how this thesis contributes to solving this problem. It is worth mentioning that another line of scene understanding research focuses on interpreting visual inputs in the way that human interprets the world. These methods build predictors that link visual features with human labels (object names or names of the entire scene). While these human-labeling interpretations are useful for consumer products and human-computer interactions, these interpretations do not provide semantic information for navigating robots. Thus, this line of research is not discussed in this chapter.

cloud with lower resolution [61, 16, 39].

A snapshot of a 3D dense point cloud is a 2D depth map, an image where its intensity reflects the 3D depth of the corresponding 2D pixel. There are three common ways to obtain a 2D depth map. First, the raw RGB-D image captured by a depth camera is a 2D depth map. However, since depth cameras use an active sensor to capture depth information, only a limited range of depth is available. Second, from a pair of calibrated stereo images, a depth map can also be constructed, and since it is based on triangulation of two views, there are less limitations on the depth range [1]. Third, from a single projected image, methods [58, 59, 46] have been proposed to reconstruct a depth map by a pre-trained estimator that maps image properties with 3D depth values.

Point clouds and depth maps are powerful representations that are flexible to represent any type of environments, including environments with irregular structures, at any desired resolution. However, man-made environments (e.g. outdoor urban and indoor scenes) are more structured, and therefore a more concise representation can be extracted from the point cloud.

Planar Model

In man-made environments, most of the points in the 3D point cloud lies on a small set of planes. Instead of representing a plane by a large amount of 3D points, it is more efficient to represent a plane by its 3D planar equation. Thus, a concise representation of man-made environments is a planar representation.

The first step towards a planar representation is to represent each 2D image pixel or superpixel with a qualitative local surface orientation (e.g. left, right and front surfaces) relative to the observing pose [30, 31]. The surface orientation map is an initial step to capture one of the most important properties of man-made environments — planes. With additional assumptions on the observing pose, a 3D pop-up model can be reconstructed from the orientation map [32]. However, the surface orientation map is still fine-grained and requires additional steps to construct a more concise planar representation representation. In addition, since the orientation map is view dependent, it may be difficult to combine orientation maps from different observing poses.

A concise planar representation is to describe the environment by a set of planes. In general, the planes follow the Manhattan-world assumption. A Manhattan-world

[1]Once the disparity falls below one pixel, it is impossible to recover the depth. This happens when a point is physically far away from the camera.

model describes a 3D scene based on the Cartesian coordinate systems. Planes in the Manhattan-world model are aligned with two of the three major axes and thus, are either parallel or perpendicular to each other. With the Manhattan-world assumption, outdoor urban environments can be represented by a set of 3D blocks that are aligned with the three major axes [23, 76]. An indoor scene is simply observing the blocks from the inside view. In other words, with the Manhattan-world assumption, an indoor environment is generally represented by the major planes — floor, walls, and ceiling [25, 45, 77, 48, 10, 13, 4, 9, 57].

The image projection of the Manhattan-world model is a set of lines that specify the projected 3D intersecting lines among the planes. These lines can be organized into three groups based on their directions in 3D, where each group corresponds to a major axis. Each group of lines contribute to a vanishing point in the projected image, and with the orthogonal properties, the three vanishing points can be determined from the projected images [45, 40, 55, 48]. Therefore, many works [48, 45, 25, 77] leverage on these vanishing points to construct a Manhattan-world model of urban or indoor environments.

For a robot moving on the ground plane in the indoor environment, the ceiling plane is much less relevant than the ground plane and the walls, so another common planar representation is a floor-wall model [14, 67]. A floor-wall model consists of a ground plane and a set of wall planes that are perpendicular to the ground plane but not necessarily to each other. The floor-wall model is more flexible than the Manhattan-world model but still captures important semantics for a navigating robot. Note that it is also common to represent the urban environment with a floor-wall planar model [6].

In this thesis, we propose the Planar Semantic Model (PSM) [71] with the same assumption (walls are perpendicular to the ground but not necessarily to each other) as the floor-wall model to represent the geometric structure of the indoor environment. PSM is a step forward from previous floor-wall models because it represents richer relations among wall segments. Unlike other planar representations, PSM is capable of representing incomplete knowledge of the local environment so that unobserved regions can be incrementally built as observations become available. Thus, PSM is a representation that fits into an on-line, incremental scene understanding framework for robot perception. The full description of PSM is presented in Chapter 6.

Clutter

For a robot to act in the world, it needs to build and maintain a simple and concise model of that world, from which it can derive safe opportunities for action and hazards to avoid. Unfortunately, the world itself is infinitely complex, containing aspects that are not well described, or even well approximated, by the simple model. An adequate explanatory model must therefore explicitly delineate the part of the environment, clutter, that it does not attempt to explain. For cluttered indoor environments, a hybrid representation that contains a concise planar model for the structure of the indoor environment and a more fine-grained model for clutter is essential to represent the environment.

One common fine-grained representation for clutter is a 3D point cloud [73]. In the projected image space, this 3D point cloud corresponds to a set of 2D image blobs [25, 77]. A more coarse-grained model that represents clutter by a set of 3D cubics has also been widely used [26, 4, 10, 9]. The Manhattan-world assumption can also be applied to the clutter regions. With the Manhattan-world assumptions, these cubics are aligned with the three major axes of the indoor structure.

In this thesis, in order to generalize to any types of cluttered environments, and since the PSM representation is less restricted than a Manhattan-world model, we represent clutter by a set of 3D point clouds. As described in Chapter 1, we first identify 3D points that cannot be described by the PSM, and cluster these 3D clutter points into a set of clutter regions, where each region is represented by a point cloud. By partitioning clutter into smaller regions, classifying the clutter objects and analyzing their functionalities become a set of independent and smaller problems.

Navigation Affordances for Mobile Agents

As described in Chapter 1, a useful representation for a robot needs to capture critical information related to its goal and its action capabilities. For an indoor navigation robot, a useful representation captures geometric structure of the environment that specifies the free-space and potential navigable directions around the robot. While a lot of representations have been proposed to represent the geometric structure of the indoor environments, very few works have focused on modeling the environment to directly reflect the action opportunities of the robot. Methods [82, 24, 37] have been proposed to model common human actions, such as sitting, but only few methods [50, 49] have been proposed to reason about the action opportunities of an indoor navigating robot. While existing works have been focused on building safety maps of environment with uneven ground planes or drop-offs, in this thesis, we propose the Action Opportunity Star (AOS) to describe a set of qualitatively distinctive opportunities for robot navigation at a given location [70]. (We discuss potential ways to incorporate the safety map into our AOS representation in our future work.) Chapter 9 introduces AOS and a method to extract AOS from PSM.

2.2 Algorithms

The algorithms of these scene understanding works can be divided into two categories: single-image approaches and multiple-image approaches. Single-image approaches take a visual snapshot of the environment and produce a 3D or 2D interpretation of the environment in view (Section 2.2.1). Multiple-image approaches take multiple snapshots of the same environment and produce a 3D interpretation of that environment and the observed pose of each snapshot in the same frame of reference of the 3D interpretation (Section 2.2.2).

2.2.1 Single-Image Approaches

From a single projected image, a common approach is to label each pixel with a geometric label. The geometric label can span from different levels of representation, from fine-grained depth maps to planar surfaces (See Section 2.1). A classifier is trained to link the appearance of image pixels or superpixels to a binary classification of ground-plane and wall [13, 14], to a classification of multiple local surface orientation [30, 33], to depth of surfaces in the environment [59], to object labels and thence to depth [46]. In addition to classifying the pixels, a Markov Random Field (MRF) is usually applied to the pixels to smooth out the estimated geometric labels. Dependence on prior training knowledge with relevant domain specific examples makes these methods difficult to generalize to different indoor environments, because appearances in indoor environments are highly variable. In addition, these prior training knowledge require a lot of human effort to provide ground-truth labels. Moreover, real-time performance may be difficult to achieve when estimations and optimizations at pixel or superpixel level are involved.

When constructing a Manhattan-world model from a single projected image, a common approach is to find a projected wire-frame model (layout) that captures the boundary lines among the planes using a hypothesize-and-test approach [48, 45, 25, 77, 57]. These approaches are commonly known as spatial layout estimation. A typical pipeline of the hypothesize-and-test approach consists of three steps: 1) scene configuration estimation; 2) layout hypothesis generation; 3) layout hypothesis testing. The first step, scene configuration estimation, estimates the camera intrinsic parameters (e.g. focal length and principal points) and the three vanishing points that correspond to the three major axis of the Manhattan-world model by clustering image line segments [45, 9]. The second step, layout hypothesis generation, generates a set of projected wire-frame model under the scene configuration by combining line segments in the image that satisfied the constraints of the three vanishing points [25, 44, 45] or by a data-driven method [57]. The third step, layout hypothesis testing, selects the layout hypothesis that best agrees with the image features by a pre-trained scoring mechanism [25, 77, 57] or by comparing with a dense orientation map constructed by these features [45]. With the best layout and the estimated scene configuration, a 3D model can be reconstructed from one single image up to a scale [2].

This hypothesize-and-test approach is vulnerable to features from clutter, image regions that are not representable by the layout, and thus various methods have been proposed to overcome this problem. One method is to train a classifier to determine whether a pixel or a feature belongs to clutter or not, and ignore features that are classified as clutter when testing the hypotheses [25, 77]. In some cases, clutter may contain objects, such as pedestrians or furniture, that are easily detectable through object detectors. Methods [9, 5] are proposed to obtain useful information from the assumption that these objects sit on the ground plane to improve the estimated scene configuration.

Although this thesis is targeting to construct a PSM model, which is less restricted than a Manhattan-world model, our method of generating hypotheses of the indoor environment is greatly inspired by these single-image hypothesize-and-test approaches. Our method generates a set of hypotheses about the 3D environment from a single snapshot by combining line segments in the projected image with certain constraints [74, 71]. Unlike these single-image methods, we test the hypotheses using information accumulated through a temporally contiguous image stream. In addition, in this thesis, we propose a method to test these hypothesized models more efficiently by focusing attention on

[2]The scale is typically the physical camera height, distance from the camera center to the ground plane.

features that are informative [72] (Chapter 8). We demonstrate our attention focusing method on our multiple-image hypothesize-and-test framework, but it is equally applicable to these single-image hypothesize-and-test methods.

There are several issues when applying a single-image scene understanding approach for a robot acquiring a temporally contiguous stream of images. Real-time performance may be difficult to achieve to keep up with the robot need because these methods are usually computationally expensive. In addition, temporally coherent results of the scene estimation may be difficult to achieve if each frame is independently processed. In fact, temporal information provides important cues for constructing the geometric structure of the environment. Furthermore, due to the limited field of view of a camera, the scene captured in the image reflects only part of the robot's immediate surrounding, so it does not provide sufficient information for the robot to make plans. Thus, multiple-image approaches are generally more applicable to visual perception for a robot.

2.2.2 Multiple-Image Approaches

The goal of multiple-image approaches is to construct a 3D structure with its choice of representation to describe the environment captured by the image snapshots and to compute the observed poses of each snapshot in the same 3D coordinate frame. There are two forms of visual input to multiple-image approaches. One form of input is a collection of atemporal snapshots of the same environment, collected from either one sensor or multiple sensors. The second form of input is a temporally contiguous stream of snapshots collected by the same sensor. A robot with vision sensors acquires visual data in the second form. Methods that apply to the atemporal visual snapshot collections can also be applied to a temporally contiguous stream of snapshots. Thus, this section discusses multiple-image approaches from both forms of inputs.

One multiple-image approach is to find the optimal poses and 3D structure from all the images through a batch process. Given a set of projected images, these works [64, 1, 42] typically use a fine-grained representation, point cloud, to describe the 3D structure. There are two major steps in these methods: 1) extract feature correspondences; 2) bundle adjustment optimization. The first step establishes sparse feature correspondences among the images. Features, like SIFT [47] and SURF [7], with view invariant descriptors are widely used to establish correspondences. The second step optimizes 3D locations of all the features and the observed poses of all images using bundle adjustment [69]. The camera intrinsic parameters (e.g. focal length and principle points) can also be estimated during the optimization process if these values are not previously determined. Bundle adjustment minimizes the re-projection error between the image locations of the observed and predicted features. Methods have been proposed to improve the efficiency of bundle adjustment [53, 2, 78], to remove the dependences of feature correspondence [15], and to improve the quality of the reconstructed point cloud [83].

Another multiple-image approach is an on-line process that simultaneously computes the observed pose and maintains a 3D interpretation of the environment that it observed so far, based on the observations that it acquires at each frame. This process is known as visual SLAM (Simultaneous Localization and Mapping) [11, 52, 41, 56]. Like most of the batch processing methods, most of these SLAM methods require a sparse set of features to relate the mapped environment and the observed image. Thus, these methods usually represent the 3D environment by a sparse point cloud. Beside these featured-based mapping, methods have been proposed to construct a dense 3D point cloud of the environment which captures more details than a sparse point cloud from a monocular image stream [51] or from a RGB-D image stream [27, 17]. Due to lack of a batch optimization process in SLAM, the estimated pose and the constructed map may drift overtime. Thus, an important aspects of SLAM is loop closing. When the robot detected that it is back to a location that is previously mapped, a loop closure is detected. Visual features are especially useful for detecting such a loop [3, 29]. Once a loop is detected, various methods [29, 41, 65] have been proposed to rectify the 3D point cloud and the estimated poses.

To allow a robot to navigate efficiently in the indoor environment, 3D interpretation of its local environment must be constructed in real-time (e.g. on-line and efficient) as the robot travels in the environment. Thus, visual SLAM is more applicable to a navigating robot than methods that require batch processes. As described above, existing visual SLAM algorithms produce a point cloud of the environment. A more concise model of the indoor environment needs to be extracted from the point cloud to support the robot to efficiently make plans. Methods [21, 20, 19, 18, 79, 4] have been proposed to extract a planar model from a 3D point cloud under the Manhattan-world assumption. However, this extra step of extracting planar models makes the scene understanding process off-line and computationally intensive, and thus these methods may be difficult to apply to real-time robot navigation. In this thesis, we propose a real-time indoor scene understanding method that directly builds a planar structure (PSM) of the indoor environment from a stream of monocular or RGB-D images.

2.3 Summary

An indoor navigating robot acquires a stream of visual data as it travels within the local environment. The robot must build a concise representation that captures the geometric structure of the indoor environment through an on-line and efficient process. Various representations have been proposed to describe the geometric structure of the 3D environment. Among these representations, planar models are commonly used to model indoor environments because they are concise and capture the major constraints of man-made environments. Thus, an important step for indoor scene understanding is to construct a planar model of the indoor environments.

There are two common paths to construct a planar model from a stream of visual data, given the existing works. The green arrows in Figure 2.1 illustrate the two paths. The first path is to take a frame (a single image) of the data steam and construct a planar model to describe the scene in view. However, temporal information is ignored and a coherent planar model may be difficult to achieve, if each frame is independently processed. The second path is to first construct a 3D point cloud from the image stream, and then, extract a planar model from the 3D point cloud. This path requires multiple iterations through the entire set of visual data making it difficult to apply to an on-line process.

This thesis demonstrates a complete method for visual scene understanding for an indoor navigation robot, and contributes both representations and algorithms toward solving the problem of scene understanding for an indoor navigating robot. The blue arrows and blue texts in Figure 2.1 illustrate our contributions. In terms of representation, we propose the Planar Semantic Model (PSM) [71] (Chapter 6) to describe the geometric structure of the indoor environment, and propose the Action Opportunity Star (AOS) to describe the navigability of the robot's surrounding environment [70] (Chapter 9). PSM is a step forward from a floor-wall model because it captures richer relations among the wall segments. While existing representation captures opportunities for human actions (e.g. sit), AOS describes a set of qualitatively distinctive opportunities for navigation (e.g. go straight or turn left) at a given location. Both PSM and AOS are capable of expressing incomplete knowledge of the local environment so that unknown areas can be incrementally built as observations become available. The ability to express incomplete knowledge makes these representation applicable to incremental scene understanding methods. In terms of algorithm, we propose an on-line generate-and-test framework [74, 71, 73, 72] to efficiently construct the PSM, without the need for prior training, from a stream of monocular images. Our method incrementally generates a set of PSM hypotheses, and test the hypotheses based on their abilities to explain 2D motion of a set of tracked features using a Bayesian filter. We also applied this framework to handle cluttered indoor environment using a RGB-D image stream. Most of this thesis is devoted to introduce and evaluate the proposed framework.

Chapter 3

Technical Preliminaries

3.1 Coordinate Systems

This section defines the coordinate systems and the transformation among the coordinates that are used in this thesis.

3.1.1 Image Space

Image space is a two dimensional coordinate system for each image in the input stream. For a raw image obtained directly by the visual sensors, the origin is at the upper-left corner of the image. The x-axis is pointing towards the right and the y-axis is pointing down. The unit of the raw image space is in pixel.

If the focal length of the physical sensor and the principal point is known in the raw image space, the origin of the image space can be set to principal point, and the unit of the image space can be scaled so that a unit equals to the focal length. We call this coordinate the calibrated image space. A point p in the raw image space $[\mathbf{p}]^{raw} = (u_{raw}, v_{raw}, 1)^T$ and the same point in the calibrated image space $[\mathbf{p}]^{cal} = (u_{cal}, v_{cal}, 1)^T$ is related by

$$[\mathbf{p}]^{raw} = \mathbf{K}[\mathbf{p}]^{col} \tag{3.1}$$

where $\mathbf{K}$ is the intrinsic matrix of the camera,

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_u \\ 0 & f_y & c_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.2}$$

f_x and f_y are the focal lengths expressed in units of horizontal and vertical pixels. (c_u, c_v) is the location of the camera center in the raw image space. The intrinsic matrix can be obtained by existing camera calibration methods [1].

In this thesis, we assume the visual sensor (camera) is calibrated. Thus, we use the calibrated image space and denote a image point as $\mathbf{p} = (u, v, 1)^T$, unless otherwise specified.

3.1.2 Camera Coordinate

Camera coordinate is a three dimensional coordinate system with the x-axis pointing to the right, y-axis pointing down, and z-axis pointing front. We define the *local camera coordinate* to be the camera coordinate with its origin at the camera center of the current frame. The z-axis is aligned to the optical direction of the camera. A point in the local camera coordinate $[\mathbf{P}]^{lc} = (x_{lc}, y_{lc}, z_{lc})^T$ and the same point in the image space $\mathbf{p} = (u, v, 1)^T$ is related by

$$[\mathbf{P}]^{lc} = \lambda \mathbf{p} = \lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = z_{lc} \begin{pmatrix} \frac{x_{lc}}{z_{lc}} \\ \frac{y_{lc}}{z_{lc}} \\ 1 \end{pmatrix} \tag{3.3}$$

where $\lambda = z_{lc}$. We call the transformation from the camera coordinate to the image space *image projection*.

We also define the *global camera coordinate* to be the camera coordinate with its origin at the camera center of the initial frame. The relation between the local and the global camera coordinate is a rigid-body transformation. The transformation can be determined by the difference in camera poses between the current frame and the initial frame.

3.1.3 3D Coordinate

We define the 3D coordinate to be a three dimensional coordinate system with respect to the ground plane. The x-y plane of the 3D coordinate is parallel to the ground plane, and the z-axis is pointing up. We define the *local 3D coordinate* to be the 3D coordinate with its origin at the camera center of the current frame. The x-axis is pointing towards

[1] In this thesis, we use a publicly available calibration toolbox: `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

the direction where the camera is facing, and the y-axis is pointing out from the left hand side of the camera. The transformation from the local camera coordinate $[\mathbf{P}]^{lc}$ to the local 3D coordinate $[\mathbf{P}]^{l3D}$ of a point $\mathbf{P}$ is

$$[\mathbf{P}]^{l3D} = R_c[\mathbf{P}]^{lc} \tag{3.4}$$

where

$$\mathbf{R}_c = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{3.5}$$

We also define the *global 3D coordinate* to be the local 3D coordinate defined by the initial frame. The transformation from the global camera coordinate $[\mathbf{P}]^{gc}$ to the global 3D coordinate $[\mathbf{P}]^{g3D}$ of point $\mathbf{P}$ is

$$[\mathbf{P}]^{g3D} = R_c[\mathbf{P}]^{gc}. \tag{3.6}$$

The transformation between the local and the global 3D coordinate is a rigid-body transformation related to the camera poses between the current frame and the initial frame.

3.1.4 Ground-Plane map

We define the ground-plane map to be a two dimensional coordinate system that describes a slice of the 3D coordinate, where the slice is along the ground plane. We define the *local ground-plane map* to be the sliced local 3D coordinate and define the *global ground-plane map* to be the sliced global 3D coordinate. The ground-plane map location of a point P is the same as the location in the corresponding 3D coordinate without the z component.

3.2 Recursive State Estimators

Recursive state estimators are useful in on-line applications. In this thesis, we use the Bayesian filter and the Extended Kalman Filter (EKF) in various steps in the proposed framework.

3.2.1 The Bayesian Filter

Given a set of hypotheses $M = \{M_1, M_2, ..., M_n\}$, the Bayesian filter is used to recursively estimate the belief among the hypotheses given a stream of observations $\{\mathbf{F}_1, ..., \mathbf{F}_t\}$. Mathematically, the belief of the hypothesis is the posterior probability distribution of the hypotheses. At each time step t, the posterior probability of each hypothesis $p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right)$ can be expressed by the Bayes rule,

$$p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right) = \frac{p\left(\mathbf{F}_t|M_i, \mathbf{F}_1, ..., \mathbf{F}_{t-1}\right) p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_{t-1}\right)}{p\left(\mathbf{F}_t|\mathbf{F}_1, ..., \mathbf{F}_{t-1}\right)} \tag{3.7}$$

or alternatively,

$$p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right) = \eta_t p\left(\mathbf{F}_t|M_i, \mathbf{F}_1, ..., \mathbf{F}_{t-1}\right) p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_{t-1}\right) \tag{3.8}$$

where

$$\eta_t = \frac{1}{p\left(\mathbf{F}_t|\mathbf{F}_1, ..., \mathbf{F}_{t-1}\right)}. \tag{3.9}$$

By making the Markov assumption [2], Equation 3.8 becomes

$$p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right) = \eta_t p\left(\mathbf{F}_t|M_i\right) p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_{t-1}\right). \tag{3.10}$$

The posterior probability at time t is the product of the likelihood function with the current observations $\mathbf{F}_t$ and posterior probability at time $t-1$. At time 0, since there are no past observations, we assign a prior distribution $P(M)$ for the Bayesian filter to initialize. With this formulation, the posterior probability at time t is simply the product of all the past likelihoods and the prior,

$$p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right) = \eta p\left(M_i\right) \prod_{j=1...t} p\left(\mathbf{F}_j|M_i\right) \tag{3.11}$$

where $\eta = \prod_{j=1...t} \eta_j$. To compute the actual posterior probability distribution of the hypotheses, we normalize the posterior probabilities of the hypotheses so they sum up to 1. By doing the normalization, η will be canceled out and thus, has no effect on the posterior probability distribution.

In practice, to compute the posterior probability probability distribution of a set of

[2]The Markov assumption suggests that the past and future observations are independent if the current belief state is known.

hypotheses, the Bayesian filter requires us to define a prior distribution and a likelihood function. Starting with a prior probability distribution, at each time step t, we compute the likelihood and update the posterior probability of each hypothesis. If there are no information for us to set the prior, we can use a uniform prior. Then, we normalize the posterior probabilities of the hypotheses to obtain the (normalized) posterior probability distribution at time t.

3.2.2 The Extended Kalman Filter

To estimate the state μ and its covariance Σ, the Extended Kalman filter is carried out in two stages, prediction and correction. Given the state from the previous frame μ_{t-1}, the prediction stage predicts the state $\hat{\mu}_t$ based on the prediction function $g(\mu_{t-1})$. The predicted covariance $\hat{\Sigma}_t$ is also computed in this stage. In general, the covariance increases in the prediction stage.

Prediction:

$$\begin{aligned} \hat{\mu}_t &= g(\mu_{t-1}) \\ \hat{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + Q_t, \end{aligned} \tag{3.12}$$

where $G_t = \frac{\partial g(\mu)}{\partial \mu}\big|_{\mu=\mu_{t-1}}$ is the Jacobian of the prediction function $g(\mu)$ at μ_{t-1} and Q_t is the prediction noise.

The correction stage incorporates the measurement z_t to update the states. We compute the Kalman gain K_t that specifies the degree to which the measurement is incorporated into the state estimation based on the measurement noise R_t and the predicted state covariance $\hat{\Sigma}_t$. Finally, the state μ_t and its covariance Σ_t is updated based on the Kalman gain and the measurement.

Correction:

$$\begin{aligned} \mu_t &= \hat{\mu}_t + K_t(z_t - h(\hat{\mu}_t)) \\ \Sigma_t &= (I - K_t H_t)\hat{\Sigma}_t \\ K_t &= \hat{\Sigma}_t H_t^T (H_t \hat{\Sigma}_t H_t^T + R_t)^{-1}, \end{aligned} \tag{3.13}$$

where $H_t = \frac{\partial h(\mu)}{\partial \mu}\big|_{\mu=\hat{\mu}_t}$ is the Jacobian of the prediction function $h(\mu)$ at $\hat{\mu}_t$ and R_t is the measurement noise.

Chapter 4

Datasets

This chapter describes the datasets that we used to evaluate the claims in this thesis. Although there are few publicly available datasets for indoor navigating robots collected by a monocular camera or a RGB-D camera [60], these datasets are not suitable or applicable for evaluating this thesis for two reasons. First, these datasets do not fulfill the assumptions that we made about the camera motion. In the thesis, we assume the camera has a fixed height with respect to the ground plane, and the tilt and roll angles are fixed and known. Second, these datasets are collected from a short robot where most of its field of view are the ground plane and are mostly taken from an open environment with objects blocking the way of the robots. Thus, there is less information about the wall planes. Since this thesis proposes methods to represent and reason about environments in terms of ground and wall planes, it is more suitable to have datasets that contain more observations about the wall planes to evaluate our methods. In addition, environments with more complex structures, such as intersections, are required to demonstrate the expressive power of our proposed representations.

We collected three datasets to evaluate this thesis, and all three datasets are made available to the public. The Michigan Indoor Corridor 2011 Video Dataset (Section 4.1) is collected in various simple empty three-wall environments by a monocular camera. This dataset is used to evaluate the general idea of on-line scene understanding and is used to compare our approach with state-of-the-art approaches. The Michigan Indoor Corridor 2012 Video Dataset (Section 4.2) is various empty indoor environments with more complex structures (e.g. intersections). This dataset is used to evaluate most of the ideas in proposed in this thesis. Finally, the Michigan Indoor Corridor 2014 Video Dataset is collected in various cluttered indoor environments by a RGB-D camera. This

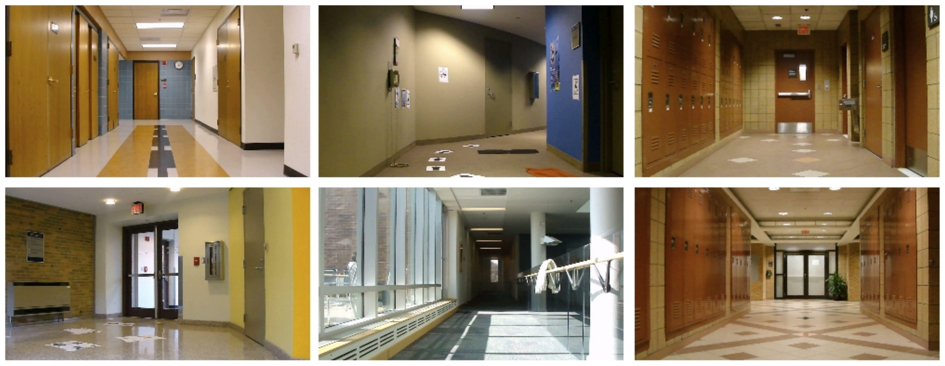


Figure 4.1: The Michigan Indoor Corridor 2011 Video Dataset. The dataset includes indoor environments that violate the Manhattan-world assumption, corridors with glass walls, reflections and partially occluded edges, and rooms with various sizes.

dataset is used to evaluate our proposed method to handle cluttered environments.

4.1 Michigan Indoor Corridor 2011 Video Dataset

The dataset [1] contains 11 video sequences with resolution 1280×720 in various indoor environments acquired with a hand-held camera placed on a wheeled vehicle. The camera was set-up so that there was zero tilt and roll angle with respect to the ground. The camera has a fixed height (about one meter) with the ground throughout the video. The number of frames in each video ranges from 300 to 380, and the frame-rate is about 30 frame per second. The overall motion in each video is about 3 meters moving forward with slight direction changes.

The goal of this dataset is to evaluate how well our on-line testing mechanism discriminates among a set of simple hypothesized planar models. Thus, the dataset were captured in various uncluttered indoor environments with at most three major walls in view. This dataset includes indoor environments that violate the Manhattan-world assumption, corridors with glass walls, reflections and partially occluded edges, and rooms with various sizes. Figure 4.1 shows the first frame of some video sequences. To evaluate scene understanding results quantitatively, this dataset provides a ground truth labeling (i.e. three walls, ground plane and ceiling plane) for all the pixels in the first frame of each video.

[1]The dataset is publicly available at `http://www.eecs.umich.edu/~gstsai/release/Umich_indoor_corridor_2011_dataset.html`.

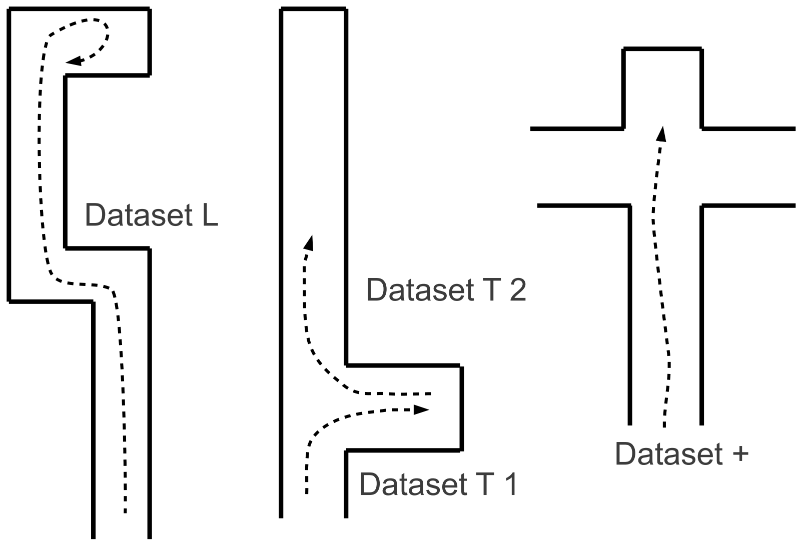


Figure 4.2: The Michigan Indoor Corridor 2012 Video Dataset. There are four video sequences in the dataset. Dataset L contains three L-intersections. In this dataset, the robot traveled through two long corridors connected by two adjacent L-intersections and finally turned at the last intersection. In dataset Dataset T 1, the robot traveled from the major corridor and turned at a T-intersection, whereas in Dataset T 2, the robot traveled from the minor corridor and turned at a T-intersection. Dataset + has one +-intersection, and the robot traveled through the intersection without turning.

4.2 Michigan Indoor Corridor 2012 Video Dataset

The dataset [2] contains four video sequences with resolution 965×400 in various uncluttered indoor environments, such as L-intersections, T-intersections and +-intersections. The videos were collected by a camera that was mounted on a wheeled robot with near zero tilt and roll angle with respect to the ground plane. The field of view of the camera is about 82°. For all datasets, the robot's pose at each frame is provided. [3] Figure 4.2 describes the structure of the environment and the robot motion in each video.

The goal of this dataset is to evaluate the ability of our proposed framework to incrementally generate hypothesized models of the environment and to test the hypothesized models. This dataset also allows us to demonstrate the expressive power of our proposed representation, the Planar Semantic Model. For each test video, we manually labeled the ground truth classification of the planes (i.e. the walls, ground and ceiling plane)

[2]The dataset is publicly available at `http://www.eecs.umich.edu/~gstsai/release/Umich_indoor_corridor_2012_dataset.html`.

[3]We use an occupancy grid mapping with a laser range finder to obtain the robot pose.

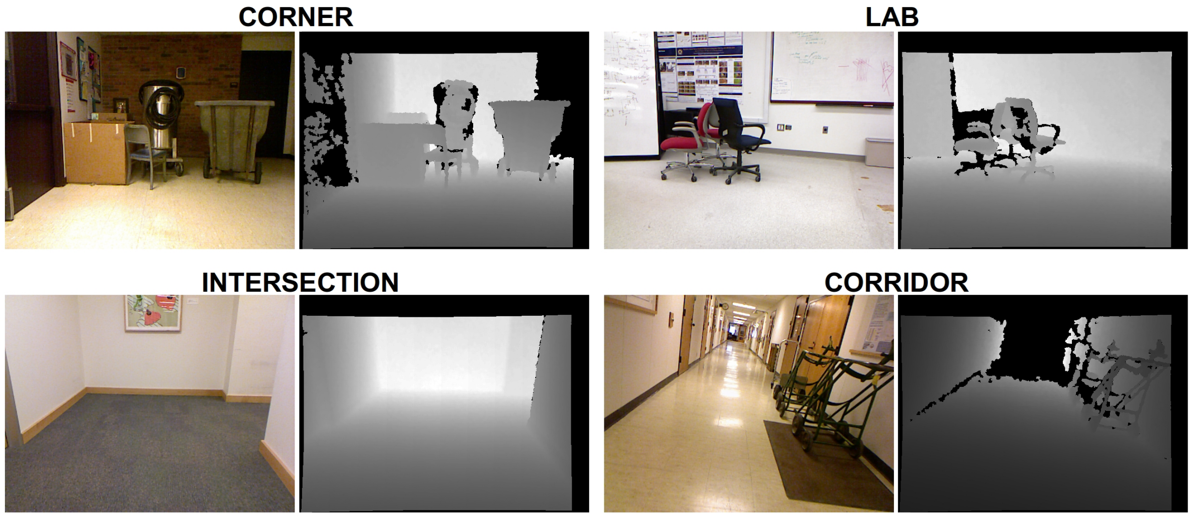


Figure 4.3: The Michigan Indoor Corridor 2014 Video Dataset. The dataset contains four RGB-D videos captured in various indoor environments.

for all pixels every 10 frames in order to evaluate our results quantitatively.

4.3 Michigan Indoor Corridor 2014 Video Dataset

The dataset [4] contains four RGB-D videos captured by a Microsoft Kinect Camera. Figure 4.3 shows some snapshots of the RGB-D videos. The goal of this dataset is to evaluate how well our framework construct an interpretation of a cluttered environment.

The camera was mounted on a wheeled device with a front facing direction. The relative poses between the camera and the ground plane are fixed [5] within each video, but different among different videos. In CORNER and LAB, the robot traveled about 1.5 meter in a very cluttered corner. In INTERSECTION, the robot made a right turn around an empty L-intersection. In CORRIDOR, the robot traveled about 5 meter in a long corridor with objects on both sides.

The videos are annotated every 10 frames to support quantitative evaluations. There are two annotations for each frame. The first annotation labels each pixel with a plane index (e.g. ground plane, ceiling, and wall planes) while ignoring the clutter. This annotation allows us to evaluate the accuracy of the planar model constructed by our

[4]The dataset is publicly available at `http://www.eecs.umich.edu/~gstsai/release/Umich_indoor_corridor_2014_dataset.html`.

[5]The actual poses between the camera and the ground plane for the videos are not provided.

framework. The second annotation is a binary label that specifies whether each pixel is part of the clutter or part of the planar structure. This annotation allows us to evaluate how well our method segments out clutter.

Chapter 5

On-line Scene Understanding

This chapter describes a method for on-line scene understanding using a stream of temporarily contiguous visual input [1]. The visual input that we use here is a short video taken from a calibrated monocular camera. Specifically, this chapter addresses the first level of interpretation

$$z_t = G_1(M, \mathbf{x}_t) + \epsilon_1 \tag{5.1}$$

and assumes the environment has no clutter $\|\epsilon_1\| \simeq 0$. Chapter 7 will discuss how to extend the method to handle clutter. The structural model M consists of a ground plane and at most three planar walls,

$$M = \{G, W_1, W_2, W_3\}. \tag{5.2}$$

where walls $W_1, ..., W_3$ are planes that are perpendicular to the ground plane but not necessarily to each other. The simple model M is useful to describe corridor-like environments but a more sophisticated model is required to describe a more complex structure, such as intersections (see Chapter 6). In addition, we assume the robot moves through the environment with fixed height and zero pitch and roll angles with respect to the ground plane. Thus, the robot pose $\mathbf{x}_t$ only has three degrees of freedom.

The proposed on-line scene understanding method is shown in Figure 5.1. A set of hypothesized models $\mathbf{M} = \{M_1, M_2, ..., M_N\}$ is generated from the first frame of the video by observing image lines (Section 5.1). With our assumptions, we can generate hypothesized models from a single frame, and given a model M_i, we can compute the

[1]Materials presented in this chapter were published in [74].

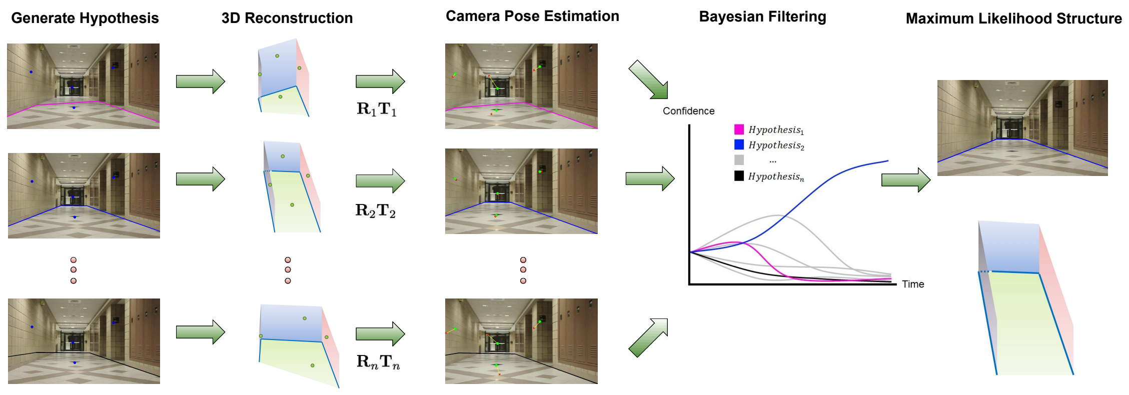


Figure 5.1: Proposed on-line scene understanding method. (Best viewed in color.) The first step is to generate a set of hypothesized model M_i from the first frame of the video. Given any hypothesis, a static model of the 3D planar environment is computed, and the trajectory of robot pose is determined based on the 3D static model and tracked features. Hypotheses are tested based on their abilities to explain the tracked features using a Bayesian filter, and finally the hypothesis with the maximum posterior probability is selected.

3D location of any image point (Section 5.2). At the meantime, a set of image point features are extracted in the first frame and reliably tracked throughout the frames of the video. This extraction and tracking step does not depend on any ground-wall boundary hypotheses. Transforming a hypothesized model M_i from the local to the global frame of reference, the model becomes static across the frames, so the tracked points can be used to estimate robot pose of each frame of the video (Section 5.4).

At this point, relative to each hypothesis M_i, we have both a static planar model of the 3D environment and knowledge of robot pose. Using these, we predict the 2D motion of the image feature points over time, and compare these predictions with the observations $\mathbf{F}_t$, the tracked locations of the point features. This comparison defines the likelihood $p(\mathbf{F}_t|M_i)$ of the observation given the hypothesis, and allows us to update the Bayesian posterior probability distribution over the set of hypotheses (Section 5.5).

This on-line scene understanding method is evaluated on a collection of videos with simple environments (Section 5.6). These experimental results demonstrate the efficiency and accuracy of our on-line method. A comprehensive comparison with state-of-the-art image scene understanding methods and 3D reconstruction methods is also presented in this chapter.

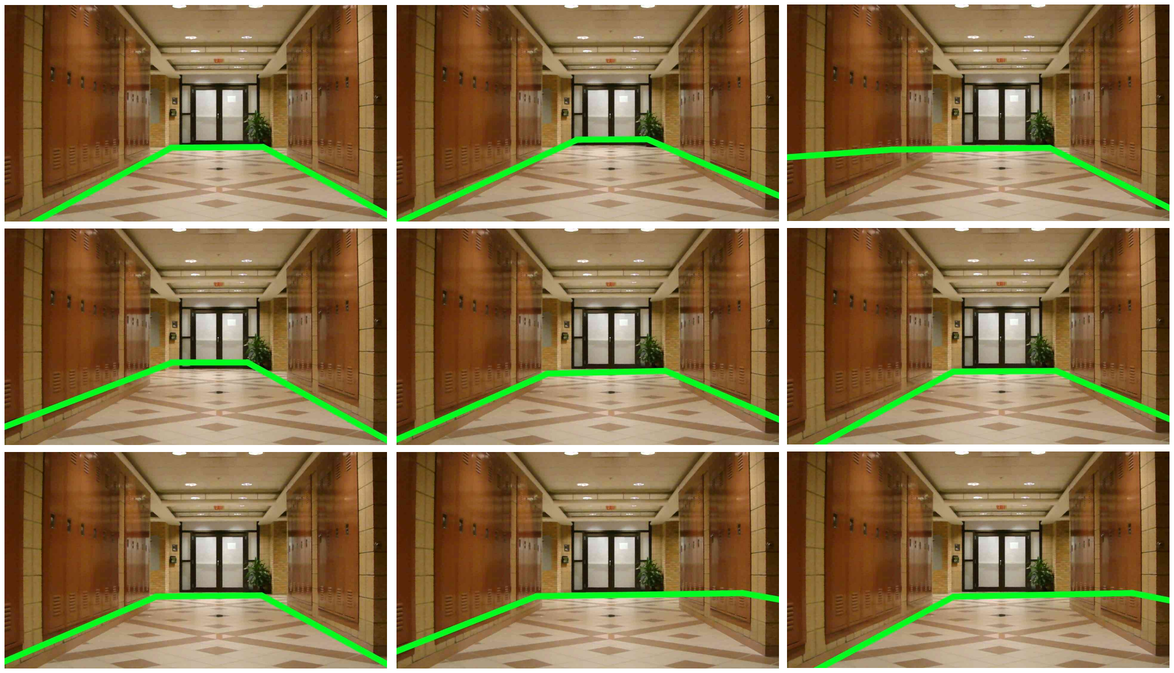


Figure 5.2: Examples of simple hypothesized model. (Best view in color.) Hypotheses are generated by combining line segments in the first frame of the video with certain constraints.

5.1 Hypotheses Generation

The image projection of the ground-wall boundary of a hypothesized model M_i, is a polyline extending from the left to the right image borders. The enclosed region of the polyline with the image lower border specifies the visible portion of the ground plane. A non-vertical line segment in the polyline is a wall plane, and vertical line segments are occluding edges. This chapter focuses on hypotheses that consist of at most three walls (i.e. left, end, and right walls) and the ground plane with no occluding edges. A more generalized model which are capable of representing intersections will be presented in Chapter 6.

To generate the hypotheses, we start by extracting lines below the camera center. The line segments are extracted by edge linking and then merging short segments to form a set of long straight lines [66]. Since the camera is always placed above the ground plane, all the lines within a feasible hypothesis are below the camera center. We remove vertical lines because vertical lines imply occluding edges. Non-vertical lines are divided into three categories (i.e. left, end and right) based on their slopes in the image.

A set of hypotheses can be automatically generated by selecting and linking at most one line from each category. However, some hypotheses are infeasible in the physical world and thus, are systematically excluded from the set.

Hypotheses with wall intersections outside the image borders are excluded because they violate the perspective geometry of indoor structures. In addition, a 3-wall hypothesis is excluded if its left and right walls intersect in front of the end wall. Furthermore, we define the *edge support* of a hypothesis to be the fraction of the length of the image ground-wall boundary that consists of edge pixels. Hypotheses with edge support below a threshold are excluded. Examples of the hypotheses are shown in Figure 5.2.

5.2 3D Construction from Single Frame

In the local camera coordinate, 3D location $\mathbf{P}_i = (x_i, y_i, z_i)^T$ of an image point $\mathbf{p}_i = (u_i, v_i, 1)^T$ is related by $\mathbf{P}_i = z_i\mathbf{p}_i$ for some z_i. If the point $\mathbf{P}_i$ lies on the ground plane [2] with normal vector $\mathbf{n}_g$, the exact 3D location can be determined by the intersection of the line and the ground plane:

$$h = \mathbf{n}_g \cdot \mathbf{P}_i = z_i \mathbf{n}_g \cdot \mathbf{p}_i \tag{5.3}$$

where h is the distance of the optical center of the camera to the ground (camera height).

With the zero pan and roll angle assumption, the normal vector of the ground plane is $\mathbf{n}_g = (0, 1, 0)^T$. By setting $\mathbf{n}_g = (0, 1, 0)^T$ in Equation 5.3, 3D location of any point on the ground plane with image location $\mathbf{p}_i$ can be determined by

$$\mathbf{P}_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = h \begin{pmatrix} \frac{u_i}{v_i} \\ 1 \\ \frac{1}{v_i} \end{pmatrix}. \tag{5.4}$$

Without any additional source of information, the constructed 3D scene is up-to-scale. The scale factor is dependent on the camera height h. If the camera height is set to $h = 1$, a unit in the constructed 3D scene equals to the camera height in the physical world. If h is set to the physical camera height in a specific unit, the constructed 3D

[2] Given a hypothesized model M, we can determine whether a point feature is on the ground-plane or on which wall plane by checking the relation of that point with respect to the projected ground-wall boundary lines.

scene is under that unit.

Within a given hypothesis, a wall plane equation is determined by its corresponding projected ground-wall boundary line segment. We determine the normal vector of a wall plane $\mathbf{n}_w$ based on the ground-wall boundary line segment,

$$\mathbf{n}_w = \mathbf{n}_g \times \mathbf{v}_b \tag{5.5}$$

where $\mathbf{v}_b$ is the 3D direction vector of the boundary line segment. The direction vector $\mathbf{v}_b$ can be constructed by any two points on the boundary line. If a point lies on the wall plane with position $\mathbf{p}_j$ in the image space, its 3D location can be determined by

$$d_w = z_j \mathbf{n}_w \cdot \mathbf{p}_j = \mathbf{n}_w \cdot \mathbf{P}_j \tag{5.6}$$

where d_w can be obtained by any point on the boundary line. Notice that if a point lies on the ground-wall boundary line, both Equation 5.3 and 5.6 must be satisfied.

The geometry described above determines the equations of the wall planes, as well as the 3D location of any given point feature on the image under a hypothesis. Once the 3D model M is determined in the local camera coordinate, we can easily transform it into the global camera coordinate given the robot pose.

5.3 Point Feature Tracking

A set of point features are extracted and tracked across frames. These tracked features are used to estimate robot motion (Section 5.4) and to serve as observations to test the hypotheses (Section 5.5). Note that this extraction and tracking process does not depend on any of the hypotheses.

There are two common methods to extract correspondences between two frames. One method is to extract SIFT [47], SURF [7] or other point features with descriptors in both frames, and then match the point features based on their descriptors. The other method is to extract Harris corners as point features and use KLT [63] to track the features frame-by-frame. In general, the former method is more robust large baselines (view point differences) between two images, while the later method is more efficient but requires small baselines. Since there are usually only very small changes between consecutive frames, it is more efficient to consider the fact that point features only move a little in the image space. Thus, in this thesis, we apply the KLT point extraction and tracking method to obtain feature correspondences across frames.

Some of the point features may be poorly tracked due to the properties of the image or the properties of the physical world. For example, point features with low image intensities may not have a robust image patch for KLT to use to refine in the next frame. Point features at depth discontinuities may be ambiguous to track when the camera is moving towards the features. These poorly tracked point features will misled the rest of the proposed method, and thus, these points need to be detected and removed. After the image location of each point feature is updated, we fit a rigid-body transformation constraint to ensure that the points are properly tracked. Specifically, we use RANSAC to fit a fundamental matrix and identify the outliers. The outliers are the points that are poorly tracked and, thus are removed. Note that even though in general, it is more efficient to fit a Homography matrix between two frames with small baselines, in indoor environments, the Homography may be dominated by a set of points that lie on the same plane.

There are various thresholds that can be set to control the point features that are extracted. We allow a maximum of 300 points to be extracted at one frame. In addition, to encourage point features to be extracted across the entire image, two features need to be at least 20 pixels apart from each other in order to be extracted. Since point features may lose track or go out of sight from time to time, new point features are extracted as they become visible in order to preserve enough observations. In this thesis, we keep track of the number of points at each frame and extract new point features when the number of features drops below a threshold (30 points). Chapter 8 presents an efficient method to select point features that makes the hypotheses testing process more efficient.

5.4 Pose Estimation

To estimate the robot motion, a set of point features are extracted and tracked across frames. Since the 3D locations of the feature points are static in the global frame of reference, robot motion between two frames can be estimated by aligning the 2D tracked points. This robot motion estimation is equivalent to estimating the rigid-body transformation of the 3D point correspondences from a still camera under the local frame of reference. In this chapter, the robot is assumed to be moving parallel to the ground plane, so the estimated rigid-body transformation of the points contains three degrees of freedom, $(\Delta x, \Delta z, \Delta \theta)$, where Δx and Δz are the translations and $\Delta \theta$ is the rotation around y-axis. With an additional assumption that there are at least two point feature correspondences on the ground plane, the robot pose between two frames can be determined by aligning the 3D points with a rigid-body transformation.

We first construct the 3D locations of the point features for both frames individually under a hypothesis M_i. Given two corresponding 3D point sets in the local camera coordinate, $\{\mathbf{P}_i = (x_i^P, y_i^P, z_i^P)^T\}$ and $\{\mathbf{Q}_i = (x_i^Q, y_i^Q, z_i^Q)^T\}$, $i = 1...N$, in two frames, the rigid-body transformation is related by $\mathbf{Q}_i = \mathbf{R}\mathbf{P}_i + \mathbf{T}$ where $\mathbf{R}$ is the rotation matrix, and $\mathbf{T}$ is the 3D translation vector. The rotation matrix $\mathbf{R}$ has one degree of freedom, of the form

$$\mathbf{R} = \begin{pmatrix} \cos(\Delta\theta) & 0 & \sin(\Delta\theta) \\ 0 & 1 & 0 \\ -\sin(\Delta\theta) & 0 & \cos(\Delta\theta) \end{pmatrix} \tag{5.7}$$

and the translation vector $\mathbf{T} = (t_x, 0, t_z)^T$ has two degrees of freedom. The rotation matrix can be estimated by the angular difference between two corresponding vectors, $\overrightarrow{\mathbf{P}_i\mathbf{P}_j}$ and $\overrightarrow{\mathbf{Q}_i\mathbf{Q}_j}$. Our estimated $\Delta\theta$ is thus the weighted average of the angular differences,

$$\cos(\Delta\theta) = \frac{1}{\sum \omega_{ij}} \sum_{i \neq j} \omega_{ij} \frac{\overrightarrow{\mathbf{P}_i\mathbf{P}_j} \cdot \overrightarrow{\mathbf{Q}_i\mathbf{Q}_j}}{\|\overrightarrow{\mathbf{P}_i\mathbf{P}_j}\| \|\overrightarrow{\mathbf{Q}_i\mathbf{Q}_j}\|} \tag{5.8}$$

where ω_{ij} is defined as

$$\omega_{ij} = \frac{(1/z_i^P + 1/z_i^Q)}{2} \frac{(1/z_j^P + 1/z_j^Q)}{2}. \tag{5.9}$$

Since the constructed 3D positions of distant points are less accurate, they are given lower weights. The translation vector $\mathbf{T}$ is then estimated by the weighted average of the differences between $\mathbf{R}\mathbf{P}_i$ and $\mathbf{Q}_i$,

$$\mathbf{T} = \frac{1}{\sum \omega_i} \sum_{i=1}^{N} \omega_i (\mathbf{Q}_i - \mathbf{R}\mathbf{P}_i) \tag{5.10}$$

where the weight w_i is defined as

$$\omega_i = \frac{1/z_i^P + 1/z_i^Q}{2}. \tag{5.11}$$

Similar to the rotation estimation, the weight is inversely proportional to the distance of the points to the robot.

The pose change of the robot in the global camera coordinate can be determined based on the estimated rotation $\mathbf{R}$ and translation $\mathbf{T}$ of the 3D feature point locations under the local camera coordinate. If we set the robot location of one frame to $(0, 0, h)^T$ with zero pan along the x-axis, the robot pose, $(x_c, z_c, \theta_c)^T$, of that frame is $(0, 0, 0)^T$. Then, the robot pose in the other frame becomes $(t_x, t_z, \Delta\theta)^T$, where t_x and t_z are the x and z components of $\mathbf{T}$ in the camera coordinates.

Since different hypotheses may assign a different set of points as ground-plane points, robot poses computed under different hypotheses may be different. For a correct hypothesis, the estimated robot pose is definitely correct (similar to the actual robot pose with reasonable amount of error). For an incorrect hypothesis, its estimated robot pose may be wrong, because the hypothesis may assign a point to be on the ground plane while the point is actually on a wall. If this occurs, the 3D relation of the point correspondences is no longer a rigid-body transformation, and thus, the estimated robot motion will be wrong. However, in our method, it is acceptable for incorrect hypotheses to have incorrect pose estimations. In fact, the hypotheses are tested based on both their abilities to constructing 3D point features correctly and their abilities to estimate robot pose (see Section 5.5).

The robot motion between two frames can also be estimated using both point features on the ground plane and point features on the wall planes. The 3D location of the wall-plane points are projected onto the ground plane to compute the robot motion. However, in our experiments, given the correct hypothesis, the 3D construction of wall-plane points are less accurate than the ground-plane points, and thus, the robot pose is less accurate when using all the point correspondences.

5.5 Hypotheses Testing using Bayesian Filtering

At this point, relative to each hypothesis, we have both a static model of the 3D environment and knowledge of robot motion. Using these, we can predict the motion of the image feature points over time, and compare these predictions with the tracked features. The set of hypotheses are tested using Bayesian filtering. The first frame of the video is defined as our reference frame and compare it with each of the other frames. For each frame, the likelihood for each hypothesis with respect to the reference frame is computed. Using a Bayesian filter allows us to accumulate the likelihoods from all the frames in order to select the hypothesis with the maximum posterior probability at the end of the video.

Given a set of hypotheses, $\mathbf{M} = \{M_1, ..., M_N\}$, the posterior probability distribution over the hypotheses at each frame t can be determined by a recursive Bayesian filter,

$$p\left(M_i|\mathbf{F}_1, ..., \mathbf{F}_t\right) = \propto p\left(M_i\right) \prod_{j=1...t} p\left(\mathbf{F}_j|M_i\right) \tag{5.12}$$

where $\mathbf{F}_t$ is the set of features whose tracked and predicted locations are compared at frame t.

If we have no information about the correctness of the hypotheses from the reference frame, the prior probability $p\left(M_i\right)$ in Equation 5.12 is uniformly distributed over all the hypotheses. If prior knowledge of the structure in the scene (e.g. width between the side walls) is given, the prior probability distribution can prefer one hypothesis over the others.

For each time step, the observation $\mathbf{F}_t$ is a set of evidence from the feature points, $\mathbf{F}_t = \{o_1^t, o_2^t, ...o_{n_t}^t\}$, that are visible in frame t. The likelihood of an individual point o_j^t at image location $\mathbf{L}(o_j^t)$ is modeled by a normal distribution with mean at the predicted image location $\hat{\mathbf{L}}(o_j^t)$ in frame t. $\hat{\mathbf{L}}(o_j^t)$ and $\mathbf{L}(o_j^t)$ are related by the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$ as described in Section 5.4. Since the likelihood is only depending on the distances between $\mathbf{L}(o_j^t)$ and $\hat{\mathbf{L}}(o_j^t)$, the individual likelihood is equivalent to modeling the prediction error between the two with a zero mean normal distribution with variance σ. By combining the likelihoods from individual points, the likelihood of hypothesis M_i at time step t is,

$$p\left(\mathbf{F}_t|M_i\right) \propto \prod_{j=0}^{n} \exp \frac{-||\hat{\mathbf{L}}(o_j^t) - \mathbf{L}(o_j^t)||^2}{2\sigma^2}. \tag{5.13}$$

5.6 Evaluation

Our on-line scene understanding method is evaluated using the Michigan Indoor Corridor 2011 Video Dataset (See Chapter 4). The efficiency of our method is shown in Table 5.1. The computational time is related to the number of point features and the number of hypotheses as shown in the table. Our method runs in real-time and the computational time (in C/C++ using an Intel Core 2 Quad CPU 2.33GHz) is less than the video length.

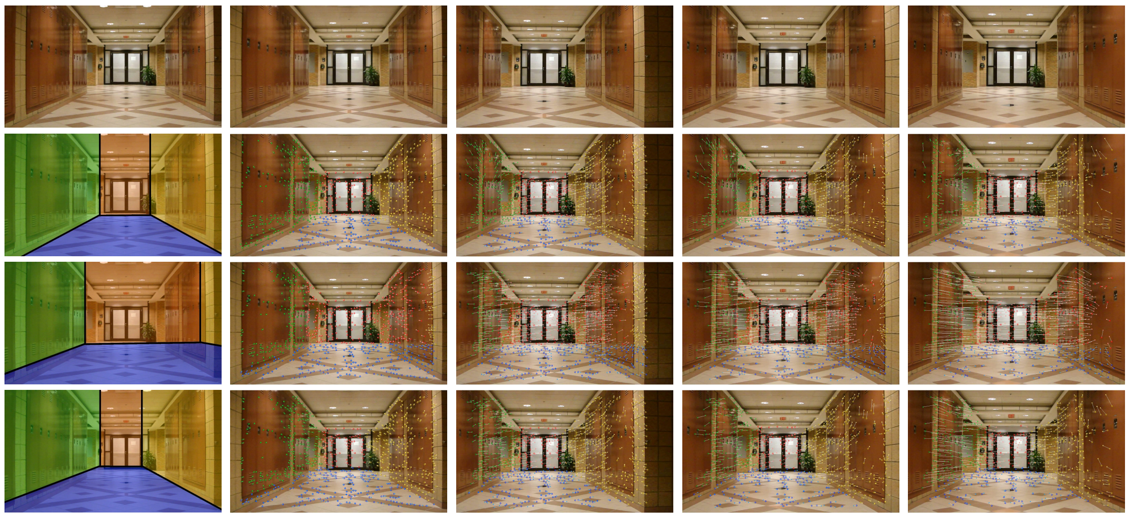
Datasets	Number of Features	Number of Hypotheses	Video Length	Computational Time
EECS Building	256	15	10 s	5.77 s
Library 1	233	12	10 s	6.18 s
Locker Room	245	16	10 s	6.70 s
Non-parallel 1	286	13	10 s	6.49 s
Non-parallel 2	268	8	11.67 s	7.60 s
Basement	282	9	10 s	7.27 s
Study Room	248	15	10 s	8.18 s
Library 2	250	20	12.67 s	7.84 s
Glass Wall	403	10	10 s	7.34 s
Object	336	15	10 s	7.50 s
Two Walls	242	10	10 s	5.37 s

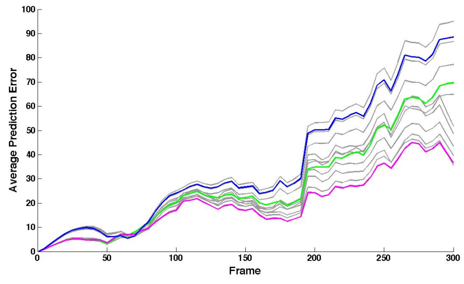
Table 5.1: Computational time analysis. The computational time is related to the number of point features and the number of hypotheses. The proposed on-line scene understanding method is real-time and efficient.

The initial frame requires the most computation because of the hypotheses generation process. In average, the time required to process the first frame is 210 ms. The rest of the frames runs in 50 Hz (about 18 ms per frame).

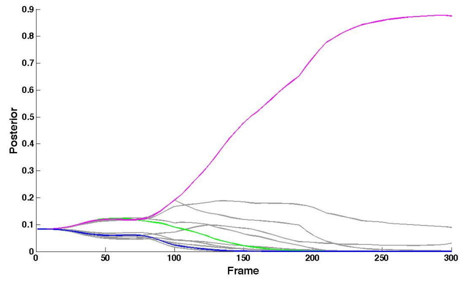
Figure 5.3 shows our results in hypothesis generation, 2D point feature motion prediction and Bayesian filtering. Since our ground-wall boundary hypotheses do not model the ceiling plane, feature points from the ceiling plane will be misleading in the evaluation. These points are excluded using essentially the technique used to identify the ground plane in Section 5.1. Even though the overall error increases with motion due to the quality of feature tracking, hypotheses that are closer to the actual indoor structure have relatively low errors compared to others since the hypotheses are evaluated based on the same set of feature points.

To compare our method to state-of-the-art scene understanding methods, we apply the indoor classifier in [31] and the box layout estimator in [25] to the first frame of each video. Furthermore, we extended the method in [31] by applying it to the same subset of frames that our method used (e.g. 60 frames out of 300), and combined the labels across frames using a spatial-temporal Markov Random Field linking superpixels within and across the frames, similar to [80]. We refer to these results as "[31]+MRF". Notice that adding temporal information to [31] does not necessarily improve the result in the first frame because incorrect labels in later frames affect the label in the first frame.


(a)



(b) Prediction Error



(c) Posterior Probability after each Frame

Figure 5.3: Example of hypotheses testing with Bayesian filtering. (Best viewed in color) The hypotheses are tested based on their abilities to explain the 2D motion of tracked features. (a) The top row are frames 1, 50, 100, 150 and 200 from the Library 1 video. The bottom rows are examples of our hypothesis generated in the first frame (first column) and the predicted locations (crosses) and tracked locations (circles) of the feature points on each frame. The discrepancy of the locations are the white lines. (b) The overall trend of the prediction error increases with the motion due to feature tracking quality. The number of existing tracked features decreases as the motion increases and these features are mostly from the distant area in the first frame which has low resolution. The abrupt increase at Frame 185 is because of the sudden camera movement which reduces the tracking quality. (c) All hypotheses are equally likely in the first frame. Hypotheses with low accuracy drop significantly in the first few frames, while the one with the highest accuracy gradually stands out among the rest. The most accurate hypothesis need not to be the one with minimum prediction error all the time in order to get the maximum posterior probability in the end.

Dataset	Our Method	[31]	[31]+MRF	[25]
EECS Building	**85.78%**	85.49%	84.57%	79.15%
Library 1	**91.13%**	88.13%	83.10%	83.32%
Locker Room	**97.91%**	71.49%	83.18%	87.72%
Non-parallel 1	**89.61%**	86.23%	86.47%	53.10%
Non-parallel 2	84.89%	**85.44%**	60.16%	66.11%
Basement	**99.71%**	72.82%	77.99%	89.79%
Study Room	**95.59%**	76.43%	72.90%	89.23%
Library 2	**88.39%**	88.38%	84.78%	78.42%
Glass Wall	85.56%	58.87%	64.39%	**87.55%**
Object	**94.73%**	94.45%	88.62%	88.16%
Two Walls	**97.71%**	92.02%	91.80%	95.63%
Average	**92.09%**	82.07%	79.62%	81.96%

Table 5.2: Quantitative analysis of the on-line scene understanding method. We compare our results quantitatively with [25] and [31], and we further extend [31] to incorporate temporal information in order to make a fair comparison. (See text for more detail) Note that while evaluating [31], the most likely label is assigned to each pixel and pixels with most likely label “sky”, “porous” or “solid” are excluded in the evaluation. While evaluating [25], pixels with label “ceiling” are excluded in the evaluation.

Thus, in order to maintain a temporarily coherent scene understanding results, visual processing must be on-line.

Figure 5.4 shows our performances in various indoor environments in which we demonstrated our capability to deal with non-Manhattan world structures, as well as noisy feature points. We also show qualitative comparisons with state-of-the-art scene understanding methods. Dataset Non-parallel 1 and Non-parallel 2 demonstrate our capability to identify non-Manhattan world structures, unlike [25]. Furthermore, our simple ground-wall models enable us to ignore objects that stick out of the wall as in Dataset Locker Room, Non-Parallel 1 and Object. Dataset Object also demonstrates our capability to deal with a partially occluded ground-wall boundary. Our method is a generalized framework that can deal with any number of walls by generalizing the hypothesis generation (Two Walls). Our method works fairly well even with noisy feature due to reflections (Glass Wall). Compared to [31] and [25], our method generalizes across different environments since we do not rely on any training from the image properties, which can be scene sensitive.

To evaluate the method quantitatively, we assign a label to each pixel in the first frame of the video according to the maximum *a posteriori* hypothesis at the final frame.

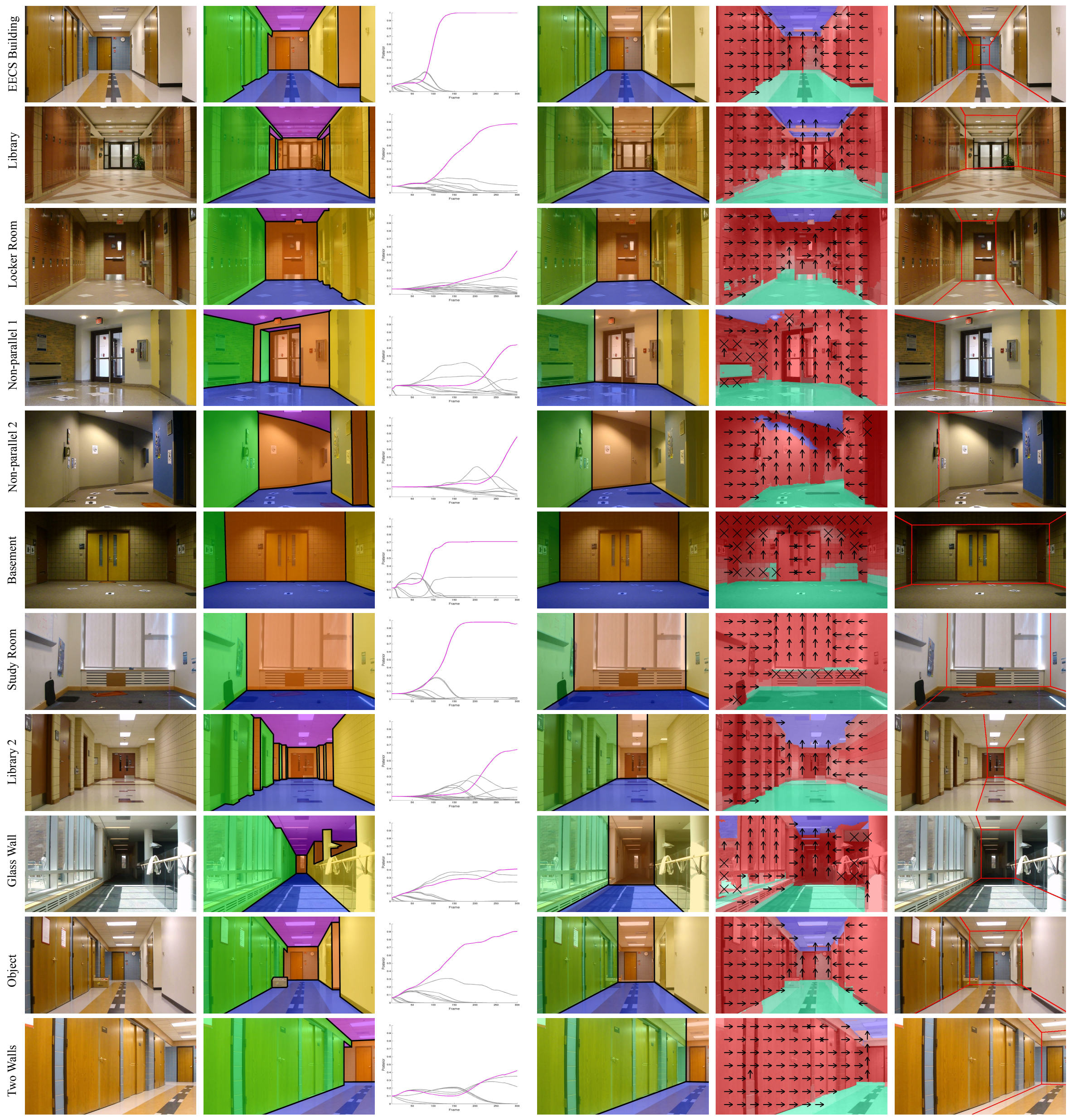
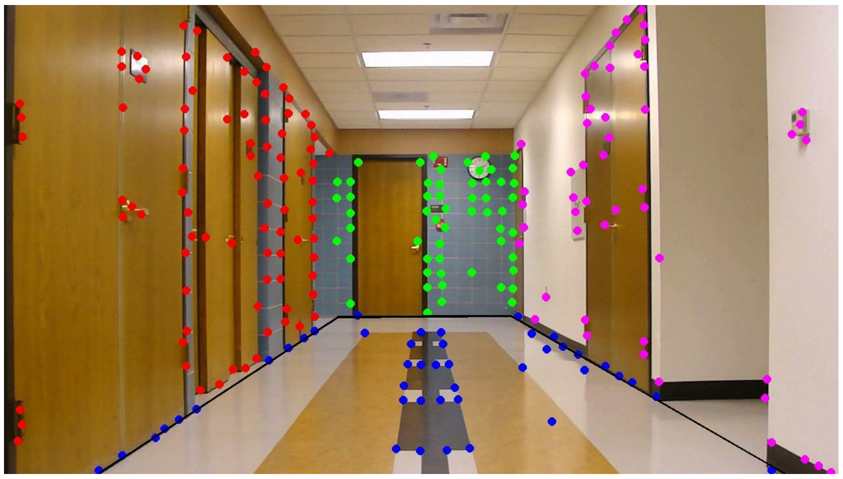


Figure 5.4: Qualitative evaluation of the on-line scene understanding method. (Best viewed in color). First column: first frame of each video. Second column: ground truth. Third column: posterior probability distribution over the hypotheses. The hypothesis with maximum posterior probability at the end of the video is shown in pink. Fourth column: visualize hypothesis with maximum posterior probability. Fifth column: results from [31] on the first frame of the video using their classifiers trained for indoor images. Last column: results of box layout estimation from [25] on the first frame of the video. (See text for more detail.)

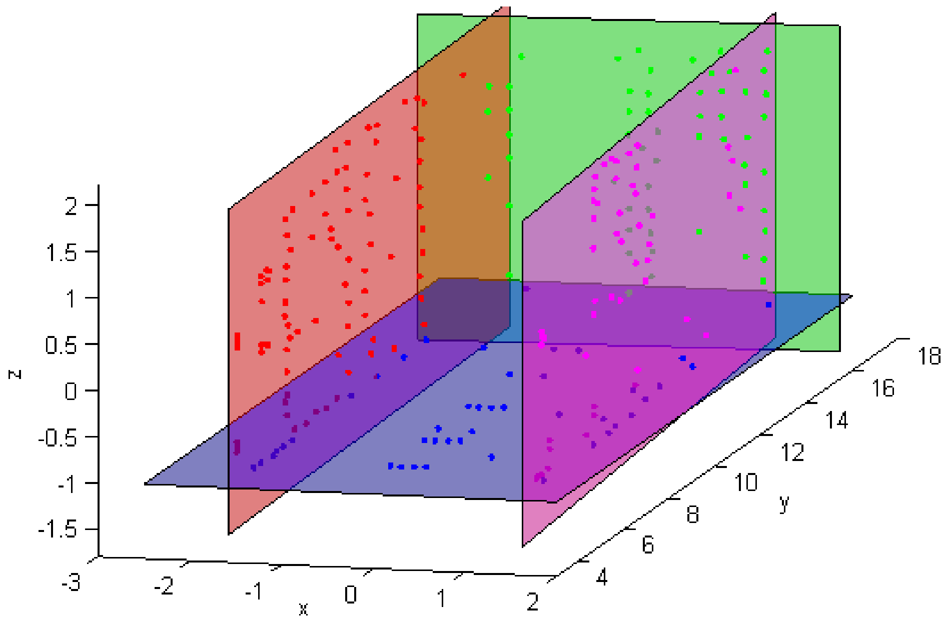
The label are compared with the ground-truth annotation from the dataset. The accuracy of a hypothesis is defined to be the percentage of the pixels that have the correct labeling in the first frame. Since the ceiling plane is not included in our hypotheses, we omitted the ceiling pixels from our evaluation.

Our quantitative results are reported in Table 5.2. Applying all four methods to the videos, we obtained a mean accuracy of 92.09% for our method, a mean accuracy of 82.06% for [31] in its original single-image mode, a mean accuracy of 79.62% for [31]+MRF and a mean accuracy of 81.96% in [25]. One reason for this substantial difference is that [31] and [25] depend strongly on training data, which is likely to be specific to certain environments. By contrast, our method applies a very general geometric likelihood criterion to planar hypotheses. In addition, [25] uses the "box" assumption, while our model M does not require the walls to be perpendicular to each other.

Even though the focus of this thesis is on scene understanding, we compare our 3D planar model with multiple image reconstruction approaches, Bundler [64] and PTAM [41], as shown in Figure 5.5 and 5.6. Bundler [64] has trouble with simple forward motion because it only considered SIFT points that frequently appear among the image set for 3D reconstructions and camera pose estimation. Thus, only the far end of the corridor was reconstructed. Our approximate 3D reconstruction is comparable with [41], but in addition to point clouds, our model provides semantic information about the indoor environments (e.g. walls and ground plane). We also apply J-linkage [68] to fit planes to the 3D point clouds from [64] and [41]. These results do not contribute meaningful information for indoor scene understanding, because the plane-fitting process is easily misled by accidental groupings within the point cloud. Our hypothesis generation process focuses on semantically plausible hypotheses for indoor environments. In addition, the extra step for plane extraction makes these method off-line, while our method is an on-line process that generates a planar model.

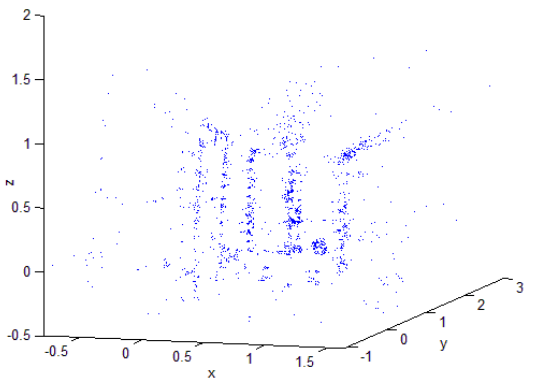


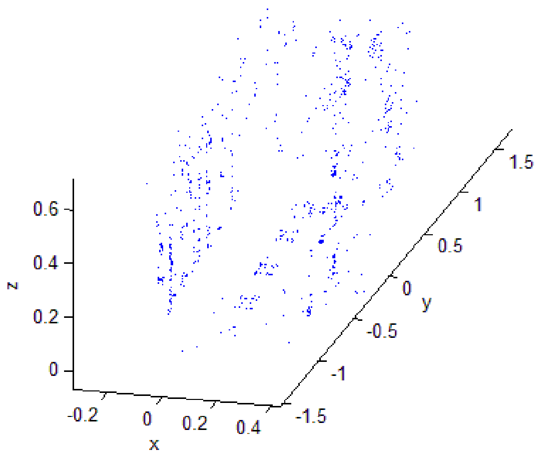
(a) Input video

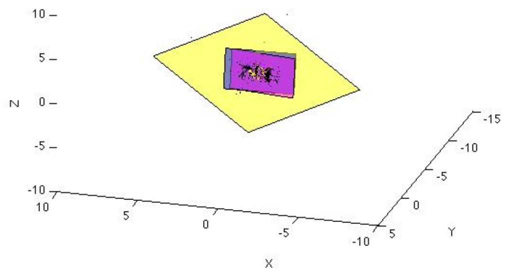


(b) Our method

Figure 5.5: 3D planar model constructed by our method. (Best viewed in color.) (a) First frame of our input video and maximum *a posteriori* hypothesis (black line) determined by our method using frame 1 to 300. Based on that hypothesis, point features are classified into ground plane (blue) and left (red), front (green) and right (pink) walls. (b) 3D planar model and 3D locations of the point features constructed using the geometry described in Section 5.2. 3D reconstruction results from other multiple-image methods are shown in Figure 5.6.


(a) Bundler [64]


(b) PTAM [41]


(c) Bundler [64]+fit plane

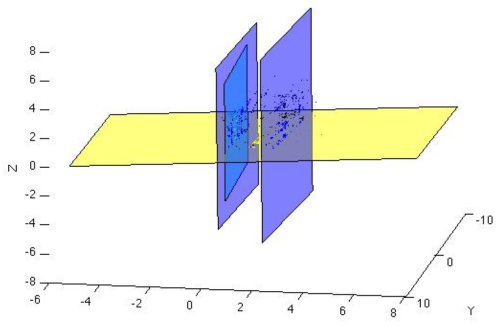

(d) PTAM [41]+fit plane

Figure 5.6: 3D model constructed by other multiple-image methods. (Best viewed in color.) Our approximated 3D planar model of the same video is shown in Figure 5.5. (a) 3D point cloud reconstructed by Bundler [64] using frame 5, 10, ..., 300 in the video. Only the distant area of the corridor are reconstructed because Bundler only considered SIFT feature points that frequently appear among the image set. (b) 3D point cloud reconstructed by PTAM [41] using frame 1 to 300 where the first 10 frames are used for initialization. (c) (d) Apply J-linkage [68] to fit planes to the point clouds. (See text for more detail.)

5.7 Summary

In this chapter, we introduce the general idea of on-line scene understanding to construct a planar model of an empty indoor environment with simple structures. We apply single-image geometric methods to the initial frame of a video to propose a set of plausible three-wall hypotheses for explaining the 3D structure of the environment. Within the context of each hypothesis, our method uses the 3D structural hypothesis plus camera motion to predict the image-space motion of a set of tracked features. A likelihood function for the observed features, given each hypothesis, can be computed from subsequent frames. The Bayesian posterior probability distribution is updated using these likelihoods from each subsequently analyzed frame in the video, almost always leading to rapid identification of the a single best hypothesis.

We demonstrate qualitative and quantitative results on (monocular) videos collected of motion in a variety of three-wall indoor environments, including non-Manhattan-world environments and ones with glass walls and windows, shadows and other difficult image features. Our experimental results suggest that our method is capable of an unprecedented combination of accuracy and efficiency. Furthermore, we compare our on-line framework to state-of-the-art single-image layout estimation methods and multiple-image 3D reconstruction methods. These comparisons demonstrate our method achieves better accuracy in terms of modeling planar environments and is more applicable to real-time applications.

Chapter 6

Incremental Scene Understanding

This chapter extends the on-line scene understanding method to an on-line generate-and-test framework to incrementally model environments with structures that are more complex than a three-wall corridor [1]. Similar to Chapter 5, this chapter addresses the first level of interpretation and assumes the environment is uncluttered $\epsilon_1 \simeq 0$,

$$z_t = G_1(M, \mathbf{x}_t) + \epsilon_1. \tag{6.1}$$

Extending from the simple three-wall model for M, this chapter presents the Planar Semantic Model (PSM) that captures more information of a wall to delimit where the wall is present and where there is an opening (Section 6.1). In order to focus our research to scene understanding, we assume the robot pose $\mathbf{x}_t$ is given. In the global 3D coordinate, the robot pose is denoted as $\mathbf{x}_t = (x_t^r, y_t^r, z_t^r, \theta_t^r, \phi_t^r, \psi_t^r)$. We assume that the robot has a fixed and known tilt ϕ^r and roll ψ^r angles with respect to the ground plane, and has a fixed height z^r from the ground plane. Thus, the robot pose is simplified to $\mathbf{x}_t = (x_t^r, y_t^r, \theta_t^r)$.

Building on top of the on-line scene understanding method presented in Chapter 5, this chapter introduces a continual, incremental process for transforming a current structural hypothesis into children hypotheses describing the same environment in more details. (Section 6.2) This incremental hypothesis generation process is a key to scene understanding for a navigating robot, because many details about the environment, such as an opening, may not be visible in the image from a distance. For example, an opening of a L-intersection at the end of a long corridor may not be visible from a distance, and

[1]Materials presented in this chapter were published in [71].

thus, evidences about that intersection can only be visible when the robot is closer to the intersection. In addition to the incremental hypothesis generation, this chapter presents our method of using the information in current frame to refine the quantitative precision of existing hypotheses. (Section 6.3) Furthermore, we discuss how to incorporate newly generated children hypotheses into the Bayesian filtering framework to continuously model the indoor environment. (Section 6.4) Evaluation of the framework is presented in Section 6.5.

Figure 6.1 illustrates the on-line generate-and-test framework. Starting from an initial set of PSM hypotheses, the Bayesian filtering framework identifies the best hypotheses and removes hypotheses with low posterior probabilities. As the robot travels along the environment, a set of children hypotheses are generated from existing hypotheses to describe the same environment with more details. The Bayesian filtering framework continuously tests each new set of hypotheses.

6.1 Planar Semantic Model (PSM)

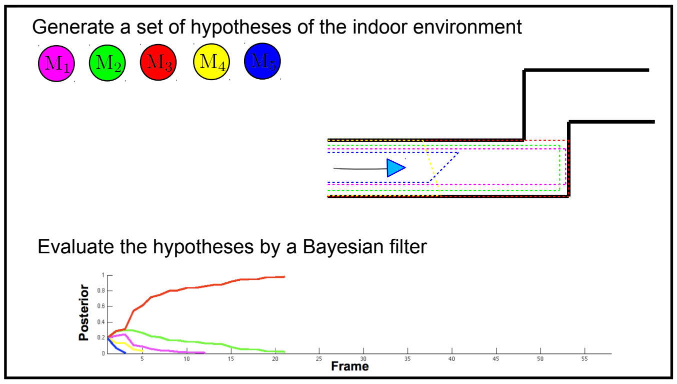
The Planar Semantic Model (PSM) is a concise and useful representation of an indoor environment that describes the environment by a set of meaningful planes — the ground plane G and a set of planar walls W_i that are perpendicular to the ground plane but not necessarily to each other [2]. Formally, PSM, M, is defined as,

$$M = \{G, W_1, W_2, W_3, ..., W_n\}. \tag{6.2}$$

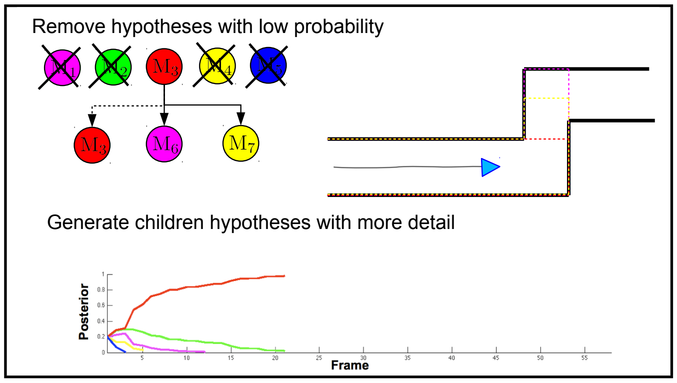
where n is the number of walls in the environment. There is a one-to-one correspondence between this representation and a set of lines (the ground-wall boundaries) in the ground plane (the ground-plane map), represented in the same 3D coordinate.

A wall W_i contains a set of disjoint wall segments that are co-planar in 3D. In the ground-plane map, the wall plane is represented by a line parametrized by (α_i, d_i). $\alpha_i \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ is the orientation of the line which implies the normal direction of the wall plane in the 3D coordinate, and $d_i \in \mathbb{R}$ is the directed distance from the origin of the ground-plane map to the line. Since the walls are perpendicular to the ground plane, the normal vector of the wall $\mathbf{N}_i$ in the 3D coordinate is $\mathbf{N}_i = (\cos\alpha_i, \sin\alpha_i, 0)$, and

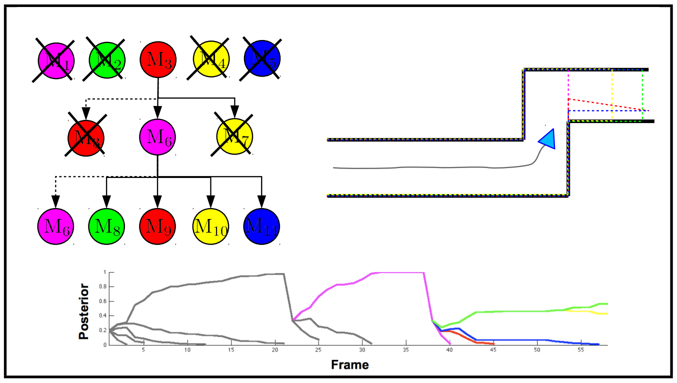
[2]For a robot moving on the ground plane, the ceiling is much less relevant than the ground plane and the walls, so it can safely be omitted from the representation. An indoor flying vehicle would require us to extend this representation to include the ceiling.


Generate a set of hypotheses of the indoor environment
M_1 M_2 M_3 M_4 M_5
Evaluate the hypotheses by a Bayesian filter
Posterior
Frame

(a)


Remove hypotheses with low probability
M_1 M_2 M_3 M_4 M_5
M_3 M_6 M_7
Generate children hypotheses with more detail
Posterior
Frame

(b)


M_1 M_2 M_3 M_4 M_5
M_3 M_6 M_7
M_6 M_8 M_9 M_{10} M_{11}
Posterior
Frame

(c)

Figure 6.1: On-line generate-and-test framework. (Best viewed in color.) A set of qualitatively distinct PSM hypotheses are generated through an incremental process, and tested through a Bayesian filter based on their abilities to explain the 2D motion of a set of tracked features. (a) Starting from a set of simple parent hypotheses, the Bayesian filtering framework identifies the best hypotheses and removes hypotheses with low posterior probabilities. (b) A set of children hypotheses are generated from existing hypotheses to describe the same environment in more detail. (c) Our Bayesian filtering framework continuously tests each new set of hypotheses.

the directed distance from the origin of the 3D coordinate to the plane is d_i. $\mathbf{N}_i$ and d_i determine the equation of the wall in the 3D coordinate.

The bound of each wall segment is defined by two vertical lines on the wall plane in 3D. By projecting the vertical lines onto the ground-plane map, the wall segment S^i_j is represented by a pair of endpoints, $(E^i_{1,j}, E^i_{2,j})$, along the corresponding ground-wall boundary line. The formal definition of a wall W_i is,

$$W_i = \langle \alpha_i, d_i, S^i_1, S^i_2, ..., S^i_{m_i} \rangle \tag{6.3}$$

where m_i is the number of wall segments along this wall. The segments and the two endpoints of each segment are ordered from from the left to the right side of the canonical view of the wall plane. Each endpoint $E^i_{s,j}$ is represented by,

$$E^i_{s,j} = \langle x^i_{s,j}, y^i_{s,j}, u^i_{s,j} \rangle \tag{6.4}$$

where $(x^i_{s,j}, y^i_{s,j})$ define the location of the endpoint in the ground-plane map, and $u^i_{s,j}$ represents the type of the endpoint. The subscript $s \in \{1, 2\}$ denotes the order of the endpoints, where $E^i_{1,j}$ is the endpoint on the left and $E^i_{2,j}$ is the endpoint on the right.

There are three types of endpoints: *dihedral*, *occluding* and *indefinite*, representing different levels of understanding of the bound of the wall segment. A *dihedral* endpoint corresponds to two visible wall segments, where the location of the endpoint is the projection of the intersection line of the two walls. An *occluding* endpoint corresponds to only one visible wall segment. An *indefinite* endpoint is the current furthest visible point of a wall segment, but its actual location has not yet been observed by the robot due to occlusions or the end of the robot's field of view. While a *dihedral* endpoint provides the full knowledge of the bound of its corresponding wall segments, an *occluding* and an *indefinite* endpoint provide different types of partial knowledge of the wall intersection. Figure 6.2(a) is an example representing a cross-intersection with the PSM.

6.1.1 Image Projection of PSM

Similar to the simple planar model in Chapter 5, given the robot pose, the image projection of the ground-wall boundary of the PSM is a polyline extending from the left to the right image borders, where the initial and final segments may lie along the lower image border. A non-vertical line segment corresponds to a wall segment in PSM and vertical segments correspond to occluding edges between walls. Figure 6.2(b) is an example of an image projection of the PSM.

6.1.2 3D Construction of PSM from Single Image

With our assumption that the robot has a fixed and known tilt ϕ^c and roll ψ^c angles with respect to the ground plane, and has a fixed height z^c from the ground plane, 3D location of ground-plane and wall-plane points can be constructed from a single image given the projected PSM. In the local 3D coordinate, the 3D location $\mathbf{P}_i = (x_i, y_i, z_i)^T$ of an image point $\mathbf{p}_i = (u_i, v_i, 1)^T$ that lies on the ground plane is related by

$$\mathbf{R}_{\psi^c}\mathbf{R}_{\phi^c}\mathbf{R}_c \begin{pmatrix} x_i \\ y_i \\ -h \end{pmatrix} = \lambda \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} \tag{6.5}$$

where $\mathbf{R}_{\psi^c}$ and $\mathbf{R}_{\phi^c}$ are the rotation matrices related to the camera tilt and roll angles, respectively. In the local 3D coordinate, the rotation matrix corresponding to the roll angle is

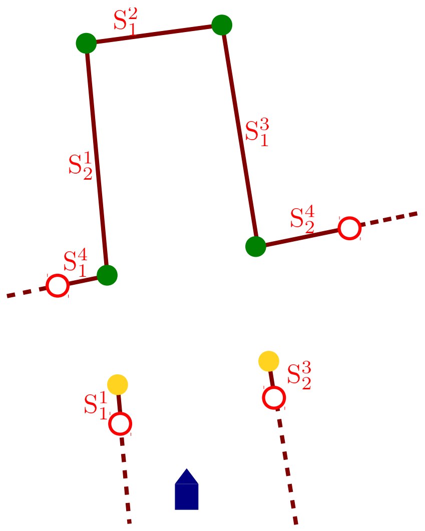
$$\mathbf{R}_{\psi^c} = \begin{bmatrix} \cos\psi^c & -\sin\psi^c & 0 \\ \sin\psi^c & \cos\psi^c & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.6}$$

and the rotation matrix corresponding to the tilt angle is

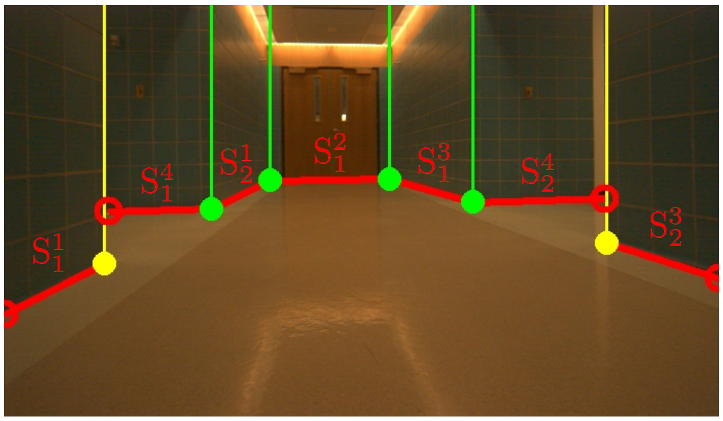
$$\mathbf{R}_{\phi^c} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi^c & -\sin\phi^c \\ 0 & \sin\phi^c & \cos\phi^c \end{bmatrix}. \tag{6.7}$$

$\mathbf{R}_c$ is the matrix that transforms the location of a 3D point from the camera coordinate to the local 3D coordinate:

$$\mathbf{R}_c = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}. \tag{6.8}$$



(a) PSM on ground-plane map



(b) Image Projection of PSM

Figure 6.2: Planar Semantic Model (PSM). (Best viewed in color.) (a) PSM representation of a cross-intersection. On the ground-plane map, the walls are represented by (red) lines and a set of endpoints delimiting where the wall is present. *Dihedral* endpoints are marked as green and *occluding* endpoints are marked as yellow. *Indefinite* endpoints are marked as red hollow points. Wall segments are notated in S_j^i where i is the wall index and j is the index of the segment within wall i. Notice that the wall segments are ordered from the left to the right of the canonical view of the wall. (b) Image projection of the PSM ground-wall boundary. The robot pose is at the blue mark in (a). The projected ground-wall boundary is a polyline connected from the left to the right image borders. Each wall segment corresponds to a line segment in the projected image. A vertical segment in the image corresponds to an occluding endpoint (yellow). Note that if the tilt and roll angle of the camera with respect to the ground plane is not zero, a vertical segment in 3D will not be vertical in the projected image. In fact, these projected vertical segments will intersect at a vanishing point.

Solving (6.5) gives us the location of the ground plane point in the local 3D coordinate,

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} \lambda_i(\cos\phi^c - v_i \sin\phi^c) \\ -\lambda_i(\sin\psi^c \sin\phi^c + u_i \cos\psi^c + v_i \cos\phi^c \sin\psi^c) \\ -h \end{pmatrix} \tag{6.9}$$

where

$$\lambda_i = \frac{h}{\cos\psi^c \sin\phi^c - u_i \sin\psi^c + v_i \cos\psi^c \cos\phi^c}.$$

To construct the wall plane in the local 3D coordinate, we start by selecting any two points along the corresponding line and reconstruct their locations in the ground-plane map, $\tilde{\mathbf{p}}_1 = (x_1, y_1)^T$ and $\tilde{\mathbf{p}}_2 = (x_2, y_2)^T$, using the geometry of ground plane points. Given the two points, α_j can be determined by,

$$\alpha_j = -\arctan\frac{x_1 - x_2}{y_1 - y_2} \tag{6.10}$$

and thus, the normal vector of the corresponding line in the ground-plane map is $\mathbf{n}_j = (\cos\alpha_j, \sin\alpha_j)^T$. The directed distance d_j from the origin to the line can be determined by,

$$\begin{aligned} d_j &= \mathbf{n}_j \cdot \tilde{\mathbf{p}}_1 = \mathbf{n}_j \cdot \tilde{\mathbf{p}}_2 \\ &= x_1 \cos\alpha_j + y_1 \sin\alpha_j \\ &= x_2 \cos\alpha_j + y_2 \sin\alpha_j. \end{aligned} \tag{6.11}$$

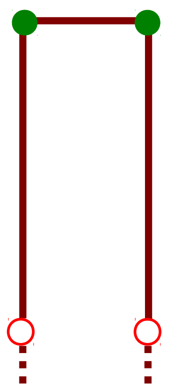
If a point lies on the wall plane with position $\mathbf{p}_i$ in the image, its 3D location in the local 3D coordinate is related by

$$d_j = \mathbf{N}_j \cdot (\lambda_j \mathbf{R}_c^{-1} \mathbf{R}_{\phi_t}^{-1} \mathbf{R}_{\phi_r}^{-1} \mathbf{p}_i) = \mathbf{N}_j \cdot \mathbf{P}_i. \tag{6.12}$$

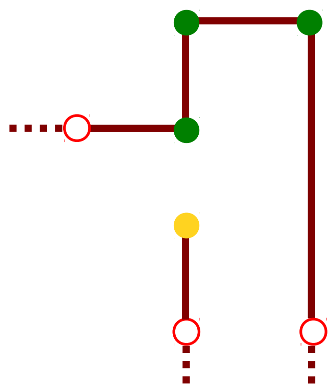
Solving λ_j in Equation 6.12 gives us the 3D location of the point $\mathbf{P}_j$.

6.2 Incremental Hypotheses Generation

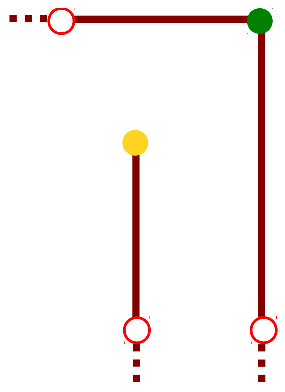
Since there is a one-to-one correspondence between a PSM model and a polyline in the image space (Section 6.1.1 and 6.1.2), a set of PSM hypotheses can be generated from 2D image features. The hypothesis generation method described in Chapter 5 is used to



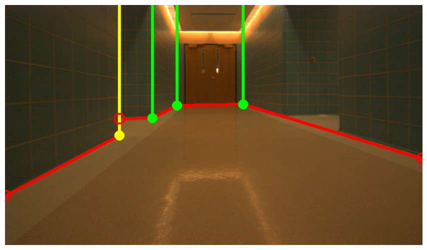
(a) Parent



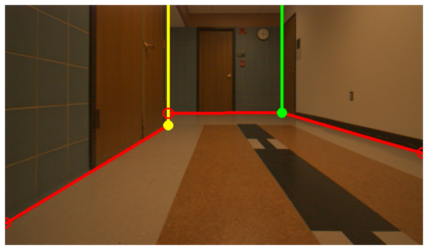
(b) Child (type 1)



(c) Child (type 2)



(d) Example Child (type1)



(e) Example Child (type2)

Figure 6.3: Types of child hypotheses. (Best viewed in color.) Given the parent hypothesis (a), two types of hypotheses can be generated to describe the environment in more details. (b) Type 1 adds two endpoints to an existing wall segment to form an opening, and adds new walls that are visible through the openings. Note that in this case, PSM captures the fact that the two wall segments are parts of the same wall plane. (c) Type 2 creates an opening between two walls that are intersecting in the parent hypothesis. Examples of the children hypotheses are shown in (d) and (e).

generate simple PSM hypotheses. A simple PSM hypothesis consists of at most three walls where each wall consists of only one segment and intersects with its adjacent walls. In other words, there are no occluding endpoints and no openings in a simple PSM hypothesis.

PSM hypotheses with openings are generated through an incremental process. We transform a current PSM hypothesis to a set of children hypotheses describing the same environment in more details by adding openings. Two types of child hypothesis can be generated from a current hypothesis. The first type of child hypothesis adds openings along walls. These children hypotheses add endpoints to the existing wall segments in the parent hypothesis to create the openings, and add new walls that are visible through the openings (Figure 6.3(b)). For each visible wall, a set of candidate openings are generated. A candidate opening consists of two endpoints belonging to two adjacent segments of the same wall to create the gap. We start by extracting image corner features along the projected ground-wall boundary line, and collect a set of corner pairs that are wide enough in 3D for the robot to pass through. For each corner pair, we combine line segments between the two corners to form a set of candidate openings. A child hypothesis is generated by selecting at most one candidate openings from each visible wall.

The second type of child hypothesis creates an opening between two walls that are intersecting in the parent hypothesis (Figure 6.3(c)). From each dihedral endpoint that corresponds to a concave wall intersection, a set of candidate openings can be generated by transforming the dihedral endpoint into an occluding endpoint for one wall segment and an indefinite endpoint for the other wall segment. Thus, we search for image corner features along both associated wall segments of the dihedral endpoint to become a candidate occluding endpoint. A candidate opening is generated by a candidate occluding endpoint that provides a feasible gap for the robot to pass through. A child hypothesis is generated by selecting at most one candidate opening from each concave dihedral endpoint.

In addition to the above two transformations that generate hypotheses with more openings, children hypotheses are also generated by merging a current hypothesis with a simple PSM hypotheses generated from the current frame. Hypotheses are merged only if they have enough overlapping and if they have no conflicting information. This type of children hypotheses is essential for incremental scene understanding after the robot makes a turn. For example, if a robot makes a turn at a L-intersection that connects two long corridors, most of the environment within the field of view is not modeled in the current hypotheses because current hypotheses only contain information about the first corridor and the intersection. Thus, we generate a set of new simple PSM hypotheses from the current frame to capture information about the second corridor. By merging a new hypothesis with a current hypothesis, we generate a child hypothesis that captures information about both corridors.

6.3 Refining PSM Model

We use the information from current observations to refine the quantitative precision of each existing hypothesis. The Extended Kalman Filter (EKF) (Section 3.2.2) is used to estimate the parameters of each wall plane and the ground-plane map location of each occluding endpoint.

For each wall W that is visible, the parameters of the plane at frame t are $\mu_t^w = (\alpha_t, d_t)^T$. Since the walls are static in the global 3D coordinate, the state prediction function is,

$$g(\mu_{t-1}^w) = \mu_{t-1}^w. \tag{6.13}$$

To obtain a 3D plane measurement z_t^w of the wall, we project the predicted ground-wall boundary line to the image space and find the best match between the boundary line and a set of lines under the camera center. Using the 3D reconstruction method described in Section 6.1.2, the measurement z_t^w can be parametrized as a 3D wall, $z_t^w = (z_\alpha, z_d)^T$. Given the robot pose $u_t = (x_t^c, y_t^c, \theta_t^c)^T$ at frame t, the predicted measurement $\hat{z}_t^w = h(\hat{\mu}_t^w, u_t)$ is obtained by

$$\begin{aligned} \hat{z}_t^w &= \begin{bmatrix} \hat{\alpha}_t - \theta_t^c \\ \hat{d}_t - \cos\hat{\alpha}_t x_t - \sin\hat{\alpha}_t y_t \end{bmatrix} \\ &\mathit{or} \\ \hat{z}_t^w &= \begin{bmatrix} \hat{\alpha}_t - \theta_t^c + \pi \\ -\hat{d}_t + \cos\hat{\alpha}_t x_t + \sin\hat{\alpha}_t y_t \end{bmatrix}. \end{aligned} \tag{6.14}$$

Once the parameters of the walls are refined, we refine the location of each occluding endpoint that is visible in the current frame. The ground-plane map location of the endpoint at frame t is represented as $\mu_t^e = (x_t, y_t)^T$. Given the refined parameters of its associated wall with $\mu_t^w = (\alpha_t, d_t)$ and uncertainty Σ_t^w, the state prediction function for the endpoint location $\hat{\mu}_t^e = g(\mu_{t-1}^e, \mu_t^w)$ projects the point μ_{t-1}^e onto the wall and the process noise Q_t is,

$$Q_t = F_t \Sigma_t^{wall} F_t^T \tag{6.15}$$

where $F_t = \frac{\partial g(\mu^e, \mu^w)}{\partial \mu^w}\Big|_{\mu^w = \mu_t^w}$. By projecting the endpoint onto the image space and matching with point features extracted along the ground wall boundary, we collect a measurement of the endpoint which is represented in the ground map space $z_t^e = (z_x, z_y)^T$. The predicted measurement of the endpoint is computed by,

$$\begin{aligned}\hat{z}_t &= h(\hat{\mu}^e_t, , u_t) \\ &= \begin{bmatrix}(x_t - x^c_t)\cos\theta^c_t - (y_t - y^c_t)\sin\theta^c_t \\ (x_t - x^c_t)\sin\theta^c_t + (y_t - y^c_t)\cos\theta^c_t\end{bmatrix}.\end{aligned} \tag{6.16}$$

The location of a dihedral endpoint is updated by finding the intersection of the two associated walls with their refined parameters. The location of an indefinite endpoint is updated based on the robot pose and its field of view. We scan through the angles within the field of view with a resolution of 0.5 degree, and obtain the furthest visible point of each wall that has indefinite endpoints. The indefinite endpoint is either at the bound of the field of view or at the blocking point from another wall.

6.4 Incremental Hypotheses Testing using Bayesian Filtering

As in Chapter 5, the hypotheses are tested based on their abilities to explain the 2D (image) motion of a set of tracked points using a Bayesian filter:

$$p\left(M_i|\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_t\right) \propto p\left(M_i\right) \prod_{j=1...t} p\left(\mathbf{F}_j|M^i\right) \tag{6.17}$$

However, unlike most Bayesian filtering applications, the number of hypotheses in the on-line generate-and-test framework changes over time. New children hypotheses are incrementally generated and added to the active set of hypotheses as the robot travels, and bad hypotheses are removed from time to time to keep the total number of hypotheses tractable. Thus, this section describes how the Bayesian filter can be applied to the on-line generate-and-test framework.

As described in Chapter 5, the likelihood function compares the predicted point feature locations with the observed point feature locations. In this chapter, at frame t, we use point features that are visible and tracked between frame $t - t_w$ in the image space to test the hypotheses. $t_w \in [5, 20]$ is automatically adjusted to ensure that the number of the point features exceeds a threshold (20 points), if possible. Chapter 8 presents a more efficient way to select the set of point features for hypothesis testing. Given a hypothesis, point features are reconstructed into the global 3D coordinate based on their image locations at frame $t - t_w$ and projected back onto the image space based on the given robot pose at frame t. The likelihood of the hypothesis at frame t is then computed by comparing the predicted and the tracked locations of the point features.

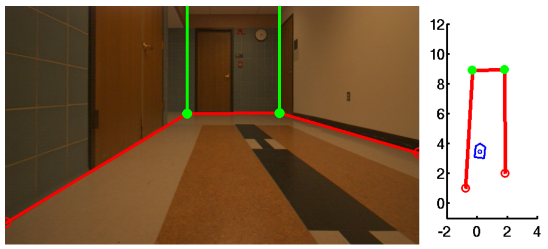
Since there is no motion information to test about the correctness of the hypotheses generated in the initial frame, their prior probabilities $p(M_i)$ is uniformly distributed over the hypotheses. A child hypothesis M_c that is added to the set at frame t has a posterior probability at frame $t-1$ equal to the posterior probability of its parent hypothesis M_p at frame $t-1$. Since the child hypothesis and its parent hypothesis only differs in the regions that are distant to the robot at the beginning of the video, we can assume that both hypotheses predict all past observations in the same way. In other words, their prior and their likelihoods are the same up till frame $t-1$, and thus their posterior probabilities at frame $t-1$ are the same:

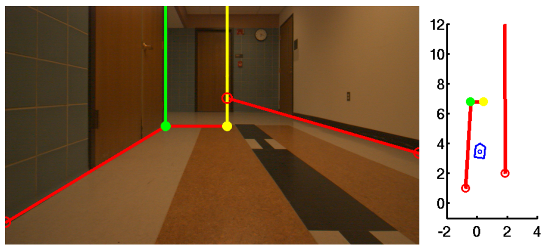
$$p(M_c|\mathbf{F}_1, ..., \mathbf{F}_{t-1}) = p(M_p|\mathbf{F}_1, ..., \mathbf{F}_{t-1}). \tag{6.18}$$

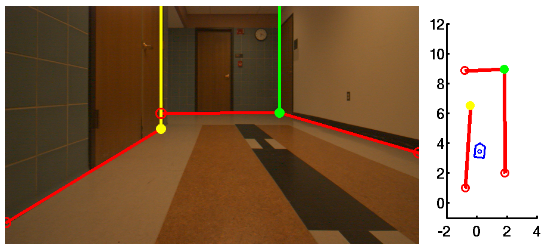
In some situations, one may have prior knowledge of the indoor environment, such as the width or length of a typical corridor. In such cases, the prior probability of the child hypothesis $p(M_c)$ may not be the same as its parent's prior $p(M_p)$, but their likelihoods from frame 1 to frame $t-1$ are still the same. This, the posterior probability of the child hypothesis at frame $t-1$ is

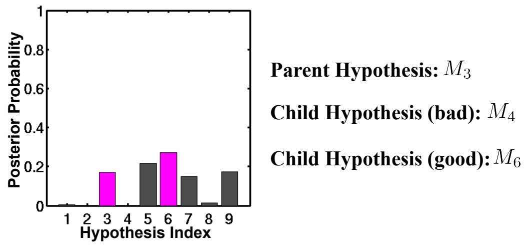
$$p(M_c|\mathbf{F}_1, ..., \mathbf{F}_{t-1}) = p(M_p|\mathbf{F}_1, ..., \mathbf{F}_{t-1}) \frac{p(M_c)}{p(M_p)}. \tag{6.19}$$

In order to keep the number of hypotheses N tractable, hypotheses with posterior probabilities lower than a threshold are removed. The threshold are set dynamically based on the current number of hypotheses N. In our experiments, the threshold is set to $\frac{0.1}{N}$. In other words, if the posterior probability of a hypothesis is lower than 0.1 of the posterior probabilities of uniform distribution among the current set of hypotheses, that hypothesis is considered to be a bad hypothesis and is removed from the set. After removing hypotheses with low posterior probabilities, we normalize the posterior probabilities among the surviving hypotheses.


(a) Parent Hypothesis M_3

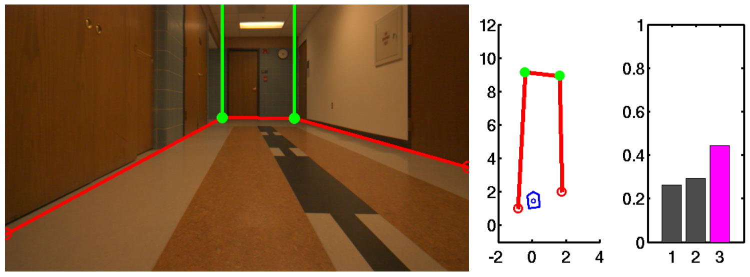

(b) Child Hypothesis (bad) M_4

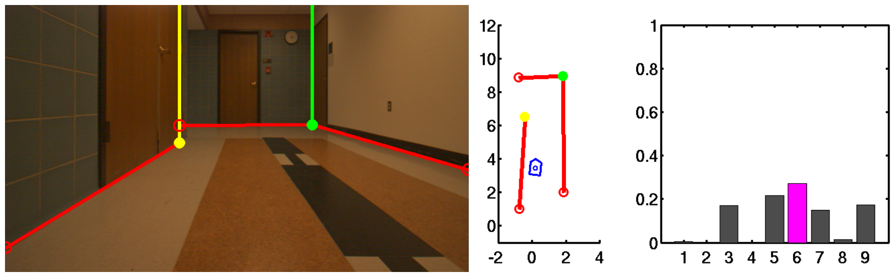

(c) Child Hypothesis (good) M_6

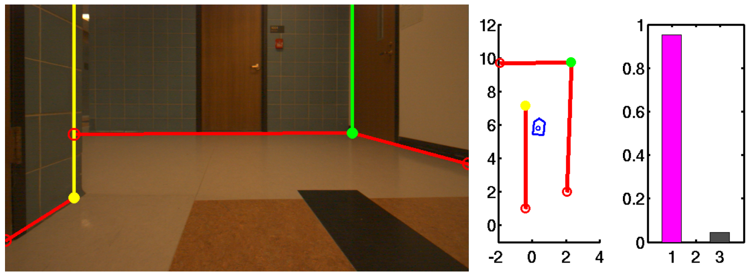


(d) Distribution

Figure 6.4: Examples of the hypotheses generated by our incremental process. (Best viewed in color.) The ground-wall boundaries are plotted as red. Color dots represent *dihedral* endpoints (yellow) and *occluding* endpoints (green). Both (c) and (b) are children hypotheses generated from (a). (d) is their posterior probabilities at the current time frame.


(a) Frame 50


(b) Frame 120


(c) Frame 200

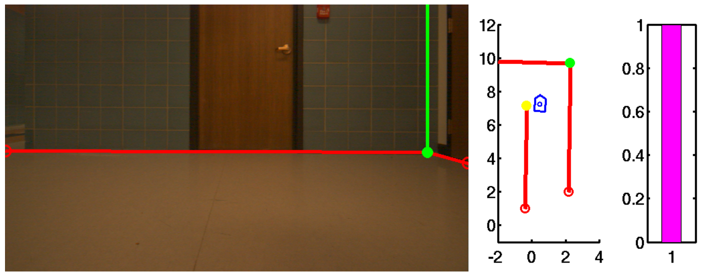

(d) Frame 245

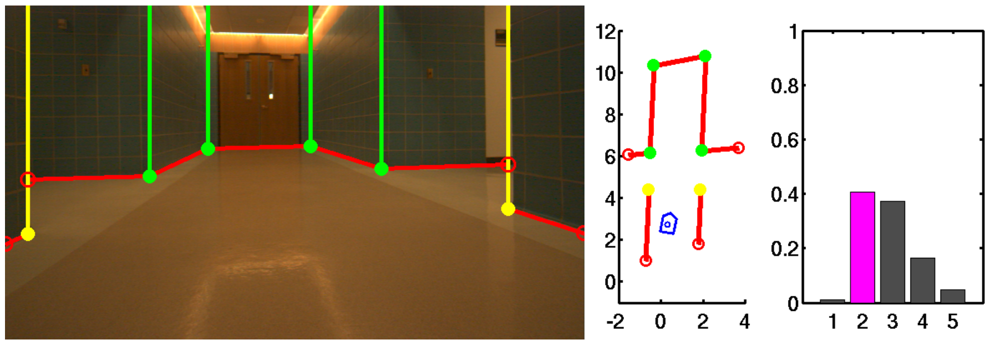
Figure 6.5: Examples of the on-line generate-and-test framework. (Best viewed in color). This example is from the first part of the Dataset L. These frames correspond to the graphs shown in Figure 6.8. (See text for more detail.)

6.5 Evaluation

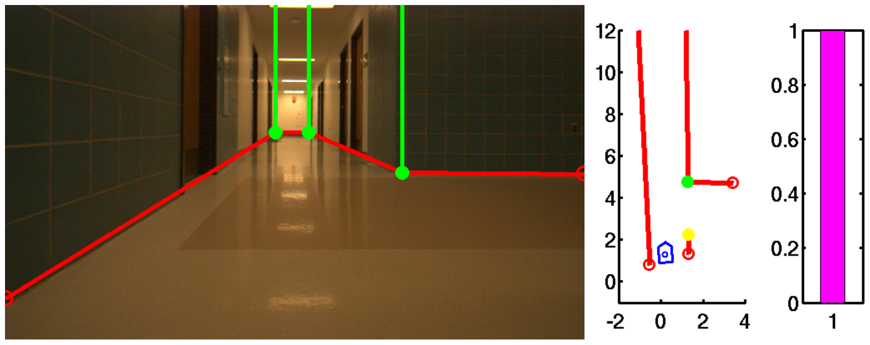
Our on-line generate-and-test framework for scene understanding is evaluated using the Michigan Indoor Corridor 2012 Video Dataset. (Details of this dataset is introduced in Chapter 4.) Examples of the hypotheses generated from our incremental process are shown in Figure 6.4. Figure 6.5 demonstrates our method for transforming a current environmental-structure hypothesis into children hypotheses and shows how the Bayesian filter converges to the best hypothesis. A set of simple PSM hypotheses (where each wall contains only one segment and intersects with its adjacent walls) were generated from the first frame of the video. At frame 50, the Bayesian filter converged to three simple PSM hypotheses. The other two hypotheses that are not shown in the figure have the same side walls but a different planar equation for the end wall from the one shown. At frame 120, hypotheses with low probabilities were removed and good hypotheses generated children hypotheses to describe the scene in more detail. From frame 200 to frame 245, our Bayesian filter continues to test all the hypotheses and gradually converges to the best hypothesis. At the end, the robot has a single current model of the surrounding environment even though much of it is not in view.

Figure 6.6 shows our results for different structures of the environment. These results demonstrate the expressive power of the PSM and demonstrate that our framework is able to generate and converge to a correct hypotheses. In all of these examples, due to the limited field of view of the monocular camera, it is impossible for the robot to realize that it is at an intersection solely from the current image. Thus, a temporally contiguous stream of images is essential for coherent visual scene understanding. Sometimes due to the lack of features and motion, the Bayesian filter might converge to a bad hypothesis as shown in Figure 6.7, but our system is still able to use the information of the current scene to generate a set of new hypotheses to describe the local environment in the next scene. However, if the ground-wall boundaries are blocked by objects, our method might fail because the best hypothesis might not be generated in the first place. Moreover, if there are some of the point features lies on clutter (part of the environment that is not representable by the PSM model), the Bayesian filter may be distracted and converge to a bad hypothesis. Chapter 7 describes methods to handle cluttered environments.

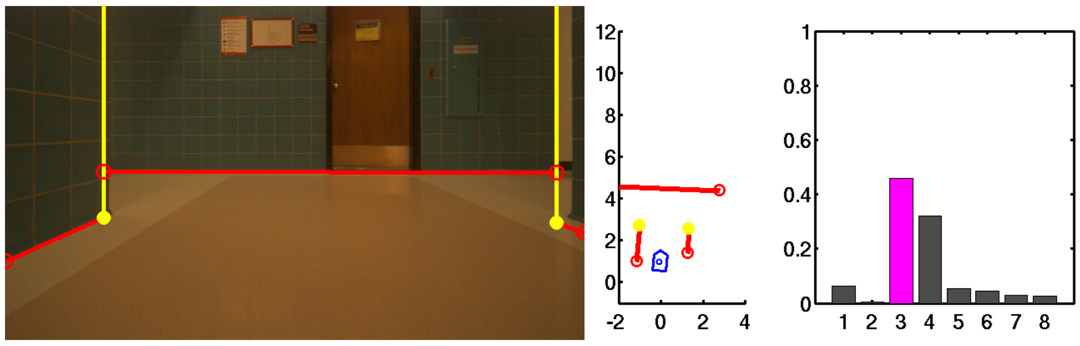
We evaluate our on-line generate-and-test framework quantitatively. In each frame, we define the accuracy of a hypothesis being the percentage of the pixels that have the correct plane classification in that frame. Since the ceiling plane is not included in our hypotheses, ceiling pixels are skipped in our evaluation. Two types of quantitative accuracy are reported for each dataset. *MAP hypothesis accuracy* is the accuracy of the hypothesis of the maximum posterior probability at each frame. *Weighted accuracy* is the weighted average accuracy of all the existing hypotheses at each evaluated frame where the weight of each hypothesis is equal to its posterior probability. Our quantitative results are reported in Table 6.1. Overall, our framework achieves 93.76% of accuracy. In order to further understand how well our framework works in different situations, Figure 6.8 shows the performance of our Bayesian filtering framework at each time frame. We showed that our framework always maintains a small set of active hypotheses where the a good hypotheses is always in the set. Our Bayesian filter is able to identify the best hypothesis most of the time.



(a) Dataset +

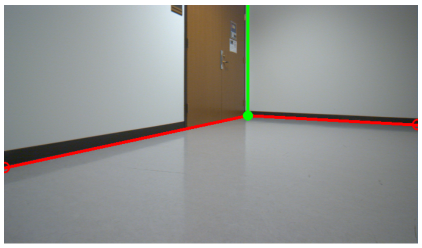


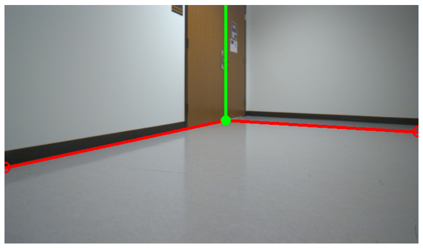
(b) Dataset T1

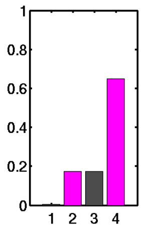


(c) Dataset T2

Figure 6.6: Results of on-line generate-and-test framework in different environments. (Best viewed in color). The second best hypothesis in Dataset + has a similar structure except with a different equation for the wall that is visible from the openings. In Dataset T 1, PSM models the right side as one wall with two endpoints that creates the opening.


(a) Best Hypothesis M_2


(b) Bad Hypothesis M_4



(c) Posterior Distribution

Figure 6.7: Examples when failing to identify the best hypothesis. (Best viewed in color.) This scene corresponds to frame 734 to frame 837 in Figure 6.8. Due to the lack of observed point features and motion, the Bayesian filter might converge to a bad hypothesis M_4 (frame 760). In fact, in this dataset, the robot did not accumulate enough information to identify the best hypothesis M_2 before the end wall went out of view. However, our system is still able to use the information of the current scene (the left wall) to generate a set of new children hypotheses to describe the local environment in the next scene (frame 843).

Dataset	Dataset L	Dataset +	Dataset T 1	Dataset T 2	Overall
MAP accuracy	91.00%	94.23%	92.71%	96.38%	93.76%
weighted accuracy	90.04%	92.77%	92.21%	95.37%	92.80%
number of frames	900	300	410	360	1970

Table 6.1: Quantitative evaluation of the on-line generate-and-test framework. To the best of our knowledge, there is no directly comparable work but as shown in Chapter 5, quantitative comparison with [31] and [25] demonstrates our on-line scene understanding method has higher accuracies on most environments. Accuracies of both [31] and [25] are about 80% on the Michigan Indoor Corridor 2011 Video Dataset.

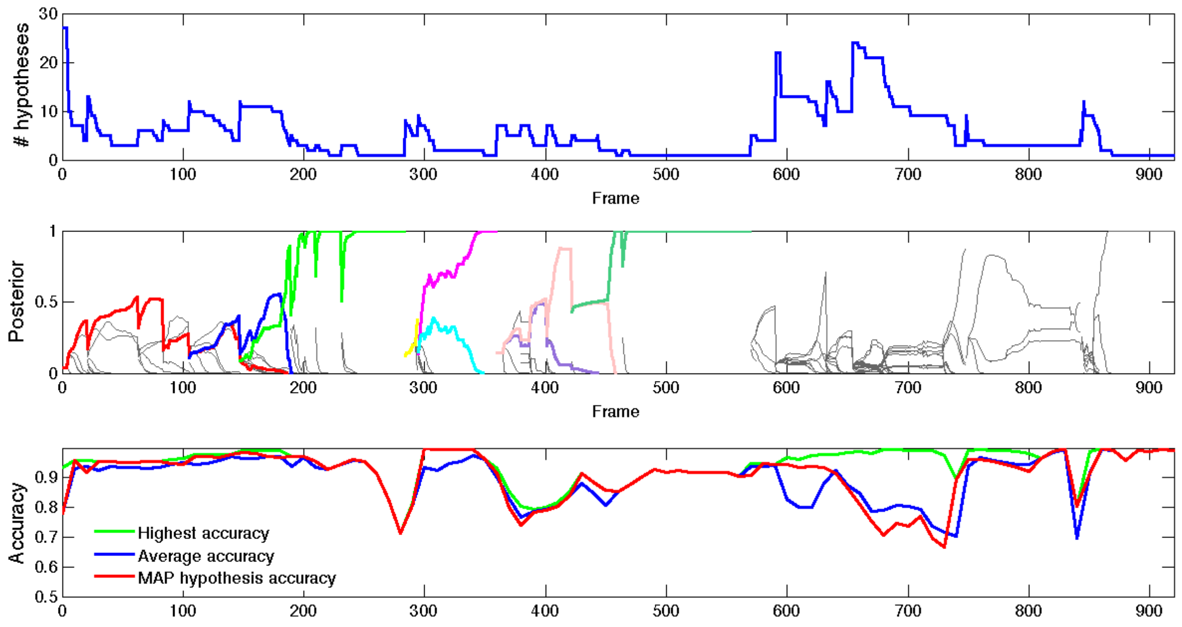


Figure 6.8: Bayesian filtering results of Dataset L. (Best viewed in color). The horizontal axis of the graphs is the time frame of the video. **The top graph** shows the number of hypotheses active in each frame. Notice that our method efficiently focuses on a small set of active hypotheses (at most 27 hypotheses in this case) at each frame. In our experiments, a set of children hypotheses are generated every 20 frames or when less than 70% of the current image is explained by existing hypotheses. **The second graph** shows the posterior probability of the hypotheses at each frame. Every time when a set of children hypotheses are generated, the posterior probability of the best hypothesis drops. We highlighted several hypotheses to illustrate our method. The red hypothesis is a simple three-wall hypothesis where each wall intersects with its adjacent walls. Both the blue and the green hypotheses (the correct hypothesis) are children hypotheses of the red one with the correct opening structure with different width generated at frame 105 and frame 147 respectively. Thus, the posterior probability of the red hypothesis started to decrease and eventually it was removed from the set, while both the blue and green hypotheses had a higher and higher posterior probability. As the robot approached the opening, more point features around the opening were observed and thus, the probability of the blue hypothesis dropped and was removed from the set at frame 190. Similar phenomena occur several times throughout the video. **The third graph** plots the accuracy (evaluated every 10 frames) of the system at each time frame. The green line shows the maximum accuracy among the existing hypotheses. The blue line shows the *weighted accuracy* among of the existing hypotheses, where the weight of each hypothesis is equal to its posterior probability. The red line shows the accuracy of the hypothesis with the maximum posterior probability. The Bayesian filter identifies the best hypothesis for most of the time. Notice a good children hypothesis was generated at frame 654 with a low prior, but as the robot approaches the opening and accumulates more evidences along the travel, the correct hypothesis becomes the winner at frame 737.

6.6 Summary

In this chapter, we extend the simple on-line scene understanding method to model environments with more complex structures, such as intersections. We present the Planar Semantic Model (PSM) to describe the environment by a set of meaningful planes — ground plane and a set of walls that are perpendicular to the ground plane but not necessarily to each other. PSM is a step forward from existing floor-wall models because it captures richer relationships among wall segments. PSM is also capable of expressing incomplete knowledge so that unobserved areas can be incrementally modeled as observations become available.

Since the full structure of the local environment may not be observable to the robot all at once, we introduced an incremental process of generating structural hypotheses to describe the same environment with more detail. This is the key to incremental scene understanding. In addition to introducing incremental hypothesis generation, we use the information in current observations to refine the quantitative precision of existing hypotheses.

We evaluate our on-line generate-and-test framework on a variety of indoor environments, including L-intersections, T-intersections, and +-intersections. Our experimental results demonstrate the expressive power of the Planar Semantic Model. We show that our framework maintains a small set of active hypotheses, and among the hypotheses, the correct hypothesis is always in the set. We demonstrate that the Bayesian filter is capable of selecting the correct hypotheses.

Chapter 7

Handling Perceptual Clutter

This chapter addresses the challenges of understanding cluttered indoor environments [1]. Unlike Chapter 5 and 6 where the environment is assumed to be empty, this chapter demonstrates the on-line generate-and-test framework for scene understanding in cluttered environments. In this chapter, ϵ_1 in the first level of interpretation is not zero:

$$z_t = G_1(M, \mathbf{x}_t) + \epsilon_1. \tag{7.1}$$

In addition, this chapter moves one step forward to the second level of interpretation,

$$z_t = G_2(M, C_1, ..., C_k, \mathbf{x}_t) + \epsilon_2 \tag{7.2}$$

by representing ϵ_1 as a set of clutter regions.

Formally, we define "clutter" as regions in the local environment that cannot be represented by the Planar Semantic Model (PSM). Since clutter is unstructured, clutter is represented by a set of 3D point clouds, where each point cloud corresponds to a 2D segment in the image space. While all observations in empty environments can be explained by the PSM (Chapter 5 and 6), in cluttered environments, PSM explains only a subset of the visual observations, and clutter is the collection of observations that are not explained by PSM.

In order to focus our research on scene understanding, we use a depth camera to collect a stream of RGB-D images. The goal of this chapter is to describe the observation in terms of PSM M and clutter $\{C_1, ..., C_k\}$ using the on-line generate-and-test

[1]Preliminary version of this chapter was published in [73].

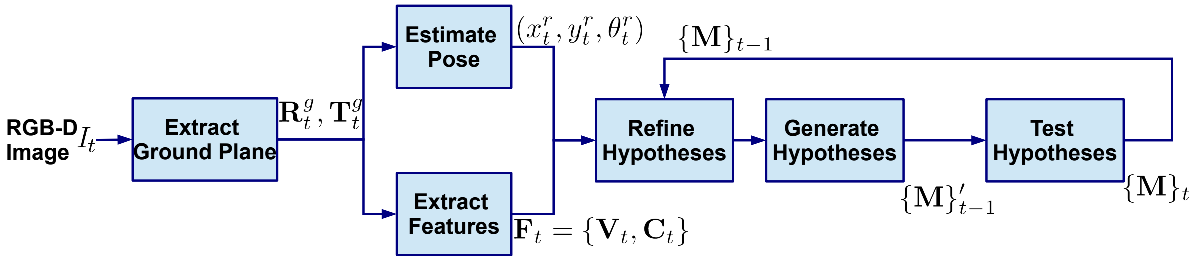


Figure 7.1: On-line generate-and-test framework with RGB-D image streams. Given the current RGB-D frame t, we first find the relation between the camera and the local 3D coordinate, and then estimate the 3dof robot pose $(x_t^r, y_t^r, \theta_t^r)$ in the global 3D coordinates. At the same time, we extract useful features $\mathbf{F}_t$ (e.g. vertical-plane features $\mathbf{V}_t$ and point-set features $\mathbf{C}_t$) from frame t. We follow the on-line generate-and-test framework described in Chapter 6, and extend the framework to handle cluttered indoor environments. The major extension of this chapter is the hypotheses testing step. This chapter presents a likelihood function that allows a good hypothesis to explain only a subset of the features $\mathbf{F}_t$. In addition, this chapter presents methods to generate and refine the PSM hypotheses in cluttered environments using RGB-D image streams.

framework described in Chapter 6. In addition, this chapter computes the robot pose from the RGB-D images without any additional sensors.

Figure 7.1 illustrates the on-line generate-and-test framework described in Chapter 6 and highlights the major aspects of the framework that need to be extended in order to handle cluttered indoor environments. At each frame t, the ground plane G is extracted and a transformation $[R_t^g, T_t^g]$ that transforms the 3D points from the image coordinate to the local 3D coordinate is computed (Section 7.1). Points that lie on the ground plane are now considered as "explained" by the ground G. The transformation $[R_t^g, T_t^g]$ captures three degrees of freedom of the full 6-degree-of-freedom robot pose $\mathbf{x}_t = (x_t^r, y_t^r, z_t^r, \theta_t^r, \phi_t^r, \psi_t^r)$, the tilt ϕ^r and roll ψ^r angles and the height z_t^r of the robot with respect to the ground plane at frame t. In contrast with Chapter 6, the tilt and roll angles and the height do not need to be fixed across the frames. The remaining three degrees of freedom is represented as $(x_t^r, y_t^r, \theta_t^r)$ on the ground-plane map. The robot pose is estimated by aligning RGB image features and the 3D non-ground-plane points between frame $t-1$ and frame t (Section 7.2).

At the same time, a set of features $\mathbf{F}_t$ are extracted from the 3D points that are not explained by the ground G. While there are other research focusing on grouping a RGB-D image or a set of 3D points into primitive shapes [34], in this thesis, we group the 3D points into a set of vertical-plane features $\mathbf{V}_t$ and a set of point-set features $\mathbf{C}_t$.

Vertical-plane features suggest the existence of walls, while point-set features suggest the existence of clutter. (Section 7.3)

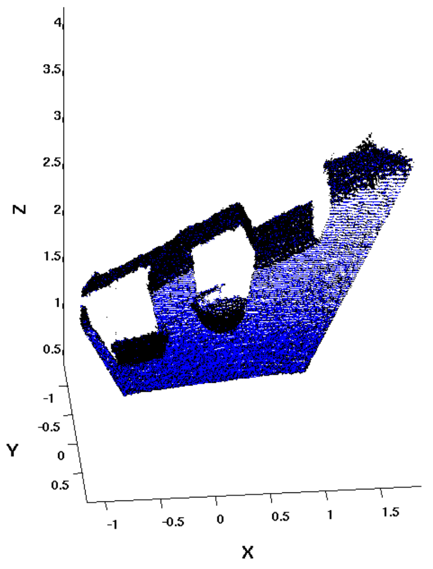
Given the robot pose and the features $\mathbf{F}_t = \{\mathbf{V}_t, \mathbf{C}_t\}$, the on-line generate-and-test framework first uses the vertical-plane features $\mathbf{V}_t$ to refine the precision of the existing set of PSM hypotheses $\{\mathbf{M}\}_{t-1}$ (Section 7.4). Then, a set of new hypotheses are generated and added to the hypothesis set $\{\mathbf{M}\}'_{t-1}$ by transforming existing hypotheses into children hypotheses describing the same environment with more details based on the vertical-plane features $\mathbf{V}_t$ (Section 7.5). Finally, we use a Bayesian filter to test the hypotheses. While Chapter 5 and 6 require a good hypothesis to explain all the observed features, in this chapter, we present a likelihood function that allows a good hypothesis to explain only a subset of the features (Section 7.6). Evaluation of the on-line generate-and-test framework in cluttered environments is presented in Section 7.7.

7.1 Ground Plane Extraction

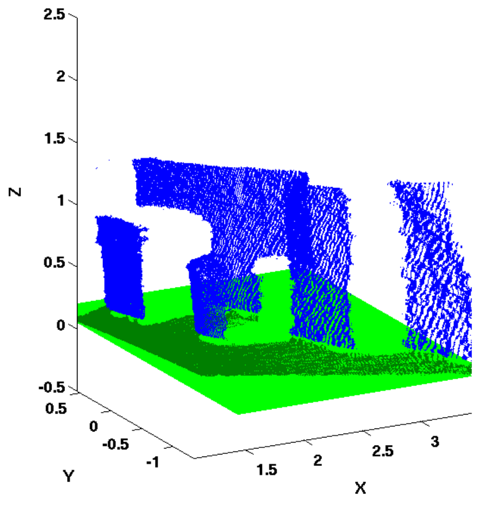
At each frame, we extract the 3D ground plane using both RGB image and depth information [2]. Figure 7.2 illustrates how the ground-plane is extracted. First, we collect the candidate pixels where their local surface normals differ from the approximated normal vector $\mathbf{N}_{appx}$ of the ground plane within ϕ_{ground}. For a front-facing camera, the approximated normal vector is $\mathbf{N}_{appx} = [0, 1, 0]$ in the image coordinate, and in our implementation, we set $\phi_{ground} = \frac{\pi}{6}$. The local surface normal of each pixel is computed using the efficient algorithm proposed by Holz *et al.* [34]. From the candidate pixels, we fit the dominant plane that has a normal vector within ϕ_{ground} of the approximated normal $\mathbf{N}_{appx}$ using RANSAC. From the inlier pixels, we perform a morphological close operation on the image space to locate a smooth and bounded region for the ground plane.

Once the ground plane is extracted, the transformation $[\mathbf{R}^g_t, \mathbf{T}^g_t]$ between the camera coordinate and the local 3D coordinate is computed. The origin of the local 3D coordinate is set to the projected location of the camera center on the ground plane. Mathematically, $\mathbf{R}^g_t$ is the rotation matrix that rotates the normal vector of the ground plane to $[0, 0, 1]$, and $\mathbf{T}^g_t = [0, 0, h_t]$ is determined by the distance h_t between the camera

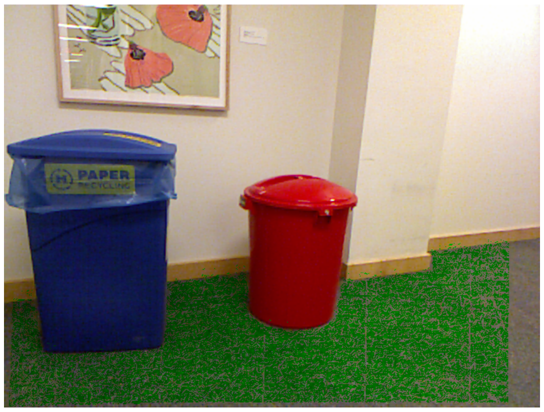
[2]We only use depth information of a pixel if it lies within a reliable range. For Kinect, the reliable range is between 0.8 to 4 meter.



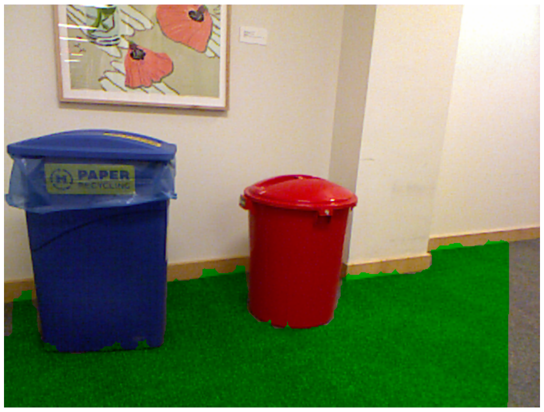
(a) Collect candidate ground points



(b) Fit plane using RANSAC



(c) Obtain inlier points



(d) Smooth ground-plane region

Figure 7.2: Ground plane extraction from RGB-D image. (Best viewed in color.) (a) From the input RGB-D image, collect candidate ground-plane points (blue) with local surface normal close to $[0, 1, 0]$ in the image coordinate. Notice that since the depth information is noisy, candidate points may lie on the ground plane and some ground-plane points may not be identified as candidate points. (b) (c) Use RANSAC to fit the dominant plane from the candidate points. Inlier points (pixels) are marked as green. From the ground-plane equation, we determine the transformation between the image coordinate and the local 3D coordinate. In (b), points are transformed to the local 3D coordinate. (d) Perform a morphological close operation on the image to obtain a smooth ground-plane region.

center and the ground-plane. The rotation matrix $\mathbf{R}_t^g$ captures two degrees of freedom (tilt ϕ^r and roll ψ^r angles) of the robot pose, and the translation vector captures one degree of freedom, z_t^r. The transformation between the local 3D coordinate and the global 3D coordinate can be inferred by the remaining degrees of freedom of the robot pose (Section 7.2).

7.2 Robot Pose Estimation

Given the ground-plane G and the transformation $[\mathbf{R}_t^g, \mathbf{T}_t^g]$ from the camera coordinate to the local 3D coordinate, the robot pose $(x_t^r, y_t^r, \theta_t^r)$ captures the remaining three degrees of freedom of the camera pose in the 3D global coordinate. Our pose estimation takes advantages of the RGB image and the depth channel. Our method uses a robust sparse feature correspondences to obtain an initial estimate of the pose change, and then apply Iterative Closest Point (ICP) to refine the pose change. On the one hand, the projected RGB image provides appearance cues to establish feature correspondence for pose change estimation, but provide less useful information for recovering the actual 3D rigid-body transformation. On the other hand, applying ICP on the depth data is vulnerable to local optimal solution and thus, a good initial pose estimation is required to avoid converging to an incorrect pose estimation. Our method combines advantages from the RGB image and the depth data.

We start by estimating the pose change between frame t and $t-1$ by aligning sparse feature correspondences between the two RGB-D images. We extract corner features in the RGB image at frame $t-1$, and obtain their corresponding image locations at frame t using KLT tracking. A rigid-body transformation $[\mathbf{R}_t^p, \mathbf{T}_t^p]$ that aligns the 3D locations of the correspondences in the two frames can be computed. Note that once the 3D points are transformed to the local 3D coordinate, the pose change between two frames has only three degrees of freedom. In other words, $\mathbf{R}_t^p$ is a rotation matrix along the z-axis, and $\mathbf{T}_t^p$ is a translation vector on the x-y plane. With this initial estimate of the robot pose [3], we use ICP to refine $[\mathbf{R}_t^p, \mathbf{T}_t^p]$ from all the 3D points that are not on the ground plane. Finally, the robot pose in the global 3D coordinate can be computed from the pose change.

[3] If there are not enough sparse feature correspondences to compute the initial estimate, we assume the robot is moving at a constant motion and use the pose change between frame $t-1$ and $t-2$ as our initial estimate.

7.3 Features Extraction

After extracting the ground-plane, we group the points (pixels) from the current RGB-D frame into a set of vertical-plane features $\mathbf{V} = \{v_j | j = 1, ..., n_v\}$ and a set of point-set features $\mathbf{C} = \{c_j | j = 1, ..., n_c\}$. Figure 7.3 is an example of the features. A vertical-plane feature v is a plane segment that is perpendicular to the ground-plane, and a point-set feature is a cluster of 3D points that cannot be grouped into vertical planar segments. Vertical-plane features suggest the existence of walls, and point-set features suggest the existence of clutter. However, not all the vertical-plane features belong to the PSM. For example, a box on the ground also generates several vertical-plane features. On the opposite, a point-set feature that is close to a wall may simply be a small instance sticking out from the wall or noise. For example, a door knob may become a point-set feature. In this chapter, we use the vertical-plane features to generate and refine the hypotheses, and we use all features to test the hypotheses.

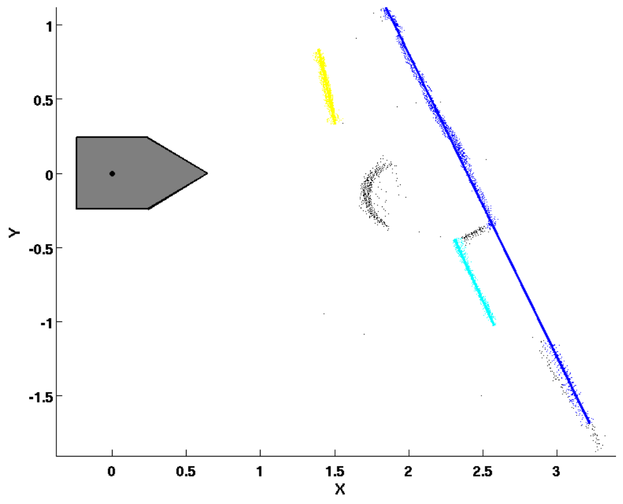
Similar to a PSM wall plane (Section 6.1), a vertical-plane feature v corresponds to a line segment in the ground-plane map. Mathematically, v is represented by

$$v = \langle \alpha^v, d^v, x_1^v, y_1^v, x_2^v, y_2^v \rangle \tag{7.3}$$

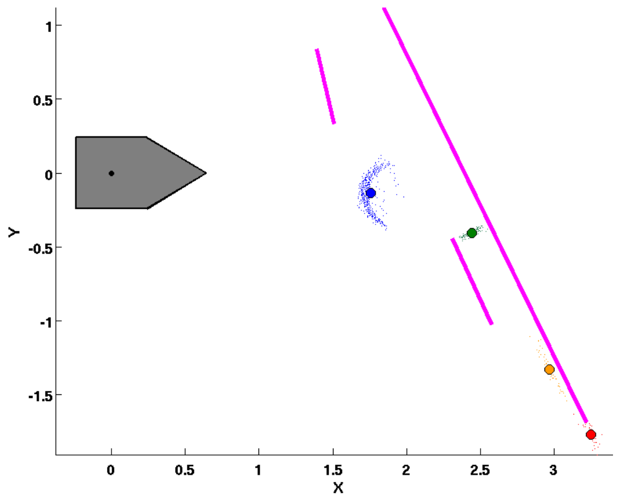
where $\alpha^v \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $d^v \in \mathbb{R}$ is the same line parametrization used to represent a wall plane, and (x_1^v, y_1^v) and (x_2^v, y_2^v) are two end-points of the line that denotes where the vertical-plane is visible.

We first extract vertical-plane features from the 3D points that are not on the ground plane. (Points are represented in the 3D global coordinate.) We project the points onto the ground-plane map, and use J-linkage [68] to fit a set of line segments. Each line segment forms a potential vertical-plane feature. Since vertical-plane features are used to generate and refine hypotheses, vertical-plane features need to be accurate. In other words, it is fine to miss a vertical-plane feature (false negative), but it is less desirable to have a vertical-plane feature that does not correspond to an actual vertical plane in 3D. Thus, we have an extra step to filter out unreliable vertical-plane features. We use a RANSAC to refine the parameters of each vertical-plane feature using the points that forms the line segments. If the number of inlier points is less than a threshold (100 points), the vertical-plane feature is not considered. The final vertical-plane feature is computed by all the inlier points.

We remove the points that form the line segments and cluster the remaining points



(a) Extract Vertical-Plane Features



(b) Extract Point-Set Features

Figure 7.3: Feature extraction from RGB-D image. This figure shows the vertical-plane features and point-set features extracted from the frame shown in Figure 7.2. (Best viewed in color.) Once the ground-plane is extracted, we extract features from the points that are not from the ground plane. (a) We first extract vertical-plane features by fitting line segments to the points in the ground-plane map. Points are colored according to the vertical-plane features that they generate, and points that cannot be grouped into line segments are marked as black. (b) We extract point-set features by clustering the remaining points based on their Euclidean distances. In general, vertical-plane features suggests the existence of walls and point-set features suggest the existence of clutter. However, some vertical-plane features may come from clutter region (e.g. the blue trash can), and some (red and orange) point-set features may be part of a wall due to noise.

based on their 3D Euclidean distances. A cluster that consists of more than 100 points forms a point-set feature. Mathematically, a point-set feature c is represented by

$$c = \langle x^c, y^c, \mathbf{P}^c \rangle \tag{7.4}$$

where $\mathbf{P}^c$ is the set of 3D points in the cluster and (x^c, y^c) is the ground-plane map projection of the 3D mean location of the points. For points that failed to form features are considered as noise and are thus, discarded.

7.4 Refining PSM Hypotheses

We use the vertical-plane feature extracted from the current frame to refine the quantitative precision of each hypothesis. Similar to Chapter 6, the Extended Kalman Filter (EKF) (Section 3.2.2) is used to estimate the parameters of each wall and the location of each occluding endpoint.

Vertical-plane features V_t are potential measurements for the walls that are visible in frame t. We associate the wall to the vertical-plane features based on the similarity of their corresponding lines on the ground-plane map. If a wall w_i is associated to a vertical-plane feature v_j, wall parameters (α_i, d_i) are updated in the correction step based on the location of the vertical-plane feature (α_j^v, d_j^v). Note that the measurement noise of the vertical-plane feature is the variance of the 3D points that form the feature.

Once the wall parameters are updated, we refine the location of each occluding endpoint. Potential measurements for an occluding endpoint are the end-points of the vertical-plane features that have similar parameters with its corresponding wall. We project these end-points onto the line of the corresponding wall, and find the nearest projected point to the endpoint. If the distance between the nearest point and the endpoint is less than 0.2 meters, the projected point is the measurement for that endpoint. An endpoint is updated in the correction step, if its measurement is available.

7.5 Incremental PSM Hypotheses Generation

In Chapter 5 and 6, hypotheses are generated by combining line segments in the image space. However, in a cluttered indoor environment, the actual ground-wall boundary may be occluded and thus, the correct PSM hypotheses may not be generated. With the RGB-D image, we can leverage on the observations from other portions of the wall, not just the ground-wall boundary, to generate hypotheses. Specifically, since a vertical-plane feature v_j suggest the existence of a wall plane, PSM hypotheses can be generated by combining vertical-plane features $\mathbf{V}_t$.

In the first frame, a set of simple hypotheses are generated. A simple hypothesis is either a PSM with two parallel walls or a PSM with at most three walls where each wall intersects with its adjacent walls. These simple hypotheses are generated by combining vertical-plane features with certain constraints. Similar to 5, hypotheses with projected wall intersections outside the image borders in the image space are infeasible because they violate the perspective geometry of the planar structure. In addition, a 3-wall hypothesis is infeasible if its left and right walls intersect in front of the end wall.

As described in Chapter 6, the key for incremental scene understanding is to generate new hypotheses by transforming existing hypotheses into children hypotheses that describes the same environment with more details as the robot travels. We use the same transformation described in Section 6.2 to generate children hypotheses. Given an parent (existing) hypothesis, its children hypotheses are generated by adding openings to the walls or by combining with a simple hypothesis generated at the current frame.

7.6 Hypotheses Testing with Partial Explanation

Hypotheses are tested using a recursive Bayesian filtering framework as described in Chapter 6. In Chapter 6, we assume that a hypothesis explains all the observed features in the current frame. This assumption works well in empty environments because all the observed features actually lie on the walls and ground planes. However, in a cluttered environment, features may be part of the clutter which cannot be explained by the PSM, as shown in Figure 7.3. This chapter presents a likelihood function that allows a good hypothesis to explain only a subset of the observed features $\mathbf{F}_t = \{\mathbf{V}_t, \mathbf{C}_t\}$.

Specifically, the likelihood function of hypothesis M_i consists of three terms,

$$p(\mathbf{F}_t|M_i) = p_c(\mathbf{F}_t|M_i)p_a(\mathbf{F}_t|M_i)p_s(\mathbf{F}_t|M_i). \tag{7.5}$$

These terms describe a three-way trade-off among the feature *coverage* by the hypothesis $p_c(\mathbf{F}_t|M_i)$, the *accuracy* of the explained features $p_a(\mathbf{F}_t|M_i)$, and the *simplicity* of the hypothesis $p_s(\mathbf{F}_t|M_i)$.

We define *coverage* to measure the amount of the observed environment that is explained by the hypothesis. Thus, coverage $p_c(\mathbf{F}_t|M_i)$ measures the number of features that M_i explains, regardless of the precision of the explanations. Formally,

$$p_c(\mathbf{F}_t|M_i) \propto \omega_v \frac{|\mathbf{V}_t^i|}{|\mathbf{V}_t|} + \omega_c \frac{|\mathbf{C}_t^i|}{|\mathbf{C}_t|} \tag{7.6}$$

where $\mathbf{V}_t^i \subseteq \mathbf{V}_t$ are the vertical-plane features and $\mathbf{C}_t^i \subseteq \mathbf{C}_t$ are the point-set features that are explained by M_i. (One can define their own metric to determine whether a feature is explained or not. Our metric is presented in Section 7.6.1 and 7.6.2.) ω_v and ω_c are the importances of explaining vertical-plane features and point-set features, respectively ($\omega_v + \omega_c = 1$) [4].

We define *accuracy* to measure the precision of the features that the hypothesis explains. Thus, accuracy $p_a(\mathbf{F}_t|M_i)$ reflects the overall error of the features that are explained by M_i. Since different hypotheses may explain different subsets of the features $\mathbf{F}_t$, we compute the weighted RMS error $error(\mathbf{F}_t, M_i)$ of the features that hypothesis M_i explains,

$$\begin{aligned} &error(\mathbf{F}_t, M_i) = \\ &\sqrt{\frac{\omega_v \sum_{v_j \in \mathbf{V}_t^i} \varepsilon_p(v_j, M_i)^2 + \omega_c \sum_{c_j \in \mathbf{C}_t^i} \varepsilon_c(c_j, M_i)^2}{\omega_v |\mathbf{V}_t^i| + \omega_c |\mathbf{C}_t^i|}} \end{aligned} \tag{7.7}$$

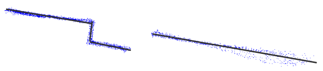
$\varepsilon_p(v_j, M_i)$ and $\varepsilon_c(c_j, M_i)$ are the error of M_i explaining feature v_j and c_j. (One can define their own error metric. Our metric is presented in Section 7.6.1 and 7.6.2.) The RMS error is modeled by a Gaussian distribution with zero mean and σ^2 variance. Mathematically,

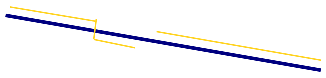
$$p_a(\mathbf{F}_t|M_i) \propto \begin{cases} 0 & \text{if } |\mathbf{V}_t^i| + |\mathbf{C}_t^i| = 0 \\ \exp \frac{-error(\mathbf{F}_t, M_i)^2}{2\sigma^2} & \text{otherwise} \end{cases}. \tag{7.8}$$

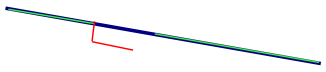
Note that both coverage and accuracy require a hypothesis to explain at least one feature in order to obtain a non-zero likelihood. In other words, a hypothesis will be filtered out by the Bayesian filter if it explains nothing in the environment.

[4] In indoor environment, a vertical-plane feature is more likely to be part of a wall plane, while a point-set feature is more likely to be clutter. Thus, we set $\omega_v > \omega_c$. In our experiments, we set $\omega_v = 0.7$ and $\omega_c = 0.3$.

(a) Frame t


(b) Vertical-plane features extracted from the observation of frame t


(d) M_2 — A simple one-wall model that explains all the features moderately well


(c) M_1 — A simple one-wall model that perfectly explains some features

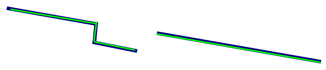

(e) M_3 — A complex four-wall segment model that perfectly explains all the features

Figure 7.4: Example of multiple valid hypotheses. (Best viewed in color.) From the environment shown in (a), assume four vertical-plane features are extracted as shown in (b) and three valid hypotheses (thick blue lines) are generated: (c) Hypothesis M_1 models the environment with one wall that perfectly explains only two of the features (green), leaving the other two unexplained (red); (d) Hypothesis M_2 models the environment with one wall that explains all the features but explains them poorly (yellow); (e) Hypothesis M_3 models the environment with four wall segments that explain all four features perfectly. Depending on the parameters in each term, the likelihood function can prefer any of the hypotheses (Table 7.1).

Coverage and accuracy define how well a hypothesis explains the observed environment. Given two hypotheses with similar levels of explanations, the likelihood function prefers a hypothesis that uses less information to achieve this level of explanation. Therefore, we define *simplicity* to prefer a hypothesis with fewer parameters.

Simplicity $p_s(\mathbf{F}_t|M_i)$ measures the amount of information that hypothesis M_i used to explain the features. Thus, simplicity is a regularization term that prevents the Bayesian filter from over-fitting the features. $p_s(\mathbf{F}_t|M_i)$ is modeled by a generalized logistic function with a negative the growth rate,

$$p_s(\mathbf{F}_t|M_i) \propto \frac{1}{1 + exp\gamma(|M_i|_t - n_{max_\gamma})}. \tag{7.9}$$

$|M_i|_t$ is the number of walls that are visible in frame t, γ is the decay rate, and n_{max_γ} is where the maximum decay occurs.

We show an example (Figure 7.4 and Table 7.1) to illustrate the trade-offs among the three terms (coverage $p_c(\mathbf{F}_t|M)$, accuracy $p_a(\mathbf{F}_t|M)$, and simplicity $p_s(\mathbf{F}_t|M)$) in the likelihood function. The importance of each term is controlled by two parameters,

Hypothesis	M_1	M_2	M_3
coverage $p_c(\mathbf{F}_t \mid M_i)$	0.50	1.00	1.00
accuracy $p_a(\mathbf{F}_t \mid M_i)$ ($\sigma = 0.2$)	1.00	0.88	1.00
accuracy $p_a(\mathbf{F}_t \mid M_i)$ ($\sigma = 0.05$)	1.00	0.14	1.00
simplicity $p_s(\mathbf{F}_t \mid M_i)$ ($\gamma = 0$)	0.50	0.50	0.50
simplicity $p_s(\mathbf{F}_t \mid M_i)$ ($\gamma = 1$)	0.88	0.88	0.27

Likelihood $p(\mathbf{F}_t \mid M_i) = p_c(\mathbf{F}_t \mid M_i)p_a(\mathbf{F}_t \mid M_i)p_s(\mathbf{F}_t \mid M_i)$			
CASE 1: $\sigma = 0.20$, $\gamma = 0$	0.25	0.44	**0.50**
CASE 2: $\sigma = 0.20$, $\gamma = 1$	0.44	**0.77**	0.27
CASE 3: $\sigma = 0.05$, $\gamma = 0$	0.25	0.07	**0.50**
CASE 4: $\sigma = 0.05$, $\gamma = 1$	**0.44**	0.12	0.27

Table 7.1: Example for the three-way trade-off among the likelihood factors. As shown in Figure 7.4, there may be multiple valid hypotheses. **Top Table:** The top half of this table shows the value of each term of each hypothesis with respect to different parameters. Depending on the parameters in each term, the likelihood function can prefer any of the hypotheses. If σ is small, accuracy $p_a(\mathbf{F}_t|M_i)$ is important since the Gaussian function in $p_a(\mathbf{F}_t|M_i)$ gives large penalties to poor explanations. Contrary, if σ is large, accuracy $p_a(\mathbf{F}_t|M_i)$ becomes less important because the Gaussian function is less discriminative between poor and good explanations. In this example, $p_a(\mathbf{F}_t|M_2)$ dramatically increases when σ increases. The decay rate γ controls the preference towards simpler hypotheses (Figure 7.5). When $\gamma = 0$, there is no preference for the simplicity of the hypothesis. As γ increases, the likelihood function will increases its preference towards simpler hypotheses. **Bottom Table:** The bottom half of this table shows the likelihood function of each hypothesis with respect to different parameters. The hypothesis with the maximum likelihood with each parameter set is shown in bold. In CASE 1, coverage is the most important factor, and since M_3 has a slightly better accuracy than M_2, M_3 has the highest likelihood. In contrast to CASE 1, CASE 2 considers both coverage and simplicity important, and thus, M_2 has a higher likelihood than the complex hypothesis M_3. In CASE 3, where only coverage and accuracy are considered, M_3 is the best because it perfectly explains all the features while M_1 and M_2 have issues with coverage and accuracy, respectively. In the above three cases, M_1 is not preferred because M_1 has a lower coverage. However, if considering all three factors (CASE 4), M_1 is the best.

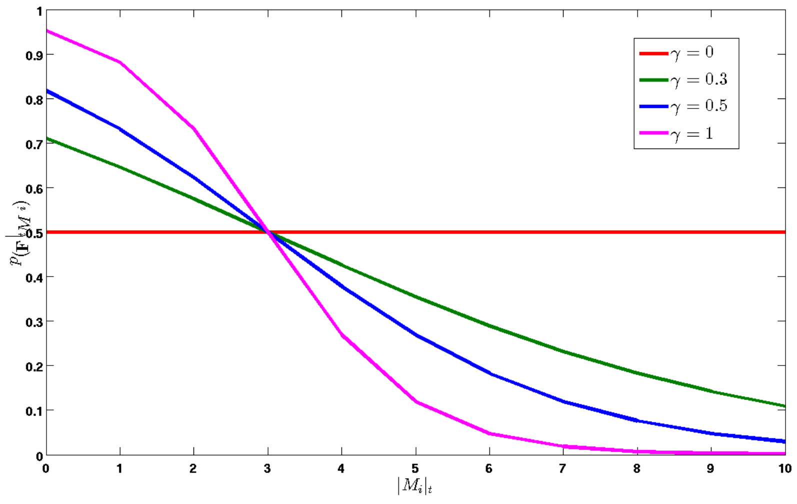
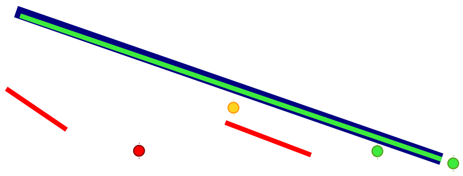


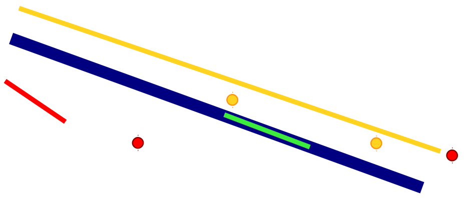
Figure 7.5: Simplicity. (Best viewed in color.) This graph visualizes simplicity $p_s(\mathbf{F}_t|M_i)$ with respect to different decay rate γ with a fixed $n_{max_\gamma} = 3$. The horizontal axis $|M_i|_t$ is the number of walls in view and the vertical axis is $p_s(\mathbf{F}_t|M_i)$. In the extreme case, where $\gamma = 0$, the likelihood function has no preference towards simpler hypotheses. As γ increases, the Bayesian filter is more likely to converge to a simpler hypothesis.

the variance of the Gaussian function σ^2 in accuracy $p_a(\mathbf{F}_t|M)$ and the decay rate γ in simplicity $p_s(\mathbf{F}_t|M)$. The variance σ^2 mainly controls the importance of accuracy. A large variance σ^2 in accuracy means that the difference between good and poor explanations is low, and therefore, the likelihood function is more reflective on the feature coverage. Contrary, a small variance σ^2 means that the penalty for poor explanation is high, and therefore, the accuracy of the explanation is highly important. The decay rate γ controls the preference towards having a simpler model as shown in Figure 7.5. In the extreme case, where $\gamma = 0$, simplicity will be the same for all hypotheses and thus, the likelihood will not prefer simpler hypotheses. As γ increases, the likelihood function will start preferring a simpler hypothesis with reasonable coverage and accuracy.

On the one hand, the parameters in the likelihood function allow the user to set their preferences based on their applications. If the purpose of exploration and mapping is to determine free-space for safe navigation, then one set of parameter values might be preferred, while if the purpose is to create an architectural CAD model of the environment, a different set of values may be preferable. On the other hand, the parameters may seem to be sensitive to the hypothesis that the Bayesian filter converges to. In fact, for the purpose of navigation, it is reasonable to converge to any of the three hy-



(a) M_4 — A good hypothesis



(b) M_5 — A mediocre hypothesis

	Full Explanation		Partial Explanation	
Hypothesis	M_4	M_5	M_4	M_5
coverage p_c	1	1	0.46	0.94
accuracy p_a	0.12	0.25	0.95	0.29
simplicity p_s	same			
Likelihood	0.12	0.25	0.44	0.31

(c) Likelihood comparisons between explaining all the features and explaining only a subset of the features

Figure 7.6: Importance of partial explanation. (Best viewed in color.) The environment in Figure 7.2 and 7.3 demonstrates the importance of allowing a hypothesis to explain only a subset of the features. Both M_4 and M_5 use one wall (blue thick line) to represent the environment, so simplicity for both hypotheses are the same. If we force both hypotheses to explain all the features, then M_5 has a higher likelihood than M_4 because it has a lower RMS error ($\omega_v = 0.7$ and $\sigma = 0.2$). If we allow the hypotheses to explain only a subset of the features, M_4 is most likely to have a higher likelihood. Green features are perfectly explained (0 error) by the hypothesis, yellow features are explained with errors, and red features are not explained.

potheses in Figure 7.4. Both M_1 and M_3 specify the same part of the environment as free-space, but they specify the free-space in a different way. M_2 is less precise in specifying the boundaries of free-space, and its accuracy $p_a(\mathbf{F}_t|M_2)$ will suggest a robot to be more cautious about the boundaries. In other words, accuracy $p_a(\mathbf{F}_t|M_i)$ provides a confidence measurement of free-space for navigation algorithms [54]. In a simple empty environment like this example, most hypotheses are reasonable. However, in a more complex or cluttered environment, many bizarre hypotheses will be generated and the likelihood function allows us to converge to a reasonable hypothesis (see Section 7.7).

Figure 7.6 is an example that shows why it is important to allow a good hypothesis to explain only a subset of the observed features. If we require a hypothesis to explain all the observed features ($p_c(\mathbf{F}_t|M_4) = p_c(\mathbf{F}_t|M_5) = 1$), then the mediocre hypothesis M_5 dominates M_4 because it has lower RMS error. However, if a hypothesis can distinguish between clutter and non-clutter features, then the higher-quality partial explanation M_4 can dominate the nearly complete but lower-quality hypothesis M_5.

7.6.1 Explaining vertical-plane features

A vertical-plane feature v is considered to be explained by hypothesis M if v can be explained by a wall segment S_j^i in M. Feature v is considered to be explained by M if the error $\varepsilon_p(v, M)$ is less than a threshold ϵ (0.1 in our experiments). The error metric $\varepsilon_p(v, M)$ is computed as follows. First, find the wall plane $W_{match} \in M$ that best matches the feature. We start by computing the displacement $dist(v, W_i)$ between feature v and each wall W_i by comparing their corresponding lines in the ground-plane map. For efficiency, we only consider a wall that has a similar angle to the vertical-plane feature ($\min(\|\alpha_i - \alpha^v\|, \pi - \|\alpha_i - \alpha^v\|) < \alpha_{same}$). If none of the walls are within the angle constraints, v is not explained by M. For each wall, we construct a coordinate for computing the displacement by an origin (x_i^g, y_i^g) and an angle θ_i^g [66]:

$$\begin{aligned} x_i^g &= \frac{l_i(x_{1,1}^i + x_{2,m_i}^i) + l^v(x_1^v + x_2^v)}{2(l_i + l^v)} \\ y_i^g &= \frac{l_i(y_{1,1}^i + y_{2,m_i}^i) + l^v(y_1^v + y_2^v)}{2(l_i + l^v)} \\ \theta_i^g &= \frac{l_i\alpha_i + l^v\alpha^v}{l_i + l^v} \end{aligned} \tag{7.10}$$

where l^v is the length of feature v and l_i is the total length of W_i that has been modeled. The x-axis of the coordinates is along the weighted average direction of the two lines and thus, the displacement $dist(v, W_i)$ of the two lines is defined as the maximum difference along the y-axis. A point $([x]^m, [y]^m)$ in the ground-plane map can be transformed to this coordinate $([x]^{g_i}, [y]^{g_i})$ by

$$\begin{aligned} [x]^{g_i} &= ([y]^m - y_i^g)sin\theta_i^g + ([x]^m - x_i^g)cos\theta_i^g \\ [y]^{g_i} &= ([y]^m - y_i^g)cos\theta_i^g - ([x]^m - x_i^g)sin\theta_i^g \end{aligned}. \tag{7.11}$$

W_{match} is found by $W_{match} = \arg\max_{W_i}(dist(v, W_i))$. Finally, if the entire feature v lies within a single wall segment in W_{match}, then $\varepsilon_v(v, M) = dist(v, W_{match})$. Otherwise, the feature is not explained by M.

7.6.2 Explaining point-set features

Similar to the vertical-plane feature, a point-set features c is considered to be explained by hypothesis M if c lies on a wall segment S_j^i in hypothesis M. We define an error metric $\varepsilon_c(c, M)$ to be the shortest distance of the point-set feature (x^c, y^c) to a wall W_i in M. Feature c is not explained by M if the error is larger than a threshold ϵ. (This is the same threshold ϵ for explaining vertical-plane features.) Feature c is also not explained if the projected location of the feature does not lie within the bound of any wall segments of W_i. Moreover, we take into consideration the distribution of the points $\mathbf{P}^c$ in c. Only if 70% of the points are within ϵ of distance to wall W_i, point-set feature c is explained.

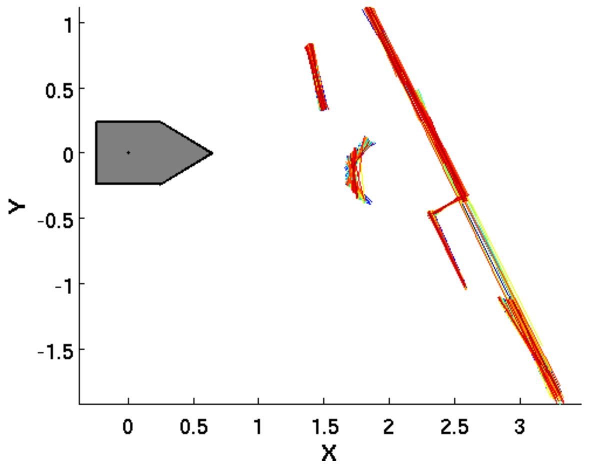
7.7 Evaluation

7.7.1 Repeatability of Feature Extraction

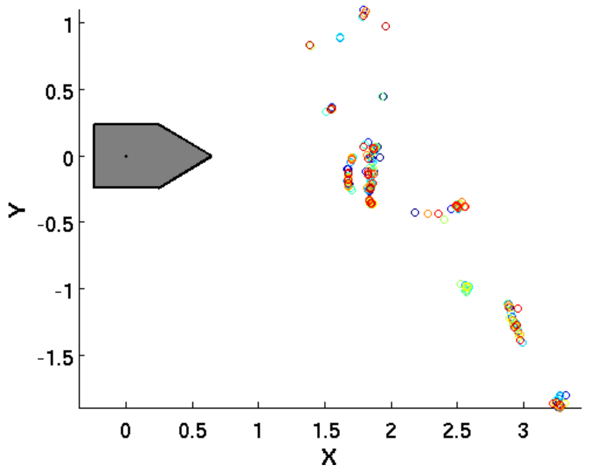
Due to the randomness in the process of extracting the ground plane and the features, we ran a small experiment to show the stability and the repeatability of these process. Note that the randomness comes from RANSAC and the clustering methods involved in these processes (see Section 7.1 and 7.3). We extract vertical-plane features and point-set features in the frame shown in Figure 7.3 50 times. At each run, we first extract the ground-plane and transform the non-ground-plane points to the local 3D coordinate. Then, we extract vertical-plane features and point-set features from the non-ground-plane points. The result of this experiment is shown in Figure 7.7. Our result shows the features extracted in each run are similar. This experiment also reveals the amount of noise we get from feature extraction.

7.7.2 Evaluation of the Framework

Our on-line generate-and-test framework for cluttered environments is evaluated using the Michigan Indoor Corridor 2014 Video Dataset (See Chapter 4). Parameters for the likelihood function are set as follows. The weights between the vertical-plane features and the point-set features are $\omega_v = 0.7$ and $\omega_c = 0.3$ because a vertical-plane feature is more likely to be part of a wall segment than a point-set feature. For accuracy $p_a(\mathbf{F}_t|M_i)$, the variance σ^2 is set to $\sigma^2 = 0.04$. The value is selected based on two factors. The first factor is the error of a single vertical plane observed from various distances and angles



(a) Extracted vertical-plane features in multiple runs



(b) Extracted point-set features in multiple runs

Figure 7.7: Extracted features in multiple runs. (Best viewed in color.) Features extracted within one run is shown in the same color. (a) Vertical-plane features from the wall, the pillar, and the blue trash can are extracted in all runs. Due to the noise in the depth sensor, multiple vertical-plane features are extracted along the wall. These vertical-plane features have very similar parameters (α^v, d^v) that specify the line in the ground-plane map but different in the two location of the two end-points (x_1^v, y_1^v) and (x_2^v, y_2^v). This figure also shows the amount of noise we generally get from our depth camera. The noise of the vertical-plane feature increases as the distance to the robot increases. The noise from the wall is about 0.1 meter. In half of the runs, one or two vertical-plan features are extracted from the red round can, and in the other half, several point-set features are extracted. (a) The majority of the point-set features are extracted from the red round can. Other point-set features are noise from the wall and around the pillar. More point-set features are extracted from the left end of the wall because that end is further away from the robot where depth measurements are less accurate.

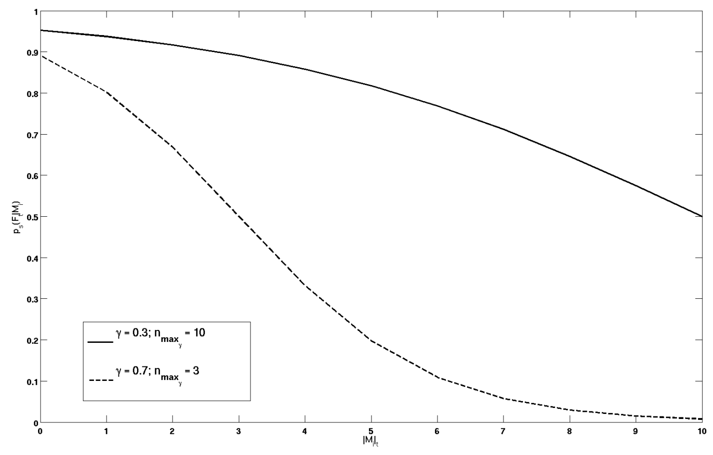


Figure 7.8: Simplicity used in our experiments. Parameters of simplicity $p_s(\mathbf{F}_t|M_i)$ is set to $\gamma = 0.3$ and $n_{max_\gamma} = 10$ (black line). Simplicity will be 0.93 if one wall is used to describe the current view and 0.81 when using five walls. Simplicity drops significantly if more than six walls are used. In Section 7.7.3, we use a set of parameters, $\gamma = 0.7$ and $n_{max_\gamma} = 3$ (dashed line), that strongly prefers simple structures to compare and demonstrate the effects of simplicity.

after converting to the local ground-plane map at a single snapshot. Based on our data, the maximum error of a point belonging to a vertical plane can be as large as 0.1 meter. Figure 7.7 shows how we collect this statistics. The second factor accounts for the pose estimation because the likelihood is computed after transforming the points to the global ground-plane map. Since the error in pose estimation is accumulative, in a more realistic setting, the variance σ^2 should increase over time. For simplicity $p_s(\mathbf{F}_t|M_i)$, $\gamma = 0.3$ and $n_{max_\gamma} = 10$. A visualization of simplicity $p_s(\mathbf{F}_t|M_i)$ with these parameters are shown in Figure 7.8. These values are selected to ensure that the likelihoods for having one to five walls are similar because these are the common numbers of walls that could appear in a single snapshot based on the field of view (57 degree horizontally) of our depth camera.

Figure 7.9 visualizes the results on these videos. Our method converges to a reasonable hypothesis to describe each environment in 3D. Based on the maximum *a posteriori* (MAP) hypothesis, we separate out clutter, observations that were not explained by the hypothesis. At each frame, each hypothesis has its own partition of explained and unexplained features. The unexplained observations are the 3D points that contribute to these unexplained features at each frame. We further cluster these unexplained observations into a set of 3D regions based on their Euclidean distances. In most cases, a cluttered region is either an object or a pile of objects, but in some situations (Dataset

INTERSECTION), the cluttered region may be part of the building, such as a pillar along a wall. In Dataset LAB, the white board is tilted about 25 degree, and since it is not perpendicular to the ground plane, it is separated out as clutter and not a part of the PSM. Notice that some of the clutter points are caused by errors in the pose estimation. The pose estimation is less accurate in Dataset CORRIDOR because a large portion of the pixels have unreliable depth measurements due to distances.

As stated in Chapter 1, once clutter points are clustered into regions, each clutter region becomes a smaller interpretation problem. Although the focus of this thesis is to parse the scene into a structural model and clutter regions, we show an example of reasoning about objects segmented out from one clutter region. We select to demonstrate this example using the functionality-based object reasoning method proposed in [28], because this method reasons about objects based on their geometric properties in 3D, and the preprocessing of this method is to identify and remove ground and wall-plane points. This example is shown in Figure 7.10. This example demonstrates that the on-line generate-and-test framework proposed in this thesis connects well with and simplifies researches on object reasoning.

To evaluate our method quantitatively, two accuracy metrics, *Plane Accuracy* and *Scene Accuracy*, are defined to measure how a hypothesis agrees with the ground-truth labeling at one frame. Although the hypothesized interpretation are in 3D, the ground-truth is manually labeled in the projected image space. *Plane Accuracy* measures the accuracy of the hypothesized PSM model and the estimated camera pose without considering clutter. The PSM is projected to the image space according to the estimated robot pose. With the projected PSM, each pixel is assigned to a plane in PSM (ground plane or a wall). *Plane Accuracy* is the percentage of the pixels that agrees with the ground-truth plane labels. *Scene Accuracy* measures the accuracy of the whole interpretation (PSM + clutter) and the estimated camera pose. To measure *Scene Accuracy*, both the PSM and the clutter points are projected to the image space. The projected clutter regions are first determined by performing a morphological close operation on the projected clutter points. Pixels from non-clutter regions are then assigned to a plane according to the projected PSM. *Scene Accuracy* is the percentage of the pixels that agrees with the ground-truth labeling in terms of plane label and clutter. When computing the accuracies, only non-ceiling pixels with valid depth data are considered, because ceiling is not modeled in PSM and pixels with invalid depth data are not used to compute the likelihood. For each dataset, two types of quantitative accuracy are

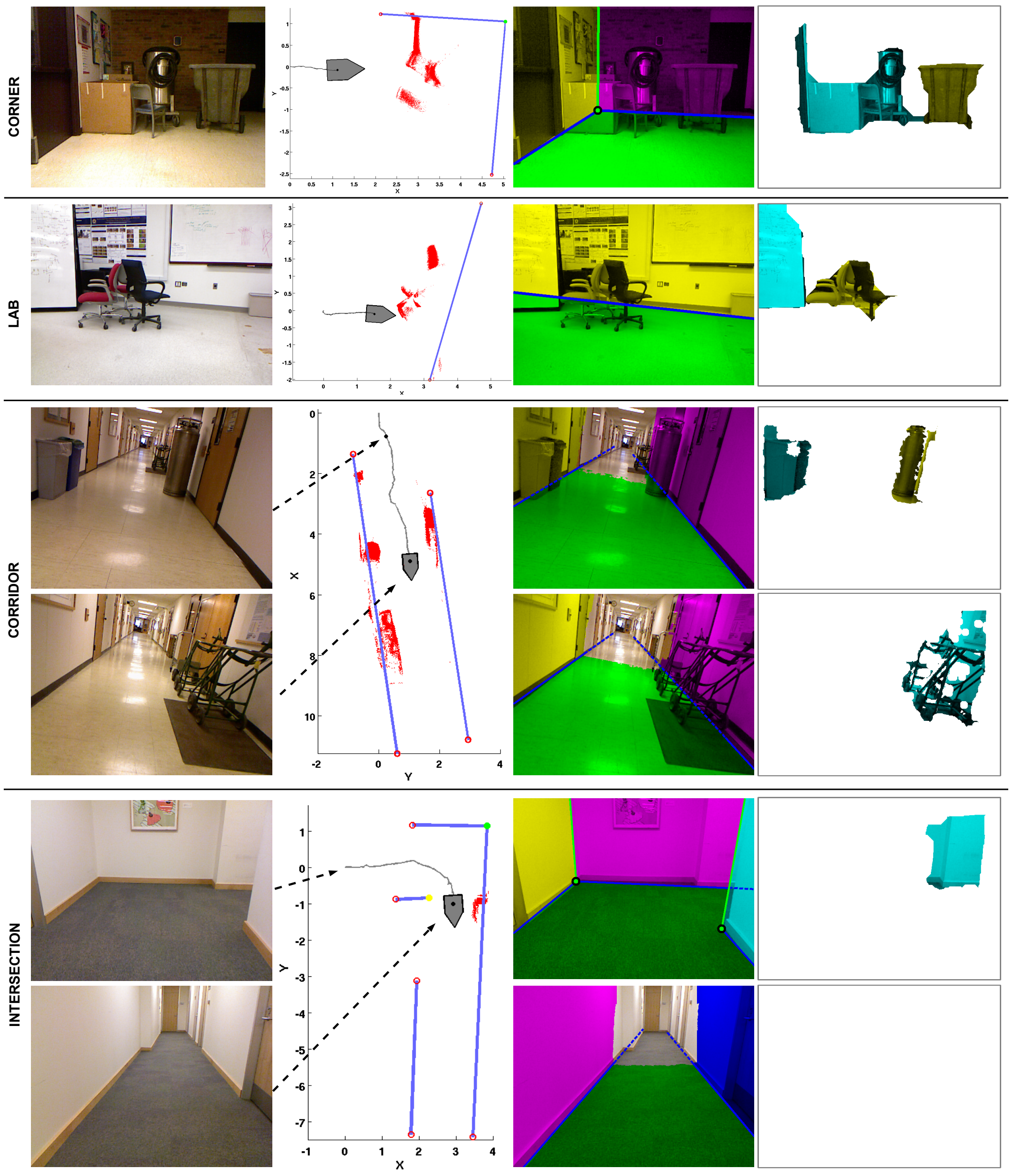
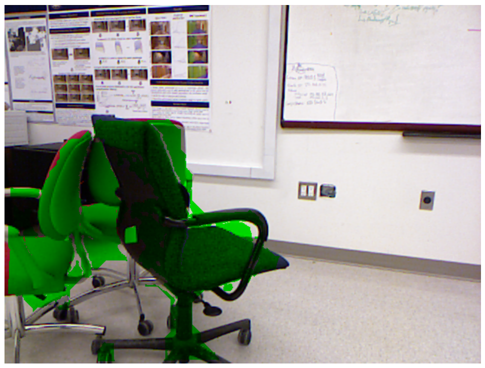
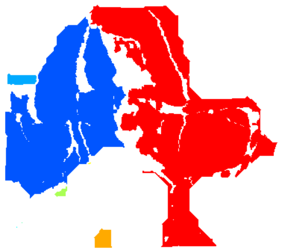


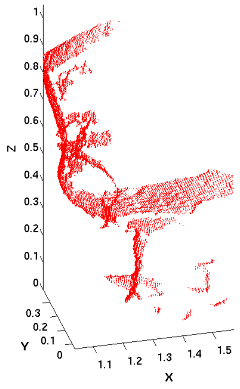
Figure 7.9: Qualitative evaluation of the our framework in cluttered environments. (Best viewed in color.) First column: image from our dataset. (The depth images are not shown.) Second column: 3D interpretation (PSM + clutter) obtained by our method visualized in the ground-plane map. For PSM, the blue lines are the wall planes and the big dots are the endpoints, color coded based on their types (green: dihedral; yellow: occluding; red: indefinite). Clutter is represented by a 3D point cloud (tiny red dots). Third column: image projection of the PSM. The ground is green and each wall is shown in a different color. The part of the images that are not painted are too far away from the robot to obtain a reliable depth data. The robot will start modeling those regions as it gets closer. Fourth column: clutter regions in the image space with different colors. (See text for more details.)



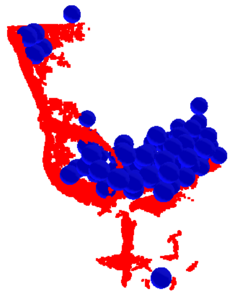
(a) Clutter region from our method



(b) Post processing



(c) 3D Points of the largest segment



(d) Physical simulation for object functionality [28]

Figure 7.10: An example of interpreting a clutter region with objects. This example demonstrates that our proposed method supports researches that reason about objects. (Best viewed in color.) (a) We select a frame in Dataset LAB that contains sufficient information to reason about the chairs. (b) In order to segment out the chair in the center, we perform connected component analysis on the 2D image and select the the largest component (red). Note that a more generalized method can be developed to better segment out individual objects. Since segmentation is not the focus of this thesis, we apply simple method that works for this specific example. (c) We collect 3D points that associate to the 2D pixels of the red component, and apply the functionality classifier [28] to this 3D point cloud to reason about the functionality of this object. (d) A physical simulation is performed by dropping balls on to the objects [28]. The classifier then uses the distribution of the balls to classify the object. Among all functionality classes ("sittable", "table-like", "cup-like", and "layable"), this 3D point cloud is classified as a "sittable" object.

Dataset		CORNER	LAB	CORRIDOR	INTER.	Overall
# Frames		49	131	106	110	396
Clutterness		34.84	20.80	9.50	10.22	16.50
Plane Acc.	*MAP*	98.18 %	99.31 %	97.83 %	98.25 %	98.49 %
	Weighted	90.18 %	91.99 %	97.03 %	91.56 %	92.97 %
Scene Acc.	*MAP*	92.82 %	95.10 %	91.76 %	98.16 %	94.83 %
	Weighted	86.47 %	86.83 %	90.89 %	89.72 %	88.68 %
% Best Selected		80.00 %	100 %	90.91 %	91.67 %	92.86 %

Table 7.2: Quantitative evaluation of our framework in cluttered environments. See text for more discussion.

reported using the two accuracy metrics described above. *MAP* hypothesis accuracy is the accuracy of the hypothesis of the maximum posterior probability at each frame. *Weighted* accuracy is the weighted average accuracy of all the existing hypotheses at each frame where the weight of each hypothesis is equal to its posterior probability. The average of these accuracies are reported in Table 7.2.

In order to understand how cluttered each dataset is, we report the *Clutterness* based on the ground-truth labeling. *Clutterness* is the percentage of pixels that are labeled as clutter. To further evaluate our method, we report how often our method selects the best hypothesis. The best hypothesis is the hypotheses with the maximum *Plane Accuracy* among the active set of hypothesis at each frame. The metric, % Best Selected, is the percentage of frames that the accuracy of the MAP hypothesis equals to the best hypothesis. In general, our method starts selecting the best hypothesis before the tenth frame. In Dataset LAB, we can even select the best hypothesis solely from the observations made in the first frame.

7.7.3 Analyzing the Likelihood Function

To empirically analyze the importance of each term in the likelihood function, we ran our framework multiple times with different likelihood settings. Table 7.3 summarizes the experiments that we ran to analyze the likelihood function. In order to have a fair comparison, these experiments are setup so that the output from the preprocessing steps (e.g. ground-plane extraction, pose estimation, and feature extraction) are exactly the same. By using the same set of features, all experiments start with the same initial set of hypotheses but the hypotheses that were incrementally generated in each experiment

Experiment	coverage p_c	accuracy p_a	simplicity p_s
Full-Likelihood	✓	✓	✓
Omit-Clutter-Concept	Omit the concept of clutter		
Omit-Coverage-Term		✓	✓
Omit-Accuracy-Term	✓		✓
Omit-Simplicity-Term	✓	✓	
Extreme-Simplicity-Term	✓	✓	$\gamma = 0.7$, $n_{max_\gamma} = 3$

Table 7.3: List of experiments that are used to analyze the likelihood. A ✓in the table means the likelihood term is used in that run, and the parameters for that term is the same as those presented in Section 7.7.2. An empty box means the term is not used. Full-Likelihood runs the framework with the proposed likelihood (all three terms), and results about this run was discussed in Section 7.7.2. Omit-Clutter-Concept assumes the environment is empty and omits the concept of clutter. This is the same assumption we made in Chapter 5 and Chapter 6. Without the concept of clutter, the hypothesized model are tested based on their abilities to explain all the observed features. In this experiment, coverage will always be one because the hypothesized model are assumed to explain all features, and accuracy is computed by combining errors from all features. Comparing the results from Omit-Clutter-Concept with those from Full-Likelihood demonstrates the importance of allowing a hypothesized model to have partial explanation of the environment. In Omit-Coverage-Term, Omit-Accuracy-Term and Omit-Simplicity-Term, we take out a term from the likelihood function. Comparing the results from each of these runs with the results from Full-Likelihood, we analyze the importance of the omitted term. Note that although we are not using coverage in Omit-Coverage-Term, this experiment still allows partial explanation because accuracy $p_a(\mathbf{F}_t|M_i)$ only considers features that are explained. Extreme-Simplicity-Term uses all three terms but sets the parameters of simplicity to have a strong preference towards very simple structures. This simplicity is visualized in Figure 7.8. (In other experiments, $\gamma = 0.3$ and $n_{max_\gamma} = 10$.)

Experiment	Plane Accuracy	Scene Accuracy
Full-Likelihood	98.49%	94.83%
Omit-Clutter-Concept	67.11%	58.93%
Omit-Coverage-Term	81.87%	74.74%
Omit-Accuracy-Term	98.16%	94.70%
Omit-Simplicity-Term	98.49%	94.83%
Extreme-Simplicity-Term	98.21%	94.99%

Table 7.4: Results for analyzing likelihood function. These experiments are described in Table 7.3. Comparing Full-Likelihood and Omit-Clutter-Concept shows the importance of allowing partial explanation. Comparing the results from experiments that omit a term, we discovered that coverage is the most important factor in the likelihood function.

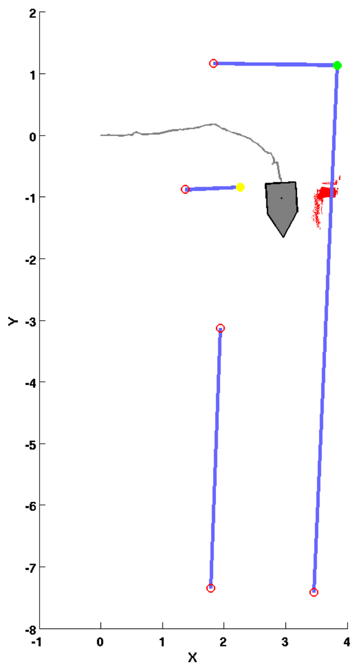
may be different. This is because the incremental hypothesis generation process depends on the active set of hypotheses, and different likelihood settings may result in a different active set of hypotheses. The overall accuracies of these experiments are reported in Table 7.4.

By comparing Omit-Clutter-Concept and Full-Likelihood, we demonstrate the importance of allowing a hypothesis to explain only a subset of the features. In Omit-Clutter-Concept, the likelihood function combines the scores from all features, and the score of each feature depends on the error between the feature and the hypothesized PSM model. Note that the likelihood function in Omit-Clutter-Concept is equivalent to the likelihood function proposed in Chapter 5 and 6. In Omit-Clutter-Concept, only INTERSECTION ends up with the correct hypothesis. The reason for INTERSECTION to have comparable result is because the only region that is not described by the PSM is the pillar. The error between the pillar and the correct PSM model is relatively small (0.3 meter) compare to the clutter regions in other datasets, and therefore, Omit-Clutter-Concept is still able to select the correct hypothesis. For the other three datasets, the MAP hypothesis at the last frame diverges greatly from the correct hypothesis. The overall MAP *Plane Accuracy* is only 67.11%, while Full-Likelihood has 98.49%. In fact, the correct hypothesis has a very bad posterior probability from the beginning of each video and thus, is removed by the Bayesian filter before the end of the video. Thus, having a likelihood function that allows partial explanation is a key to handle cluttered environments.

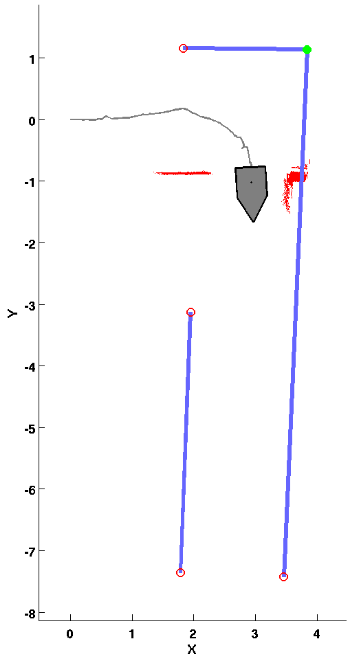
The most important term in the likelihood function is coverage $p_c(\mathbf{F}_t|M_i)$. Among Omit-Coverage-Term, Omit-Accuracy-Term and Omit-Simplicity-Term experiments, Omit-Coverage-Term performs the worst while the other two experiments have similar results to Full-Likelihood. In Omit-Coverage-Term, only one of the dataset (LAB) converges to the correct hypothesis at the end of the video. However, unlike Omit-Clutter-Concept, the correct hypothesis still remains in the active set of hypotheses, except it does not have the maximum posterior probability. The overall MAP *Plane Accuracy* for Omit-Coverage-Term is 81.87%, while Full-Likelihood has 98.49%.

Simplicity is also an important term in the likelihood function. Since the parameters we set in simplicity is less discriminative when the number of walls is less than five (the maximum number of visible walls in our dataset is three), the overall MAP *Plane Accuracy* in No-simplicity is the same as that in Full-Likelihood. To really demonstrate the importance of simplicity, we run Extreme-Simplicity-Term, which has a simplicity that strongly prefers a simple structure. In Extreme-Simplicity-Term, three datasets (LAB, CORNER, CORRIDOR) converges to the correct hypothesis. Since there is only one wall in LAB, Extreme-Simplicity-Term converges to the correct hypothesis earlier than Full-Likelihood. However, Extreme-Simplicity-Term converges to a wrong hypothesis in INTERSECTION because the initial part of INTERSECTION has three visible walls. Both Extreme-Simplicity-Term and Full-Likelihood end up with the same two hypotheses at the last frame, but they have a very different posterior probability distribution over the two hypotheses. These results are shown in Figure 7.11.

The effect of accuracy $p_a(\mathbf{F}_t|M_i)$ is relatively small compare to coverage and simplicity in our experiments. In Omit-Accuracy-Term, the overall MAP *Plane Accuracy* is 98.16%, which is very similar to that in Full-Likelihood. Since we are using a depth camera, most of the explained features are pretty accurate. Thus, the difference in accuracy among the hypotheses at each frame is low. However, if a different sensor is used, the importance of accuracy may increase.



(a) Hypotheses M_1



(b) Hypotheses M_2

Experiment	$p(M_1 \vert \mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_t)$	$p(M_2 \vert \mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_t)$
Full-Likelihood	0.87	0.13
Omit-Simplicity-Term	0.90	0.10
Extreme-Simplicity-Term	0.18	0.82

(c) Posterior Probability Distribution

Figure 7.11: Final hypotheses from INTERSECTION. In Full-Likelihood, Omit-Simplicity-Term, and Strong-Simplicity, the framework end up with two hypotheses (M_1 and M_2) at the last frame of INTERSECTION. However, their posterior probability distributions are different. In Full-Likelihood, M_1 is the MAP hypothesis. In Omit-Simplicity-Term, M_1 is also the MAP hypothesis but the posterior probability of M_1 is slightly higher. The reason for Omit-Simplicity-Term to have a higher preference towards M_1 is because Omit-Simplicity-Term only tests the hypotheses based on their ability to explain the features without any preference of the simplicity of the structure. M_1 has a better coverage than M_2. Contrarily, in Extreme-Simplicity-Term, M_2 is the MAP hypothesis because in Extreme-Simplicity-Term, simplicity drops significantly when there are more than two walls in view. Thus, M_1 has a much lower likelihood than M_2 at the beginning of the video, when the robot is at the first corridor.

7.8 Summary

In this chapter, we generalize the on-line generate-and-test framework to handle cluttered indoor environments. We define the concept of "clutter" as regions in the local environment that cannot be represented by the Planar Semantic Model (PSM). Thus, in a cluttered environment, a PSM hypothesis explains only a subset of the observed environment, and different hypotheses have different partitions between regions that are explained by the model and regions that are clutter. We propose a likelihood function that allows a good hypothesis to provide a partial explanation of the observations. The likelihood function makes explicit a three-way trade-off among coverage of the observed features, accuracy of the explanation, and simplicity of the hypotheses. We evaluate the effectiveness of each factor analytically and empirically.

We implement our on-line generate-and-test framework with partial explanation using a stream of RGB-D images. Our experimental results on a variety of videos demonstrate that our method is capable of interpreting cluttered environment by a Planar Semantic Model and clutter. Finally, we compare our framework with and without the concept of clutter. This comparison demonstrates the importance of allowing partial explanation for model-based interpretation in cluttered environments.

Chapter 8

Focusing Attention on Features that Matter

As described in Chapter 5, we extract and track point features in the image space to serve as observations for testing the PSM hypotheses [1]. The typical way to collect features is extracting features that are easily detectable with a fix threshold. In fact, some of the most informative features to discriminate the hypotheses may not be extracted if features are detected by fixed thresholds, because the most informative regions may not have high image contrasts for features to be detected.

This chapter demonstrates that by focusing attention on features in the informative regions, we can test the hypotheses more efficiently. We divide the image into regions based on the expected information gain that each feature provides, which we call *informativeness*. The idea of focusing on informative regions of the image space is inspired by the idea of saliency detection [36, 35, 62]. While these works typically define saliency regions based on image and motion properties of the pixels in the images [36, 35] or based on human fixations [62], our informative regions are defined in terms of the robot's current set of hypotheses about the geometric structure of the indoor environment. We adapt the threshold for extracting features for each region based on its informativeness. If a region is more informative, features with lower image contrasts are allowed to be used for hypotheses testing. We demonstrate the attention focusing method on the on-line generate-and-test framework described in Chapter 6 [2].

[1]Materials presented in this chapter were published in [72].

[2]The attention focusing method is equally applicable to other scene understanding problems (e.g. [45, 25, 57, 44]).

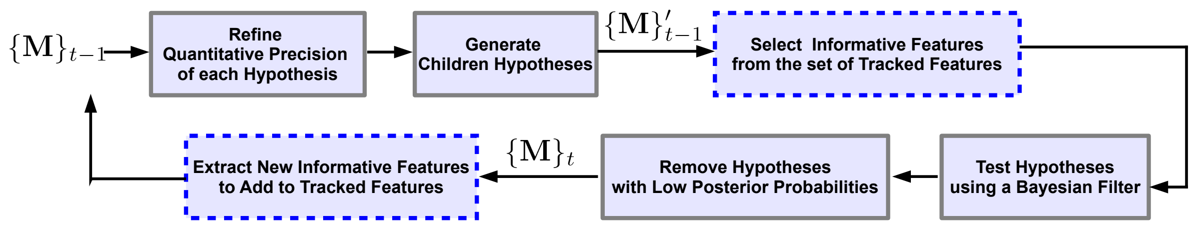

$\{\mathbf{M}\}_{t-1}$
Refine Quantitative Precision of each Hypothesis
Generate Children Hypotheses
$\{\mathbf{M}\}'_{t-1}$
Select Informative Features from the set of Tracked Features
Test Hypotheses using a Bayesian Filter
Remove Hypotheses with Low Posterior Probabilities
$\{\mathbf{M}\}_{t}$
Extract New Informative Features to Add to Tracked Features

Figure 8.1: Framework with attention focusing method. (Best viewed in color.) The steps with solid gray blocks are demonstrated in Chapter 6, and the steps with dashed blue blocks show where we select and extract the informative features. After generating children hypotheses ($|\{\mathbf{M}\}'_{t-1}| \geq |\{\mathbf{M}\}_{t-1}|$) , we select point features from the current set of tracked features that are informative to discriminate among the hypotheses $\{\mathbf{M}\}'_{t-1}$. Once the posterior probabilities of the hypotheses are updated and hypotheses with low probabilities are removed ($|\{\mathbf{M}\}_t| \leq |\{\mathbf{M}\}'_{t-1}|$), we identify new informative features based on the current set of hypotheses $\{\mathbf{M}\}_t$ to add into the tracking set. These features will be used to test hypotheses in future frames.

Figure 8.1 describes the on-line generate-and-test framework with the attention focusing method. This chapter describes a method for selecting the set of point features $\mathbf{P}$ that are most informative for testing the set $\mathbf{M}$ of hypothesized models. The goal is to select features $\mathbf{P}$ that maximize the information gain $IG(\mathbf{M}, \mathbf{P})$:

$$IG(\mathbf{M}, \mathbf{P}) = H(\mathbf{M}) - H(\mathbf{M}|\mathbf{P}) \tag{8.1}$$

where $H(\mathbf{M})$ is the entropy of the current set of hypotheses and $H(\mathbf{M}|\mathbf{P})$ is the entropy given the set of point features $\mathbf{P}$. To explicitly maximize Equation 8.1, we need to test the hypotheses with all combinations of all possible features, and then select the combination that returns a minimum expected entropy $H(\mathbf{M}|\mathbf{P})$. This process is very costly. However, we observe that a point feature will increase $IG(\mathbf{M})$ only if at least two hypotheses have different explanations about its 2D motion. In other words, a point p_j is "informative" if it lies in a region where at least two hypotheses make different predictions. We define $I(p_j, \mathbf{M}) \in [0, 1]$ to be the *informativeness* of point p_j, measuring its discriminating power among the set $\mathbf{M}$,

$$I(p_j, \mathbf{M}) = \log(|\mathbf{M}|) - H(\mathbf{M}^u|p_j), \tag{8.2}$$

where $H(\mathbf{M}^u|p_j)$ is the expected entropy of the set $\mathbf{M}$ with uniform prior. Higher informativeness $I(p_j, \mathbf{M})$ means the point is able to provide larger information gain.

If all hypotheses explain the 2D motion of point p_j in the same way, the point is not informative $I(p_j, \mathbf{M}) = 0$. Section 8.1 describes how the informativeness $I(p_j, \mathbf{M})$ of a point is computed, and Section 8.2 describes how we identify a set of informative points $\mathbf{P}$ for a set $\mathbf{M}$ of hypotheses. Our experimental results (Section 8.3) demonstrate that this bias of the search toward the most informative point features helps the Bayesian filter to converge to a single hypothesis more efficiently, without loss of accuracy.

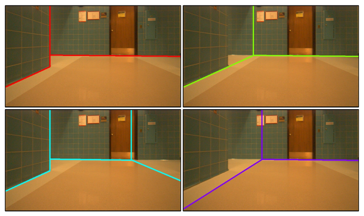
8.1 Compute Informativeness of a Point Feature

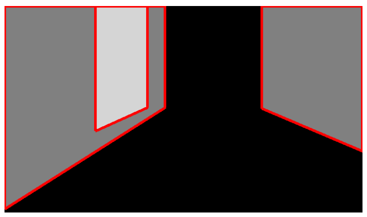
For any point p_j in the image space, its informativeness $I(p_j, \mathbf{M})$ reflects how informative that point is for testing the current set of hypotheses $\mathbf{M} = \{M_1, M_2, ..., M_N\}$. $I(p_j, \mathbf{M}) \in [0, 1]$ is positive if the point is capable of discriminating at least two hypotheses, and is zero if the point does not provide any information to discriminate among any hypotheses.

Given two hypotheses, if a point is informative $I(p_j, \mathbf{M}) > 0$, the two hypotheses have different explanations about its 2D motion. A hypothesis predicts the 2D motion of point p_j by reconstructing the point in 3D based on the 3D plane that the point is on, and then projects the point onto another frame (Chapter 5). Thus, the key for the two hypotheses to have different predictions is when the two hypotheses assign the point to different 3D planes. If there is a difference from this pair of hypotheses, $I(p_j, \mathbf{M})$ of point p_j increases. At the end, $I(p_j, \mathbf{M})$ is the sum of scores from all possible pairs of the current hypotheses.

In fact, the informativeness $I(p_j, \mathbf{M})$ of all the points can be divided into several regions, where all points within each region have the same $I(p_j, \mathbf{M})$. Figure 8.2(b) is an example of these regions. For efficiency, instead of computing the precise boundaries of these regions, we approximate these regions with a set of non-overlapping boxes that specify which portions of the image are informative. The upper bounds of these boxes are the top image border. All points within each box are set to the same $I(p_j, \mathbf{M}) > 0$ value, and any point that is outside the boxes has $I(p_j, \mathbf{M}) = 0$. Figure 8.2(c) is an example of our box approximation. Note that it is possible that a point that originally has zero informativeness becomes non-zero in our box approximation, but all informative points remain informative. Thus, we do not lose any information by using this approximation.

Formally, we represent our box approximation based on a set of non-overlapping boxes $\{b_1, b_2, ..., b_{n_b}\}$. The informativeness $I(b_k, \mathbf{M})$ of each box b_k is proportional to the


(a) Hypotheses


(b) Informative Regions

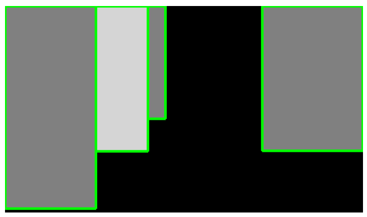

(c) Box Approximation

Figure 8.2: Informative regions. This figure is an example of the informative regions and our box approximation for those regions. (Best viewed in color.) (a) The current set of hypotheses at this frame. (b) The gray-scale value reflects the informativeness $I(p_j, \mathbf{M}) \in [0, 1]$ of each pixel p_j in the current image based on the four hypotheses shown in (a). Since the hypotheses are qualitatively distinctive, the image divides into several regions based on the informativeness. However, to precisely compute the exact boundary of these regions can be computationally expensive. Thus, we use a set of boxes to approximate these regions as shown in (c). All points within each box are set to the same $I(p_j, \mathbf{M}) > 0$ value, and any point that is outside the boxes has $I(p_j, \mathbf{M}) = 0$. The informativeness of each box is set to the maximum informativeness among all pixels within the box, so no information is lost by using the box approximation.

number of hypothesis pairs that the point can discriminate,

$$I(b_k, \mathbf{M}) = \begin{cases} \frac{1}{n_b(n_b-1)/2} \sum_{M_m, M_n \in \mathbf{M}} \delta(b_k, M_m, M_n) & \text{if } N > 1 \\ 0 & \text{otherwise} \end{cases} \tag{8.3}$$

where $\delta(b_k, M_m, M_n) \in \{0, 1\}$ equals to 0 if hypotheses M_m and M_n are the same within box b_k, and equals to 1 if the hypotheses differ. Two hypotheses are the same if the associated 3D wall that is projected to the box area b_k is the same for the two hypotheses. We check whether the two walls are the same in 3D. Since the walls are perpendicular to the ground, a 3D wall is parameterized by a line $W = (\alpha, d)$ on the ground plane, where α is the orientation of the line which implies the normal direction of the wall plane in 3D, and d_j is the directed distance from the origin of the ground-plane map to the line. With this parameterization,

$$\delta(b_k, M_m, M_n) = \begin{cases} 0 & \text{if } |\alpha_m - \alpha_n| < \alpha_{same} \text{ and } |d_m - d_n| < d_{same} \\ 1 & \text{otherwise} \end{cases} \tag{8.4}$$

where (α_m, d_m) and (α_n, d_n) are the walls in hypothesis m and n, respectively. α_{same} and

d_{same} are the thresholds for considering the two walls to be the same. In our experiments, $\alpha_{same} = 0.00872$ radian and $d_{same} = 0.05$ meter.

To find the boxes, we start by finding their left and right bounds, and then find the lower bound. (The upper bound lies along the top image border.) The left and right bounds of the boxes correspond to a set of break points along the image columns. These break points only occur at the projected image locations of the vertical wall borders of the current hypotheses. We sort all the break points from the left to the right to form the bounds of the boxes, and form a set of candidate boxes using adjacent bounds. We then compute the informativeness of each box using Equation 8.3, and remove boxes that have $I(b_k, \mathbf{M}) = 0$. For each informative box, the lower bound is the lowest horizontal line that encloses the ground-wall boundary segment of all hypotheses that pass through this box. Note, if the lowest horizontal line of a box is below the border of the image, the lower bound is set at the image border.

8.2 Select Informative Point Features

To test the hypotheses $\mathbf{M} = \{M_1, M_2, ..., M_N\}$, a common approach is to extracts point features that have high corner responses from the entire image $\mathtt{I}_t$. The corner response $V(p_j)$ of a point p_j is defined as the minimum eigenvalue of the covariance matrix of derivatives over its neighborhood $S(p_j)$ [63]

$$\begin{bmatrix} \sum_{S(p_j)}(\frac{d\mathtt{I}_t}{dx})^2 & \sum_{S(p_j)}(\frac{d\mathtt{I}_t}{dx}\frac{d\mathtt{I}_t}{dy}) \\ \sum_{S(p_j)}(\frac{d\mathtt{I}_t}{dx}\frac{d\mathtt{I}_t}{dy}) & \sum_{S(p_j)}(\frac{d\mathtt{I}_t}{dy})^2 \end{bmatrix} \tag{8.5}$$

However, efforts are being wasted when points with high corner responses lie within uninformative regions, and opportunities may be missed when points in informative regions have relatively low corner responses. Thus, we need to adjust the threshold for extracting point features in the informative regions to allow point features to be extracted even if they have lower corner responses. Moreover, when testing the hypotheses, instead of using all the tracked features, we only use point features that are capable of discriminating among the current hypotheses to reduce the computational cost.

An informative feature may not necessarily be a good feature to track if we adjust the feature extraction threshold. The tracking quality of the point will greatly affect the hypotheses testing process, because an ill-tracked point may not agree with the predicted 2D motion of a correct hypothesis. Thus, there is a trade-off between the informativeness of a feature and the quality of feature tracking. We introduce a cost term $C(p_j)$ to penalize the system for using point p_j,

$$C(p_j) = 1 - \frac{V(p_j)}{V_{max}} \tag{8.6}$$

where $V(p_j)$ is the corner response of point p_j, and V_{max} is the value of the maximum corner response from the current image $\mathtt{I}_t$.

Given a set of candidate point features $\mathbf{P}_c$ in the current image $\mathtt{I}_t$, we determine which points are to be added into the tracking set for testing the hypotheses in a later frame [3]. (We will discuss how these candidate point features are extracted later.) Inspired by [12], the most efficient way to test hypotheses is to use a diagnosis method that can well discriminate the hypotheses and has a low cost at the same time. Thus, to select the set of point features for testing, we maximize

$$\sum_{p_j \in \mathbf{P}_c} (I(p_j, \mathbf{M}) - C(p_j))\delta(p_j) \tag{8.7}$$

where $\delta(p_j) \in \{0, 1\}$ equals to 1 if the point is selected to be tracked and 0 if the point is not going to be used. Maximizing Equation 8.7 is equivalent to selecting all the points that have more informativeness than cost $I(p_j, \mathbf{M}) > C(p_j)$. By maximizing Equation 8.7, a more informative point can be selected even with lower corner response, and a point that is less informative needs to have high corner response in order to be selected. For efficiency, in our experiments, we only allow at most 20 points to be added at each frame. If $\sum_{p_j \in \mathbf{P}_c} \delta(p_j) > 20$, we add 20 points with the highest gain $I(p_j, \mathbf{M}) - C(p_j)$.

The set of candidate point features $\mathbf{P}_c$ are extracted in the non-zero informativeness regions with a minimum corner response τ,

$$\tau = \min(V_{max}(1 - \max_{p_j}(I(p_j, \mathbf{M}))), \tau_{min}). \tag{8.8}$$

If the corner response is less than this threshold, it is impossible to be used based on Equation 8.7. We set a hard threshold τ_{min}, to avoid using unreliable points to ensure the quality of hypotheses testing. In our experiments, we set $\tau = 0.0000001$. In addition, a candidate point is not considered if that point is too close (less than 20 pixels) to an

[3] A point feature needs to be tracked for at least one frame in order to be used to test the hypotheses.

existing tracked point in the image space.

Besides the informative features, we also extract corner features with high corner responses as they become available because these features can potentially be informative for testing the hypotheses that are generated in future frames. For the same reason, we keep track of a point feature as long as it is trackable even when it is not informative for the current set of hypotheses. Thus, at each frame, we select the subset of the tracked points $\mathbf{P}_t$ with non-zero informativeness $I(p_j, \mathbf{M}) > 0$ to test the hypotheses. To test the hypotheses, we extract point correspondences between frame t_s and the current frame $t > t_s$. Given a hypothesis, we construct the 3D location of a point feature in the global frame of reference given its tracked location in t_s, and then, project the point onto the current frame t to compare with the observation, the tracked location of the point at frame t (re-projection error). The likelihood of that hypothesis is a function of the re-projection error. The likelihood function is more informative when t_s is larger, so we automatically adjust $t_s \in [5, 20]$ to ensure the number of features exceeds a threshold [4].

8.3 Evaluation

We implement our attention focusing method within our on-line generate-and-test framework described in Chapter 6. We compare results of our framework with and without the attention focusing method. We call the framework without the attention focusing method *baseline*. The baseline framework uses point features (with high corner responses) that are extracted by a fixed threshold. The evaluation is done using the Michigan Indoor Corridor 2012 Video Dataset (See Chapter 4). Our implementation uses the same parameters for the one with the attention focusing method and the one without, except for those that are related to point feature extraction.

We compare the effectiveness of our method with the baseline by computing the informativeness of the selected features at each frame. We define the informativeness $I(\mathbf{P}, \mathbf{M})$ of a set of point features $\mathbf{P}$ relative to a set of hypotheses $\mathbf{M}$ as

$$I(\mathbf{P}, \mathbf{M}) = \log(|\mathbf{M}|) - H(\mathbf{M}^u|\mathbf{P}). \tag{8.9}$$

where $\mathbf{M}^u$ is the set $\mathbf{M}$ of hypotheses at the current frame, but with uniform prior. We then compute the likelihood of each hypothesis based on $\mathbf{P}$, and update the posterior

[4] We only use points that can be tracked for at least five frames to ensure that the point is reliable.

probabilities of the hypotheses. $H(\mathbf{M}^u|\mathbf{P})$ is the entropy of the posterior probability distribution.

We use the set of hypotheses $\mathbf{M}_a$ that exist at each frame when running the attention focusing method. At the meantime, we tracked two sets of point features. The first set $\mathbf{P}_b$ is obtained by tracking features with high corner responses as it is in Chapter 6, and the second set $\mathbf{P}_a$ is obtained by the attention focusing method where features are extracted in the informative regions even when their corner responses are low [5]. Point features in both sets may overlap. At each frame, we compute the informativeness by using each set of features ($I(\mathbf{P}_b, \mathbf{M}_a)$ and $I(\mathbf{P}_a, \mathbf{M}_a)$) to test the current set of hypotheses $\mathbf{M}_a$. We repeat the same comparison using the set of hypotheses $\mathbf{M}_b$ when running the baseline, and compute the informativeness ($I(\mathbf{P}_b, \mathbf{M}_b)$ and $I(\mathbf{P}_a, \mathbf{M}_b)$) at each frame. Notice that since $\mathbf{M}_a$ and $\mathbf{M}_b$ may consist of different hypotheses, the set of informative points $\mathbf{P}_a$ may be different in the two comparisons. However, the set of baseline features $\mathbf{P}_b$ in both runs are the same because these features are extracted independent of the hypotheses. These comparisons are shown in Table 8.1.

In most cases, the informative set $\mathbf{P}_a$ provides more informativeness than the baseline set $\mathbf{P}_b$. This is because more features in the informative set $\mathbf{P}_a$ lie in the region where the current hypotheses give different predictions than those in the baseline set $\mathbf{P}_b$. In some extreme cases, none of the baseline points lie in the informative regions. Figure 8.3 shows examples of these situations. The attention focusing method achieves higher informativeness because more point features that are capable of discriminating the hypotheses are tracked. In general, there are 1.5 to 6.5 times more point features that are capable of discriminating the hypotheses in the informative set $\mathbf{P}_a$ than in $\mathbf{P}_b$. In some extreme cases (last row), the baseline set $\mathbf{P}_b$ does not contain any features to discriminate the hypotheses so the informativeness $I(\mathbf{P}_b, \mathbf{M})$ at those frames are zero. Sometimes the baseline features $\mathbf{P}_b$ provide equal or more informativeness than the informative features $\mathbf{P}_a$. This happens at the first few frames when a large set of children hypotheses are generated. Because the informative set $\mathbf{P}_a$ is so focused on discriminating the parent hypotheses, $\mathbf{P}_a$ may not contain features that can discriminate the children hypotheses at the first few frames when they are generated. However, the attention focusing method adds new informative features that discriminate the children hypotheses at the frame when they are generated, so after tracking these features for some times (at least 5 frames), $\mathbf{P}_a$ provides more discriminative power than $\mathbf{P}_b$. Figure 8.4 is an

[5] Subscript $_a$ represents attention and $_b$ represents baseline.

Dataset	L	+	T 1	T 2	Overall
$\sum I(\mathbf{P}_a, \mathbf{M}_a)$	11.67	5.08	5.5	1.21	23.46
$\sum I(\mathbf{P}_b, \mathbf{M}_a)$	4.34	3.81	3.98	0.03	12.16
$I(\mathbf{P}_a, \mathbf{M}_a) > I(\mathbf{P}_b, \mathbf{M}_a)$	79.92%	63.41%	61.02%	72.99%	71.80%
$I(\mathbf{P}_a, \mathbf{M}_a) < I(\mathbf{P}_b, \mathbf{M}_a)$	10.81%	19.02%	20.34%	0%	10.76%
% Frames $\lvert\mathbf{M}_a\rvert > 0$	75.95%	65.92%	15.09%	51.34%	50.48%
MAP $\mathbf{M}_a$ Accuracy	**94.31%**	93.8%	**92.8%**	**95.47%**	**94.12%**
Weighted $\mathbf{M}_a$ Accuracy	92.79%	**93.08%**	**92.57%**	**94.79%**	**93.35%**
$\sum I(\mathbf{P}_a, \mathbf{M}_b)$	34.04	13.88	6.06	21.36	75.34
$\sum I(\mathbf{P}_b, \mathbf{M}_b)$	11.48	4.45	4.9	0.9	21.73
$I(\mathbf{P}_a, \mathbf{M}_b) > I(\mathbf{P}_b, \mathbf{M}_b)$	70.62%	67.98%	52.16%	52.82%	61.71%
$I(\mathbf{P}_a, \mathbf{M}_b) < I(\mathbf{P}_b, \mathbf{M}_b)$	8.25%	17.98%	7.23%	2.82%	8.28%
% Frames $\lvert\mathbf{M}_b\rvert > 0$	88.86%	73.31%	17.65%	94.89%	68.09%
MAP $\mathbf{M}_b$ Accuracy	94.30%	**94.03%**	91.65%	94.62%	93.68%
Weighted $\mathbf{M}_b$ Accuracy	**93.33%**	92.82%	92.27%	92.38%	92.67%

Table 8.1: Quantitative comparison with and without attention focusing. The top half of the table is the results of running the attention focusing method, and the bottom half is the results of running the baseline. $\sum I(\mathbf{P}, \mathbf{M})$ reports the sum of informativeness over all the frames with more than one hypothesis ($|\mathbf{M}| > 0$). In all videos, the informative set $\mathbf{P}_a$ captures more informativeness than the baseline set $\mathbf{P}_b$. To further analyze the effectiveness of the informative features, $I(\mathbf{P}_1, \mathbf{M}) > I(\mathbf{P}_2, \mathbf{M})$ reports the percentage of frames when feature set $\mathbf{P}_1$ is more informative than $\mathbf{P}_2$ among all the frames that have more than one hypothesis ($|\mathbf{M}| > 0$). These statistics show that the informative set $\mathbf{P}_a$ provides more informativeness than the baseline set $\mathbf{P}_b$ in most of the frames. In a small amount of frames, the baseline set $\mathbf{P}_b$ provides equal or more informativeness than the informative set $\mathbf{P}_a$. These are frames when a large set of children hypotheses are generated. Since $\mathbf{P}_a$ is so focused on discriminating the parent hypotheses, $\mathbf{P}_a$ may not have features to discriminate the children hypotheses for the first few frames when they are generated. We also report the accuracy of the framework with and without the attention focusing method. MAP Accuracy is the average accuracy of the hypothesis with the highest posterior probability at each frame. Weighted Accuracy is the average weighted accuracy of the set of hypotheses at each frame, where the weight is the posterior probability of each hypothesis. By comparing these accuracy statistics, we show that the bias towards informative features reaches comparable or even better accuracy. Thus, by focusing attention on point features that are informative, our framework converges to a single hypotheses more often without loss of accuracy.

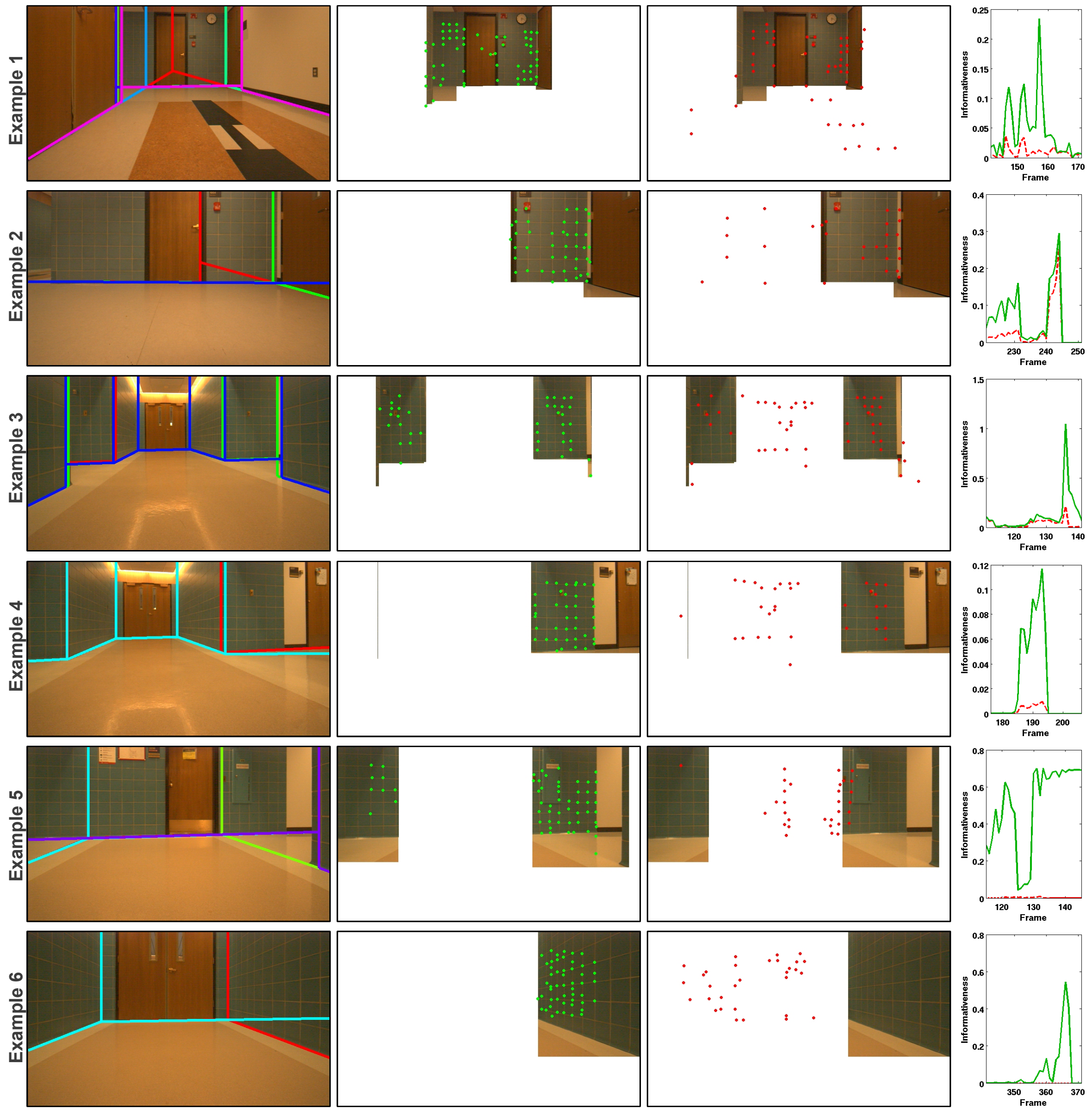


Figure 8.3: Qualitative evaluation of the attention focusing method. (Best viewed in color.) Each row is a snapshot of one of the four datasets. The first column is the set of hypotheses $\mathbf{M}$ at that frame. The second column visualizes the point features (green) that are used to test the hypotheses from the informative set $\mathbf{P}_a$. The third column shows the point features (red) from the baseline set $\mathbf{P}_b$ that are visible at the current frame. For the second and the third column, only the informative regions ($I(b_k, \mathbf{M}) > 0$) are shown, and non-informative regions are shown in white. The last column is the informativeness of using each feature set. The attention focusing method $I(\mathbf{P}_a, \mathbf{M})$ is shown in green solid lines, and the baseline framework $I(\mathbf{P}_b, \mathbf{M})$ is shown in red dashed lines.

example of this situation.

Since the point features that are used in the attention focusing method and the baseline framework are different, the hypotheses that the two methods tested may differ. The total number of hypotheses tested in our method is larger than those in the baseline method. This is due to a threshold on the posterior probability for determining whether a hypothesis is good enough to generate children hypotheses. The attention focusing method tests the hypotheses more efficiently than the baseline and thus, more hypotheses exceeded this threshold and generated children hypotheses. Even though the attention focusing tests more hypotheses, it converges to a single hypotheses more often. As shown in Table 8.1, about 50% of the time, the attention focusing method converges to a single hypothesis while only about 30% of the time, the baseline framework converges to a single hypothesis.

In Table 8.1, we report the accuracy of the attention focusing method and the baseline framework based on this implementation. The accuracy of the attention focusing method reaches similar accuracies as those shown in Chapter 6. This suggests that by focusing attention on regions that are informative, regions where the current hypotheses have different explanations of the point features, we can converge to a single hypothesis more efficiently with no loss of accuracy.

Although we demonstrated the effectiveness of our attention focusing method on point features, the method can also be applied to other types of features that are used in Chapter 7. The informativeness of a vertical-plane feature can be approximated by the average informativeness of the pixels enclosed in the image region of the projected vertical-plane feature. The informativeness of a point-set feature can be computed by the average informativeness of the projected 3D points.

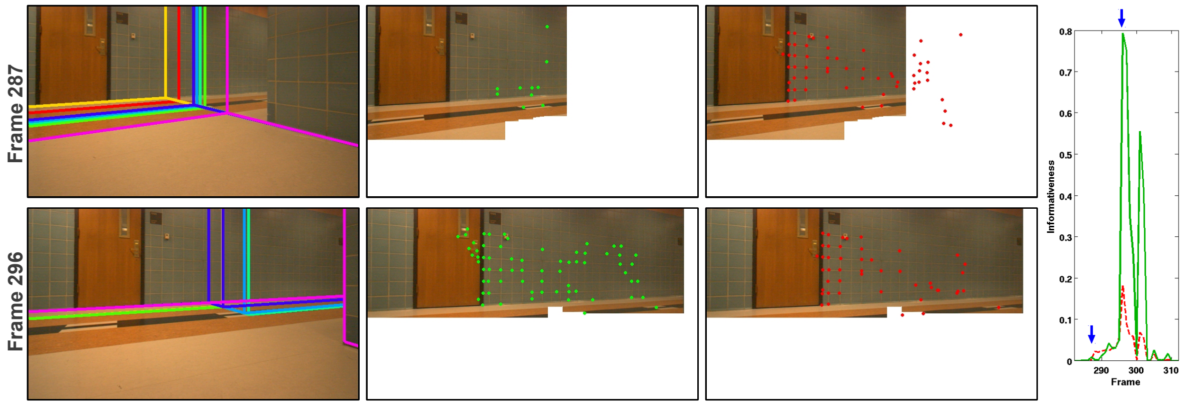


Figure 8.4: Example when baseline framework is more informative. An example where the informative set $\mathbf{P}_a$ provides less informativeness than the baseline set $\mathbf{P}_b$. (Best viewed in color.) Since there is only one hypothesis prior to frame 287, our attention focusing method does not add in new points to the informative set $\mathbf{P}_a$ (green dots) prior to frame 287. At frame 287, a set of children hypotheses are generated so our method adds new informative features into the informative set $\mathbf{P}_a$. Once these points are tracked for some frames (second row), these informative points are used to discriminate the hypotheses. This phenomenon is reflected on the large increasing slope of informativeness $I(\mathbf{P}_a, \mathbf{M})$ (green solid line) between the two blue arrows. At the meantime, the baseline set $\mathbf{P}_b$ (red dots) continues to add in points with high corner responses so at frame 287 the baseline set happens to have more points that are capable of discriminating the hypotheses.

8.4 Summary

In this chapter, we demonstrate that by focusing attention on visual features that are informative, we can test a set of hypothesized models of the environment more efficiently. A feature is informative if it is capable of discriminating among the hypotheses. Specifically, we define *informativeness* of a point feature mathematically and propose a method to identify informative features. We show that by using informativeness to control the process of feature acquisition, we can use computational resources more efficiently to discriminate among the hypothesized models of a visual scene, with no loss of accuracy. Informativeness allows our method to focus computational resources on regions in the scene where different hypotheses make different predictions.

We implement the attention focusing method on our on-line generate-and-test framework (Chapter 6), and compare the results with and without the attention focusing method. Our experimental results demonstrate that this bias of search towards informative features allows the framework to converge to a single correct hypothesis more often, with no loss of accuracy.

The attention focusing method presented in this chapter is an important step towards active vision. While this chapter selects a set of informative features from the current view, in active vision, the goal is to compute a robot motion to maximize the informativeness of the upcoming frames. For example, it is more informative to move towards a region where two hypotheses have different explanations than to move along the region where all hypotheses have similar explanations. (More discussions on future work is presented in Chapter 10.)

Chapter 9

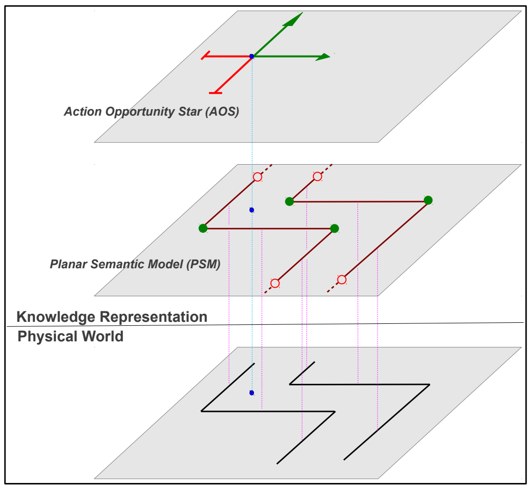
Application — From Structure to Opportunities for Action

In addition to modeling the geometric structure of the indoor environment, this chapter takes one step forward to reason about the opportunities of robot action [1]. In this chapter, we present a two-layer representation of the local indoor environment. Each layer represents a different level of understanding of the environment. The first layer models the geometric structure of the local environment by the Planar Semantic Model (Chapter 6). The second layer is a pure symbolic representation that describes the opportunities for robot action, based on the PSM determined in the first layer. Figure 9.1 illustrates the two-layer representation.

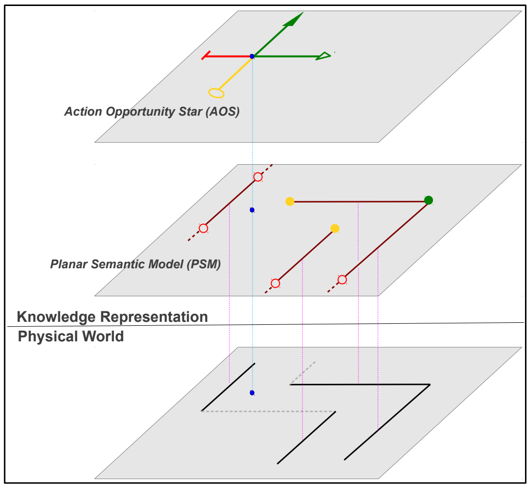
To represent the opportunities for robot action, we present the Action Opportunity Star (AOS) to describe a set of qualitatively distinctive opportunities for action at a given location (Section 9.1). An *opportunity* represents a group of trajectories that can be described by the same semantic meaning of the robot's action. AOS captures where each opportunity is valid in the PSM and the relationships among these opportunities. Since AOS is an abstracted representation, if the PSM of the robot's surroundings in two locations are similar, AOSs extracted in both locations will be the same. Finally, we demonstrate an algorithm that extracts AOS from PSM at a given location.

As discussed in Chapter 1, a semantic meaningful representation needs to reflect the robot's action opportunities. The proposed representation for action opportunities, AOS, is designed for a robot that navigates on the ground plane. A robot with different

[1]Materials presented in this chapter were published in [70].



(a) Two-layer representation with complete knowledge



(b) Two-layer representation with incomplete knowledge

Figure 9.1: Two-layer representation. (Best viewed in color.) The representation is illustrated on the ground-plane map. Each layer represents a different level of understanding of the local environment. The first layer models the geometric structure of the local environment. The second layer is a pure symbolic representation that describes the opportunities for robot action at a given location, based on the geometric structure determined in the first layer. Both layers are capable of representing incomplete knowledge as shown in (b). In the physical world, black solid lines represent the part of the environment that is observed by the robot, and the gray dashed lines represent the part of the environment that is not observed by the robot. The first layer is the Planar Semantic Model (PSM), which models the geometric structure of the local environment in terms of meaningful planes — the ground plane and a set of walls that are perpendicular to the ground plane but not necessarily to each other. (PSM was introduced in Chapter 6.) Each wall contains a set of disjoint wall segments (red lines), delimiting where the wall is present and where is an opening. Each wall segment is represented by two endpoints, and each endpoint has its properties indicating the level of understanding of the bound of the wall segment. While a *dihedral* endpoint (green dot) provides the full knowledge of the bound of its corresponding wall segment, an *occluding* endpoint (yellow dot) and an *indefinite* endpoint (red hollow dot) provide different types of partial knowledge of the wall intersection. The second layer is the Action Opportunity Star (AOS), describing the robot's surrounding environment by a structured set of qualitatively distinct opportunities for robot action. Each opportunity is visualized by an arrow pointing towards its associated direction, and the tip of the arrow reflects its type. The opportunity type represents different purposes or different levels of understanding of the opportunity. A green arrow means that the opportunity is *observed* and navigable, while a red line means that the opportunity is *unnavigable*. A green hollow arrow means that the opportunity is navigable but the actual boundary of the opportunity is only *partially observed*.

motion capabilities or with different goals may require a different representation for its action. For example, a robot arm may require a representation that captures the graspable regions in the environment rather than the paths for traveling.

Both layers of the representation, PSM and AOS, are capable of representing incomplete knowledge of the local environment so that the robot can plan to explore unknown regions to incrementally fill in missing information. Our representation is useful to the robot in making plans at different levels. While PSM allows the robot to precisely generate a trajectory to get from one pose to another, AOS allows the robot to make plans at a higher level, such as turning right at an intersection or going forward (rather than reverse) in a corridor. Moreover, our representation is useful for building a topological map [8]. For example, from the AOS, we can easily detect whether a robot is at a topological place, such as a hallway intersection, or on a path that links two places. Finally, our representation is not limited to monocular vision sensors. The AOS representation is available for any sensors that provide sufficient information to extract the ground plane and the walls to build the PSM. AOS extraction depends solely on the PSM and is independent of the input sensors.

9.1 Action Opportunity Star

An Action Opportunity Star (AOS) is a qualitative description of the small finite set of opportunities for robot action abstracted from an infinite number of trajectories that are available within the region around the robot (the field of interest). An *opportunity* is an abstraction, representing a group of trajectories that have the same qualitative effect on the robot's state. An opportunity for action is intended to be similar to the concept of an *affordance* [22]. We define a *gateway* as a line segment on the metric map, PSM, that specifies which trajectories belong to an opportunity. All trajectories that cross a particular gateway from the side closer to the robot to the side farther from the robot belong to the same opportunity.

In addition to representing individual opportunities, the AOS models the relationships among opportunities in terms of the *paths* they are on. Two opportunities that unambiguously represent opposite directions from the same field of interest are considered to be on the same path. We say that when the robot has exactly two opportunities unambiguously representing opposite directions, then the robot is *on a path.* In any other situation, the robot is *at a place*, which typically involves the need to make a decision, selecting among the available opportunities [8]. For example, when the robot is at a T-intersection, it is surrounded by three opportunities, associated with two paths, one of which passes through the place, while the other ends at that place.

Formally, at a given robot location, the AOS, ς is defined by a list of opportunities A_i circularly ordered in the counter-clockwise direction,

$$\varsigma = \{A_1, A_2, ..., A_k\} \tag{9.1}$$

where k is the number of opportunities around the given location. Each opportunity, A_i, is defined as,

$$A_i = \langle \pi_i, \rho_i, \tau_i, \mathbf{g}_i \rangle \tag{9.2}$$

where $\pi_i \in \{0, 1, ..., N_p\}$ is the path that the opportunity is on, among the N_p paths that pass through the field of interest. $\rho_i \in \{+, -\}$ is the direction along the path that the opportunity is leading onto. The path π_i and the direction ρ_i specify the relation between opportunity A_i and another opportunity A_j where $\pi_i = \pi_j$ and $\rho_i = -\rho_j$. $\mathbf{g}_i$ is the gateway associated to the opportunity A_i, which is a line segment ϕ_i parameterized by two ends $(\mathbf{p}_i^1, \mathbf{p}_i^2)$ in the ground-plane map, and the qualitative traveling direction ψ_i of the opportunity is the normal direction of the gateway pointing away from the robot. Finally, τ_i specifies the type of the opportunity.

There are six different types of opportunity: *observed*, *partially observed*, *unnavigable*, *potential*, *beginning* and *exiting* representing different purposes or different levels of understanding of the opportunity. An *observed* opportunity is navigable and leads the robot into or out of a path intersection where more than two unaligned opportunities are presented. Both ends of an *observed* gateway, $\mathbf{p}_i^1$ and $\mathbf{p}_i^2$, are fully determined. A *partially observed* opportunity plays the same role as an *observed* opportunity, except only one of the two gateway-ends is determined, which leaves the actual width of the gateway undetermined. An *unnavigable* opportunity prohibits the robot to travel along the path due to obstacles. A *potential* opportunity exists when the region that the opportunity is leading to has not yet been observed by the robot, and thus, there is a potential opportunity for traveling along that direction. Similar to a *potential* opportunity, a *beginning* opportunity crosses the boundary between observed and unobserved regions, except a *beginning* opportunity leads the robot into the observed region. A *beginning* opportunity only occurs at the beginning of an episode (the first few frame of an image stream), where the robot only observes the environment in front of it, instead

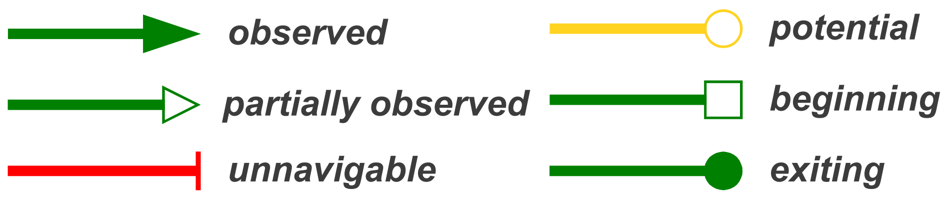
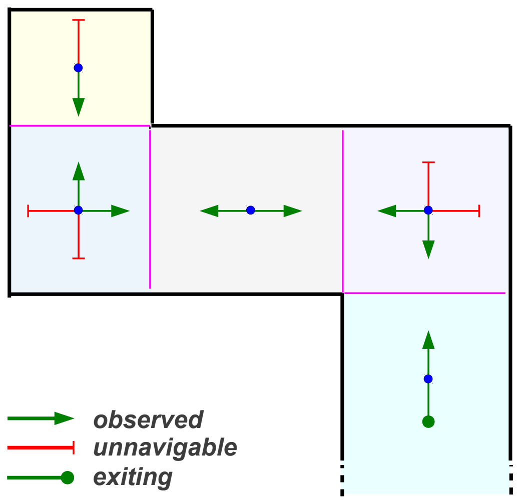


Figure 9.2: Visualization of different opportunity types. (Best viewed in color.) A filled arrow means the opportunity type captures the full knowledge of that opportunity, while a hollow shaped arrow means the opportunity contains incomplete knowledge. Navigable opportunities are marked as green, while opportunities that are marked as red are *unnavigable*. Yellow arrows represent a potentially navigable direction, and by acquiring more observations around the opportunity, it can become an *observed* or an *unnavigable* opportunity. From the color and the shape code, an *exiting* opportunity is navigable (green) and the knowledge of that opportunity is complete. A *partially observed* opportunity is also navigable (green) but it contains incomplete knowledge (the actual width of its gateway has not yet been determine). A *beginning* opportunity occurs at the beginning of each video sequence because the robot only observes the environments that are in front of it. Thus, a *beginning* opportunity is navigable but contains incomplete knowledge.

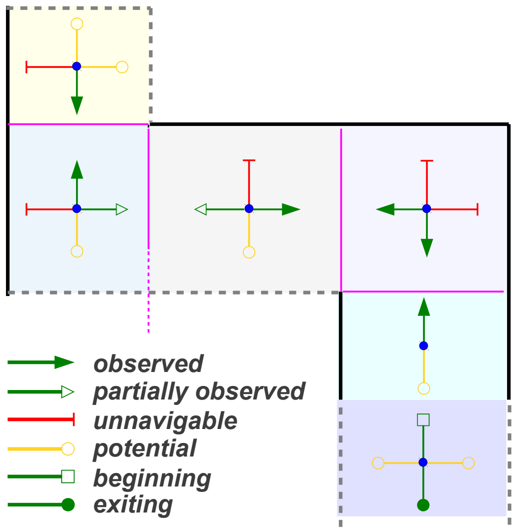
of its surrounding environment. The ends of the associated gateway of a *potential* or a *beginning* opportunity are specified so that the right side of the vector $\overrightarrow{\mathbf{p}_i^1\mathbf{p}_i^2}$ is the observed region while the left side of the vector $\overrightarrow{\mathbf{p}_i^1\mathbf{p}_i^2}$ is unobserved. While the above five opportunity types reflect the structure of the local environment, an *exiting* opportunity leads the robot out of the robot's field of interest. This type of opportunity usually appears when a robot is traveling on a long corridor where going forward and turning backward are the only two possible qualitative actions. Visualization of different types of opportunities are shown in Figure 9.2. Examples of the AOS in different situations are shown in Figure 9.3.

9.1.1 Extracting Opportunities from PSM

Since a gateway links its associated opportunity to the geometric structure of the environment, we start by extracting a set of gateways within the field of interest, given a robot location. In this section, we determine only the gateway $\mathbf{g} = (\mathbf{p}^1, \mathbf{p}^2)$ and the type τ of each opportunity. In Section 9.1.2, we determine the other two elements, π and ρ, of the opportunities by comparing the gateways to see which opportunities are well-aligned to be on the same path.



(a) w/ complete knowledge



(b) w/ incomplete knowledge

Figure 9.3: Examples of the AOS in different locations. (Best viewed in color.) AOS is an abstract representation of the local surrounding in terms of the action opportunities for a navigating robot at a given robot location. AOS captures where each opportunity is valid in the metric map and the relationships among the set of opportunities Since AOS is an abstract representation, AOSs extracted at all location within a region that has the same surrounding geometric structure are the same. The regions are shown in different colors. In (b), due to the limited field of view of a monocular camera, it is quite often that the robot has incomplete observations of the local environment. Black solid lines represent the part of the environment that is observed by the robot, and the gray dashed lines represent the part of the environment that is not observed by the robot. Each opportunity is illustrated by an arrow and the tip of the arrow reflects the type of the opportunity (see Figure 9.2).

Given the PSM, there are two major steps to collect the set of gateways within the field of interest. The first step extracts gateways that reflect the structure of the surrounding environment. All of these gateways have at least one end anchored at a PSM endpoint. The type of the PSM endpoint at which a gateway is anchored affects the type of its associated opportunity. Possible opportunity types at this stage are *observed*, *partially observed*, *potential* and *beginning*. The rest of this section describes how we extract these gateways in detail. Given the gateways from the first step, the second step extracts *exiting* opportunities from regions that are not explained by either a PSM wall segment or an existing gateway, through a circular scan around the field of interest. The gateway of an *exiting* opportunity is perpendicular to, and intersects with, a PSM wall segment or another gateway.

By using each endpoint in the PSM as an anchor, two gateways can be extracted with their directions $\hat{\mathbf{g}}$ parallel to the associated PSM wall of the endpoint, and another two gateways can be extracted with their directions perpendicular to the wall segment. A gateway is valid only if it lies along the free space of the PSM. For a gateway anchored at an *occluding* or an *indefinite* endpoint, it must lie on the free space side of the associated wall of the PSM endpoint. If a gateway anchors at a *dihedral* endpoint, it must lie on the free space side of both walls associated to the PSM endpoint. Furthermore, gateways that are not within the field of interest are not considered.

Given $\mathbf{p}^1$ and the direction $\hat{\mathbf{g}}$ of the gateway, we find the other gateway-end $\mathbf{p}^2$. If the gateway anchors at an *occluding* endpoint that connects a wall segment and a wall opening along the same wall, $\mathbf{p}^2$ is the other PSM endpoint that associates to the opening. Otherwise, $\mathbf{p}^2$ is the closest intersection point of a ray pointing from $\mathbf{p}^1$ in $\hat{\mathbf{g}}$ direction and a wall segment in the PSM. In the case where no wall segment intersects with the ray, $\mathbf{p}^2$ is left undetermined and thus, the gateway width is also undetermined. We exclude a gateway if $\mathbf{p}^2$ is determined but the gateway is too narrow for the robot to pass through. Finally, a gateway is removed, if its direction and its gateway-ends are too similar to another gateway.

From each remaining gateway, we form an opportunity and determine its type τ by: 1) the type of the PSM endpoint at which the gateway is anchored; 2) whether $\mathbf{p}^2$ is determined; and 3) the robot's location. From a gateway that anchors at a *dihedral* or an *occluded* endpoint, an *observed* opportunity is formed if $\mathbf{p}^2$ is determined, and a *partially observed* opportunity is formed if $\mathbf{p}^2$ is undetermined. A gateway that anchors at an *indefinite* endpoint is a boundary line between an observed and an unobserved region in the PSM, and thus forms a *potential* or a *beginning* opportunity. We arrange the order of the two gateway-ends $(\mathbf{p}^1, \mathbf{p}^2)$ so that the observed region is on the right side of the vector $\overrightarrow{\mathbf{p}_i^1\mathbf{p}_i^2}$ and the unobserved region is on the left side. A *potential* opportunity is formed if the robot is located on the right side, the observed side, of the $\overrightarrow{\mathbf{p}_i^1\mathbf{p}_i^2}$, and a *beginning* opportunity is formed otherwise.

9.1.2 Extracting AOS from Opportunities

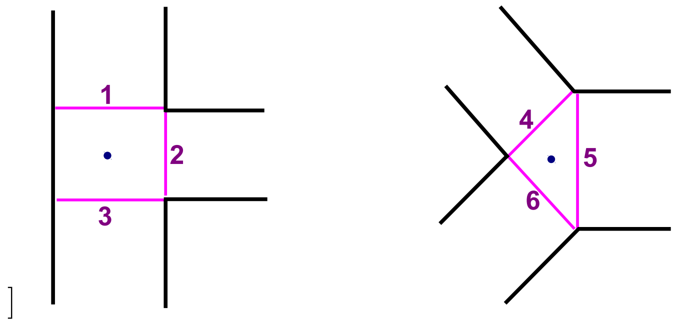
Given a set of opportunities with each provided with only the gateway $\mathbf{g}$ and the opportunity type τ, this section determines the other two elements $\langle \pi, \rho \rangle$ of each opportunity and the ordering among the opportunities to construct the complete AOS. Since $\langle \pi, \rho \rangle$ of the set of opportunities capture the relationships among them, the complete AOS is extracted by pairing up opportunities if their gateways are well-aligned to form a path. In other words, AOS is extracted by determining the number of paths N_p passing through the field of interest.

First, we define a *bounding box* to represent the smallest bounding box enclosing all gateways. Second, for each pair of opportunities, their gateways $(\mathbf{g}_i, \mathbf{g}_j)$ are compared using the similarity measurement,

$$sim(\mathbf{g}_i, \mathbf{g}_j) = -\cos(\psi_i - \psi_j) * max(0, \frac{l_{\mathbf{g}_i,\mathbf{g}_j}}{l_{\mathbf{g}_i}}) \tag{9.3}$$

where ψ_i is the normal direction of gateway $\mathbf{g}_i$ pointing away from the robot. $l_{\mathbf{g}_i}$ is the length of the *bounding box* edge that intersects by a line in the opposite direction of ψ_i, and $l_{\mathbf{g}_i,\mathbf{g}_j}$ is the shortest distance from the gateway line ϕ_i to the center of gateway $\mathbf{g}_j$. Note that this quantity is not symmetric, $sim(\mathbf{g}_i, \mathbf{g}_j) \neq sim(\mathbf{g}_j, \mathbf{g}_i)$. The similarity measurement is designed to account for two factors. The first metric considers how similar the gateway directions are. Orthogonal gateways are not on the same path, while gateways with ψ_g pointing in opposite directions may be on the same path. The second metric considers the amount of overlap between the gateways relative to the size of the *bounding box* enclosing all gateways. More overlap between the two gateways is better. If there is no overlap at all, then the gateways will not be on the same path.

Starting from an empty set of paths Π that pass through the field of interest, we carry out an exhaustive search among the opportunities to find unambiguous matches using the similarity measurement. If a single, unambiguous match is found between two



(a) aligned opportunities

(b) unaligned opportunities

Figure 9.4: Matching opportunities to form AOS. (Best viewed in color.) Given the robot location (blue dot), three *observed* opportunities are extracted and their gateways (pink lines) are determined from the PSM. If a single, unambiguous match is found between two gateways, then their associated opportunities are considered to be on the same path. (a) Gateway 1 and Gateway 3 are an unambiguous match, and thus their associated opportunities are on the same path. Gateway 2 does not aligned to any other gateways, thus, it is on its own path. (b) In a Y-intersection as shown, Gateway 4 is aligned with Gateway 5 but Gateway 5 is aligned to both Gateway 4 and Gateway 6. Thus, there are no unambiguous matches among the three gateways. The opportunities associated with all three gateways are on separate paths.

opportunities, they are considered to be on the same path, and thus the path is added to the set Π. If an opportunity belongs to no paths or to more than one path in the existing path set Π, a separate path is created for the opportunity. Figure 9.4 are examples for aligned and unaligned gateways. After the search is done, if a path in Π is associated to only one opportunity A_i, an *unnavigable* opportunity A_j is generated with $\pi_j = \pi_i$ and $\rho_j = -\rho_i$ to describe the opposite side of the path. Once all the opportunities are fully determined, the complete AOS is formed by ordering the opportunities so that the normal directions of their gateways are sorted in the counter-clockwise direction.

9.2 Evaluation

We implement the Action Opportunity Start (AOS) on top of the on-line generate-and-test framework presented in Chapter 6 using The Michigan Indoor Corridor 2012 Video Dataset. (See Chapter 4 for more description on the dataset.) For each frame t, we select the maximum *a posteriori* PSM hypothesis at the current frame and extract the AOS from the PSM at the current robot location. For each example (Figure 9.5,9.6,9.7,9.8), the first column is the image projection of the ground-wall boundaries of the PSM onto the indicated frame of the video. The second column visualizes the PSM in the ground-plane map with the robot pose plotted in blue. In the PSM, a green dot represents a *dihedral* endpoint, and a yellow dot represents an *occluding* endpoint. A red hollow dot represents an *indefinite* endpoint. Each wall in the PSM has an index automatically assigned by the implemented system, and all the wall segments contained in that wall are marked by the same index. The third column is the AOS at the current robot location. Each opportunity A_i in the AOS is shown directed along its associated path with an arrow reflecting its type (Figure 9.2), and a label for its path index and its direction along the path $\langle \pi_i, \rho_i \rangle$.

Figure 9.5 demonstrates the AOSs extracted in various locations within the PSM constructed from the on-line generate-and-test framework. At Frame 20, a simple PSM is constructed to describe the current view. Due to the field of view of the monocular camera, no information of the PSM around the robot's immediate surrounding is available when the process begins. Only the environment in front of the robot is observed. Thus, the *beginning* opportunity leads the robot into the region that has been modeled by the PSM. At Frame 141, as more observations become available, PSM with more details (the first L-intersection) of the environment are incrementally built. Although there is still incomplete knowledge in the PSM in the distance, the robot is now in a long corridor with full knowledge of its current surrounding. Thus, in the AOS, the *observed* opportunity leads the robot towards the L-intersection, while the *exiting* opportunity leads the robot out of its field of interest. At Frame 170, the robot is at the first L-intersection and has full knowledge of the opportunities available at the intersection. At Frame 200, our method continues to capture the geometric structure of the environment with the PSM representation. At the current frame, the robot has incomplete knowledge of its surrounding. The wall on the left side of the robot is unobserved. Thus, the *potential* opportunity leads the robot towards the unobserved region. At Frame 235, the robot is in the second L-intersection but unlike the first one, the robot has incomplete knowledge of the intersection. Finally at the last frame, the final PSM of the local environment is constructed. Notice that, at this point, the system cannot yet conclude that the endpoint of Wall 4 is an *dihedral* endpoint that connects to Wall 1.

Figure 9.6 demonstrates that PSM is a step forward from a pure planar model because it represents a richer set of relationships among planar segments. At Frame 147, PSM models the +-intersection by three walls (Wall 0, 2, and 3). Each wall contains two

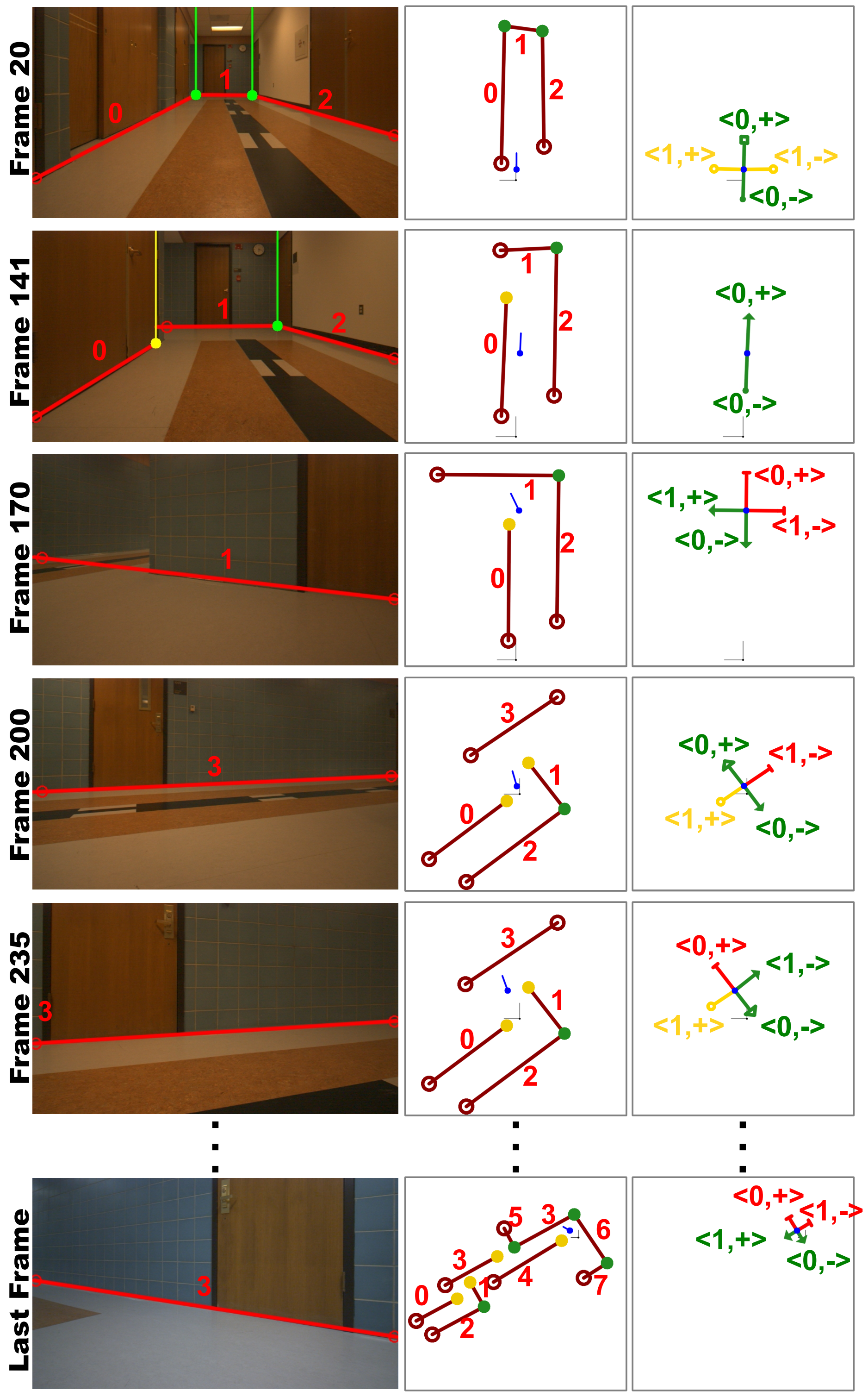


Figure 9.5: Examples of the two-layer representation on a long video. (Best viewed in color.) These examples are results from Dataset L, where there are three L-intersections. The robot traveled through two long corridors connected by two adjacent L-intersections and finally made a U-turn at the last L-intersection. (See text for more details.)

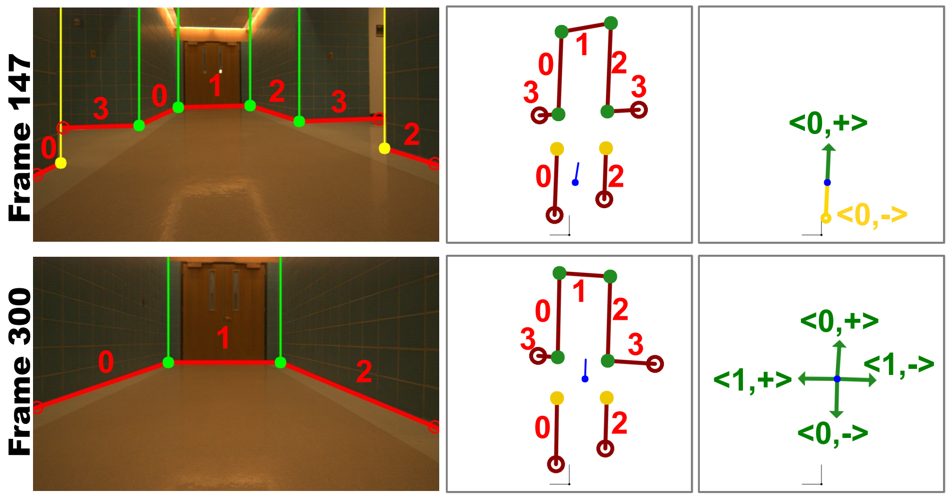


Figure 9.6: Examples of the two-layer representation in a cross intersection. (Best viewed in color.) This dataset has one +-intersection along a long corridor, and the robot traveled from one end of the corridor to the intersection without making any turns. (See text for more details.)

disjoint wall segments that share the same plane equation in 3D. At the current frame, the robot is at the long corridor as illustrated by the AOS. At Frame 300, the robot is at the +-intersection, and has full knowledge of the intersection. Notice that it is impossible for the robot to realize that it is at a +-intersection solely from the current image due to its limited field of view. Thus, a temporally contiguous stream of images is essential for coherent visual scene understanding.

Figure 9.7 compares results from two sequences acquired around the same intersection with different trajectories. At (T1)Frame 200, PSM models the T-intersection by three walls (Wall 0, 2 and 3). Wall 2 contains two disjoint wall segments, and the gap between the wall segments is the opening of the T-intersection. The AOS captures the opportunities for actions at the T-intersection with full knowledge. At (T1)Frame 410, PSM continues to model the dead-end at the minor corridor. Due to lack of observations, the video sequence contains no clue for Wall 5 to intersect with Wall 2. Thus, in the AOS, the *potential* opportunity captures the incomplete knowledge of the missing information between Wall 5 and Wall 2. At (T2)Frame 110, the robot is at approximately the same location as the robot in (T1)Frame 410 but in the opposite direction. Since the observations of the two sequences of the same environment are different, PSMs from the two sequences captures different forms of partial knowledges of the environment. Consequently, the AOSs extracted from the two sequences captures partial knowledge of

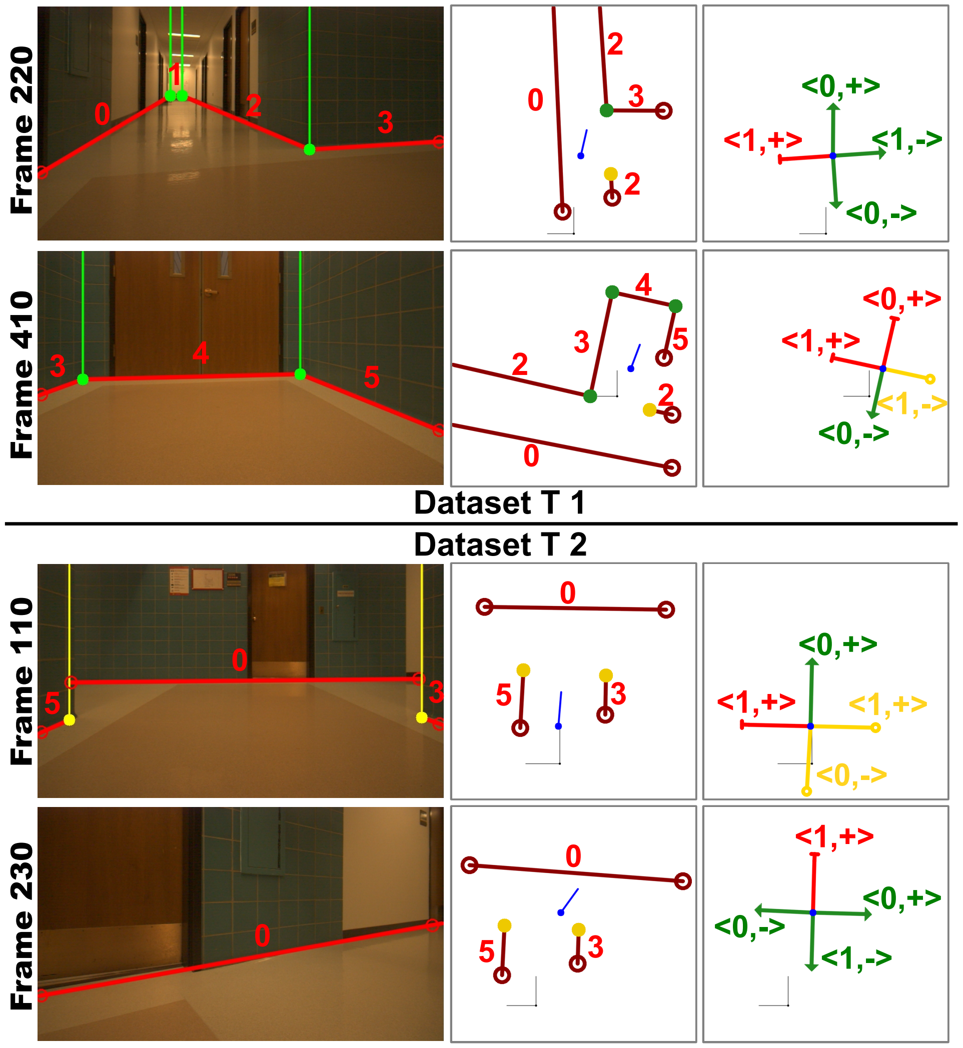
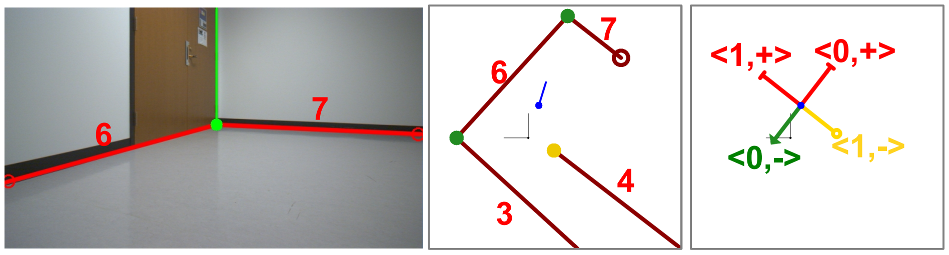
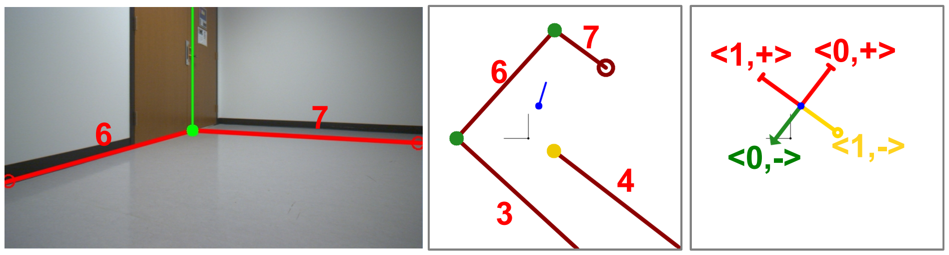


Figure 9.7: Examples of the two-layer representation at T intersections. (Best viewed in color.) These examples are from Dataset T 1 and Dataset T 2. In the two video sequences, the robot traveled around the same T-intersection in the physical world. In Dataset T 1, the robot traveled from the major corridor and made a right turn onto the minor corridor at the intersection, whereas, in Dataset T 2, the robot traveled from the minor corridor to the major corridor. We process each sequence independently and compare the results from the two. To clarify the comparison, we aligned the wall indices so that the same wall in the physical world has the same index. (See text for more discussions.)



(a) Qualitative best hypothesis



(b) Maximum posterior hypothesis

Figure 9.8: Example of incorrect PSM but correct AOS. (Best viewed in color.) Due to lack of feature points, our method may fail to identify the correct PSM hypothesis. (The maximum *a posteriori* hypothesis is not a correct PSM hypothesis.) In this case, the actual location of Wall 7 is not correctly identified. However, if the incorrect PSM has the correct structure layout, the extracted AOS will still be the same as the correct PSM hypothesis.

different part of the robot's surrounding. The two AOSs contains no conflicting opportunities. Note that since the robot was facing in opposite direction in the two sequences, one of the AOSs needs to be rotated at about 180° in order to match the other one. Thus, by acquiring more observations around a *potential* opportunity, it can become an *observed* or an *unnavigable* opportunity. At (T2)Frame 230, the robot is at the T intersection and has full knowledge of the intersection. Since the robot is at the same T-intersection as the (T1)Frame 200, the AOSs in both situations are the same. In fact, any location within the T-intersection will have the same AOS. Moreover, if the structure and the knowledge of the robot's surrounding of two locations are similar, AOSs extracted in both locations will be the same.

The maximum *a posteriori* PSM hypothesis is correct [2] 92% of the time. The main reason that our method fails to select the correct hypothesis is lack of feature points.

[2]We consider a PSM hypothesis correct if the geometric structure within 4 meters of the vision cone is correctly modeled. Thus, for a given frame, it is possible to have more than one correct PSM hypothesis, if the differences are further than 4 meters away.

However, among the frames with an incorrect PSM, our method extracted the correct AOS 73% of the time. This happens because the incorrect PSM hypothesis has the same structure layout of the correct one, except the actual locations of the walls are off. Figure 9.8 is an example of this situation.

9.3 Summary

In this chapter, we move one step forward from modeling the geometric structure of the indoor environment to reason about opportunities for navigation. We propose the Action Opportunity Star (AOS) to describe a set of qualitatively distinctive opportunities for robot action at a given location. An *opportunity* represents a group of trajectories that can be described by the same semantic meaning of the robot's action. AOS captures where each opportunity is valid, the property of each opportunity, and the relationships among these opportunities. We presented a method to extracts AOS from PSM.

Since AOS is an abstract representation, if the surrounding PSM at two locations are similar, AOSs extracted at both locations will be the same. Similarly, if two PSM hypotheses have the same structure but differs in the exact wall parameters, AOS extracted from the two PSM hypotheses will be the same. Therefore, AOS allows the robot to make plans even when there are ambiguities in modeling PSM.

AOS opens up several interesting future research directions that can be build on top of this thesis. First, the action opportunity star supports topological mapping [8] because the robot can detect whether it is at a topological place which requires making a decision (e.g. hallway intersection) or not by checking the number of paths in the AOS. Second, AOS also supports active sensing where a robot actively explores the unobserved parts of its local environment. Our representations, AOS and PSM, allows the robot to make plans in different levels. The robot can select an opportunity from the AOS based on its active exploration strategy. From the selected opportunity, the robot can identify a target pose that leads the robot towards that opportunity. A motion planning algorithm [54] can then be applied to find the optimal trajectory that gets the robot from its current pose to the target pose within the free-space of PSM. (More discussions on future work is presented in Chapter 10.)

Chapter 10

Conclusion and Future Work

In this chapter, we summarize our contribution to solving the problem of scene understanding for indoor navigating robots (Section 10.1). We then suggest extensions to our proposed method and future research directions for indoor navigating robot to build on top of this thesis (Section 10.2).

10.1 Summary

An indoor navigating robot with vision sensor needs to understand the geometric structure of its local environment in order to navigate. The input to the visual perception is a temporally contiguous stream of images, and the output of visual scene understanding must be a representation of the local environment that is useful for the robot to make plans. Moreover, visual scene understanding must be an on-line and efficient process so that the robot's representation can be incrementally updated as visual data is acquired.

In this thesis, we propose the Planar Semantic Model (PSM) to represent the locally-sensed indoor environment. PSM is a concise planar representation that describes the environment in terms of meaningful planes — ground plane and walls. PSM is a step forward from existing floor-wall models because it captures richer relationships among the wall segments. PSM is capable of capturing incomplete knowledge so that missing information can be incrementally assimilated when observations become available. In addition, we move one step forward from modeling the geometric structure of the indoor environment to reason about opportunities for robot actions. We propose the Action Opportunity Star (AOS) to describe a set of qualitatively distinctive opportunities for action at a given location. We demonstrate a method to extract AOS from PSM.

We demonstrate an on-line generate-and-test framework to efficiently construct PSM of the local environment as the robot travels within it. Our framework includes two key elements: 1) incremental hypothesis generation, and 2) on-line hypothesis testing. Hypothesized PSM models are incrementally generated by transforming the current set of hypotheses into a set of new hypotheses that describe the same environment with more details. These hypothesized models are tested through a recursive Bayesian filter based on their abilities to explain the 2D motion of a set of tracked features in the image when using a monocular camera, or their abilities to explain a set of 3D features in the 3D coordinate when using a depth camera. In order to interpret cluttered environments, we introduced a hypothesis testing mechanism that addresses a three-way trade-off among the coverage of the hypothesized model, the degree of accuracy with which the model explains the features, and the simplicity of the model. This testing mechanism not only allows us to converge to the correct PSM in cluttered environment, but also segment out clutter regions that cannot be represented by that PSM. In addition, we propose an attention focusing method to select informative features (observations) to discriminate more efficiently among the current active set of hypotheses.

We evaluate our on-line generate-and-test framework in three phases. First, we evaluate the effectiveness of the on-line testing mechanism to select the best model from a set of hypotheses about a simple empty three-wall environment generated at the first frame of the video. We compare our results with existing single-image layout estimation methods and demonstrate that by using temporal information, we can construct the geometric structure of the local environment more accurately. We also show that using a planar structure is more concise and more meaningful than a 3D point cloud, which is what most existing on-line methods produce. Second, we evaluate the whole on-line generate-and-test framework in an empty environment with more complex structure (hallway intersections) and demonstrate the expressive power of the PSM representation. We also compare our framework with and without the attention focusing method and demonstrate that with this bias search towards informative features, our framework converges to a single hypothesis more often, with no loss of accuracy. Finally, we evaluate our framework in cluttered environments, where portions of these environments are not representable by the PSM and demonstrate the effectiveness of the three-way trade off among converge, accuracy and simplicity. A potential weakness of our evaluation is that we only tested on the datasets that we collected with the same robotic platform in the same campus. A more extensive evaluation on our framework using different robotic platforms in a larger variety of environments is part of our future work.

To summarize, this thesis contributes to solving the problem of visual scene understanding for an indoor navigating robot from both the representation aspect and the algorithm aspect. In terms of representation, we propose the Planar Semantic Model to concisely represent the geometric structure of the indoor environment and propose the Action Opportunity Star to describe qualitative actions available at a given location. In terms of algorithm, while existing plane-based indoor scene understanding algorithms use either a single-image approach or a batch multiple-image approach, we propose an on-line generate-and-test framework to incrementally and efficiently construct a planar structure of the indoor environment.

10.2 Future Work

In this section, we propose a few ways to extend or improve our current on-line generate-and-test framework (Section 10.2.1). We also identify research directions related to indoor navigating robots that could be built on top of this thesis (Section 10.2.2).

10.2.1 Extension of Proposed Method

Combine Pose Estimation and Scene Understanding

This thesis assumes that the robot pose between consecutive frames is either computed by the visual data or provided by additional source (e.g. laser). In other words, robot poses are treated as known values in the on-line generate-and-test framework for scene understanding. However, robot pose estimation may be noisy, and the overall robot trajectory may drift over time. One common way to improve pose estimation is to use the geometric structure to provide constraints for refining the estimated poses in the global frame of reference. Thus, one extension to our framework is to add the uncertainty of pose estimation into our scene understanding process.

There are several potential methods to incorporate pose estimation into our framework. The simplest method is to maintain an uncertainty measurement for the estimated pose, and then refine the pose according to the hypothesized model with the maximum posterior probability. A threshold on the posterior distribution is needed to make sure the maximum *a posteriori* hypothesis outperforms the rest of the hypotheses substantially to avoid misleading the pose estimation due to the noise in hypothesis testing. It is possible to have multiple compelling hypotheses that vary slightly. In this case, the portion that these hypotheses agree on should be used to refine the pose.

Another method to incorporate the uncertainty of pose estimation is for each hypothesis to not only maintain the precision of the planar model but also maintain an estimate of the robot pose. In other words, each hypothesis is an interpretation of the 3D environment and the robot pose within that environment. Each hypothesis is an independent SLAM process where the robot pose and the planar model are jointly updated in each frame. The role of the Bayesian filter is to test which combination of pose and model best explains the observed features. A potential issue of this method is that a combination of bad pose and bad model may explain all observations very well, and thus, a likelihood function may need to be designed to overcome this issue.

Handle Dynamic Obstacles

The method proposed in this thesis assumes the world is static. However, it is quite common to have dynamic objects (e.g. pedestrians) moving relative to the static environment. Features from dynamic objects do not provide the same constraints as those from static environments, and thus, may mislead the Bayesian filter to converge to an incorrect hypothesis. Thus, one extension to our framework is to handle dynamic objects.

One simple way is to detect features from potential dynamic objects and remove these features from the hypothesis testing step. If the robot pose is provided by external sources, features that do not fulfill the rigid-body constraints given by the robot pose are dynamic. If the robot pose is estimated from the visual data, dynamic features may also be detected as part of the pose estimation process. Pose estimation is usually obtained by fitting a 3D rigid-body transformation. One could apply an outlier rejection method, like RANSAC, to identify outliers that are not part of the dominant rigid-body transformation. These outliers are either noise or dynamic features, and should not be used to test the hypothesized models.

Handle Stairways and Ramps

The Planar Semantic Model assumes that the ground plane is flat. However, indoor environments usually contain stairways and ramps, which cannot be expressed by a single ground plane. Thus, one extension to our framework is to extend the PSM to allow the ground to consists of multiple planes.

The ground should be specified by a base plane, the plane that covers the majority of the ground, and a set of sub-planes that capture stairways and ramps. A stairway can be represented by a sequence of sub-planes that are parallel to the base plane. Each sub-plane is represented by its height from the base plane along with a set of vertices on the ground-plane map specifying where the plane is present. One could also apply constraints (e.g. constant step heights or relative locations between two steps) to the sub-planes to further reduce the parameters required to represent a stairway. A ramp is a sub-plane that are tilted from the base plane. This sub-plane can be represented by an angle specifying the slope of the ramp and a set of vertices on the ground-plane map specifying where the ramp is present.

Given a good calibration or estimation between the camera and the base plane of the ground, our on-line generate-and-test framework can be extended to generate children hypotheses with ramps and stairways. Geometric properties about the ramps and stairways can also be derived.

Since stairways and ramps may be dangerous for certain types of navigating robots, such as a wheelchair robot, we can further extend our representation to capture the safeness for the robot to navigate through these regions. Starting from the extended PSM, a safety map [50, 49] of the indoor environment can be constructed. Then, the AOS can be extended to capture the safeness about each opportunity. For example, an opportunity that leads to a ramp can be navigable but unsafe.

Apply to Indoor Flying Robots

Another interesting future work is to apply this thesis to indoor flying robots, like a quadrotor. Unlike wheeled robots, a semantically meaningful representation for a flying robot must express the maximum height of the building in order to determine the limits of the flying height. The Planar Semantic Model needs to be extended to include the ceiling. Including the ceiling raises several interesting questions for the representation. First, unlike the ground-plane, ceiling may need to be represented by multiple planes, instead of one. Second, while assuming the ceiling is parallel to the ground plane reduces the degrees of freedom of the representation, in many real-world environments, ceilings are not parallel to the ground. Third, the representation must be able to express incomplete knowledge about the boundary between the ceiling and the walls.

10.2.2 Future for Scene Understanding for Indoor Navigating Robots

Detect Scene Changes

Assume a robot has obtained an understanding of the local environment using our framework. An understanding means the robot has a set of hypothesized interpretations (PSM and clutter) of this environment and a posterior probability distribution over the set of hypotheses. Ideally, there is one hypothesis with a relatively high posterior distribution among the set. When the robot revisits this environment, the robot needs to localize itself in the coordinate frame of the previous understanding and update its previous understanding based on observations from current visit.

There are two ways to localize the robot to its previous understanding. A simple way is to compare current observations with the hypothesis with the maximum posterior probability to determine its pose. We can also extend our hypothesis to include pose estimation as described in Section 10.2.1. In this case, each hypothesized interpretation will maintain its own localization. Once the robot is localized within its previous understanding, our generate-and-test framework can be applied to update its understanding of the local environment. The framework will now start with the set of hypotheses from the previous understanding, and use the previous posterior distribution as the prior distribution.

The local environment may change between the two visits. For example, an object may be placed in a different position, or may even be moved in or out from that environment. Thus, an important issue about updating the scene interpretation is to identify these quasi-static regions, regions that are static at each visit but may not remain in the same position between visits. These quasi-static regions usually consist of objects, like furniture, that are not part of the building. One can segment out these regions by comparing the previous clutter regions with the current ones.

A door, which is a part of the building and contains important properties for navigations, is also quasi-static. While a closed door can be explained by a wall segment, an open door creates an opening along the wall and the door itself will be identified as part of clutter. The best interpretation that describes the same environment depends on the status of the door. Thus, an interesting research direction is to extend our PSM representation to include doors and extend our hypothesis generation process to generate hypotheses with doors. AOS also needs to be extended to include opportunities that are "quasi-navigable", opportunities that are navigable at some visits and are unnavigable at other visits.

Active Vision: Explore the Indoor Environment

In this thesis, we process the visual data that the robot acquires and update its understanding of the local environment. This is considered as passive visual processing, because the robot does not decide where to look. In fact, scene understanding becomes more efficient, if the robot can control where to look for observations that best update its knowledge of the local environment. Thus, an interesting research direction is active exploration for indoor scene understanding, and this thesis provides a useful foundation for active exploration.

From a high-level point of view, active exploration requires the robot to know which portions of the environment are not fully explored. Our proposed representations, PSM and AOS, capture incomplete knowledge of the local environment, and provide information for planning in different levels. With the AOS, the robot can easily select an action opportunity that is navigable but contains incomplete knowledge (e.g. partially observed, potential, and beginning opportunities) to explore. With the selected action opportunity, the robot can refer to the PSM to plan a trajectory that leads to an unexplored portion of the environment while avoiding obstacles in the modeled portion.

From a lower-level point of view, having more informative features that can discriminate among the hypotheses allows our generate-and-test framework to converge to a single hypothesis more frequently. In Chapter 8, we defined an informativeness measurement to identify regions that are informative for testing the hypotheses. In this thesis, we adjust the threshold for extracting visual observations based on the informativeness to allow more informative features to be extracted and to avoid wasting efforts on uninformative regions. This informativeness measurement is also useful for active exploration. For active exploration, the goal is to compute the motion that maximizes the informativeness of the entire image. A simple example is to compute the predicted informativeness by slightly varying each dimension of the robot pose and then selecting the dimension that produces the maximum predicted informativeness. A more sophisticated method can be developed to maximize the ratio between the predicted informativeness and the amount of robot motion.

Bibliography

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Communications of the ACM*, 2011.

[2] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. *ECCV*, 2010.

[3] A. Angeli, S. Doncieux, J.-A. Meyer, and D. Filliat. Real-time visual loop-closure detection. *ICRA*, 2008.

[4] S. Y. Bao, A. Furlan, L. Fei-Fei, and S. Savarese. Understanding the 3d layout of a cluttered room from multiple images. *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[5] Y. Bao, M. Sun, and S. Savarese. Toward coherent object detection and scene layout understanding. June 2010.

[6] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin. Fast automatic single-view 3-d reconstruction of urban scenes. *ECCV*, pages II: 100–113, 2008.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *CVIU*, 110:346–359, 2008.

[8] P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the Hybrid Spatial Semantic Hierarchy. *IJRR*, 29(4):428–459, 2010.

[9] Y.-W. Chao, W. Choi, C. Pantofaru, and S. Savarese. Layout estimation of highly cluttered indoor scenes using geometric and semantic cues. *International Conference on Image Analysis and Processing*, 2013.

[10] W. Choi, Y.-W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. *CVPR*, 2013.

[11] A. J. Davison. Real-time simultaneous localization and mapping with a single camera. *ICCV*, 2003.

[12] J. De Kleer and B. C. Williams. Diagnosing multiple faults. *Artificial intelligence*, 32:97–130, 1987.

[13] E. Delage, H. Lee, and A. Y. Ng. Automatic single-image 3d reconstructions of indoor Manhattan world scenes. *ISRR*, 2005.

[14] E. Delage, H. Lee, and A. Y. Ng. A dynamic Bayesian network model for autonomous 3d reconstruction from a single indoor image. *CVPR*, pages 2418–2428, 2006.

[15] F. Dellaert, S. M. Seitz, C. E. Thorpe, and S. Thrun. Structure from motion without correspondence. *CVPR*, 2000.

[16] I. Dryanovski, W. Morris, and J. Xiao. Multi-volume occupancy grids: An efficient probabilistic 3d mapping model for micro aerial vehicles. *IROS*, 2010.

[17] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. *ICRA*, 2012.

[18] A. Flint, C. Mei, D. Murray, and I. Reid. Growing semantically meaningful models for visual slam. *CVPR*, 2010.

[19] A. Flint, D. Murray, and I. Reid. Manhattan scene understanding using monocular, stereo, and 3d features. *ICCV*, 2011.

[20] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-world stereo. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[21] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Reconstructing building interiors from images. *ICCV*, 2009.

[22] J. J. Gibson. *The Senses Considered as Perceptual Systems.* Houghton Mifflin, Boston, 1966.

[23] A. Gupta, A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *European Conference on Computer Vision*, 2010.

[24] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. *CVPR*, 2011.

[25] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. *ICCV*, 2009.

[26] V. Hedau, D. Hoiem, and D. Forsyth. Recovering free space of indoor scenes from a single image. *CVPR*, 2012.

[27] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *IJRR*, 2012.

[28] L. Hinkle and E. Olson. Predicting object functionality using physical simulations. *IROS*, 2013.

[29] K. L. Ho and P. Newman. Detecting loop closure with scene sequences. *IJCV*, 2007.

[30] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. *ICCV*, 2005.

[31] D. Hoiem, A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, October 2007.

[32] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005.

[33] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *CVPR*, 2:2137–2144, 2006.

[34] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using RGB-D cameras. *RoboCup 2011: Robot Soccer World Cup XV*, 2012.

[35] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. *CVPR*, 2007.

[36] L. Itti, C. Koch, and E. Niebur. A model of saliency-based visual attention for rapid scene analysis. *PAMI*, 1998.

[37] Y. Jiang and A. Saxena. Infinite latent conditional random fields for modeling environments through humans. *RSS*, 2013.

[38] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach. Real-time compression of point cloud streams. *ICRA*, 2012.

[39] B. Kim, P. Kohli, and S. Savarese. 3d scene understanding by voxel-crf. *ICCV*, 2013.

[40] Z. Kim. Geometry of vanishing points and its application to external calibration and realtime pose estimation. *Institute of Transportation Studies Research Report*, UCB-ITS-RR-2006-5, 2006.

[41] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. *ISMAR*, 2007.

[42] A. Koutsoudis, B. Vidmar, G. Ioannakis, F. Arnaoutoglou, G. Pavlidis, and C. Chamzas. Multi-image 3d reconstruction data evaluation. *Journal of Cultural Heritage*, 2014.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[44] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. *NIPS*, 2010.

[45] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. *CVPR*, 2009.

[46] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. *CVPR*, 2010.

[47] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[48] B. Micusik, H. Wildenauer, and M. Vincze. Towards detection of orthogonal planes in monocular images of indoor environments. In *ICRA*, 2008.

[49] A. Murarka. Building safety maps using vision for safe local mobile robot navigation. *Doctoral dissertation, Computer Sciences Department, The University of Texas at Austin.*, 2009.

[50] A. Murarka and B. Kuipers. A stereo vision based mapping algorithm for detecting inclines, drop-offs, and obstacles for safe local navigation. *IROS*, 2009.

[51] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. *ICCV*, 11.

[52] P. Newman, D. Cole, and K. Ho. Outdoor SLAM using visual appearance and laser ranging. *ICRA*, 2006.

[53] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. *ICCV*, 2007.

[54] J. J. Park, C. Johnson, and B. Kuipers. Robot navigation with model predictive equilibrium point control. *IROS*, 2012.

[55] C. Rother. A new approach to vanishing point detection in architectural environments. *Image and Vision Computing*, 2002.

[56] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, and M. Devy. Rt-slam: a generic and real-time visual slam implementation. *Computer Vision Systems*, 2011.

[57] S. Satkin, J. Lin, and M. Hebert. Data-driven scene understanding from 3d models. *BMVC*, 2012.

[58] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. In *Neural Information Processing Systems (NIPS) 18*, 2005.

[59] A. Saxena, M. Sun, and A. Ng. Make3d: learning 3d scene structure from a single still image. *IEEE Trans. PAMI*, 30:824–840, 2009.

[60] A. Schmidt, M. Fularz, M. Kraft, A. Kasiński, and M. Nowicki. An indoor rgb-d dataset for the evaluation of robot navigation algorithms. *Advanced Concepts for Intelligent Vision Systems*, 2013.

[61] R. Schnabel and R. Klein. Octree-based point-cloud compression. *Eurographics Symposium on Point-Based Graphics*, 2006.

[62] H. J. Seo and P. Milanfar. Static and space-time visual saliency detection by self-resemblance. *Journal of vision*, 2009.

[63] J. Shi and C. Tomasi. Good features to track. *CVPR*, 1994.

[64] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3d. *SIGGRAPH*, 2006.

[65] H. Strasdat, A. J. Davison, J. Montiel, and K. Konolige. Double window optimisation for constant time visual slam. *ICCV*, 2011.

[66] J. M. R. S. Tavares and A. J. M. N. Padilha. A new approach for merging edge line segments. *In Proceedings of 7th. Portuguese Conference of Pattern Recognition*, 1995.

[67] C. J. Taylor and A. Cowley. Parsing indoor scenes using RGB-D imagery. *RSS*, 2012.

[68] R. Toldo and A. Fusiello. Robust multiple structure estimation with J-linkage. *ECCV*, 2008.

[69] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmenta modern synthesis. *Vision algorithms: theory and practice*, 2000.

[70] G. Tsai, C. Johnson, and B. Kuipers. Semantic visual understanding of indoor environments: from structures to opportunities for action. *CVPR workshop on Vision Meets Cognition*, 2014.

[71] G. Tsai and B. Kuipers. Dynamic visual understanding of the local environment for an indoor navigating robot. *IROS*, 2012.

[72] G. Tsai and B. Kuipers. Focusing attention on visual features that matter. *BMVC*, 2013.

[73] G. Tsai and B. Kuipers. Handling perceptual clutter for robot vision with partial model-based interpretations. *IROS*, 2014.

[74] G. Tsai, C. Xu, J. Liu, and B. Kuipers. Real-time indoor scene understanding using Bayesian filtering with motion cues. *ICCV*, 2011.

[75] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, and W. Buntine. Unsupervised object discovery: A comparison. *IJCV*, 2010.

[76] C. A. Vanegas, D. G. Aliaga, and B. Benes. Building reconstruction using manhattan-world grammars. *CVPR*, 2010.

[77] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. *ECCV*, 2010.

[78] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. *CVPR*, 2011.

[79] J. Xiao and Y. Furukawa. Reconstructing the worlds museums. *ECCV*, 2012.

[80] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. *ICCV*, 2009.

[81] C. Xu. *Steps towards the Object Semantic Hierarchy.* PhD thesis, University of Texas at Austin, 2011.

[82] Y. Zhao and S.-C. Zhu. Scene parsing by integrating function, geometry and appearance models. *CVPR*, 2013.

[83] S. Zhu, T. Fang, J. Xiao, and L. Quan. Local readjustment for high-resolution 3d reconstruction. *CVPR*, 2014.