Attributed Object Maps:
Descriptive Object Models as High-level Semantic Features for Mobile Robotics

by

Jeffrey A. Delmerico

September 1, 2013

A dissertation submitted to the
Faculty of the Graduate School of the
State University of New York at Buffalo
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science and Engineering

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Acknowledgements

First and foremost, I have my family to thank upon achievement of this milestone. I have learned from them, both directly and indirectly, about the importance and value of education throughout my life. From a young age, my family members have helped to inspire my curiosity and cultivate in me a life-long passion for learning. In seeing the lasting impact that my mother Ricki's teaching had on her students, I understood the importance of sharing that curiosity with others through teaching. And in watching both my father Bob and brother Alan complete their doctoral degrees, I had positive role-models for the career trajectory I was plotting for myself. During my meandering studies, through physics, mathematics, secondary education, and eventually computer science, my entire family has enabled me to pursue my dreams and my curiosity through their encouragement and support. I would especially like to thank Jean, my partner in life and in graduate studies, for her love, companionship, and understanding throughout my journey to the conclusion of this degree.

I owe a significant debt of thanks to my advisor, Jason Corso, whose belief in my ideas and abilities inspired much of the work that went into this dissertation. Even before I had formally enlisted as a student of his, he recognized some sort of potential in me, and perhaps an unrealized aptitude for robotics. His patient guidance and mentoring have helped me to grow from a role of student, to that of collaborator, and eventually to an independent researcher. In Jason, I have had a consistent role model for a faculty member who balances active engagement with the students in his research group, as well as teaching and departmental obligations, with the pursuit of new research opportunities and the satisfaction of his own intellectual curiosity. I will miss our frequent discussions

about research as well as our regular social interactions. Due to my esteem for Jason and his accomplishments as a young researcher, I am somewhat apprehensive about now assuming a role of peer, but I know that it is a role for which he has prepared me well.

My development as a roboticist would not have been possible without the opportunities I received in working with the United States Army Research Laboratory (ARL). he mentoring of Philip David, which extended from my summers as an intern at ARL through this dissertation research, was invaluable in my development as an investigator in robotic perception. Our discussions have been profoundly influential on my understanding of the current research in that area. Without the ARL field experiences that Stuart Young enabled me to participate in, I would certainly not have developed an understanding of the challenges and rewards of experimental field robotics. The opportunity to work closely with many individuals at ARL, including Dave Baran, Brendan Byrne, Jon Fink, Jason Gregory, Rick Gregory, Sean Ho, Cem Karan, John Rogers, Ethan Stump, and Jeff Twigg, provided me with many great learning experiences and a very positive research environment over the years.

Over the course of five years as a graduate student in the University at Buffalo Department of Computer Science and Engineering, I have been privileged enough to learn from, study with, and teach alongside a number of great people. Among my other mentors at UB were my committee members Robert Platt and Venkat Krovi, who helped to diversify my understanding of the robotics field, and who have generated momentum for robotics research within the School of Engineering. My labmates Albert Chen, Caiming Xiong, Sagar Waghmare, Ananth Sadanand, David Johnson, Ran Xu, Kevin Keane, Gang Chen, and Chenliang Xu have all provided me with friendship, inspiration, and feedback about my research ideas over the years that we worked together. In particular, Julian Ryde and my fellow robotics students Vikas Dhiman and Suren Kumar helped me to grow in my understanding of the existing research problems in robotics. Discussions with them have broadened my knowledge, challenged and strengthened my ideas, and helped me to feel a part of

a robotics team within the Vision and Perceptual Machines Lab. I would also be remiss if I did not mention the opportunities that were provided for me by Vipin Chaudhary and the Center for Computational Research early on in my studies of computer science. Additionally, Atri Rudra acted as a mentor for me during the semesters I served as his teaching assistant, and I enjoyed the professional experience of that role, as well as Atri's personal friendship during our time working together.

I owe my final thanks to all of the other people in my life that have been supportive to me during my long journey to this point. Your influence has been critical to my success, and I am grateful to have you in my life.

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

This dissertation presents the concept of a mid-level representation of a mobile robot's environment: localized class-level object models marked up with the object's properties and anchored to a low level map such as an occupancy grid. An *attributed object map* allows for high level reasoning and contextualization of robot tasks from semantically meaningful elements of the environment. This approach is compatible with, and complementary to, existing methods of semantic mapping and robotic knowledge representation, but provides an internal model of the world that is both human-intelligible and permits inference about place and task for the robotic agent. This representation provides natural semantic context for many environments—an inventory of objects and structural components along with their locations and descriptions—and would sit at an intermediate level of abstraction between low level features, and semantic maps of the whole environment. High level robotic tasks such as place categorization, topological mapping, object search, and natural language direction following could be enabled or improved with this internal model.

The proposed system utilizes a bottom-up approach to object modeling that leverages existing work in object detection and description, which are well developed research areas in computer vision. Our approach integrates many image-based object detections into a coherent and localized model for each object instance, using 3D data from a registered range sensor. By observing an object repeatedly at the frame-level, we can construct such models in an online way, in real time. This ensures that the models are robust to the noise of false-positive detections and still extract useful information from partial observations of the object. The detection and modeling steps we present

do not rely on prior knowledge of specific object instances, enabling the modeling of objects in unknown environments. The construction of an *attributed object map* during exploration is possible with minimal assumptions, relying only on knowledge of the object classes that might be contained therein. We present techniques for modeling objects that can be described with parameterized models and whose quantitative attributes can be inferred from those models. In addition, we develop methods for generating non-parametric point cloud models for objects of classes that are better described qualitatively with semantic attributes. In particular, we propose an approach for automatic foreground object segmentation that permits the extraction of the object within a bounding box detection, using only a class-level model of that object's scale.

We employ semantic attribute classifiers from the computer vision literature, using the visual features of each detection to describe the object's properties, including shape, material, and presence or absence of parts. We integrate per-frame attribute values into an aggregated representation that we call an *object attribute descriptor*. This method averages the confidence in each attribute classification over time, smoothing the noise in individual observations and reinforcing those attributes that are repeatedly observed. This descriptor provides a compact representation of the model's properties, and offers a way to mark up objects in the environment with descriptions that could be used as semantic features for high-level robot tasks.

We propose and develop a system for detecting, modeling, and describing objects in an unknown environment that uses minimal assumptions and prior knowledge. We demonstrate results in parametric object modeling of stairways, and semantic attribute description of several non-parametric object classes. This system is deployable on a mobile robot equipped with an RGB-D camera, and runs in real time on commodity compute hardware. The object models contained in an *attributed object map* provide context for other robotic tasks, and offer a mutually human and robot-intelligible representation of the world that can be created online during robotic exploration of an unknown environment.

Chapter 1

Introduction

1.1 Overview

A robot's internal representation of the world provides the context for all of its tasks and goals. Metric map representations such as occupancy grids or feature maps utilize low level sensor responses or features to guide the robot's behavior, but do not incorporate higher level understanding of the environment. The result is an internal model of the world that is useful for the robot, as it is at the same level of abstraction as its sensor input, but is either difficult to understand for humans, or provides limited information about the semantics of the robot's surroundings. Efforts in semantic mapping have been effective at introducing such high level context into the robot's internal representation, but often these approaches abstract directly from low level features to high level semantic concepts (e.g. [1, 2]). A mid-level representation would enable inference from more semantically meaningful elements of the environment than features, but would also represent a more naturally human model of the world. Indeed, evidence from studies of human spatial memory in the cognitive science and neuroscience literature suggests that humans maintain an internal representation of their surroundings that organizes the spatial arrangement of semantically meaningful landmarks and components of the scene [3–6]. A variety of human-robot interaction tasks could be facilitated if both entities had relateable internal representations of their surroundings.

A key shortcoming of most existing methods that utilize object detection for semantic mapping or for estimating the pose of objects in the environment is that they rely on instance-level object models (e.g. [7, 8]). When exploring a previously unknown space, a robot is unlikely to have a priori knowledge of the *particular* objects that are to be found there, unless it is performing a search for *a particular object*. In general, though, instance-level object detection is not appropriate for arbitrary environments, whereas class- or category-level object detection permits exploration of place classes that are likely to contain the modeled object types. For example, office environments are likely to all contain the same classes of furniture, and many similar types of objects such as computers and personal effects, but each office will contain different instances of these object classes. This generality is a primary motivating factor in the use of class-level object detection in our approach.

Object attributes have been the focus of much recent investigation in the computer vision community [9–11]. Attributes present a multi-labeling problem, where classifiers are trained on low level features in order to describe an object detected in an image with many binary meta-features. For example, attributes such as 'vertical cylinder', 'plastic', and 'shiny', might describe a travel coffee mug. Much like the object's class, the attribute labels provide additional semantic information about the object instance, and can help to distinguish a particular object instance from other members of the same object class. Consider two detected chairs in different frames captured from a camera; with the object detections alone, the two detections could simply be observations of the same chair from different perspectives. However, if the objects are labeled with attributes, the colors, materials, and shape descriptions could be used to distinguish them if they are in fact distinct object instances.

While object types and locations provide some semantic information, object descriptions with attributes permit a more discriminating characterization of the things in an environment. For example, two offices might both contain a desk, a chair, and a computer monitor. If these objects are

arranged similarly in both rooms, their locations and classes do not provide distinguishing features between the two offices. However, if one office contains a black leather chair, a wooden desk, and a black monitor, and the other contains a gray plastic chair, a metal table, and a white monitor, the attributes of the objects provide the necessary discriminating power to determine that the two rooms are distinct. If we are interested in class-level place categorization [12] instead of place recognition, we can still use the object types, frequencies, attributes, and locations to determine the function of a space. For example, the number and type of chairs that are found in offices are often different than that of the chairs found in a dining room or hotel lobby. Additionally, object attributes may provide enough context for a task of object search with only a textual description and not a detailed object instance model.

1.2 Project Scope

The scope of this project will cover the development of the attributed object map representation and a deployed implementation thereof, which will perform object detection from RGB-D data, label the detected objects with their attributes, build localized and attributed object models from these detections, and anchor these models to a low-level map being built simultaneously with SLAM. The end goal will be a system that can automatically construct this representation in real time when deployed on a mobile robot (Chapter 5). The diagram in Figure 1.1 represents a manually drawn version of this type of representation; one that captures the objects in the environment and some of their descriptive properties, and allows both humans and robots to extract useful information and context from it.

We consider the problems of modeling both objects that can be described with parameterized generative models (Chapter 3), and freeform objects with no such models (Chapter 2). We develop methods for aggregating and localizing multiple detections of object instances, such that the re-

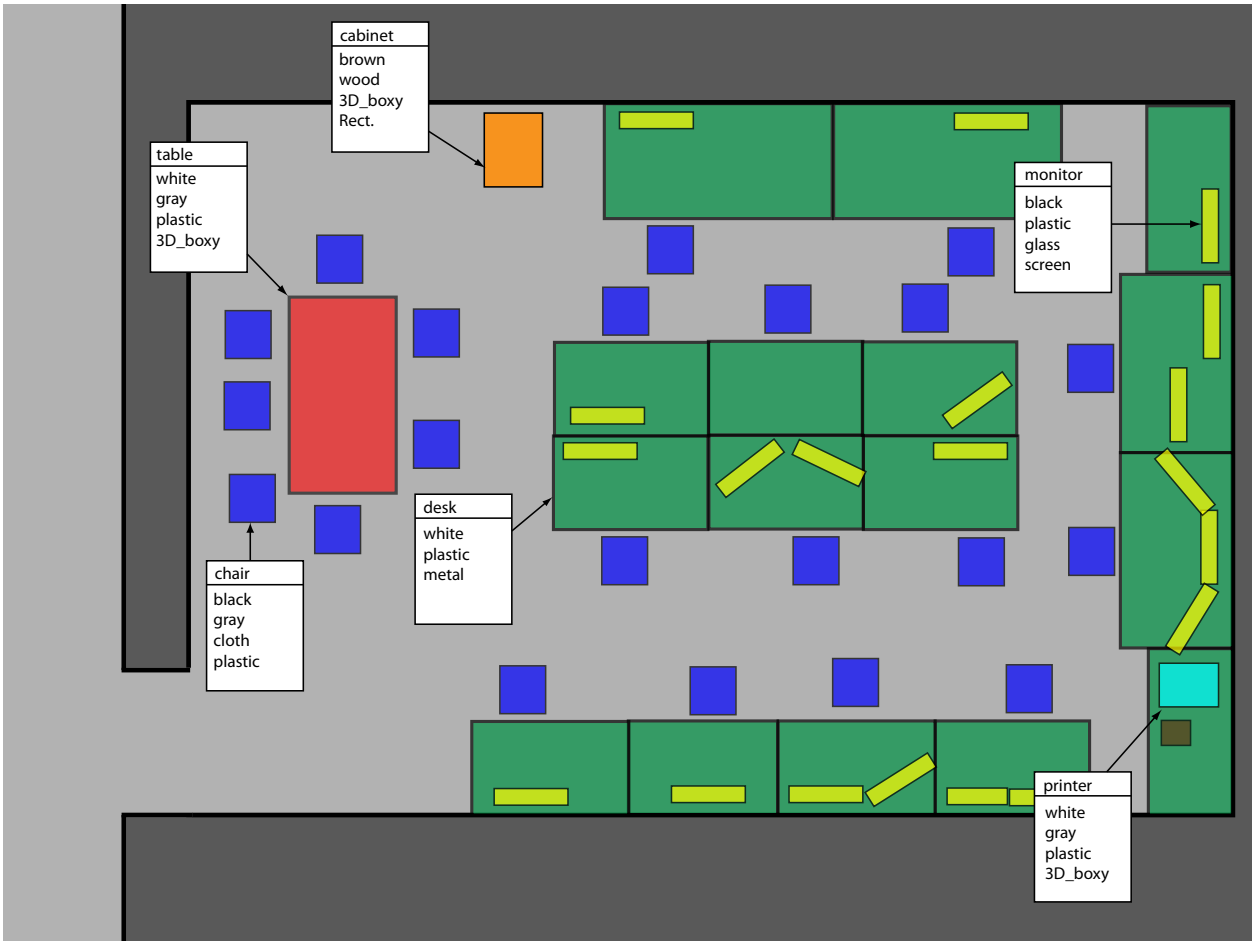

Figure 1.1: Diagram showing a conceptual example of an Attributed Object Map. In this case, it shows the old VPML lab in Davis 301B.

sulting models are consistent and stable in their locations and attributes within each trial, and that these models are consistently reproducible over multiple trials. We expand on existing techniques in attribute classification in problem-specific ways, and design a new descriptor for the attributes of a constructed model (Chapter 4). One of the applications we hope to enable with this work is place categorization and topological mapping in indoor environments using an attributed object map. In the Appendix, we present a preliminary study as a proof-of-concept of using descriptive objects as features of the environment for such a task. Figure 1.2 shows a diagram of the components of our system and the flow of data between them.

We consider a small set of relevant object classes and attributes in the work presented here, but also design the system to be flexible and easily extensible to more object and attribute types. The object classes and attributes we use are summarized in Table 1.2. We have selected these due to the availability of training data, and for compatibility with existing methods of object detection and attribute classification, which we incorporate in our system. In particular, the object classes are those relevant to our task for which trained models were available for the object detector we use, and for which we have training data for learning the attribute classifiers. This set of classes is the group from PASCAL VOC 2008 [13] that are relevant to indoor object mapping. The attributes are a subset of the 64 proposed in [10], selected for their relevance to our problem scenario. Modeling the stairway parametric object class was a motivating example for this system, but also a separate, more problem-specific project, and at this time we do not consider any other parametric objects. It would certainly be possible and straightforward to train detectors or design other generative models for new object classes, as well as learn other attributes, but we restrict our work to the development of the system using the existing classes, since the contributions of this work are not in object detection or attribute classification.

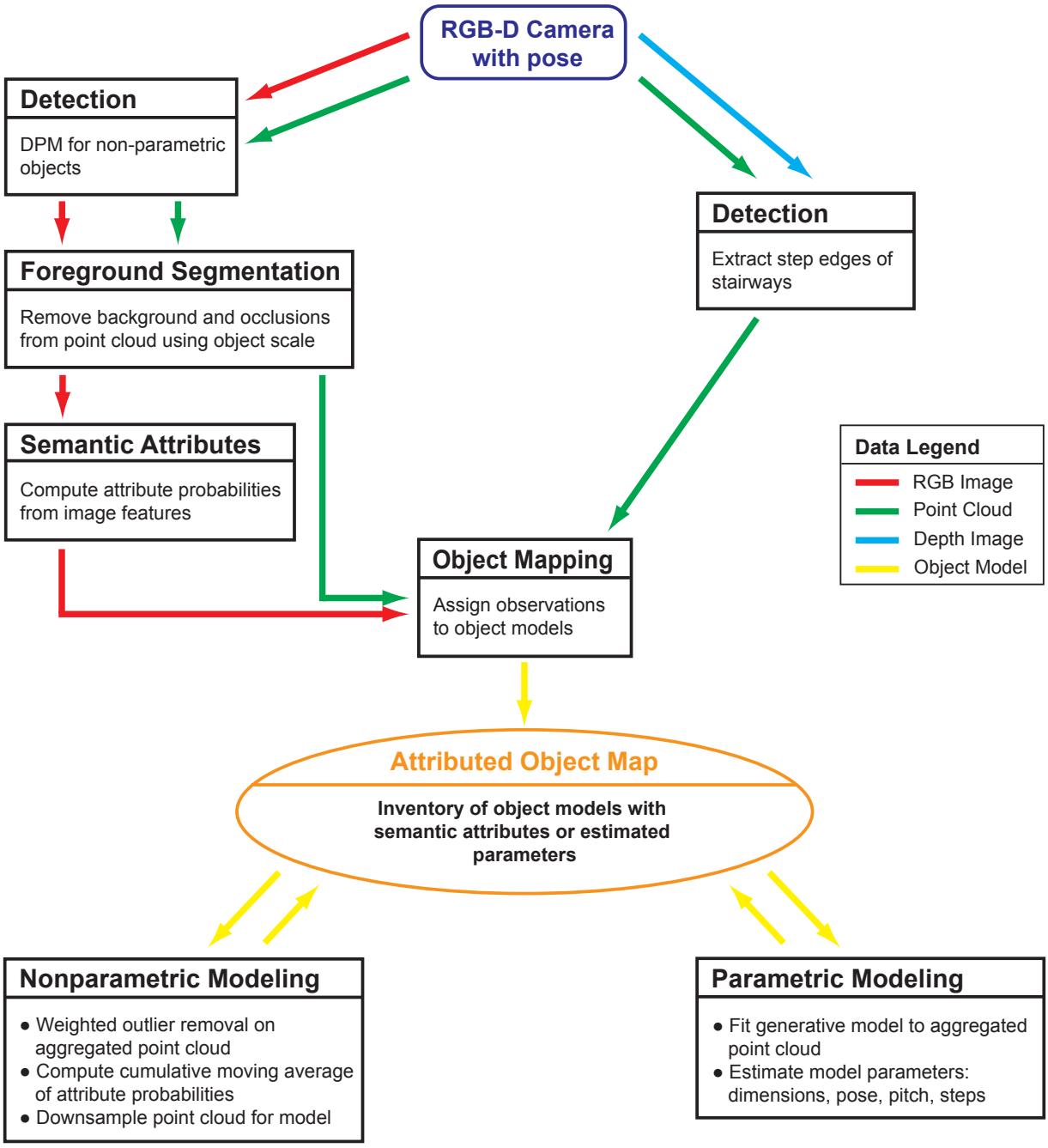

Figure 1.2: Overview block diagram for the proposed system. Each rectangle encapsulates an algorithm for performing a specific component task. Color coded arrows (see **Data Legend**) show the data transfer for inputs and outputs of these algorithms. The orange ellipse represents the data structure we construct, the *attributed object map*.

Table 1.1: Table of object classes and attributes implemented in our system.

Nonparametric Objects	**Material Attributes**	**Geometric Attributes**
Bottle	Metal	2D Boxy
Chair	Plastic	3D Boxy
Sofa	Cloth	Round
Table	Glass	Vertical Cylinder
TV/Monitor	Wood	Horizontal Cylinder
	Wool	
	Leather	
Parametric Objects	**Appearance Attributes**	**Parts Attributes**
Stairway	Clear	Furniture Arm
	Shiny	Furniture Leg
	Text	Furniture Back
		Furniture Seat
		Screen

1.3 Literature Review

Semantic mapping has received increasing attention in the research community in recent years as robotic platforms and sensors have become more sophisticated, and new algorithms have enabled the introduction of high level understanding of the environment into mapping and exploration tasks. However, many of the existing representations of the robot's environment operate on low-level data (features, sensor responses) which are unintelligible to humans or do not have strong semantic meaning associated with them. Although some approaches end up producing high-level conclusions from these low-level inputs, they abstract directly from low to high level. This dissertation presents the idea of a mid-level representation that instead abstracts from low-level data to semantically meaningful components of the environment, and permits inference from these semantic entities that are at a moderate level of abstraction.

The existing methods that do incorporate high-level understanding of the world typically do not

generalize well to unknown environments because they use instance-level object models instead of class-level models, or require manual intervention in volumetric modeling of robot task environments. The other primary idea that we present here is that objects can be useful landmarks in semantic mapping of unknown spaces, but they must be considered at a class or semantic category level, and not at an instance level.

1.3.1 Semantic mapping

Semantic mapping has been applied to the problems of place recognition (identifying a *particular* place) and place categorization (determining the *type* of place), with the goal of using semantic cues from the environment as well as sensor inputs to identify the type of place or specific location of the robot. Many of these approaches use features extracted from the sensor data to perform inference about place [14–18]. Although effective at the task, these methods skip over scene elements that are between the concrete features and the abstract concept of place, including objects and structural elements of the environment. The resulting representation thus does not provide a framework for tasks beyond place recognition/categorization.

Other works incorporate a map hierarchy with different levels of abstraction: they layer conceptual and spatial representations over metric maps [14, 19–22], and include a layer for objects. However, these hierarchies are presented as a framework for high level reasoning, and the actual object detection is done in a cursory way or assumed to be provided by some other system [21, 23]. Commonly, objects are detected at an instance level or a detected object is fit to a small database of 3D models or templates [19, 20, 22, 24], or geometric primitives [25], limiting the generality of such approaches and their effectiveness in unknown environments. Limitation to instance-level object detection is also common among the non-hierarchical mapping [16] and generalized robotic reasoning systems. Such ontologies and logic frameworks enable the contextualization of tasks and conceptual knowledge representation, or keep track of object instances in a dynamic environment

[25], but often do not explicitly detect the objects that they use [26–29]. There is no existing method for object detection with associated spatial information that robustly detects objects at the class or category level in an unknown environment.

Volumetric mapping systems are also capable of performing object detection natively in 3D, and this task can be performed at a categorical model level for object classes [25, 30–33]. However, working in 3D is currently not scalable to large environments at the resolution that is necessary for geometry-based object detection. Class-level object detection in 3D "in the wild" is also a much more challenging problem without appearance information, due to the large variance in geometric properties of some object classes. Mozos et al. circumvent this limitation by enabling a robot with a large library of object models (specifically furniture) that can be downloaded from the web for a more robust model-based approach, however the other limitations of working in 3D still apply [34]. In general, these 3D approaches are geared more towards the acquisition of detailed maps for persistent interaction with a fixed environment, and not exploration of unknown environments.

1.3.2 Object detection in semantic maps

One of the shortcomings of the object detection methods that do incorporate spatial information, whether natively in 3D [30], in RGB-D space [15], or from images with layout inferred from perspective cues [35] is that they rely on templated object models or an object database. In an unknown environment, even the classes of the objects therein may not be known a priori, and so although these approaches may work well in more constrained scenarios, they do not generalize to arbitrary object classes or places.

One approach to overcoming this generalization problem is the "tabletop assumption". A number of works utilize the principle that, often, objects of interest within a human environment will be

supported by a planar surface, such as a table or desktop [25, 33, 36, 37]. This assumption is valid for many object classes and in many situations. By finding and mapping the planar surfaces in the robot's immediate locale [38], objects or object hypotheses can be discovered even without explicit class-level detection or instance-level recognition [37]. Other regularities of a known environment type can be leveraged as well. For example, Rusu et al. [32] use the expectation that a kitchen environment will contain cabinets and drawers to focus a domestic robot's search for handles to manipulate. Rogers and Christensen [36] use the tabletop assumption, combined with knowledge of adjacent room types and the object categories that are likely to be found in those room types, to detect and recognize known objects in a domestic environment during an object search task. The use of prior knowledge about the robot's environment can greatly reduce the search space for specific items or object classes, if that knowledge is available.

But what if the object of interest *is* the table? What if the robot is in an environment about which it does not have any prior knowledge? In these situations, the scenario of this proposed work, we aim to relax even these assumptions, and leverage only object classes. Situational context can certainly offer refinement to our bottom-up approach by providing a restricted search space for object detection or a restricted set of objects to search for depending on place. However, we present this work as a means of *using* objects to provide that context, and not the other way around. This approach is very much in line with that of [36], which explores a space and infers the place context from the objects it detects, but feeds that inference back into the controller guiding the robot's actions. Jointly modeling the objects and places in this way is compatible with our perspective of objects, and our bottom-up, class-level approach could overcome the restriction of the approach in [36] to object instances in a precomputed database. One of the primary positions in our proposed approach is that supportive objects (tables, desks, etc.) as well as the objects they support, are important for understanding the context of place. The limitations in generality of the Rogers and Christensen approach could be overcome with the class-level modeling approach we advocate, and could be extended to unknown environments and by moving away from the use of

object instances.

There are several works that focus on class-level object detection and infer the spatial layout of the scene from perspective and contextual cues, but these are evaluated on only a small number of object classes [39, 40] or rely on a model database [41]. Lee et al. [42] avoid this problem by using a generative approach and modeling objects as cuboids, but the output is a scene layout that is agnostic of object classes. But this concept is extended with [43] who recover both object class and pose from the arrangement of aspect parts. Limketkai et al. [44] detect structural object classes (walls, doors, etc.) in indoor environments and use them for semantic mapping. Structural objects are among those we seek to model as well. The work by Saenko et al. [45] presents a robust appearance based object detection system that is capable of detecting object categories as well as object instances if there is enough support from local features. Other work extends the generality even further to a generic "object" detector that detects objects without specifying their categories [46]. Such detections could then be classified and described using attributes.

Several works have demonstrated object detection and pose estimation from multiple observations [7, 47], but these approaches seek to match a repeatedly observed object to a database of learned instances. In contrast, we propose building object instance models from multiple object class observations, which is more in line with approach [48] that aims to infer a 3D model from a single RGB-D view. Given an object detection in image space, in order to aggregate multiple detections, we must segment the object from any occlusions or background in the bounding box. None of the existing methods are general enough for segmentation of arbitrary instances of arbitrary object classes. Some of the work by Lai et al., although including many object classes and instances, still relies on a model database [49–51]. However, [52] does present methods that use depth information to extract the 3D points corresponding to the object in the bounding box with high accuracy, but in an offline manner. Similarly, the work by Sun et al. [8, 53] provides some techniques for extracting partial 3D observations of an object based on image-wise detections. These sorts of approaches

may be leveraged to perform our object segmentation, but practical concerns like computational cost may require new techniques.

1.3.3 Object description with attributes

The problem of object description with physical properties or semantic meta-features, what have been called "attributes", has received much attention in recent years in the computer vision community. These studies have investigated description of objects, inference about class or category from those descriptions, ranking of objects based on their relative presentation of those attributes, or fine-grained attribute localization on the object [9–11, 54–58]. Despite the appeal of their descriptive power and their ability to be understood by humans as well as robots, their use has not crossed over extensively into the robotics community.

Quite a few works, though, have introduced some type of semantic feature into systems for robotic behavior. For example, Rogers et al. [59] detect and read signs in office environments to serve as semantic landmarks in performing SLAM. And Pronobis and Jansfelt [16] use geometric features of rooms such as shape and size, which are derived from laser scan data, to provide semantic context for place recognition.

The recent work of Sun et al. in object identification [60] represents a robot-oriented application of image-based attributes. They use sparse coding techniques to learn a number of visual attributes similar to those proposed in [10] and used in this project, including shape and material, but also include the more subtle concept of object name. This idea goes beyond object class to include the variety of names that humans might use for these objects in natural language, as in [61]. For example, a bag of pretzels might be referred to as "bag", "bag of pretzels", "Rold Gold", or any number of other variations. The goal of this work is to recognize object instances from their natural language descriptions, given a set of segmented images of objects. This approach to object

description has applications in human-robot interaction, but their attribute classification methods, and especially the object name attribute, would be advantageous in increasing the descriptive power of our object models.

1.3.4 Objects as features of the environment

The presence of objects, without explicit attribute descriptions, have been used to inform a variety of vision tasks, including scene classification [62], textual scene description [63], as well as image understanding and scene layout [39]. In the robotics domain, object detection has been used for place recognition and place categorization, and the objects have been described by their class and/or instance label, but not with further attributes. The systems in [14] and [16] include object detection among the components that provide features of the environment for inference, but their detection systems are not extensible to arbitrary classes and they rely more on other types of features. Torralba et al. [64] go in the other direction and use the inferred place as a prior for determining which object detectors to run. The work by Viswanathan et al. [65, 66] is perhaps most in line with the object mapping approach we present here, if applied to the place categorization problem. They perform class-level object detection and then infer the location class for place categorization from the distribution of detected objects. However, their system is validated strictly with simulated scenes, and has not been deployed for actual robotic applications. Consequently, many of the challenges of a live system have not been considered, including object model localization and maintenance of continuity for an object model over multiple observations.

The more recent work of Rogers and Christensen [36] represents a deployed system that does use objects as features of the environment. They model both the likelihood of finding an object class in a particular place, and the probability of a room being classified as a particular type of place given the objects it contains. This work demonstrates the utility of using object locations and types as features in performing inference about the robot's environment. Although currently restricted

to object instances in a database and a few broad categories of object types, the performance of their system at the joint place categorization and object search problem could be improved and generalized with the use of attributes and class-level models like those proposed here.

1.3.5 Outdoor semantic mapping

Although much of the semantic mapping literature focuses on indoor environments, approaches for outdoor environments seek to leverage object detection [67] or scene labeling [68] to inject semantic information into the map. However, we restrict our work to the indoor problem.

1.3.6 Comparison with *worldmodel*

The most similar work to the proposed system is that of Mason and Marthi [37], and the *worldmodel* software package. This work presents an approach to automatically discovering, mapping, and describing objects as elements of a semantic map, built while a robot explores a known, but possibly dynamic environment. It shares many of the goals of this work, and already implements some of the applications of interest for *attributed object maps*. However, there are several important differences in both the motivation and execution of the two works, and there is nothing mutually exclusive about them, to the point that they could work in complementary fashion. It should also be noted that [37], as well as several other works that intersect with the goals of this project [36, 60], have been published since our work was originally proposed.

Mason and Marthi have a PR2 robot autonomously navigate a large indoor environment, extract supporting planar surfaces that it encounters, such as tables, and segment objects that rest upon these planes from dense point clouds provided by an RGB-D sensor. In this case, they define an object as "a geometrically distinct occupied region atop a plane", and they specifically avoid explicit object detectors. They extract and inventory these "objects" of ambiguous class and maintain

a database of them, permitting queries and change detection over time. In order to facilitate these tasks, the objects that this system extracts use the color and spatial information in the segmented point cloud to compute a small set of semantic attributes for each object. They compute the dominant color of the object with an HSV histogram, and the approximate diameter and planar/curved shape from the cloud's spatial data. A location attribute is also associated with each object, using the object's pose and distance to a set of predetermined landmark locations. These attributes enable the detected objects to be differentiated and they demonstrate that such semantic properties can indeed provide useful context for high-level semantic robotics tasks, given an inventory of described objects.

However, the advantages of avoiding *a priori* models or detectors also come with the downside that there is no associated semantic class for each detected object. So a small, green, curved object with an *office* location tag in the *worldmodel* system could either be a green coffee mug or a small potted plant on someone's desk. The set of simple attributes introduced in [37] is sufficiently descriptive for the tasks they consider, but the lack of object class and the small set of attributes may make disambiguation of more complex objects or tasks such as object recognition difficult. One of the primary motivations for the bottom-up approach taken in our proposed system is the ability to leverage the existing techniques in image-based attribute classification. This frame-based approach also permits robust attribute association for each object because the attributes can be assessed from many observations over time, either by computing an averaging descriptor as we propose, or in a top-down way from an object's aggregated point cloud model.

One strong similarity between [37] and this project is the approach to merging multiple observations of a single object. Mason and Marthi also propose a method of merging colocated observations of an object, along with the merging of their associated point clouds and attributes. However, this is implemented as a way to avoid double-counting object instances, and not as a design feature. In our case, we include explicit dependence on a bottom-up approach for robustness and because

we, like Mason and Marthi, expect observations to be partial and potentially sparse, but we aim to enable more complete object observation and modeling by also performing outlier removal.

Considering their similarities, the proposed work and [37] could be competitive approaches to the same problem, but few aspects of these implementations are contradictory, and indeed they could serve complementary functions. For example, the perception pipeline of [37] could produce object candidates, and the *attributed object map* pipeline could produce object detections for known object classes. The intersection of these sets would be object observations with known classes, while the set of object candidates with no corresponding object detection would be of unknown class, and the detections with no corresponding object candidate could be pruned as false positives. Attributes from both the image-based detection and point cloud object hypothesis could describe the resulting object, and the robot could actively or passively pursue further observation and modeling of the object depending on its current task.

1.3.7 Contributions

This work makes the following contributions:

- A new mid-level representation for storing objects and their descriptions that can provide semantic features of a mobile robot's environment and human-intelligible context for a variety of robotic tasks. This representation is unique in the literature in its passive acquisition of multi-view object models and attribute description over multiple observations.
- Techniques for robust, bottom-up object modeling of both parametric and nonparametric object classes from many partial observations.
- A method for automatic foreground segmentation—removal of both background and occlusions—of objects in bounding boxes of RGB-D imagery.
- A novel generative model for a stairway and a robust step edge detector that permit estima-

tion of stairway dimensions and traversability.

- A simple attribute descriptor for capturing an aggregate description of an object model from multiple image-based attribute observations.
- A real time, deployable implementation of a system for object detection, automatic segmentation, attribute classification, object modeling, and mapping. This system is prepared for public release to the semantic mapping community.

Chapter 2

Non-parametric Object Modeling

2.1 Introduction

A primary and necessary task in developing a map of objects in a robot's environment is the localization of those objects. Most of the object classes we consider here in this project do not have a parametric model associated with them, and so we do not have a predetermined geometric structure for instances of these classes. Instead, we seek to discover the parts of the environment that "look" like each object class of interest, and extract a segmented object in three dimensions from RGB-D image-based observations. The robot must then represent each object internally in a way that permits assessment of the object's properties. In order to construct these object models in a way that allows the robot to make use of the information contained within, we desire a system for isolating and localizing these objects in an online way. These models should capture the physical properties of the object in a compact form that allows for updates upon further observation.

To that end we present a bottom-up method for 3D object segmentation from RGB-D data that leverages existing robust class-level object detectors. We consider the following tasks: 1) given an object bounding box on an RGB-D image, segment out the foreground object; 2) given a sequence of these segmentations, reconstruct a volumetric representation of the object. We learn a

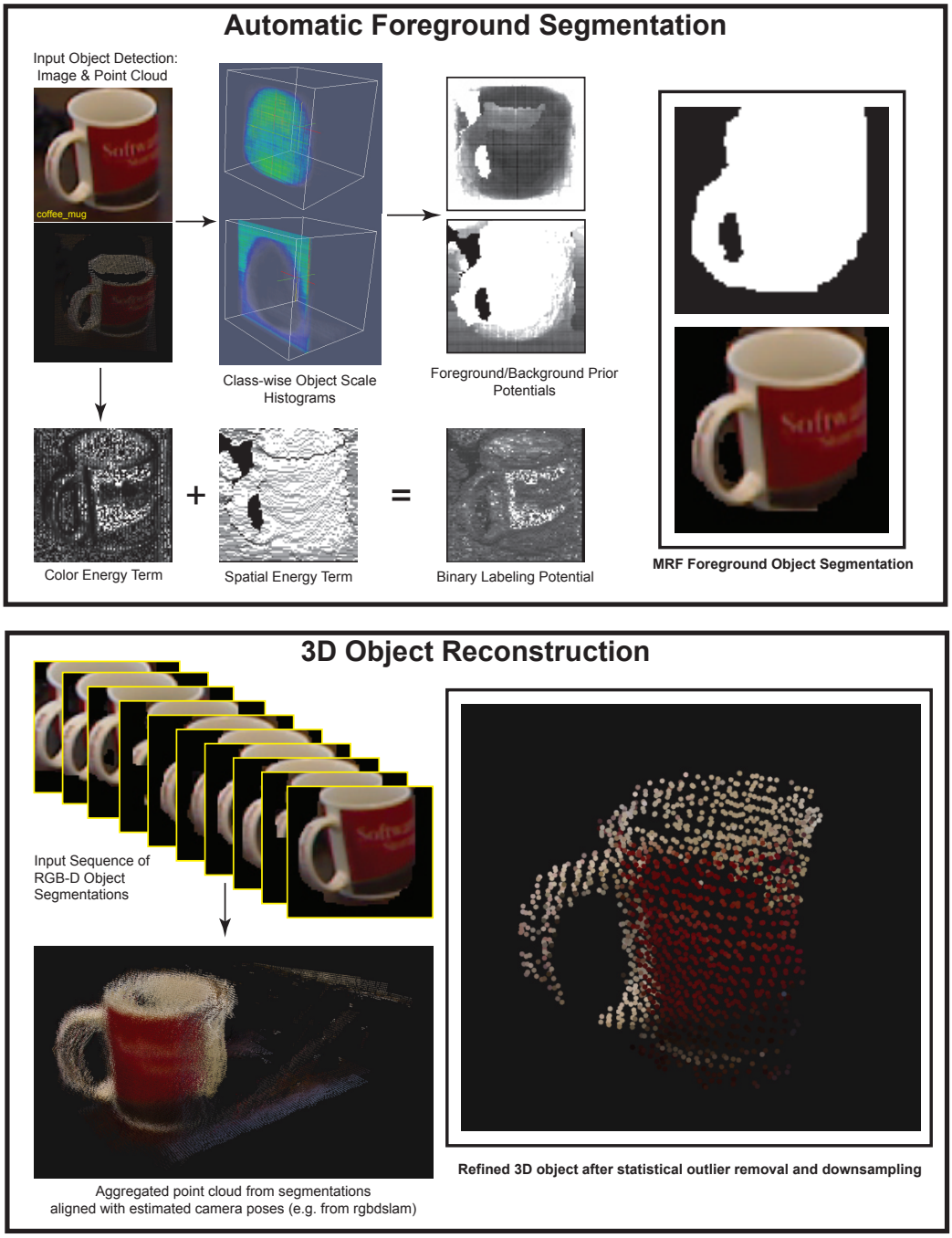

Figure 2.1: Method overview. Automatic segmentation is performed on an RGB-D bounding box input using a Markov Random Field. Multiple segmentations are aggregated in 3D and then the reconstruction is refined based on local point density.

non-parametric model of object scale and spatial distribution for each class and use it to estimate the likelihood of object presence at each pixel of a detection bounding box. This serves as the prior for a Markov Random Field energy function whose pairwise term enforces consistency in 3D spatial layout and color for foreground objects. This energy is minimized to produce a pixel-wise segmentation of the foreground object within the bounding box on both the image and its corresponding 3D point cloud. We aggregate multiple segmentations in 3D using estimated camera poses, and refine the volumetric model by performing outlier removal based on local point densities. We evaluate the performance of our automatic segmentation approach on several object classes, and we compare our 3D reconstruction to existing model-free approaches in volumetric object segmentation.

Appearance-based object detectors that produce bounding boxes are well developed and robust, with state of the art methods such as Discriminatively Trained Part Based Models [69] capable of achieving high accuracy. However, bounding boxes capture background and occlusions in addition to the object in question, so the actual object must be segmented from that bounding box in order to be isolated, manipulated, etc. Although bounding boxes are often roughly centered on the object they are detecting, this is not always the case, and non-convex objects that are centered may not actually occupy the geometric center of the bounding box (e.g. a torus). Additionally, a general class-level object detector will not make assumptions about object color or pose, or spatial layout if it is an RGB-D detector. Indeed, the minimal output we can expect from an object detector is a bounding box in the image and a class label, and our approach to automatic foreground object segmentation relies only on these inputs.

We seek to employ these foreground object segmentations in a bottom-up approach to 3D object reconstruction without an instance-level object model or class-level template so that arbitrary instances of an object class can be modeled in unknown environments. However, as with any computer vision system, there will be some noise and error in the segmentations: some visible parts of

the object may be missing and some background or occlusions may be included. The challenge of this problem is the separation of object points from outliers using only the information contained in a collection of these segmentations.

An implicit approach to foreground segmentation is background subtraction, which can be employed to detect and segment a novel object in a previously viewed scene [70]. However, we aim to solve this problem with minimal assumptions, so we avoid the use of explicit background images. Instead, we leverage the spatial information provided in the depth channel of RGB-D imagery to use object scale for extracting the foreground. Existing methods for 3D object reconstruction or object recognition often rely on geometric templates [8, 71], or instance-level models [72]. Although these strategies are appropriate for their respective applications, they do not capture the intra-class variance in object geometry or generalize to arbitrary instances.

Similarly, some approaches use the "table-top" assumption to impose a strong prior on 3D object location and pose [45,73]. Again, although this is an appropriate assumption for some applications, notably object discovery for robot grasping [73], we seek a more general solution that can be used to reconstruct arbitrary objects, including tables themselves. Top-down methods such as [52] are also inadequate for our problem scenario, as they require that the full scene be reconstructed before objects can be segmented from its volumetric representation. A bottom-up approach would be capable of online object reconstruction concurrent with 3D reconstruction of the scene.

We seek to relax these assumptions and avoid the aforementioned limitations in enabling foreground object segmentation for 3D object reconstruction. We have developed an approach that is general, extensible to arbitrary classes and scenarios, and can work online. A visual overview of our approach can be found in Fig. 2.1.

We want to leverage the existing work on appearance-based object detection in the literature. We therefore only assume that some other system provides object detections as bounding boxes in RGB-D images along with a class label. We make no assumptions about the specific appearance,

geometry, or pose of a detected object. Instead, we learn a non-parametric model that captures the distribution of the foreground class in a histogram encoding location within the bounding box and object scale (see Fig. 2.2). The scale of the object is inferred from the apparent size of the object in the scene and its depth. We also learn a model of the spatial distribution and scale of the background, and use these models as priors for a Markov Random Field (MRF) that is used for two-class (foreground/background) labeling. We use spatial and color cues from the D and RGB channels of the RGB-D images to enforce segmentation boundaries when we perform energy minimization on the MRF to obtain automatic foreground segmentations.

For 3D object reconstruction, we combine multiple segmentations together, extracting the 3D points from each foreground segmentation and aggregating them into one point cloud. The resulting cloud will have many erroneous points from imperfect segmentations. However, regions in space that are repeatedly observed (the object) will have a high local point density relative to regions that are erroneously segmented as foreground in a few of the input images. Based on this intuition, we expect the density of the point cloud in the location of the object to be reinforced over multiple observations, allowing us to filter out occlusion and background outliers based on their local point density. Our priors and constraints can be computed quickly from the image, and the segmentation routine is fast enough to be run in real time. Periodic refinement of the 3D reconstruction would allow this approach to be implemented online.

2.1.1 Related Work

In this chapter, we primarily compare our results to the work of Lai et al. in [52]. This is the most similar work to ours in terms of approach — using object detections in individual frames to extract the object in the assembled scene. However, they take a top-down approach, reconstructing the scene from many RGB-D frames while performing object detection and learning a multiclass likelihood for each occupied voxel from the class labels of object detection points that lie within

it. They use an MRF over this volumetric representation to then do multiclass segmentation of the scene into object classes and background. Their approach produces good results in terms of both precision and recall, but it is an offline algorithm and requires specialized depth-encoded object detectors. Our problem of 3D reconstruction is the bottom-up equivalent of their segmentation problem, but in addition we have developed our automatic foreground segmentation approach, which may be of interest beyond this 3D application.

The work of Sun et. al. [8, 53] is also similar to our approach to the reconstruction problem in the sense that they also attempt to use partial object views to obtain a full object model. They use a technique that they call Depth-Encoded Hough Voting (DEHV) that enhances object detection, but primarily allows them to extract some 3D foreground points from within the bounding box. To these sparse points (not a full segmentation) they fit 3D templated models in order to reconstruct 3D objects from as little as one image. However, their reliance on models makes their approach inadequate for our purposes.

A related result from Meger and Little [41] performs multi-view 3D object recognition on indoor scenes. They produce occlusion masks, the inverse of which is similar to our foreground object segmentations, except that they mask only occluding objects and include both the desired object and the background.

A recent RGB method [74] for automatic object detection and segmentation uses inpainting to detect salient regions of the image that are very different from the original when inpainted from their surroundings. Combined with a class-level object shape mask (similar to our scale histogram, but without a depth component), they produce automatic segmentations of pedestrians for which they claim state of the art performance. However, we do not compare to their quantitative results since they only perform this method on the pedestrian class, and not on indoor objects.

Ückermann et al. [73] perform model-free tabletop scene segmentation from RGB-D, intended to enable a grasping robot to observe a table and pick up arbitrary, previously unknown objects.

They separate the image into object edges and surface patches based on local normals and perform grouping and splitting to infer the independent object, without assumptions about their shapes or poses. Aside from the tabletop assumption, there are also no semantics attached to the resulting segments.

Although we consider only passive perception of objects in this work, the approach presented in [75] uses active perception to construct a 3D model of an object that is grasped by a manipulator and moved to new poses, using several partial views to construct the model. Our work shares its goal of full object modeling without templates, but we do not perform any manipulation of the object. However, their surfel-based models and explicit alignment procedure with ICP do produce cleaner, more complete models. This work represents a potential post-processing step should our passive modeling system be deployed on a platform capable of mobile manipulation. An approximate location and model can be developed passively, perhaps accurate enough for an initial grasp of the object, and a precise model can be generated using this active perception approach.

Our object scale histograms are inspired by the use of inferred object size in [45, 52, 76]. The intuition for this approach is based on apparent object size in images: an object of height h at some distance d will appear to be of height $h/2$ at distance $2d$. For an object with known depth from the camera, its true scale can be estimated based on how large it appears in the image. This sort of feature is used in these other methods, and we leverage it in a new way in a 3D histogram that encodes the scale along with the distribution of the object and background within the bounding box.

We know of no other work utilizing an MRF on individual RGB-D images, using all four channels in the data and smoothness terms. MRFs have been used on RGB-D data in other ways, for example on volumetric maps reconstructed from RGB-D images [52] and on scene change detection and object discovery [77], but these methods apply MRFs to the scene as a whole.

2.1.2 Contributions

We make three primary contributions in this chapter:

- An algorithm for automatic foreground object segmentation from a bounding box detection using RGB-D cues in a Markov Random Field. Compatible with existing state of the art appearance-based detectors.
- A novel approach to modeling the likelihood of foreground and background classes within a bounding box based on relative location within the box in x and y and object scale inferred from its depth.
- A 3D object reconstruction algorithm that merges multiple partial observations. The segmentation algorithm runs fast enough to enable real time, online performance for this reconstruction approach.

2.2 Methods

The presented approach is designed to apply the utility of appearance-based object detection to 3D object segmentation and modeling by using RGB-D data. We assume the input to our algorithm to be a class-labeled bounding box produced by some object detector (e.g. Discriminatively Trained Deformable Parts Models [69]) operating on the RGB (and possibly D) channels of RGB-D imagery. For each object class, we learn a model of where the foreground object exists within the space of the bounding box. We train a 3D histogram that we call an *Object Scale Histogram* (OSH) on the location of foreground pixels in the bounding box (normalized by bounding box size) and a measure of the object scale that is estimated from the depth of each pixel and the size of the bounding box relative to the image. We train a similar model for the background pixels in the bounding box. The backprojection of an input object detection to these histograms provides priors

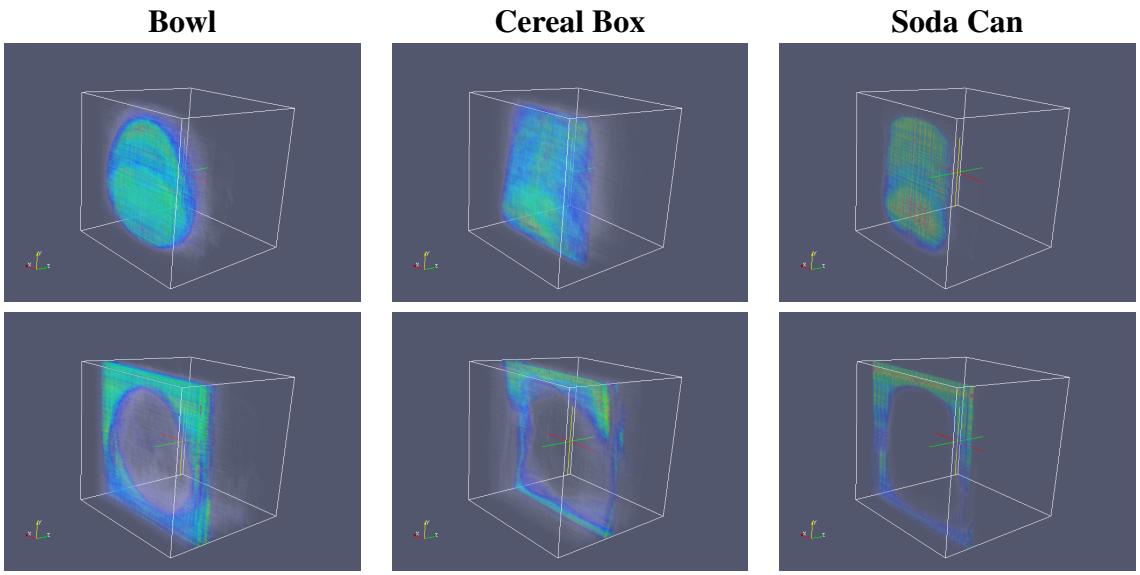

Figure 2.2: Object Scale Histograms for several of the object classes we consider (foreground top, background bottom). Note that the histogram is populated in different parts of the scale direction (to the right) for these classes.

for the foreground and background classes at each pixel in the bounding box.

We use a Markov Random Field (MRF) model to incorporate this prior with appearance and 3D cues in an energy function that we minimize with Graph Cuts [78] to provide a segmentation of foreground from background within the bounding box for a given class. This segmentation produces both an RGB segment in the image frame and a corresponding point cloud in 3D space, each of which can be used in further applications (e.g. robot grasping, foviation on salient objects). The application of interest to us is multi-view 3D object reconstruction, and we present an algorithm that utilizes many partial 3D object segmentations to construct a full object point cloud without relying on a geometric model or object template. We aggregate multiple segmentations and then perform statistical outlier removal based on local point densities to produce a refined volumetric representation of the object.

2.2.1 Object Scale Histograms

Object detectors produce bounding boxes as output with the idea that the detected foreground object is captured somewhere within that box. For arbitrary objects, we can not assume that the foreground object is centered in the bounding box, and thus we seek to model the distribution of foreground pixels within the space of the bounding box. Since we use RGB-D data, we can also model the spatial distribution of the foreground object in the depth field. However, since the pixel resolution and depth sensitivity of an RGB-D camera is relatively low, most indoor objects, especially small ones like we consider here, are likely to only be detected in a small region of the camera's usable range. Therefore, we encode this spatial information in the estimated size of the object as described below. We use these normalized data to train two histograms for each object class, providing us with distributions of foreground and background pixels for the class in the coordinate space of the bounding box and object scale.

We assume a cropped bounding box of RGB-D data, and we preprocess the data as follows. Given a bounding box of dimensions $w \times h$ pixels, with the data captured by a sensor with image size $W \times H$ pixels, normalize the coordinates of each RGB-D pixel (u, v, d) in the bounding box:

$$(u_n, v_n, d_n) = \left(\frac{u}{h}, \frac{v}{w}, \frac{d\sqrt{w^2 + h^2}}{\sqrt{W^2 + H^2}} \right) \tag{2.1}$$

The depth is scaled by the apparent size of the bounding box relative to the image as whole, on the intuition that the same object at twice the distance would have a bounding box diagonal ($\sqrt{w^2 + h^2}$) that is half as large, and would thus have the same scale by this metric. For each class, we learn an object scale histogram for the foreground, and one for the background, from labeled training data that is preprocessed as above. We use a histogram with $100 \times 100 \times 100$ bins, normalizing by the total number of (u_n, v_n, d_n) data points it contains. This resolution proved to be adequate for all of our trials.

Figure 2.3: Examples of foreground object segmentation with our method. Note the occlusions that are segmented out in the coffee mug and cap examples.

With trained OSHs for each object class, we can then determine the foreground and background likelihoods for a given $\mathbf{x} = (u, v, d)$ pixel in an input bounding box by preprocessing as above and then computing its backprojection in each of the OSHs. The trained foreground and background Object Scale Histograms for each are visualized in Fig. 2.2.

2.2.2 Markov Random Field Formulation

We model the problem of foreground segmentation in two dimensions, where the pixels of a bounding box to be segmented are organized in a 2D lattice, Λ. However, we incorporate the spatial information provided by dense RGB-D data by associating the corresponding depth value and (x, y, z) coordinate with each lattice site, as well as the pixel's color information. We seek to label each

lattice site y_i as belonging to either the foreground class or the background class: $y_i \in \{f, b\}$. The distribution of labels in our model derives from a Markov Random Field with the following energy function:

$$E(y_1, \ldots, y_{|\Lambda|}) = \sum_{\mathbf{x_i} \in \Lambda} \Psi(\mathbf{x_i} \mid y_i) + \sum_{\{r \sim s\} \in \Lambda} \Phi(y_r, y_s) \tag{2.2}$$

The unary term (Ψ) incorporates the foreground prior developed in Sec. 2.2.1, and the pairwise term (Φ, summed over all pairs of neighboring pixels in our 4-connected lattice) uses spatial and color similarity to enforce consistency in the labeling of neighboring pixels. Minimizing this energy function over the lattice produces the optimal labeling of foreground pixels, and a segmentation of the foreground object from the background in both the bounding box and in the associated dense 3D data.

Our energy function leverages the foreground prior in the unary term to encode the distribution of the object class' scale, as well as its geometric distribution in the normalized bounding box. We use the negative log of OSH backprojection for each class with α as a balancing constant:

$$\Psi(\mathbf{x} = (u, v, d) \mid y) = \alpha(-\ln OSH_c(u_n, v_n, d_n)) \tag{2.3}$$

For the pairwise term, we consider the consistency of neighboring labels in terms of spatial and color information. From the RGB-D image and associated dense point cloud, we extract a feature vector $\mathbf{w_i} = (x_i, y_i, z_i, h_i, s_i, \rho_i)$ for each pixel, where x_i, y_i, and z_i are the 3D spacial coordinates of the pixel's point in the point cloud, and h_i, s_i, and ρ_i are the hue, saturation, value for the pixel in HSV color space. We utilize these data in following pairwise term:

$$\Phi(y_i, y_j) = D_{XYZ}(x_i, x_j) + D_{color}(x_i, x_j) + \kappa \tag{2.4}$$

We encode the spatial relationship of 3D points that correspond to neighboring pixels by penalizing

inconsistent labels for points that are close together in space. However, the penalty is low if the points are distant in 3D, as can happen at an occlusion edge in the depth field. This term of our pairwise energy is defined as:

$$D_{XYZ} = \beta \cdot \frac{\mathbf{1}_{\mathbf{y_i} \neq \mathbf{y_j}}}{d((x_i, y_i, z_i), (x_j, y_j, z_j))} \tag{2.5}$$

where $d()$ is $L2$ distance, and $\mathbf{1}_{\mathbf{y_i} \neq \mathbf{y_j}}$ is a delta function that evaluates to one if the neighboring labels are different.

Lastly, we also include the color relationship of neighboring pixels by computing their distance in HSV color space. This is based on the intuition that objects are often consistent in their coloring, or have transition in their color over several pixels, even when they have texture.

We compute this term with the distance in cylindrical coordinates of two points in HSV color space:

$$D_{color} = \gamma \cdot \frac{\mathbf{1}_{\mathbf{y_i} \neq \mathbf{y_j}}}{\sqrt{s_i^2 + s_j^2 - 2 s_i s_j \cos(h_i - h_j) + (\rho_i - \rho_j)^2}} \tag{2.6}$$

Here we penalize inconsistent labeling of neighboring pixels if they are close in the color space. The last term κ enforces standard Ising model smoothness on the labeling with a constant penalty for inconsistent labels.

We have four free parameters $\{\alpha, \beta, \gamma, \kappa\}$ that balance the penalties for labelings that violate the scale we expect for a pixel at a given depth, or the spatial, color, or smoothness relationships that we expect for adjacent pixels of the same class. In this work, we manually set these parameters, and we leave learning their optimal values for future work. Several examples of segmentations produced by this approach are shown in Fig. 2.3.

2.2.3 Foreground Segmentation

We perform foreground object segmentation using the above MRF model as described in Algorithm 2.1.

Algorithm 2.1 Automatic foreground object segmentation

1: Input: RGB-D image cropped to object bounding box, associated cropped point cloud, and object class label c
2: Construct 4-connected lattice Λ over bounding box
3: Normalize pixel coordinates to bounding box and compute object scale with each pixel's depth: (u_n, v_n, d_n)
4: Compute backprojection for all (u_n, v_n, d_n) in foreground and background OSH models for class label
5: For all pairs of adjacent nodes in Λ, compute $\Phi(y_i, y_j)$
6: Minimize $E(y_1, \ldots, y_{|\Lambda|})$ using graph cuts to determine label set $y_1, \ldots, y_{|\Lambda|}$
7: Output: binary foreground mask based on label set

2.2.4 Object Reconstruction

In order to enable 3D reconstruction from multiple partial segmentations, we observe a scene using an RGB-D camera, and concurrently perform RGB-D SLAM [79] in order to estimate the camera pose. We assume that an object detection system is also running and providing bounding box object detections with class labels as input to Algorithm 2.2. We then perform the following steps to extract foreground segments for the object and combine them into a single point cloud representing a full 3D rendering of the object.

We perform outlier removal based on one of two methods, and in our experiments we evaluate both. The method by Rusu et al. [30] computes the mean distance to the nearest K points for each point in the point cloud, essentially measuring the inverse of the local point density. They assume a normal distribution of these mean distances, and therefore filter them by removing any points that have a mean distance more than s standard deviations away from the mean μ of the mean distances for the cloud. Although this may be a valid assumption for full point clouds captured by a sensor, upon analysis of the distribution of mean distances in one of our assembled clouds

for 3D reconstruction, we found that the assumption of normality is not adequate. Many of the points in our assembled clouds come from different image segmentations, but are close together in space due to the redundant nature of multiple views of the same object. However, any background points that have erroneously been filtered into the cloud are likely much more sparse due to the background not being the focus of repeated observations. Thus, our distribution has a large peak at a small mode, but has a heavy right tail representing the sparse outliers (see Fig. 2.4). We propose an alternative approach to statistical outlier removal that is more appropriate for our data given its distribution: compute mean distance to the K nearest neighbors as in [30] but filter out points more than s standard deviations away from the mode m of the distribution.

Algorithm 2.2 Multi-view 3D object reconstruction

1: Input: Stream of RGB-D images cropped to bounding boxes, associated cropped point clouds, object class labels, and estimated camera poses
2: Initialize an empty point cloud P_r to hold the reconstruction
3: **for** each image cropped to its bounding box **do**
4: Perform foreground segmentation as in Alg. 2.1
5: Apply the output mask of foreground pixels to the associated point cloud, removing background points
6: Transform point cloud to the world coordinate frame, and add points to P_r
7: **end for**
8: Perform statistical outlier removal on P_r (as in [30] or Sec. 2.2.4) to filter out sparse, noisy points
9: Downsample P_r to voxel grid
10: Output: P_r, point cloud reconstruction of object segmented from 3D scene

2.3 Experiments

We aim to compare quantitative results with the methods in [52], so we restrict our experiments to the classes {bowl, cap, cereal_box, coffee_mug, and soda_can}. For training, we use class-wise ground truth foreground/background masks for each object bounding box. For evaluation, we are not testing object detection performance, and our system assumes that bounding boxes are provided by some other method, so we use the ground truth object bounding boxes in the evaluation sets.

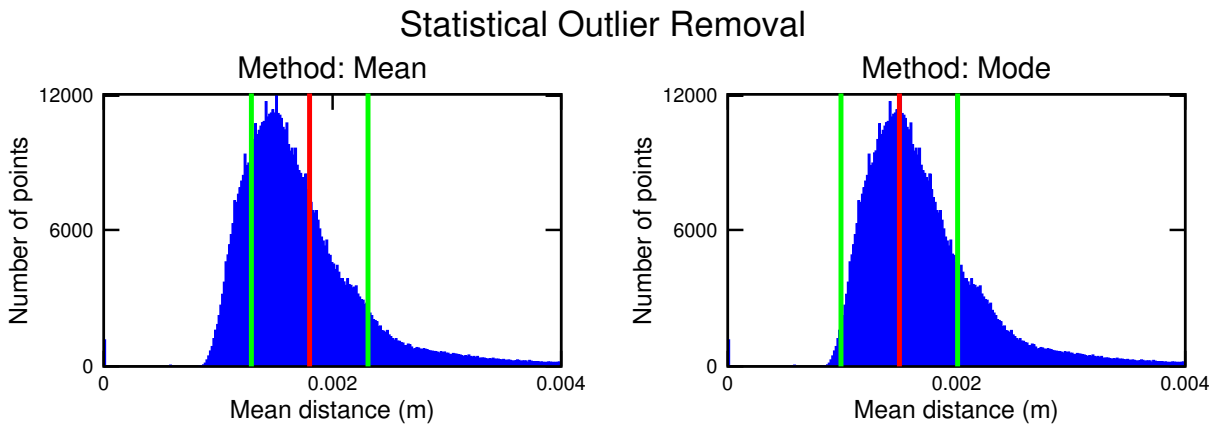

Figure 2.4: Distribution of mean distances to neighboring points in one of our assembled point clouds. The Mean method [30] of statistical outlier removal filters out points lying outside of the green bars (at $\pm s$ standard deviations from the mean (red line)). The Mode method we propose filters out points lying more than $\pm s$ standard deviations from the mode. Tail is cropped for clarity, but extends to $>$ 1.0 m along the horizontal axis.

2.3.1 Datasets

For training, we use labeled images from all of the available RGB-D datasets that contain the classes in question: RGBD Object [50], Berkeley (B3DO) [76], and NYU Depth [80]. RGBD Object contains over 50 classes, with several instances of each class observed a total of several thousand times with an RGB-D camera while rotating on a turntable. Since these training views do not capture the objects in natural indoor scenes with realistic backgrounds, we only sample a few images (four per view) for each object instance, for a total of 12 images per instance, per object class. We are only considering object scale from depth, so the redundant views offered by these sets (with a fixed camera pose and centered but rotating object) do not provide much information for our histograms.

The B3DO dataset contains RGB-D images captured in realistic indoor environments with many classes of objects labeled by bounding boxes. There were 91 images of the *bowl* class and 9 images for the *soda_can* class. The provided annotations are bounding boxes, so we created binary masks

for these images using Grabcut [81].

The NYU Depth dataset contains over 1400 RGB-D frames of indoor scenes that are densely labeled with almost 900 classes of objects. We extracted all relevant annotations by labeling any segments corresponding to our object classes with a bounding box and converting the dense labels to a binary mask.

In all, we were able to train OSHs for the foreground and background for each class using 410 images for the *bowl* class, 83 for the *cap* class, 302 for *cereal_box*, 96 for *coffee_mug*, and 95 for *soda_can*.

2.3.2 Foreground Segmentation

We tested the performance of our foreground segmentation algorithm on the evaluation sets from the RGBD Object dataset. In each of the 8 sets, an RGB-D video stream captures a cluttered indoor scene, and ground truth bounding boxes are provided for each of the object classes. In order to compute quantitative results, we randomly selected two object instances for each class from among these sets, and manually annotated ground truth binary segmentation masks using Grabcut, amounting to over 1000 masks. We ran our foreground segmentation algorithm (see Alg. 2.1) on these bounding boxes with their associated class labels and then computed the class-wise and average precision and recall.

We tested five different variations of our MRF formulation. The method described in Sec. 2.2.2 represents our intended version (Histogram+Color+Spatial+Smoothness, or HCS), but we also evaluate versions with some of the terms omitted: Histogram+Color+Smoothness (HC), and Histogram+Smoothness (HS). We also consider a naive prior that assumes the object is centered in the bounding box and a 2D Gaussian with $\sigma_x = W/3$ and $\sigma_y = H/3$ provides the foreground likelihood. We pair this prior with our color, spatial, and smoothness binary terms for the GCS trial.

Table 2.1: Precision/recall for Automatic Foreground Segmentation (top performers in **bold**)

Method	bowl	cap	cereal	coffee	soda	Overall
GCS	73.7/**94.5**	70.4/**83.6**	66.3/**96.5**	68.6/**86.3**	**63.4/95.8**	68.8/**91.7**
HS	74.2/86.3	71.1/67.8	80.1/91.6	77.0/78.2	51.8/85.0	71.6/82.1
HC	76.5/86.2	73.7/63.9	80.0/92.4	78.3/78.1	50.4/83.0	72.6/81.0
HCS	78.7/84.4	75.4/59.5	80.4/92.5	79.3/77.2	52.8/80.9	74.3/79.2
HCSN	**79.2**/84.0	**75.7**/58.3	**80.8**/92.5	**79.4**/77.2	54.6/80.4	**74.7**/78.7

Lastly, we implement the convexity potential as in Lai et al. [52] along with our other constraints for the Histogram+Color+Spatial+Smoothness+Normal (HCSN) trial. Due to the different ranges for potentials provided by each type of constraint, we used parameter values of $\alpha = 500$, $\beta = 0.5$, $\gamma = 2000$, and $\kappa = 250$, and we used these values consistently for all trials. Additionally, for the normal constraints from [52], we use values of $\lambda = 1$ and $\varepsilon = 0.1$.

Our results are summarized in Table 2.1. The top scoring methods in precision and recall show a clear trend: the assumption of a centered object leads to statistically high recall for the GCS method, but low precision, whereas the histogram-based methods with more complicated potential functions produce notably higher precision at the expense of recall. Indeed, the Gaussian prior scores well on the basis of these bounding boxes being human-annotated, and therefore generally being centered on the object. Qualitatively, the more sophisticated methods are more faithful to the object boundaries, but this is often achieved at the expense of segmentation completeness.

Another dimension of this comparison is the computational performance, which is summarized in Fig. 2.5. Processing time versus number of pixels per bounding box is plotted for each method, with the HCSN values truncated out due to their rapid growth. The computation of local normal estimates in the HCSN method is too expensive for an online algorithm, and only appropriate for postprocessing as in [52], although the normal information does improve precision slightly over the HCS version. However, based on their performance in unoptimized C++ code on a MacPro, all of the other methods could be implemented in a real time, online system operating at $> 5Hz$.

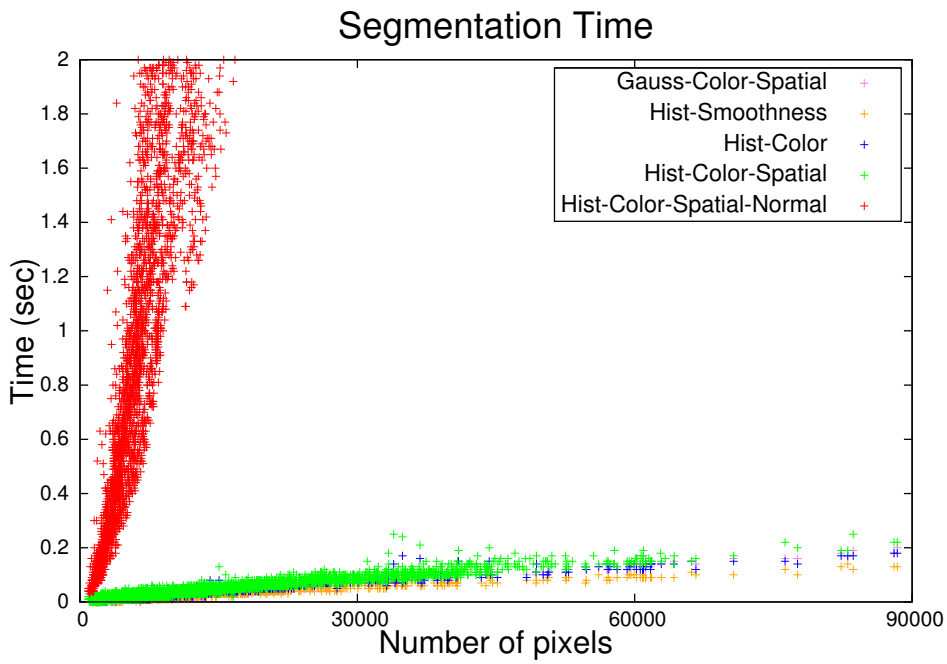

Figure 2.5: Processing time vs. pixel count per bounding box.

We advocate the use of the HCS approach due to the qualitative performance in producing segmentations that *look* good, as well as its balance of precision and recall performance. It is also fast enough to be implemented in real time.

2.3.3 Object Reconstruction

We performed object reconstruction on every object (37 in total) in all 8 evaluation sequences from the RGBD Object dataset and computed the precision and recall of the resulting 3D segmentations as compared to manually annotated ground truth. We used the ground truth object bounding boxes as our input, and performed HCS segmentation on each one. Estimated camera poses were provided by RGB-D SLAM [79]. We followed Alg. 2.2 in performing reconstruction and used both outlier removal methods (the mean-based one from [30], and the mode-based one we propose in Sec. 2.2.4). The resulting refined model was then downsampled to a 1 cm voxel grid for comparison to the results in [52]. The quantitative results are tabulated in Table 2.2 along with those

Table 2.2: Precision/recall for 3D object segmentation (top performers in **bold**)

Method	bowl	cap	cereal	coffee	soda	Overall
HCS+Mode	76.1/79.2	76.5/71.3	**95.5**/90.1	74.1/72.7	**97.2**/75.7	83.9/77.8
HCS+Mean	75.2/79.0	72.3/73.4	**95.5**/90.0	74.7/72.4	95.5/81.1	82.6/79.2
DetOnly	46.9/**90.7**	54.1/90.5	76.1/90.7	42.7/74.1	51.6/**87.4**	54.3/86.7
PottsMRF	84.4/**90.7**	74.8/91.8	88.8/94.1	87.2/73.4	87.6/81.9	84.6/86.4
ColMRF	**93.7**/86.9	81.3/**92.2**	91.2/89.0	88.3/73.6	83.5/86.5	87.6/85.6
Det3DMRF	91.5/85.1	**90.5**/91.4	93.6/**94.9**	**90.0/75.1**	81.5/**87.4**	**89.4/87.3**

Note that the overall values for the methods from [52] (DetOnly, PottsMRF, ColMRF, and Det3MRF) are averaged from the five object classes and do not include the background class.

from [52] and several visual examples of the reconstructed scenes and the objects segmented from them are shown in Fig. 2.7. Our methods (HCS+Mode,HCS+Mean) do bottom-up reconstruction and do outlier removal beyond 0.1 standard deviations from the mean/mode, while Lai et al.'s methods (DetOnly, PottsMRF, ColMRF, and Det3DMRF) do top-down segmentation of the objects from the reconstructed scene in post-processing.

To study finer-grained performance of our algorithm, we computed precision and recall for all of the objects in the evaluation set while varying the threshold s for statistical outlier removal. The data are grouped by object instance in each of the class-wise plots in Fig. 2.6, as well as averaged, and the results from [52] are plotted as points. The variance in performance on an instance-level indicates that some objects are more difficult to model with this approach than others.

2.4 Conclusions

We have presented two algorithms that, in conjunction, are capable of producing 3D object reconstructions using only bounding box object detections in RGB-D imagery. We relax the assumptions of other methods with similar goals, and design our approach so that it can work online, in real time, and can integrate with existing state of the art methods for object detection.

Although our quantitative performance does not consistently beat the top-down, offline approaches,

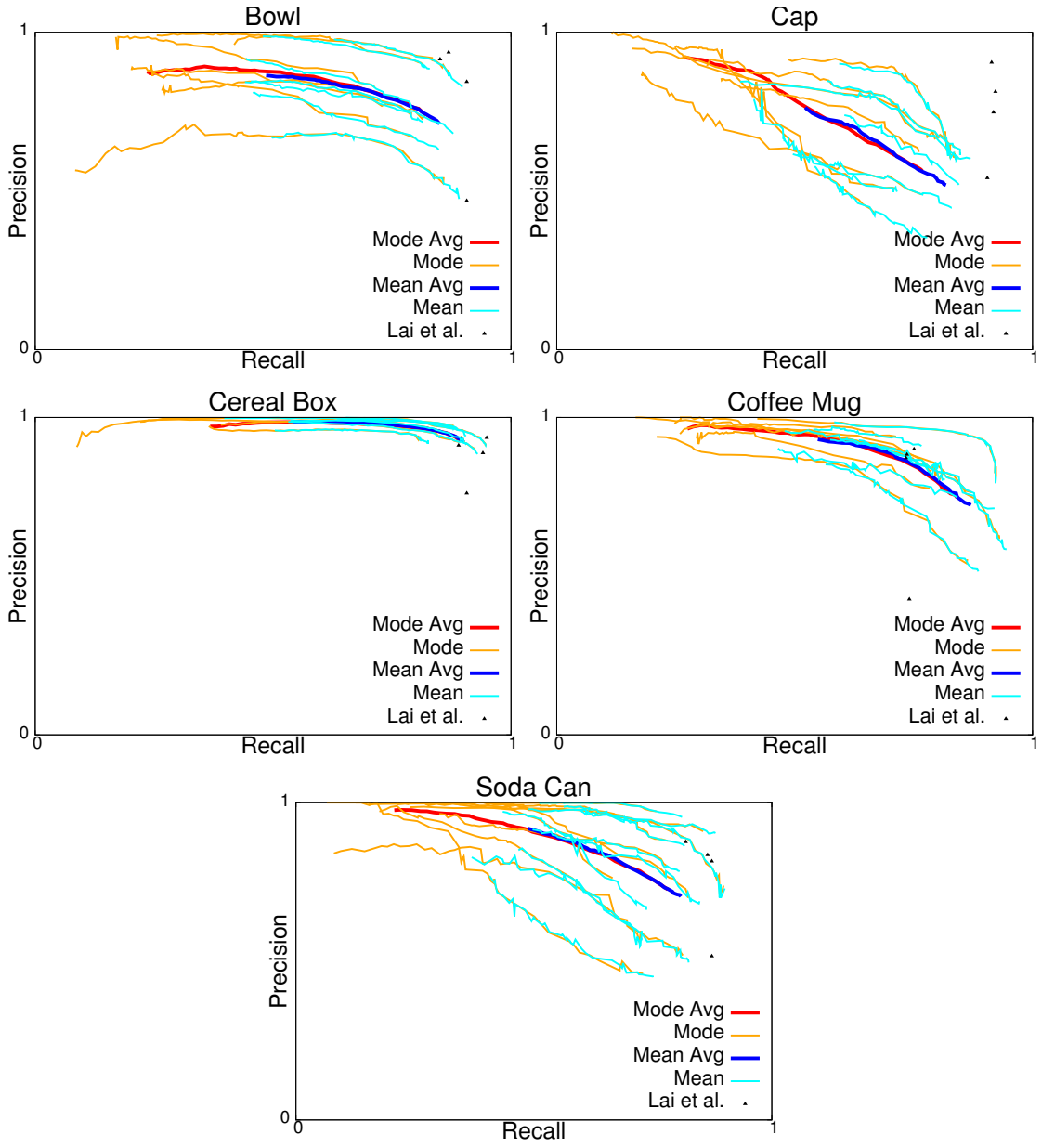

Figure 2.6: Precision-recall curves for object reconstruction using both the Mode and Mean versions of outlier removal. Class-wise results are averaged in the thick lines, but each thin line represents the results for a particular object instance.

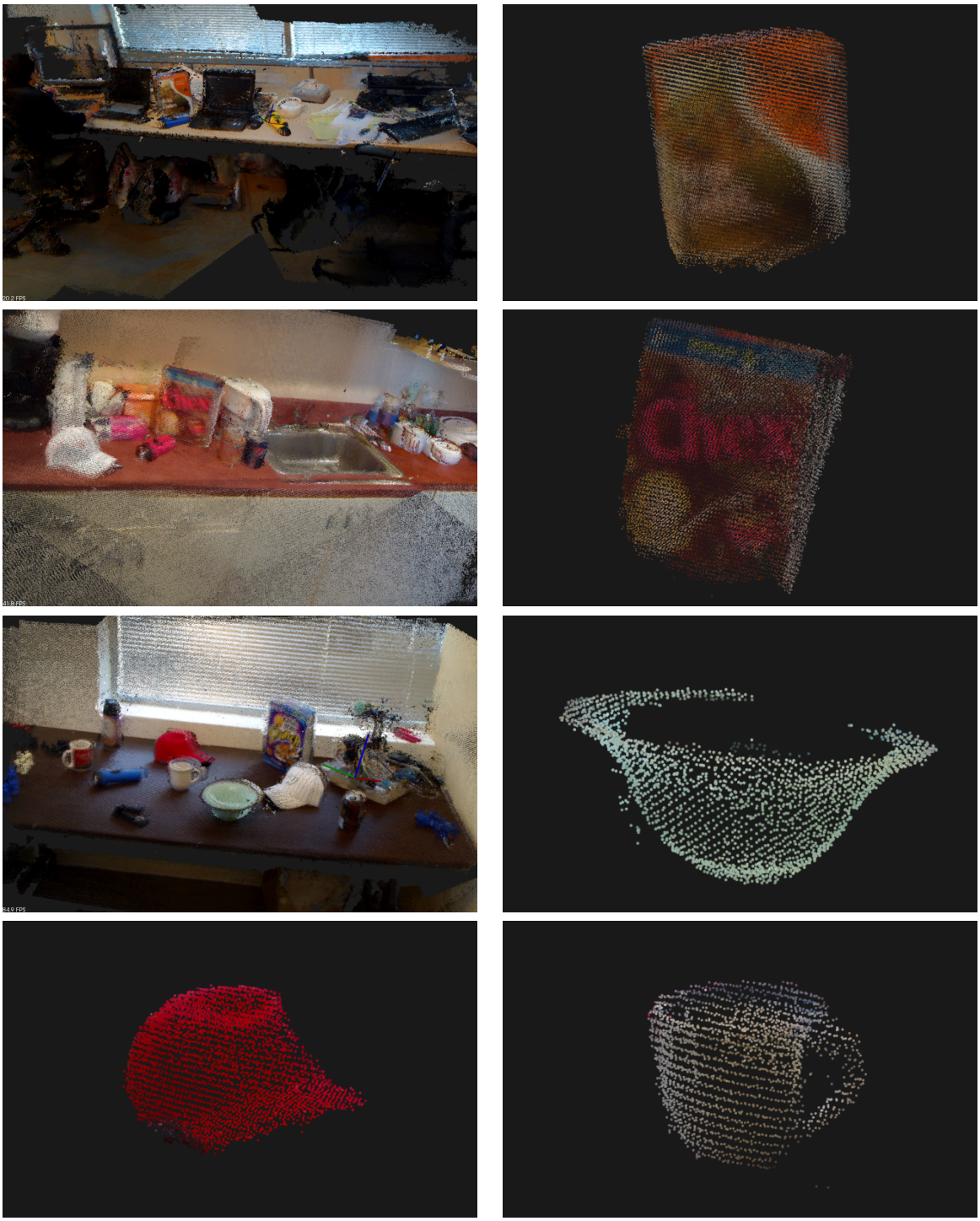

Figure 2.7: Examples of object reconstructions and the scenes from which they came.

our results are comparable, and the 3D reconstructions we produce are accurately localized and robust to outliers. This approach makes minimal assumptions, and can therefore generalize well to arbitrary object classes and instances.

Chapter 3

Parametric Object Modeling

3.1 Introduction

Many objects in a robot's environment are adequately described with qualitative attributes, and their corresponding models can be qualitative in nature as well. As demonstrated in Chapter 2, point cloud models can capture the shape and color properties of many object classes, but a model as simple as a bounding box could describe the location and rough extent of such objects. However, other object classes are best described with quantitative attributes or properties, and their corresponding models should be parametric in nature, such that the system for detection and modeling can estimate those properties from the model. As a case study of this type of object modeling, we propose a system for the quantitative assessment of stairway traversability.

A stairway's presence in a part of the environment carries quite a bit of discriminating power about that space's function, but qualitative attributes of the stairway provide little additional semantic information. However, the quantitative attributes of a stairway permit reasoning about a robot's potential interaction with it: if the steps have appropriate dimensions, some robot platforms may be able to climb them. We therefore focus this chapter on the development of a generative model for a stairway that enables parameter estimation in order to assess traversability, and we evaluate

this approach on the accuracy of the model.

Autonomous ground robots have traditionally been restricted to single floors of a building or outdoor areas free of abrupt elevation changes such as stairs. Although autonomous traversal of stairways is an active research area for some humanoid and ground robots, the focus within the vision and sensor community has been on providing sensor feedback for control of the mechanical aspects of stair traversal, and on stair detection as a trigger for the initiation of autonomous climbing, rather than on stair traversability.

The restriction to a single floor presents a significant limitation to real-world applications such as mapping of multi-floor buildings and rescue scenarios. Our work seeks a solution to this problem and is motivated by the rich potential of an autonomous ground robot that can climb stairs while exploring a multi-floor building.

A comprehensive indoor exploration system could be capable of autonomously exploring an environment that contains stairways, locating them and assessing their traversability, and then engaging a platform-specific climbing routine in order to traverse any climbable stairways to explore other floors. The physical properties of a stairway may limit the platforms that are capable of climbing it. For example, a humanoid robot may not be able to climb some stairways due to step height, and a ground robot may be restricted to stairways with a low pitch due to its weight distribution. In addition to its relevance to this project as an example of parametric object modeling, this proposed approach is also an effort to integrate the existing work in autonomous stair climbing with autonomous exploration: a system to detect and localize stairways in the environment during the process of exploration, and model any identified stairways in order to determine if they are traversable by the robot.

With a map of the environment and estimated locations and parameters for the stairways, the robot could plan a path that traverses the stairs in order to explore the frontier at other elevations that were previously inaccessible. For example, a robot could finish mapping the ground floor of a building,

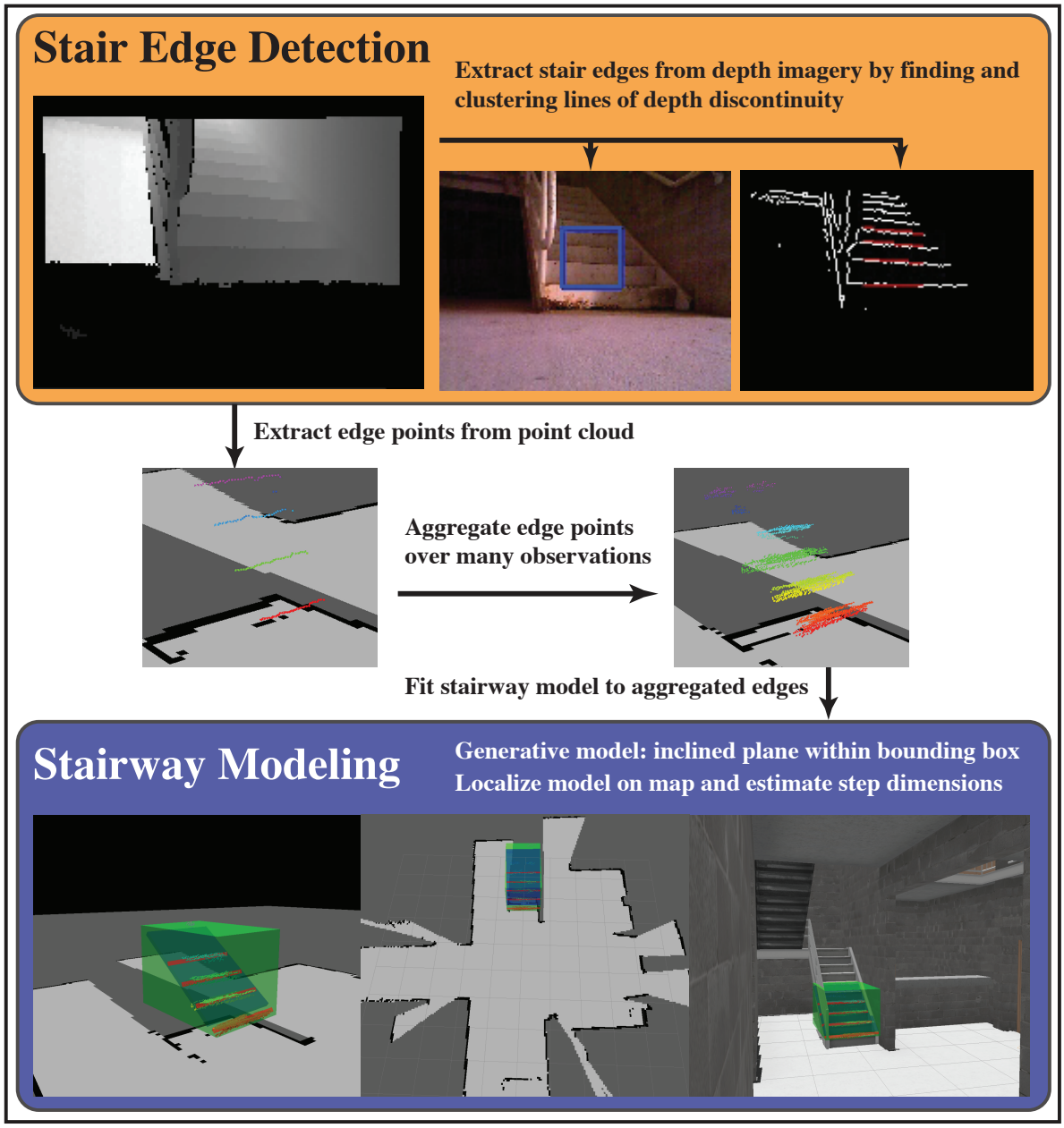

Figure 3.1: High level workflow of the proposed system, consisting of two modules: stair edge detection and stairway modeling. Stair edges are extracted from depth imagery and collected over many observations into an aggregated point cloud. Periodically, a generative model of a stairway is fit to the aggregate cloud and its parameters re-estimated. The result is a model localized with respect to the robot's map of its environment. (Data: Building 1 Front trial of the MOUT dataset)

return to a stairway that it had previously discovered, and ascend to the second floor to continue exploring if that stairway is of dimensions (e.g. step height, width, pitch) that are traversable by that particular platform.

Our proposed system directly addresses the needs of an exploratory platform for solving the problem defined above. We seek to answer the question "Is this traversable?" when a stairway is discovered during exploration. The questions "When should I traverse that?" and "How do I traverse that?" are left for the path planner and stair climbing routines, respectively. The system is composed of a stairway detection module for extracting stair edge points in 3D from depth imagery and a stairway modeling module that aggregates many such detections into a single point cloud from which the stairway's dimensions and location are estimated (see Fig. 3.1).

Our detection algorithm leverages the property of range discontinuity that step edges exhibit in the depth field. Modeling the stairway over many detections allows the system to form a complete model from many partial observations. We model the stairway as a single object: an inclined plane constrained by a bounding box, with step edges lying in the plane. As new observations are added to the aggregated point cloud, the model is re-estimated, outliers are removed, and well-supported stair edges are used to infer the dimensions of each step. These physical properties can then be used by a path planner to determine the traversability of stairs in relation to the specific robotic platform. For example, the humanoid robot in [82] and the hybrid claw-wheel robot in [83] are both capable of climbing stairs, but they require different constraints on the sizes of the steps they are able to traverse. Section 3.2 contains a more in-depth description of our approach. Autonomous multi-floor exploration is a new behavior for ground robots, and we present this work as a first step toward the realization of that capability.

3.1.1 Related Work

Other systems have been proposed for related tasks, including the actual platform-specific autonomous climbing procedure, but no existing work approaches the problem in the context of the aforementioned scenario. The problem we are considering is the evaluation of stairways as traversable terrain for path planning, and as such, we aim to localize and then estimate the physical properties of a stairway to determine if it is traversable. A path planning algorithm such as [84] could incorporate these stairway properties and the robot's climbing capabilities into its decisions about obstacle traversal. Previous measures of stair traversability [85] considered stairways as generic obstacles and only evaluated the height of the first step to determine whether to attempt traversal. Unlike that system, which does not differentiate individual obstacles and groups of stairs, we attach the semantic concept of *stairway* to the obstacle because it likely leads to unexplored areas. Traversal of an *obstacle* does not have such an association, and therefore it is important, from a path planning perspective, to know both the class of the object and its traversability.

Several existing methods [86–88] perform stairway detection but immediately initialize a traversal procedure with their respective platforms, which is not necessarily desirable in an exploration scenario. These works assume that the robot is located near the stairway, but not aligned with it. As such, these methods do autonomous exploration until they detect the stairway, and only serve to trigger the autonomous traversal phase of their systems. Rather than model the stairway or assess the traversability, they provide only a bearing, and in some cases a distance, to the stairway relative to the robot's pose, in order to facilitate alignment and subsequent climbing. Although these capabilities are related to our problem scenario, immediate climbing is not necessarily compatible with exploration. Path planning for multi-floor exploration should take the stairway into account as a portal to more unexplored regions, but traversing stairs immediately upon a single detection makes exploring the low-cost frontiers of the original floor more difficult and may fail if the detection was erroneous.

The works by Hernandez and Jo [89] and Hernandez et al. [90] represent the most similar approaches to ours in terms of the goal of detecting and localizing sets of stairs, but are independent of modeling or mapping. In [89], they segment outdoor staircases from single monocular images using line detection and vanishing point analysis, and in [90] they use some of the same line techniques (Gabor filtering) along with motion stereo to detect and localize indoor stairways. However, the scope of the works are similarly limited to detection and a computation of bearing relative to the robot's pose.

Our proposed system overcomes these limitations by modeling the stairway and anchoring the model to a simultaneously constructed map. Since immediate climbing is not necessary, the stairway can be considered as traversable terrain for path planning, and these same climbing procedures can be initiated at a later time when the robot's path requires traversal of the stairway.

A number of existing approaches perform modeling of individual steps or sets of stairs. However, these methods produce fine-grained models for humanoid robot stair climbing [82, 91], or for obstacle negotiation for the visually impaired [92]. Another method proposes a minimal inclined plane model, but uses it to produce a 3D reconstruction for robot obstacle detection [93], and does not localize the model with respect to a map nor estimate its physical parameters. However, these detailed models and 3D reconstructions are not mutually exclusive of our proposed approach. Since precise step locations are often needed for stair climbing (for humanoid robots, for example), these models could always be produced once a path planner has decided to climb a traversable stairway.

In [91], Oßwald et al. assemble 3D point clouds by tilting a 2D ladar mounted on a humanoid robot's head, and then extract planes for the steps and risers, and estimate the average step dimensions. However, this modeling is done while the robot is already close to and manually aligned with the stairway and is repeated periodically during its ascent. Our approach, on the other hand, observes the stairs passively during exploration. For those platforms that require a more detailed

plane-based model, the approach in [91] could always be performed while the robot climbs the stairway. This is the only other approach that performs parameter estimation for step dimensions, and in Sec. 3.3 we demonstrate comparable accuracy with our minimal stairway model. However, we produce our results passively, at a distance, and without explicit alignment to the stairway.

The aforementioned climbing methods use edge detection from camera imagery [88], and range discontinuity detection from horizontal ladar [86] or vertical ladar [87] data, to detect stairs. Other stair edge detection systems have been proposed in the context of controller feedback for stair traversal [94, 95] and object detection from monocular [96] or stereo imagery [93]. Our range discontinuity-based detector incorporates line extraction ideas from many of these systems, but we apply these techniques to dense depth data provided by an RGB-D camera. The registered point cloud that is output from these cameras alongside the depth image allows for extraction of 3D data corresponding to step edges, and enables our modeling approach.

Some recent work in multi-floor mapping may provide some of the tools for implementing our desired comprehensive system. Shen et al. [97] have demonstrated that multi-floor exploration is possible in open indoor environments with an unmanned aerial vehicle. Although their platform by nature avoids the need for stair detection and traversal, their approach for mapping may one day be applicable for ground vehicles. The barometric method presented by Ozkil et al. in [98] for measuring elevation, and therefore distinguishing floors of a building, would likely also be useful in implementing a multi-floor exploratory system.

The most comprehensive system yet presented is also one that aims to perform the complementary task to our detection and localization of ascending stairways: detection and traversal of descending stairways. Hesch et al. [99] use a combination of texture, optical flow, and geometry from a monocular camera to detect candidate descending stairwells, navigate to them, and then align with and traverse them. Although they do not perform any explicit mapping of stairway location or present quantitative accuracy results, their implementation runs in real time and the detector mod-

ule from their implementation could be extracted and paired with our ascending stair detector in a comprehensive multi-floor mapping system. No system has yet been proposed for both ascending and descending stairway detection and traversal.

3.1.2 Contributions

We have deployed our stairway detection and modeling system on an iRobot PackBot as well as a Turtlebot, both fitted with Microsoft Kinect depth sensors. In principle, this modeling system could be deployed on any platform with an RGB-D sensor, such as our Turtlebot, for stairway detection and traversability analysis. For stair climbing robots, such as the PackBot or Aldebaran Nao, this system can potentially be paired with more fine-grained and platform specific modeling approaches for facilitating the act of stair climbing. Our system runs in real time and demonstrates robust and accurate performance in both localization and parameter estimation for a wide variety of stairways (see Sec. 3.3).

This paper presents the following contributions:

- Initial step toward new ground robot behavior: autonomous multi-floor exploration. Locate stairways during mapping of environment, assess their traversability, and later ascend them to explore new frontiers.
- A minimalist generative stairway model: an inclined plane constrained within a bounding box. This provides enough detail to determine if the stairway is traversable by the robot, and if necessary, more detailed modeling can be performed in the context of subsequent stair traversal.
- Aggregation of many partial views into a coherent object model. Re-estimation and outlier removal permits estimation of a robust aggregate model in the presence of imprecise alignment for each observation.

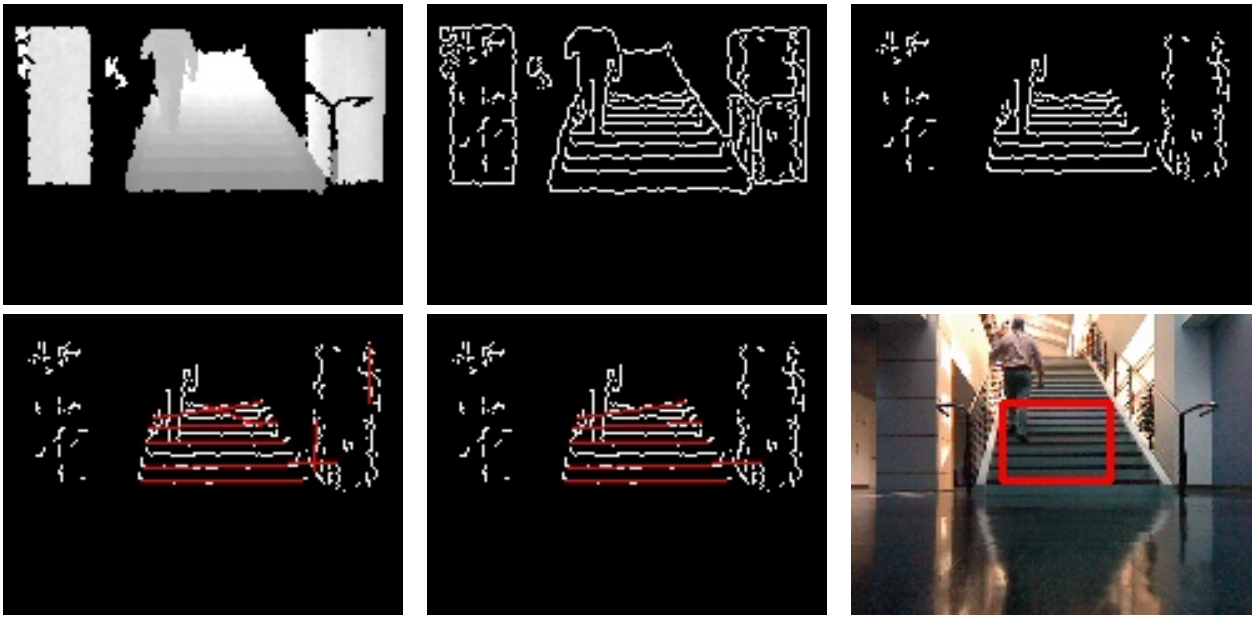

Figure 3.2: An example frame from an indoor testing video. Top row (L to R): source depth image, edge image, edge image with boundary lines removed. Bottom row (L to R): candidate lines (in red) before filtering for orientation and clustering, candidate lines after filtering, marked up camera image with bounding box.

3.2 Methods

3.2.1 Stair Edge Detection

Inspired by some of the techniques used in other methods [82, 87–89, 93–95], we have developed an ascending stairway detector that exploits the geometric properties that steps display in depth images. On a deployed system, it runs in real time with high accuracy and robustness. In particular, we find lines in a depth image that represent discontinuities where the depth from the camera changes abruptly. In the depth field, a set of stairs will have a discontinuity at the edge of each step that is above the height of the sensor. The tops of lower steps will be visible in the sensor's field of view, and may not exhibit a strong enough depth discontinuity to be detected as edges. Regardless of the horizontal rotation of the camera relative to the stairs, these discontinuities will form a set of

nearly parallel lines (with some perspective foreshortening effects) for all but tight spiral staircases. We leverage this distinct depth signature by detecting all such lines of discontinuity in the image, filtering and clustering them to find a near-parallel set, and ultimately fitting a plane to the extracted stair edge points to confirm or reject the stairway candidate hypothesis if they lie on an inclined plane of traversable pitch. By detecting these lines of discontinuity in the depth field rather than a camera image, our detector is robust to appearance.

Given an input depth image, our algorithm proceeds following the steps in Alg. 3.1, further details of which can be found in [100]. Please refer to Fig. 3.2 for visual reference. We enforce several physical constraints to restrict the lines of range discontinuity to stair edges, and extract the 3D points that correspond to the edges that satisfy all of them. We additionally fit a planar model to each set of extracted points, and only confirm a positive detection if the plane is at a traversable angle. Observations that pass these tests are then provided to the stairway modeler.

Our depth image and point cloud based approach was motivated by our ultimate goal of 3D modeling. However, many of the existing line-based stair detection methods could be modified to produce point cloud observations of extracted step edges if they operated on data from an RGB-D camera, and could in principle provide the observations for our modeler in place of this detector. Since this chapter is focused on modeling and parameter estimation, and not on stair detection, we do not fully evaluate our detector against these other approaches here.

3.2.2 Stairway Model

We propose a generative model to represent a stairway as a single object. For localization and path planning purposes, piecewise planar models provide more detail than necessary for traversability analysis, when the questions "Are these steps too tall?" or "Are these stairs too steep?" can be answered with a simpler model. Although more detail does not detract from the model, the need

Algorithm 3.1 Stair edge detection

1: Input: depth image D and co-registered point cloud P provided by depth sensor
2: Do Canny Edge Detection on D to produce edge image E
3: Remove boundary edges from E (bordering 0 valued depth)
4: Generate a set of candidate lines L using the Probabilistic Hough Transform on E
5: Merge collinear lines and compute slope histogram for L
6: Extract lines in the bin with largest frequency into L'
7: Compute bounding box B for maximal set of vertically overlapping lines in L'
8: Remove all lines from L' that do not fall within B
9: Reject if $\mid L' \mid < 3$ (enforce multiple steps)
10: Extract from P the points on the lines in L' into P'
11: Fit a least squares plane p to the points from P'
12: Reject if dihedral angle ($\phi = \arccos(n_p \cdot n_{horiz})$) from the horizontal is $>$ the robot's maximum climbable stair pitch
13: Return P'

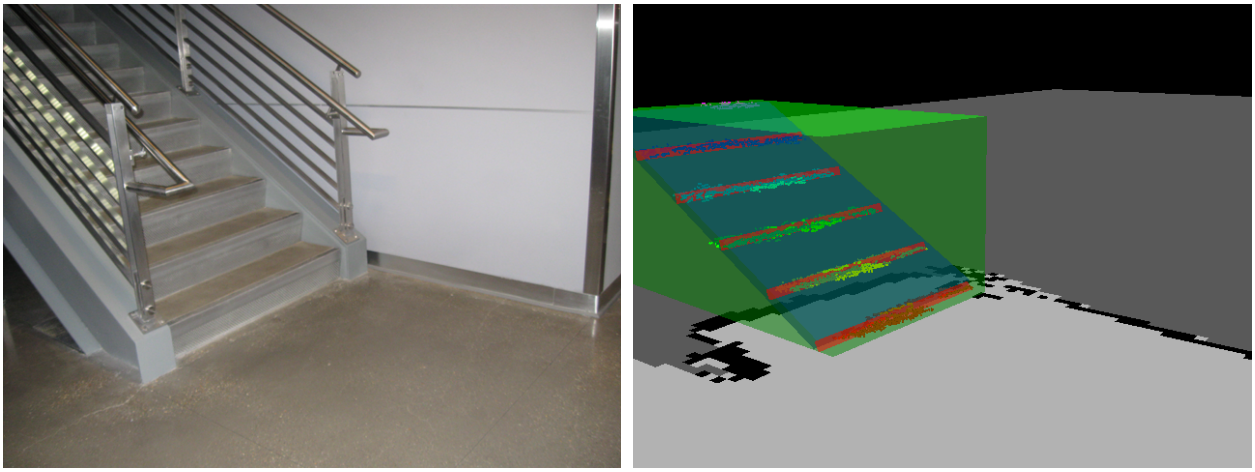

Figure 3.3: Stairway with corresponding model consisting of bounding box (green), planar model (blue), and step edges (red), as well as edge point cloud support for step edge lines (rainbow). (Data: Davis Hall Front trial from UB dataset)

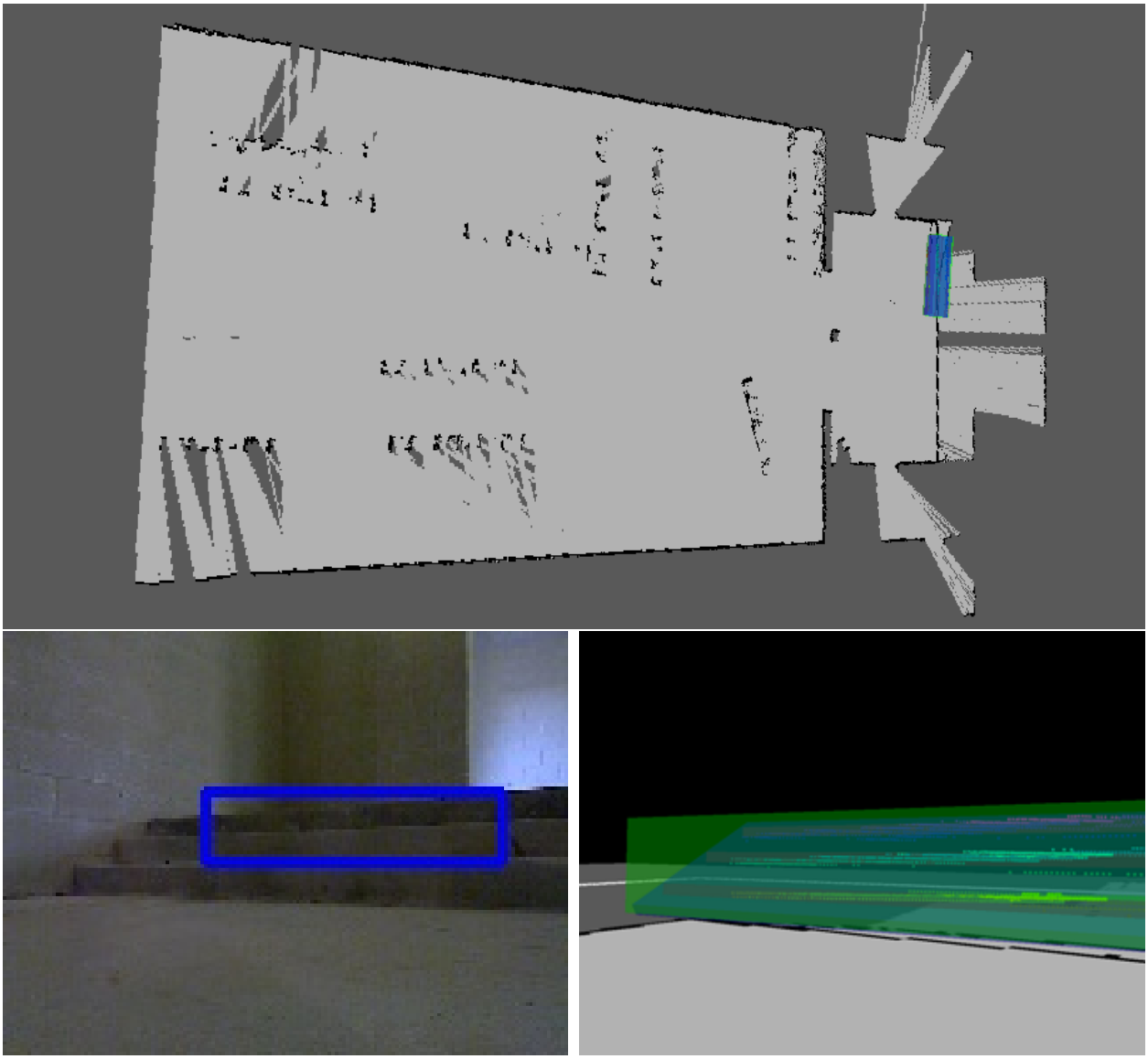

Figure 3.4: Figure showing a map marked up with a marker for the model of the detected stairway, in addition to a camera image and model view from the robot's perspective. (Data: Building 7 Interior trial from MOUT dataset)

for close proximity and alignment with the stairway, as in [91], limits the effectiveness of other approaches for this purpose, even if the computational cost is not restrictive. We instead aim to enable passive sensing of the stairway from a distance, such that modeling can be performed concurrently with exploration. Localization on a map should enable the robot to return to ascend the stairs at a later time if they are determined to be traversable.

Our model consists of an inclined plane constrained by a bounding box, with stair edges wherever there are well-supported clusters in the plane (see Fig. 3.3). This model is parameterized by the bounding box centroid (B_x,B_y,B_z) and dimensions (H,W,D), pitch relative to the ground plane (P), and step dimensions (h,d). We assume that stair steps are approximately parallel to the ground plane, so the bounding box top and bottom are parallel to the XY plane. For an inclined plane model of:

$$ax + by + cz + d = 0 \tag{3.1}$$

the planes constituting the bounding box are given by:

$$z = B_z \pm \frac{H}{2} \tag{3.2}$$

$$a(x - B_x) + b(y - B_y) \pm \frac{D}{2}\left(\sqrt{a^2 + b^2}\right) = 0 \tag{3.3}$$

$$-cb(x - B_x) + ca(y - B_y) \pm \frac{W}{2}\left(c\sqrt{a^2 + b^2}\right) = 0 \tag{3.4}$$

for the top/bottom, front/back, and sides, respectively. Here, the pitch P is computed as the dihedral angle of the planar model and the ground plane: $P = \arccos\left([a,b,c]\,[0,0,1]^T\right)$. We infer the parameters of the model from the extracted points corresponding to step edges.

3.2.3 Localization and Modeling

In order to build up a complete model of a stairway, we piece together many incomplete views, potentially from many different perspectives, and estimate the parameters of the model from the aggregate pool of data. Our modeling system is capable of modeling multiple stairways in the same environment. Input observations are assigned to stairway models based on proximity; if an observation's centroid is within $2m$ of the centroid of an existing model, it is added to that model, otherwise a new model is spawned. In practice, this distance threshold is adequate for differentiating most real world stairways. However, a more sophisticated approach to the assignment of observations, and a thorough evaluation of the modeling of multiple stairways is left for future work on autonomous stairway discovery. Algorithm 3.2 details the steps in parameter estimation for each model in the environment.

Starting with an empty point cloud representing the stair edges for a new model, we add to it the extracted edge points from each subsequent observation. We do not explicitly align the detected edges, but instead rely on the robot's estimated pose to approximately align the independent observations, and implement a number of statistical techniques to ensure that the resulting model is robust to outliers and imprecise point cloud alignment. Ultimately, the quality of the model will depend on the quality of the robot's estimated pose, but individual false positive detections or misaligned observations are removed as statistical outliers, as detailed below. However, since we consider the task of stairway modeling in the context of exploration, the robot's performance at both map building and stair detection will be dependent on the quality of its odometry.

Since each observation only adds a partial view of the stairway, we periodically re-estimate the parameters of the model (in our experiments, after k = 5 or 10 observations). When deployed on the PackBot and during post-processing of recorded data on a Mac Mini, our detector operates at over $20Hz$ on average, including the time to fit the model (compared to the Kinect's $30Hz$ frame

rate). Although more frequent modeling is possible, we expect the model's parameters to converge over many observations, so we do not anticipate a need for more frequent updates if the information is to be used for traversability analysis. We perform the following steps in order to estimate the model's parameters.

To prevent our aggregate edge point cloud E from growing without bound, we first downsample E to a 1cm voxel grid. We perform statistical outlier removal using the algorithm in [30] in order to reduce sensor noise in the extracted points in E. To the remaining points we fit a planar model p with RANSAC [101] and remove any outliers from the model from E.

We then infer the parameters of the stairway model from the remaining points in E. We determine the bounding box centroid and dimensions by fitting a rectangular prism to the data that is aligned with the ground plane but rotated in the XY plane to match the alignment of p. We next compute the cross-sectional orthogonal plane that passes through (B_x,B_y,B_z) and project the points of E to it. When accounting for alignment errors and unequal observation of each step, we would expect there to be a cluster of projected points around each step edge. We therefore find Euclidean Clusters on the projected plane using the Point Cloud Library's Segmentation Module [102] and treat each well-supported cluster center ($n > 250$ points) as a stair edge. We compute the differences in height and depth between each pair of adjacent cluster centers, and then average these differences to determine the step dimensions (h and d).

3.3 Experiments

Our system has been tested on data collected at a Military Operations in Urban Terrain (MOUT) site, on all of the available stairway types and on numerous negative examples. It has also been tested at a building at the SUNY at Buffalo (UB) campus. These datasets consist of 5 recorded trials (3 and 2, respectively) with ground truth stairway dimensions. The set of trials included stairways

Algorithm 3.2 Stairway modeling

Initialize point cloud E to be empty for model M_i
Define the modeling period k: number of observations between model fittings
for Each detection P' (see Alg. 3.1) that is assigned to M_i **do**
    Add points from P' to E
    if # of detections is divisible by k **then**
        Downsample E to fine voxel grid
        Perform statistical outlier removal (as in [30])
        Fit a plane m to E using RANSAC and compute pitch relative to ground plane
        Remove outliers of m from E
        Fit bounding box B to E and compute stairway dimensions (H,W,D)
        Project E onto cross-sectional plane x, orthogonal to m and passing through bounding box centroid
        Find Euclidean Clusters C from projected cloud E_p and compute their centroids
        Sort C by ascending height, and compute differences in height and depth between adjacent centroids with $> n$ points of support
        Average height and depth differences to compute step dimensions (h, d)
    end if
end for

of a variety of step sizes and building materials (metal, concrete, etc.), ranged from a few steps to a full flight, and included one outdoor stairway. In each case, the robot was teleoperated around an environment, observing both the stairway and its surroundings from a variety of perspectives. Development of techniques for integrating this modeling approach into autonomous exploration is ongoing, but the scope of this study is limited to an evaluation of modeling performance.

Our experiments use an iRobot PackBot mounted with a Microsoft Kinect depth sensor for the MOUT trials, and a Turtlebot (also with a Kinect sensor) for the UB data. Our system is implemented in C++ in the Robot Operating System (ROS) environment, with image processing performed using OpenCV, and point cloud processing with the Point Cloud Library (PCL) [102]. Although the Kinect restricts the usable range of the detector and limits outdoor use to shaded areas, the dense depth image it produces provides high quality input data for our system. The outdoor data captured at the MOUT site indicated good performance with even somewhat compromised depth data. In principle, our approach could be applied to dense stereo depth data, with appropriate adjustments to the parameters of the algorithm, but this is as yet untested.

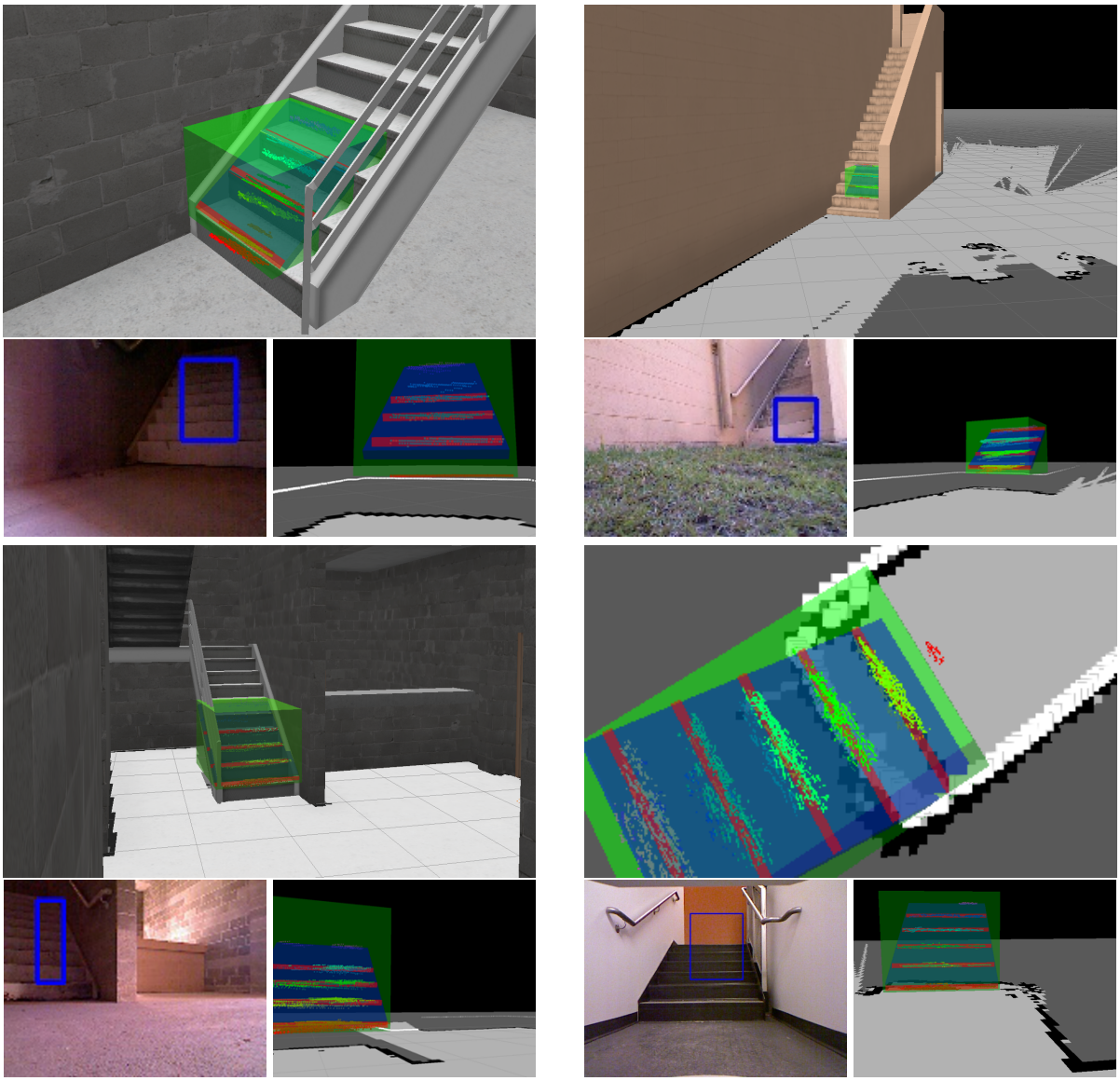

Figure 3.5: Results of several runs from our datasets: Building 1 Rear (top left), Building 3 (top right), Building 1 Front (bottom left), and Davis Hall Rear (bottom right). Model components shown are the bounding box (green), planar model (blue), and step edges (red), as well as edge point cloud support for step edge lines (rainbow).

Table 3.1: Table of model step estimates and **ground truth values (GT)** (in *cm*)

Trial	**Height**	**Depth**	**Width**	**Pitch** (°)
B.1 Front	17.4	25.7	97.7	34.0
GT	**19.6**	**25.4**	**96.5**	**37.6**
B.1 Rear	16.6	25.2	71.5	33.0
GT	**19.6**	**25.4**	**96.5**	**37.6**
B.3	16.9	24.5	68.9	35.4
GT	**19.2**	**26.3**	**101.5**	**36.2**
Davis Front	17.5	32.3	107.6	28.3
GT	**18.1**	**30.5**	**122.6**	**30.7**
Davis Rear	16.9	31.1	104.8	29.4
GT	**16.5**	**29.2**	**117.5**	**29.5**
Mean Error	1.7	1.2	17.3	2.3
Std. Dev.	1.4	1.6	12.8	1.88

Visual results of modeling for all trials in the two datasets can be found throughout this chapter in Figs. 3.1, 3.3, and 3.5. Where possible, rendered 3D models of the corresponding buildings were superimposed and aligned with the map such that the stairway model is overlaid. In each image of a model, the bounding box is represented in green, the planar model in blue, and any step edges in red.

3.3.1 Modeling Accuracy

We measured ground truth step dimensions and pitch for our trials, and we present those results in Table 3.1. Each of these results was achieved with < 100 observations. The model estimates for step dimensions are accurate to within $2cm$ and the pitch to within $3°$, on average. However, one frequent source of inaccuracy is underestimation of the step width, with a mean error of $17cm$. This is expected, though, based on the stair detection procedure, which only extracts edge points in a horizontal window of the depth image where all of the edge lines overlap, leading to observations that are always narrower than the lines producing them, especially for the lower steps (see Fig. 3.2). Each trial's results indicate sufficient accuracy for a robotic platform to assess whether that

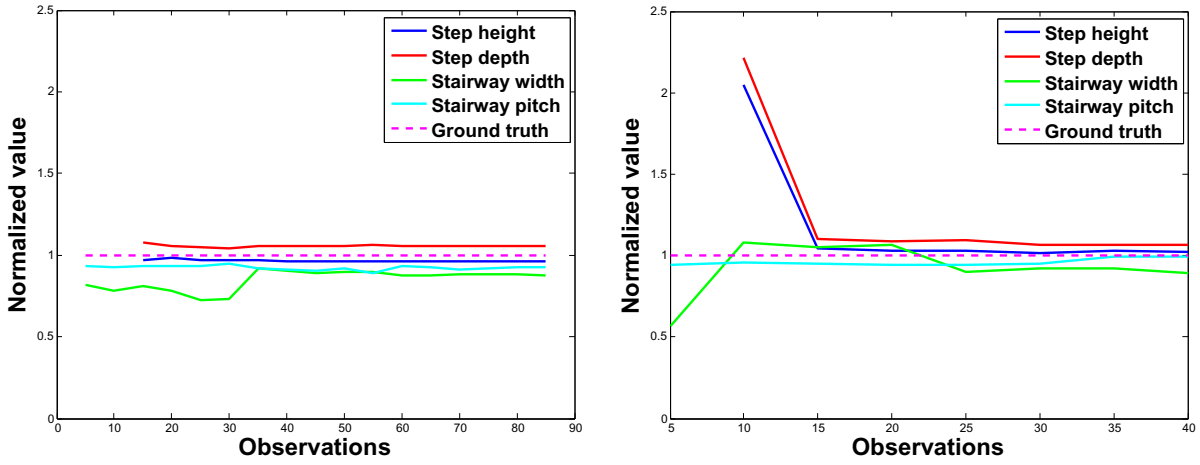

Figure 3.6: Convergence of normalized model parameters for Davis Front (top) and Davis Rear (bottom) trials from the UB dataset.

stairway's physical dimensions would be traversable.

We also present some results showing the convergence of the models for several trials. Fig. 3.6 shows the evolution of the model parameters over time for the two UB trials. Here, all of the parameters are normalized by their ground truth values, so each quantity should tend toward 1 over time. Both trials indicate that after a small number of detections, the models approach their final state.

3.3.2 Comparison

The only other stairway modeling approach to present quantitative results on parameter estimation is [91]. In this work, two plane fitting algorithms are implemented (Scan-Line Grouping and Two-Point Random Sampling) for modeling stairways from point clouds for humanoid robot climbing. For their trials, the robot is aligned at a distance of 70cm from the base of the stairs when it captures a point cloud and fits a stairway model using one of the two algorithms, in the end estimating its step dimensions. The procedure is then repeated with the robot on odd-numbered steps as it climbs. Each point cloud was acquired by nodding a head-mounted 2D laser scanner to produce a 3D point

cloud with approximately 130,000 points.

We compare the mean modeling errors for our trials with the results in [91], using the respective datasets. Neither the data nor the source code for their system is publicly available, so a more direct comparison using a common dataset is not possible. Additionally, the input data for the two approaches are of different modalities. A tabulated error comparison of their two methods with ours can be found in Table 3.2. Since the stairway being modeled in [91] is specially designed so a small humanoid robot can climb it, its steps are of significantly smaller size than a regular stairway that adheres to building codes. Our trials were conducted on five distinct full-size stairways, so we give percent error for comparison for the step height, depth, and width dimensions, and for our data it is averaged over our five trials. However, since these datasets are different, some visual characteristic idiosyncrasies may render the numbers not in perfect correspondence.

In step height and depth, our results are comparable, if not marginally better. The error for step width, however is $3-4$ times higher for our method. This is due to the nature of our detector's rectangular bounding boxes. In practice, unless the robotic platform is wider than a human being, the width of the step is less restrictive to its traversability than the step height or pitch, and for robotic traversal it safer to underestimate this parameter.

In their plane-based methods, Oßwald et al. [91] measure angular errors by deviations for planes that are supposed to be parallel and those that are supposed to be perpendicular. However, our planar model measure pitch relative to the ground plane. Both types of angular errors have been presented in Table 3.2. Although they are measures of different model properties, the angular errors for all three methods are comparable as well. The Scan-Line Grouping algorithm runs at approximately $40Hz$ and the Two-Point Random Sampling method at $0.32Hz$, compared to our method at $20Hz$ (concurrent with mapping).

Table 3.2: Table of step modeling errors for this method and the Scan-Line Grouping (SLG) and Two-Point Random Sampling (TPRS) methods in [91] (avg $\pm$ std)

Quantity	**This Method**	**SLG**	**TPRS**
Height Error (cm)	1.7 ± 1.4	0.42 ± 0.31	0.68 ± 0.54
Percent	8.9	6.0	9.7
Depth Error (cm)	1.2 ± 1.6	1.17 ± 0.67	0.90 ± 0.61
Percent	4.2	6.5	5.0
Width Error (cm)	17.3 ± 12.8	3.40 ± 1.95	2.25 ± 1.97
Percent	16.5	5.7	3.8
Pitch Error	2.3 ± 1.9		
Plane Error (parallel)		2.22 ± 2.17	1.14 ± 1.13
Plane Error (90°)		4.97 ± 2.13	3.12 ± 1.47

3.4 Conclusions

We present a novel, minimal, generative model for a set of stairs, as well as a system for fitting that model to data extracted and aggregated from many observations of a stairways with an RGB-D sensor. Our model is sufficiently detailed to permit the robot to determine the traversability of a set of stairs, while simple enough to be computed in real time and robust to errors. Providing the observations for the modeling module is a stair detector that uses image processing techniques to find lines of depth discontinuity and enforce geometric constraints on them in order to extract the points on just the lines corresponding to stair edges. We have tested our system on a variety of stairways in both indoor and outdoor environments, and we are able to achieve high accuracy in estimation of a stairway's physical parameters. Our results from passive sensing during exploration are comparable to more detailed models that require initial alignment with the stairway. Thus, our approach can serve to assess stairways that are discovered while a robot is exploring a new environment before such detailed models are used to facilitate stair climbing by the robot.

Ultimately, we want this work to enable a new robot behavior: fully autonomous multi-floor exploration by ground robots. With the localization and modeling system presented here, we aim to make some advancement toward that goal. Other problems that would still need to be solved

include incorporation of elevation measurements into both mapping and exploration algorithms, execution of an autonomous stair climbing routine after a stairway is found, and modification of path planning algorithms to set stair traversal as a path with high, but finite, cost. In addition to serving as a case study in parametric object modeling in a way that would be compatible with the proposed Attributed Object Map framework, this work represents an initial step toward autonomous multi-floor exploration by unmanned ground robots.

Chapter 4

Semantic Attributes

4.1 Overview

A recent shift in computer vision research has been from the detection of objects in natural images to the description of those objects with semantic attributes. This interest in object description has manifested itself as methods for class or category inference [10, 54], quantitative measurement of attribute presentation [55], and localization of those attributes to specific regions of the object [58], among a host of other applications.

Within the scope of this project, we do not concern ourselves with inference using object attributes, but instead consider just the attribute classification. The following is a sampling of the many classifier types applied to the object description problem. Farhadi et al. [10] use L1-regularized logistic regression to fit a linear model to a vector of image features. This approach avoids overfitting the data by penalizing the complexity of the model and selecting a sparse set of non-zero weights for the feature vector entries [103]. Parikh and Grauman [55] learn a wide-margin ranking function that similar to an SVM but with pairwise constraints that enforce the relative nature of the attributes they model. Sun et al. [60] implement unsupervised feature learning with K-SVD [104] and learn sparse codebooks from RGB-D data. Their attribute classifiers are linear decision functions over

sparse code features pooled in spatial cells of the image. Mason and Marthi [37], compute attribute-like features of objects from a segmented point cloud rather than imagery. In particular, they measure the dominant color, approximate size, and curvature from the points in the point cloud.

We utilize the classification approach presented in [10] for computing attribute scores because Matlab code for their system is publicly available, and allowed for adaptation to our system. The RGB-D approach in [60] may be a more natural fit given our input data, and might provide better results, but this work was not published until after our attributes module was already implemented. We compute visual features for an input image, including color, texture and histogram of oriented gradients (HOG). For any object detection bounding boxes on the image, we assemble a feature vector from the bounding box region of interest and compute a classification score using an attribute's linear model. This score can then be converted to a probability of that attribute's presence in the bounding box with the logistic function. Unlike previous work in describing objects, a primary feature of our system is the multi-view nature of the object modeling process, so we combine the attributes from individual images in a descriptor that computes aggregated attributes for an object model (see Section 4.3).

4.2 Implementation

The approach taken by Farhadi et al. [10] is to compute a large set of features on an input image including color, texture, and histogram of oriented gradients, then compute histograms for each feature type by quantizing the features to learned k-means centers and then normalizing them. The resultant feature vector is input to a feature selection process that fits an L1-regularized logistic regression for each attribute within each class. The selected features are pooled to learn a single classifier for the attribute over all object classes. Attribute classification proceeds by computing a feature vector for an object bounding box in an input image and computing the probability of an

attribute's presence from its logistic regression model (see Section 4.2.2).

Our implementation was adapted from public code released by the authors of [10] in order to enable real-time, deployed attribute classification. The original code is in Matlab, with several C and C++ functions utilized through the MEX interface. We ported the full system to C++, incorporating the attribute classification code into a ROS node (further details in Section 5.1). We rely heavily on the OpenCV library [105] for image processing functions, parallelize the feature and histogram computation using OpenMP [106], and improve the efficiency of multi-attribute classification by using optimized matrix operations (see Section 4.2.2. Our resulting attribute subsystem operates efficiently enough to enable real-time performance from the full system, which is performing detection, attribute classification, and modeling.

4.2.1 Features

The feature vector used in [10] is a set of concatenated, normalized histograms. The procedure we describe here is a slight deviation from the method presented in this paper; we adapt their public code, and the system they have released is not consistent with their proposed one, since it omits an edge detection histogram as a component of the feature vector. Algorithm 4.1 summarizes the following steps, which describe the construction of a feature vector for each detected object in an input image. First, features are computed for the whole image, and then subsets of these features corresponding to the detection bounding boxes are eventually summarized in the histograms.

RGB images are converted to LAB color space, and the raw values form points in $\mathbb{R}^3$. A texture descriptor is computed for each pixel, using the approach described in [107]. This descriptor describes the texture at each pixel with a 4-dimensional vector. Lastly, Histogram of Oriented Gradients (HOG) descriptors are computed for the image, according to the Dalal and Triggs method [108]. This descriptor is a 576 element vector for each 8×8 pixel block.

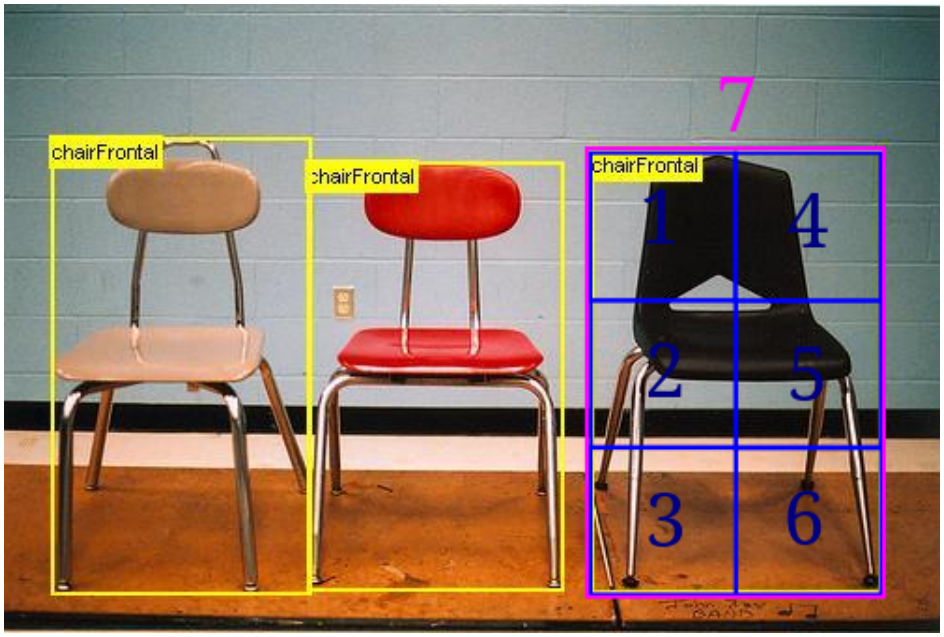

Figure 4.1: Example object detection showing decomposition of bounding box into 7 subregions for feature summarization. Separate histograms for each numbered rectangle are computed for color, texture, and HOG features, and then concatenated to form the final feature vector.

Each of these points in $\mathbb{R}^n$ for color, texture, and HOG features (where n is 3, 4, and 576, respectively), is quantized to the nearest k-means center, selected from a set trained on the whole of the training data. For color they use 128 centers, for texture 256, and for HOG 1000. The color histogram is computed using all of the pixel color values that are within the object bounding box, and then it is normalized over its 128 entries, with the texture histogram computed similarly. The HOG histogram is computed using all of the HOG blocks that lie within the object bounding box, since we do not have pixel-level features.

This histogramming procedure is repeated for several subregions of the bounding box. The full bounding box rectangle is broken into a 3 x 2 grid of subimages (see Figure 4.1) and color, texture, and HOG histograms are computed for the pixel or block values lying within each. Ultimately, each object detection has seven normalized histograms each for color, texture, and HOG. These histograms are concatenated, first within each feature type in the sequential order indicated in

Figure 4.1, and then all together to form a 9688-element feature vector:

$$[\ \textbf{color:}\quad 7\times 128 = 896\ \mid\ \textbf{texture:}\quad 7\times 256 = 1792\ \mid\ \textbf{HOG:}\quad 7\times 1000 = 7000\ ]$$

One feature vector is computed for each object detection bounding box in the training data and when performing attribute classification.

4.2.2 Classifier

The classifier type used in [10] is L1-regularized logistic regression. This approach seeks to learn a set of parameters θ, such that

$$p(y = 1|\mathbf{x}, \theta) = \frac{1}{1 + \exp(-\theta^{\mathbf{T}}\mathbf{x})} \tag{4.1}$$

models the conditional probability that given our input feature vector $\mathbf{x}$, the attribute y is present ($y = 1$). Training seeks to optimize the following:

$$\arg\max_{\theta} \sum_{i=1}^{m} \log p(y_i|x_i, \theta) - \alpha R(\theta) \tag{4.2}$$

subject to the regularization term

$$R(\theta) = \|\theta\|_1 = \sum_{i=1}^{n} |\theta_i| \tag{4.3}$$

on the model coefficients, which in this case are the weights on the normalized histogram bins [103]. The vector of weights we wish to learn includes a bias term, such that the input feature vectors have a 1 appended to them, and we add a θ_{bias} to the end of the θ vector.

We use the training method implemented by Farhadi et al. to learn a model for each attribute.

Because our feature computation is performed in C++ instead of Matlab, we retrained all of the attribute models using our features. Despite the computations being the same algorithmically, the feature values are subtly different at the pixel level due to slight differences in the Matlab and OpenCV utility functions involved.

Algorithm 4.1 Attribute probability computation

1: Input: RGB image I with d object detection bounding boxes, $m \times k$ matrix of model coefficients ■
2: Convert RGB to LAB and quantize to learned k-means color centers
3: Convolve I with a texton filterbank to compute texture descriptor, quantize to learned k-means texture centers
4: Compute HOG descriptor on I with 8×8 blocks, 4 pixel step size, and 2 scales per octave; quantize to learned k-means HOG centers
5: **for** each of d object detection bounding boxes **do**
6: Histogram the color, texture, and HOG features for pixels and blocks within the bounding box
7: Repeat for 3 rows and 2 columns of non-intersecting subregions whose union is the bounding box
8: **end for**
9: Concatenate histograms for each subregion by feature type, then concatenate feature type histograms per bounding box
10: Concatenate feature vectors into an $m \times d$ feature matrix $\mathbf{X}$
11: Multiply model coefficients and feature matrix for classifier scores: $■^T\mathbf{X} = \mathbf{S}$
12: Compute matrix of attribute probability estimates $\mathbf{P} = \frac{1}{1+\exp(-\mathbf{S})}$
13: Columns of $\mathbf{P}$ represent vectors of conditional probabilities for the set of k attribute models, one column for each object detection in I

Determining the conditional probability for an input object detection involves computing the linear model score:

$$s = \sum_{i=1}^{n} \theta_i * x_i + \theta_{bias} \tag{4.4}$$

and then computing the probability using the logistic function:

$$p(y = 1|\mathbf{x}, \theta) = \frac{1}{1 + \exp(-s)} \tag{4.5}$$

We implement our attribute classification for multiple attributes for each detected object, so we utilize matrix operations to compute all of the probabilities for one input image efficiently. Suppose we have d detected objects in an input frame, and we wish to compute the probability for k attributes each. Our model size is determined by our feature vector with a bias, so we have an $m \times d$ feature matrix $\mathbf{X}$ and an $m \times k$ model matrix Θ, where $m = n + 1(\text{bias}) = 9688 + 1$. We

compute the set of linear model scores as follows:

$$\Theta^T \mathbf{X} = \mathbf{S} \tag{4.6a}$$

$$\begin{bmatrix} & k \times m & \\ & \text{Models} & \end{bmatrix} \begin{bmatrix} & n \times d & \\ & \text{Features} & \\ & & \\ 1 & \cdots & 1 \end{bmatrix} = \begin{bmatrix} & k \times d & \\ & \textit{Scores} & \end{bmatrix} \tag{4.6b}$$

And use the logistic function element-wise to compute the conditional probability:

$$\mathbf{P} = \frac{1}{1 + \exp(-\mathbf{S})} \tag{4.7}$$

In output matrix $\mathbf{P}$, element $p_{i,j}$ is the conditional probability that the i^{th} attribute applies to the j^{th} object bounding box. Algorithm 4.1 describes the full set of steps for computing features and classifying attributes on an input image.

4.3 Object Attribute Descriptor

Although we reproduce the attribute classification method proposed in [10] for describing object attributes at the frame level, there is no existing work that incorporates image-level semantic attributes into an object model acquired over many observations. Some recent work in semantic mapping [37] computes object attributes from an object's point cloud, but does not consider image-based features. These object model-level features could be computed from our object models as well, but we also do not wish to discard the rich appearance information contained in the

image-based object detections.

Extending our bottom-up approach to object modeling to attribute description, we propose a way to combine attribute probabilities from many observations in individual images that we call the *Object Attribute Descriptor* (see Figure 4.2). We require a data structure that is easy to update in an online way as we acquire new observations of an object, and that permits the merging of object models should two previously separate models turn out to be co-located. We also want to capture the confidence in the attribute classification, rather than a binary label of {present/not present}.

One common assumption about image data is that the noise in the value at each pixel is normally distributed. Gaussian noise permits the use of techniques like the Kalman Filter [109], but in our case, this requires several further assumptions. Although it is still valid under the linear transformation from RGB to LAB color space, there is no documentation in the literature showing that the texture and HOG features we use have Gaussian noise. If we make the assumption that our features do indeed have normally distributed error, the use of a logistic regression model means that the errors in the output labels (0 or 1, since our training data is discrete) are binomially-distributed. Even if we ignore the nature of the noise in the input data and output labels, and assume that the probability of the 1 label is indeed normally-distributed, we do not have a direct way of estimating its mean or variance for use in a Kalman Filter.

In order to address these design constraints, we propose the *Object Attribute Descriptor* (OAD) as a vector of attribute probability values that are averaged over all of the observations that have been incorporated into its object model. This method, a *cumulative moving average*, permits simple updates and merging, and relies on minimal assumptions about the error in the input. The OAD data structure consists of a k-element vector of attribute probabilities and an integer n that counts the number of observations.

Let $c_{i,t}$ be the cumulative moving average for the i^{th} attribute after t observations, with $c_{i,0} = 0$ for all i. If $o_{i,j}$ is the j^{th} observation of the i^{th} attribute, the average after t observations is simply

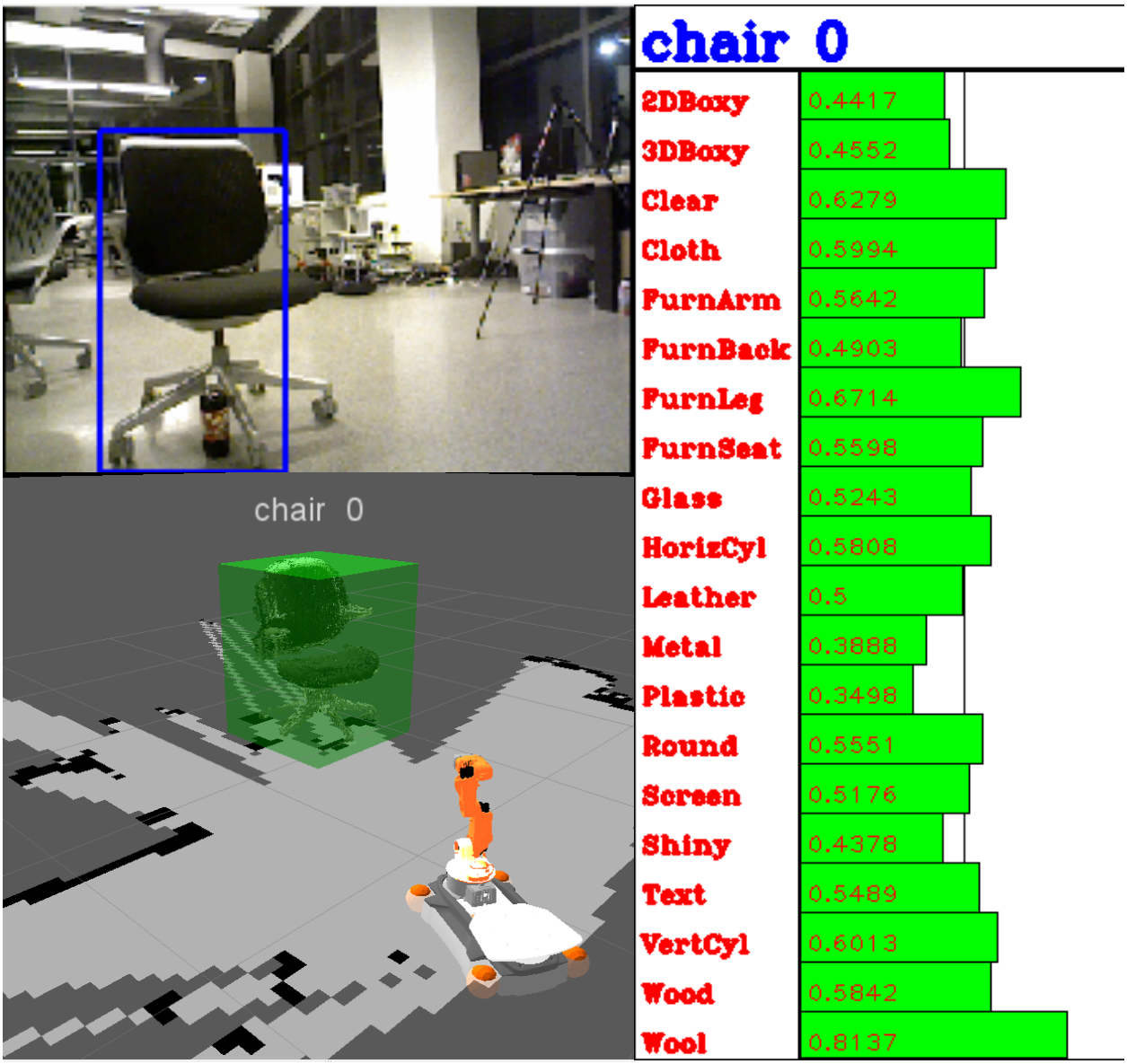

Figure 4.2: Example visualization of an *Object Attribute Descriptor* (OAD) for the model of the pictured object. The OAD is an array of cumulative moving averages over all of the attribute probabilities of the observations that were used to construct the model.

$c_{i,t} = \frac{\sum_{j=1}^{t} o_{i,t}}{t}$. Updating an attribute probability for observation $t+1$ only requires the value of the average after the previous observation:

$$c_{i,t+1} = \frac{t\, c_{i,t} + o_{i,t+1}}{t+1} \tag{4.8}$$

In the event that two different models actually describe the same physical object and are to be merged, the combined attribute probability can also be computed easily with this representation. If one model has $c_{i,g}$ after g observations, and the other has $c_{i,h}$ after h observations, the new value is:

$$c_{i,g+h} = \frac{g\, c_{i,g} + h\, c_{i,h}}{g+h} \tag{4.9}$$

Although a more complicated error model for an attribute's conditional probability for an observation might be possible, we posit that this approach is straightforward to compute online, robust to errors, and uses minimal assumptions. In particular, and in keeping with the bottom-up philosophy of our system, repeated observation of an object's attributes will enable stability in the average, since consistent values with be reinforced, and the influence of any outlier observations will diminish as the number of observations increases. We also expect that by modeling the average values, the attribute descriptor will provide repeatability when observing the same object instance in different trials. In other words, the same object should have the similar attributes even if we observe it somewhat differently, and we propose that our Object Attribute Descriptor satisfies that. We present qualitative results of our attribute modeling approach in Sections 5.2.3 and 5.2.4.

Chapter 5

Attributed Object Maps

5.1 Framework and Implementation

The system we present incorporates many different components into a single pipeline. We utilize the popular middleware platform Robot Operating System (ROS) in our implementation, in order to take advantage of the abundance of available modules and also to make the software for our system easy to distribute and deploy on other platforms. ROS operates with a *publisher/subscriber* model, facilitating interprocess communication via serialized messages passed over TCP/IP [110]. Each process running on a possibly heterogeneous network of physical machines, can publish and subscribe to topics, allowing data to flow between different components of the robot's software system. This approach permits self-contained computation in task-specific modules (e.g. one node to publish images from a camera, one to receive velocity commands and operate a motor, etc.) and efficient and scalable deployment of diverse software on a variety of robotic platforms. Figure 5.1 shows a block diagram illustrating the structure of our system, its dependencies, and the data flow through the pipeline.

One of the advantages of using ROS as a framework is that many powerful components are publicly available and easy to integrate with new applications, either as separate nodes, or as linked-in

libraries. In particular, we make extensive use of the Point Cloud Library (PCL) [102] and OpenCV [105] for all 3D and image processing, respectively. We are also able to easily capture RGB-D data from an OpenNI-compatible camera using the ROS wrapper for that driver, and we visualize our results with the ROS package *rviz*.

We distribute our processing pipeline into four complementary steps: observation, mapping, model fitting, and visualization. Within each of these areas, ROS nodes handle task-specific processing, as described below.

5.1.1 Observation

The nonparametric object detector node *object_segmenter* performs object detection, foreground segmentation, and attribute classification for objects of a specified class. Detection is performed using an online implementation of an accelerated Deformable Part-Based Model (DPM) detector [111]. This detector, called FFLD, utilizes properties of the Fourier transform to optimize the computational performance of a sliding window detector, without sacrificing exactness. We adapted this approach to one that processes an online stream of images within ROS. For each object detected with FFLD, we also perform automatic foreground segmentation, as described in Section 2.2. Lastly, an Object Attribute Descriptor is generated for each detected object, according to the approach in Chapter 4. This node is currently designed to be class-specific: each instance of this node handles a single object class. Some efficiency could be gained by avoiding redundant feature computation for multiple classes, but this approach permits compartmentalization of any class-specific settings. As such, each instance of the *object_segmenter* requires inputs of an FFLD model for the detector, foreground and background object scale histograms for the segmenter, and a list of model files describing the desired attribute classifiers. Each node subscribes to a camera image and registered point cloud from an RGB-D camera, and publishes a topic of observations (see the description of Object.msg in Fig. 5.1.1).

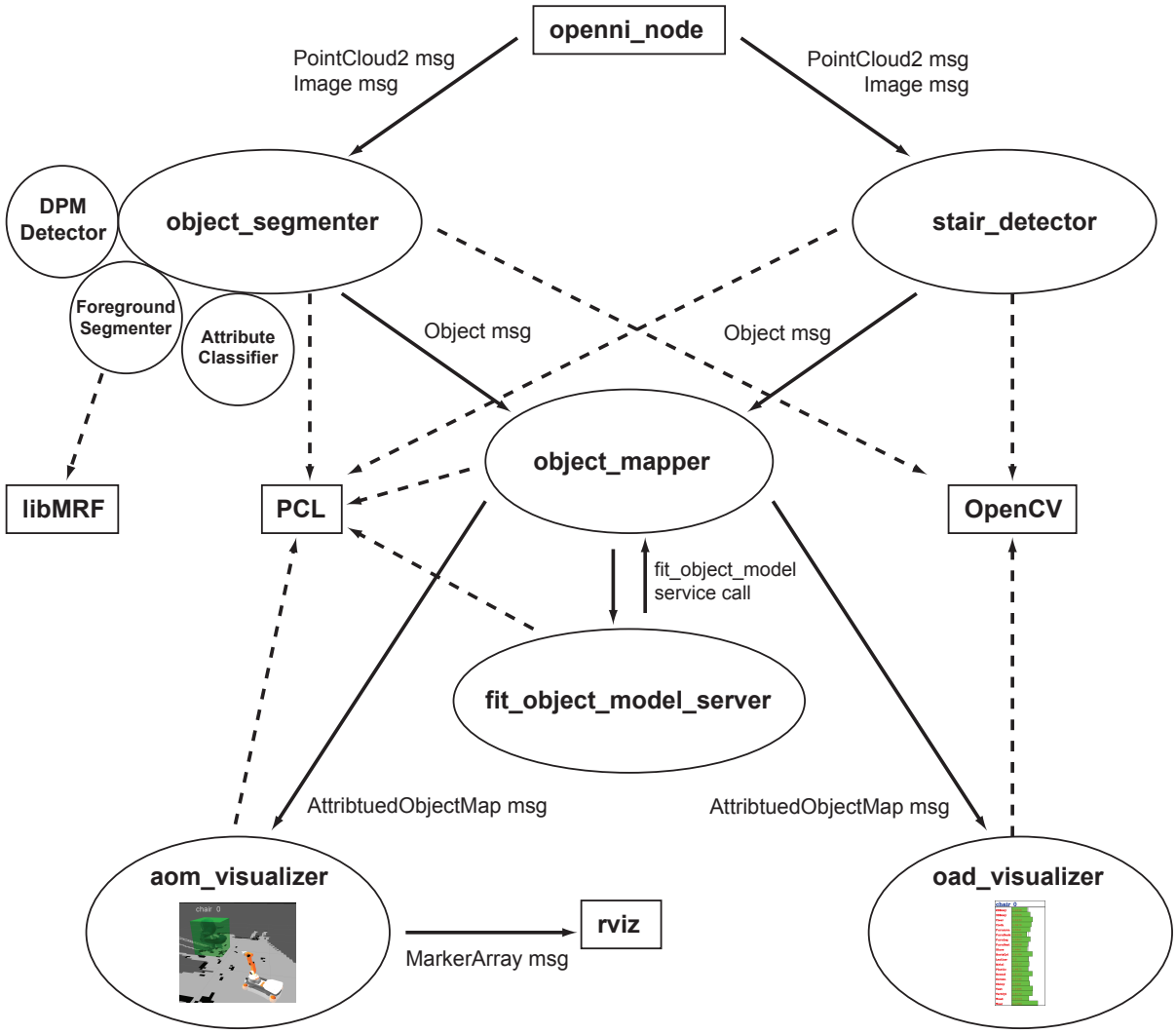

Figure 5.1: Overview diagram for *attributed object maps* ROS stack. Ellipses represent ROS nodes from our system, circles represent components of those nodes, rectangles and dashed lines represent external software dependencies or ROS nodes, and arrows show the flow of data in the forms of ROS messages or service calls.

Object.msg

```
# Object - pose, attributes, and extracted sensor data
#          can represent a single observation or the parameters
#          of the object model

string object_class               # Object class
BoundingBox box                   # Pose and dimensions
                                  # of bounding box

ObjectAttributeDescriptor oad     # Stores attribute descriptor:
                                  # attribute names and
                                  # probabilities, or model
                                  # parameter names and values

sensor_msgs/PointCloud2 pc        # Extracted point cloud data
                                  # for object. Can be left
                                  # empty for nonparametric
                                  # object classes.
```

Figure 5.2: ROS msg file with specifications for Object message type.

Stair edge detection is performed by the *stair_detector* node as described in Section 3.2. Due to the problem-specific nature of parametric object detectors within our bottom-up modeling paradigm, each detector of this type will likely need to be distinct based on the physical nature of the object and its generative model. The details provided here are specific to modeling stairways, but the same principle could be applied to other parametric object classes. The *stair_detector* node does not require any *a priori* inputs, but it subscribes to the RGB and depth images, and registered point cloud, from an RGB-D camera. Like the nonparametric object detector, the stair detector also publishes a stream of observations on its own topic.

Both parametric and nonparametric object detectors publish a common message type, *Object*, that contains the data about one observation of an object instance (see Fig. 5.1.1). Nested within the *Object* message is an *ObjectAttributeDescriptor* message, defined by the specification in Figure 5.1.1.

ObjectAttributeDescriptor.msg

```
#ObjectAttributeDescriptor - stores attribute or parameter
#                            values for one or more object
#                            observations

AttributeMetaData meta      # Array of strings with names
                            # for each attribute or parameter

float32[] values            # Attribute probabilities or
                            # parameter values

uint16 count                # Number of observations in
                            # this OAD
```

Figure 5.3: ROS msg file with specifications for ObjectAttributeDescriptor message type.

5.1.2 Mapping

The *object_mapper* node is responsible for taking in object observations and assigning them to object models based on class and proximity. This process maintains an internal array of object models and subscribes to a topic of observations, to which multiple detectors can publish messages from different classes. These object models store an *Object* message type for the model, an array of observations that it has accumulated since its last refitting, and some meta data about the model. The mapper assigns the observations, and periodically the model fitting module uses the accumulated observations and a previously refit model to fit a new model. The *ObjectModel* message type is detailed in Figure 5.1.2.

When the *object_mapper* receives a new observation, it follows Algorithm 5.1 in deciding whether this observation belongs to an existing model or a new one. For each object model, if the model has previously been fit, the bounding boxes of the new observation and the modeled object are checked for 3-dimensional intersection. If it has not been previously fit, the new observation is similarly checked for intersection with all of the accumulated observations of this object. We assign the observation to the first object it intersects with, and if it does not intersect with any of the existing

ObjectModel.msg

```
# ObjectModel.msg - Model representing generic object
#                   instance. Collects observations in
#                   observations array until the model is
#                   re-estimated. Stores object type,
#                   bounding box and attributes (or point
#                   cloud and parameters if parameterized)

Object model                    # Object message storing
                                # aggregated, model-level
                                # object data

bool modeled                    # Has object model been
                                # fit to this yet, or are
                                # we still waiting for
                                # enough observations

uint32 number_of_observations   # Number of observations
                                # that have gone into this
                                # model

time initial_observation        # Timestamp of first
                                # observation

duration length_of_observation  # Difference between first
                                # and last observation

Object[] observations           # Collection of observations
                                # since last model refitting
                                # Periodically, this is
                                # flushed and the observations
                                # in it are used to refit the
                                # model.
```

Figure 5.4: ROS msg file with specifications for ObjectModel message type.

object models, a new object model is spawned to contain it.

Algorithm 5.1 Assignment of new observations to models

```
Input: object observation (Object.msg) obs_i, cached array of n existing object models {O}
Compute axis aligned bounding box of obs_i point cloud obs_i.pc and assign to obs_i.box
if {O} is empty then
    Create empty ObjectModel as O_1 and add obs_i to O_1.observations
else
    for each object model O_j in {O} of of class obs_i.object_class do
        if O_j.modeled then
            if obs_i.box and O_j.model.box intersect then
                Add obs_i.box to O_j.observations
                return
            end if
        else
            for each observation o_k in O_j.observations do
                if obs_i.box and o_k.box intersect then
                    Add obs_i.box to O_j.observations
                    return
                end if
            end for
        end if
    end for
    obs_i.box does not intersect with any existing model or observation
    Create empty ObjectModel as O_{n+1} and add obs_i to O_{n+1}.observations
end if
```

Using a greedy approach to assigning observations to objects may create ambiguous results if a new observation intersects with multiple object models. Additionally, as they receive more observations and grow, the object model bounding boxes may eventually intersect as well. To deal with these situations, we implement a merging algorithm (Alg. 5.2 below) to combine object models if they are found to intersect in 3D space. It is a reasonable assumption, and one that is also made in [37], that if two objects of the same class are closely adjacent or abutting, they can be considered to be one object for the purposes of object modeling using only visual perception (e.g. books stacked on one another may appear as one unit until they are manipulated to disambiguate them). Whenever one of the models reaches the threshold for accumulated observations and is refit, we also check for model intersection for the purposes of merging.

Whenever a model is refit, the set of object models is republished as an *AttributeObjectMap* mes-

Algorithm 5.2 Merging of object models

Input: cached array of n existing object models $\{O\}$
for each object model O_j in $\{O\}$ **do**
  for each object model O_k for $k = j+1$ to n **do**
    if $O_j.model.box$ intersects $O_k.model.box$ **then**
      Add $O_k.model$ and $O_k.observations$ to $O_j.observations$
      $O_j.number\_of\_observations$ = $O_j.number\_of\_observations$ + $O_k.number\_of\_observations$
      $O_j.initial\_observation = \min(O_j.initial\_observation, O_k.initial\_observation)$
      Set $O_j.length\_of\_observation$ to be union of $O_j.length\_of\_observation$ and $O_k.length\_of\_observation$
      Let $count_j = O_j.model.oad.count$ and $count_k = O_k.model.oad.count$
      for each attribute l in $O_j.model.oad$ **do**
        Let $value_j = O_j.model.oad.values[l]$ and $value_k = O_k.model.oad.values[l]$
        $O_j.model.oad.values[l] = \frac{count_j * value_j + count_k * value_k}{count_j + count_k}$
      end for
      $O_j.model.oad.count = count_j + count_k$
      Delete O_k from $\{O\}$
    end if
  end for
end for

sage type. This message contains a timestamped header and the array of object models that the *object_mapper* node stores internally, representing the state of the map at the time of publication. Visualization nodes (as described below) can subscribe to this message topic, in order to obtain the map state for display. Additionally, and more importantly, other processes performing path planning, topological mapping, place categorization, among many other potential applications, can subscribe to this topic to extract the object models that the robot has acquired. Each of the models contains its spatial information (both bounding box and model point cloud, anchored to the underlying metric map) and attributes or parameter values.

5.1.3 Model Fitting

Object model fitting is performed by the *fit_object_model_server* node. For both parametric and nonparametric objects, we perform model fitting after k observations have been accumulated, where k is a parameter set by the user. Each time an observation is assigned to a model, as described in Algorithm 5.1, the model that the observation is assigned to is checked for the size of its

observations array. Once it reaches k accumulated observations, the mapper node makes a blocking service call to *fit_object_model_server* to fit the appropriate model to this object. If the object is of a nonparametric object class, it is refit using the approach described in Section 2.2.4. In order to accommodate online object modeling with periodic model fitting, we modify that approach to introduce a weighting scheme for performing our proposed outlier removal algorithm. Weighting the points that remain in the model after downsampling by the number of points contributing to each voxel permits the overall point densities in the model to be maintained despite the removal of many of the individual points (See Algorithm 5.3). Counting each point in the downsampled model according to the aggregated "population" of its voxel ensures that the repeatedly observed model points are persistent, even if new observations are of other parts of the object.

Each parametric object type requires a separate subroutine for object modeling, implementing its own algorithm for its particular generative model. In our current implementation, we only have a module for fitting stairway models, as described in Algorithm 3.2. These procedures each return a new *ObjectModel* message that has a refit model and has had its observations array purged, and the new model replaces the original in the mapper's internally cached *AttributedObjectMap*.

Algorithm 5.3 Online multi-view 3D object reconstruction

Input: Stream of object observation point clouds assigned to a particular model, modeling period n
Initialize an empty point cloud P_r to hold the reconstruction
for each point cloud **do**
    Add point cloud to queue of observations
    if size of queue = n **then**
        Add all points from observation in queue and any points already existing in the model to P_r
        Compute mean distance to k nearest neighbors for each point in P_r
        Compute weighted histogram of mean distances for P_r: points from observations in queue are counted once, points in existing model are counted m_i times, where m_i is the weight for that voxel
        Perform statistical outlier removal on P_r (as in [30] or Sec. 2.2.4) to filter out sparse, noisy points
        Downsample P_r to voxel grid, maintaining weighted count of points lying within each voxel (observation points count as 1, model points as m_i)
    end if
end for
Output: P_r, point cloud reconstruction of object segmented from 3D scene

5.1.4 Visualization

Two additional nodes handle visualization of the data in an *AttributedObjectMap*: *aom_visualizer* for displaying the models in ROS's *rviz* visualization program, and *oad_visualizer* for displaying *ObjectAttributeDescriptors*. In *rviz*, we can easily visualize messages from the *Marker* message type, and the *aom_visualizer* node subscribes to an attributed object map topic and publishes an array of markers corresponding to various aspects of it models. These markers can be visualized simultaneously with the robot's metric map in *rviz*, along with other data that the robot acquired during its run. For each model, *aom_visualizer* publishes a rectangular prism marker for the model's bounding box with a label for the object's class and index in the attributed object map's array of models. The model's point cloud is also published, offering a more detailed level of visualization than just the bounding box. For stairway objects, it also publishes markers for the parametric model (a flattened rectangular prism for the plane and a set of lines for the step edges) and some text labels for the model parameters. Examples of these visualizations can be found in Chapter 3 and below in Section 5.2.

The *oad_visualizer* node also subscribes to the *AttributedObjectMap* topic, but extracts the attribute values for the models in the map and displays them in a separate window. For each nonparametric object, a bar plot indicating the current attribute probability estimates and their names is displayed. An example of this output is shown in Figure 4.2.

5.1.5 Building an Attributed Object Map

The system presented here handles modeling, mapping, and visualization, but does not integrate directly with high-level controllers for the robot's actions. Consequently, the object models throughout this work have been obtained from pre-recorded data or during teleoperation of the robot. Some preliminary experiments in autonomous control, with feedback from the attributed object

map, have been performed and are discussed in Section 6.3.1.

Object detection in the wild presents many additional challenges that were not addressed in Chapter 2, where ground truth object detections were used as input. The qualitative experiments presented in the next section demonstrate the system's performance in real-world scenarios, where the performance of the object detector affects every processing step after it in the pipeline. In general, we tune our detection threshold such that we achieve high precision, even if it is at the expense of having high recall. Even with true positive detections, the detected bounding box is often not as tight as a human-annotated one, or may be misaligned from the actual object. In these situations, the utility of the automatic foreground segmentation step is especially important in the quality of the resulting models.

5.2 Qualitative Experiments

We evaluate our full system by conducting several qualitative trials, extending the quantitative results presented in Chapter 2 to demonstrate deployed performance and attribute description. We do not further address the performance of our stairway modeling component, since it has been subjected to thorough quantitative evaluation already in Chapter 3. The localization accuracy, model completeness, and attribute stability are analyzed. We demonstrate the effectiveness of our bottom-up modeling approach in producing visually appealing models that accurately capture the semantic attributes of each of the object classes we consider.

In all of the following experiments, the system was deployed on a KUKA youBot mobile manipulation platform (See Figure 5.5). This robot consists of a holonomic mobile base that uses omniwheels to translate along hard indoor surfaces. Attached to the base is a 5 degree of freedom manipulator arm that ends in a two-finger gripper. An ASUS Xtion Pro Live RGB-D camera is mounted at the wrist for perception. A Hokuyo URG-04LX-UG01 laser scanner attached to the

Figure 5.5: KUKA youBot mobile manipulation platform. Equipped with ASUS Xtion Pro Live RGB-D camera mounted at the wrist of the manipulator arm, and Hokuyo URG-04LX-UG01 laser scanner mounted on the front bumper.

front of the robot allows for localization and mapping. For all tasks, we perform visual data processing on an external laptop (with a quad-core 2.4 GHz Intel i7 processor and 8 GB of memory), and run mapping and control processes on the robot's onboard PC. We teleoperate the robot and observe each scene from a number of viewpoints, and analyze the final attributed object map.

When operating the RGB-D sensor at 320×240 resolution, per-frame processing time is approximately 0.8 seconds when the detector fails to find any object instances. The additional processing time for foreground segmentation and attribute classification when objects are found is typically between 1 and 3 seconds more, for an average overall frame processing time of about 2.5 seconds. Many factors, including the number of detected objects and the sizes of their bounding boxes, determines the processing time of each particular frame. However, in all of the following experiments, the total processed frame rate is approximately 0.4 Hz, which is responsive enough for real-time object mapping. Currently, OpenMP shared-memory parallelization is implemented for detection and attribute classification, but these computations could be performed on a GPU if faster performance is desired.

All of these examples were constructed using the same parameter settings. In particular, we use a

detection threshold of -0.85 for all of the DPM object detectors, and the same parameter values for foreground segmentation as the experiments in Section 2.3.2. We downsample to a $0.5cm$ voxel grid during model fitting and use a modeling period of 10 observations. The robustness of our system is illustrated by its flexibility in modeling object classes of widely varying size and appearance using the same settings.

For all object classes we consider, we utilize deformable part model detectors trained on PASCAL VOC data [13]. For the object classes that were not evaluated in Chapter 2, we trained new object scale histograms using RGB-D data from the NYU Depth [80] dataset. Classifiers for the semantic attributes we consider are described in Chapter 4.

5.2.1 Localization

The presented system is capable of localizing multiple instances of an object class, and multiple object classes simultaneously. Figure 5.6 illustrates the constructed attributed object maps and robot-perspective camera images for both scenarios. In each case, the robot constructed an occupancy grid map as it was teleoperated around the scene, observing both objects repeatedly. Any detected objects were segmented and assigned to the appropriate model, based on the procedure in Algorithm 5.1. The object models are accurately localized to the underlying metric map. Although mapping in these experiments is done in a small area, the same approach to object localization is taken in the stairway-specific experiments in Chapter 3, where we demonstrate scalability to large maps.

5.2.2 Class Model Examples

This experiment evaluates the accuracy of the attribute probabilities computed in the object attribute descriptor. Here we consider the agreement of the object model's attributes with its physical

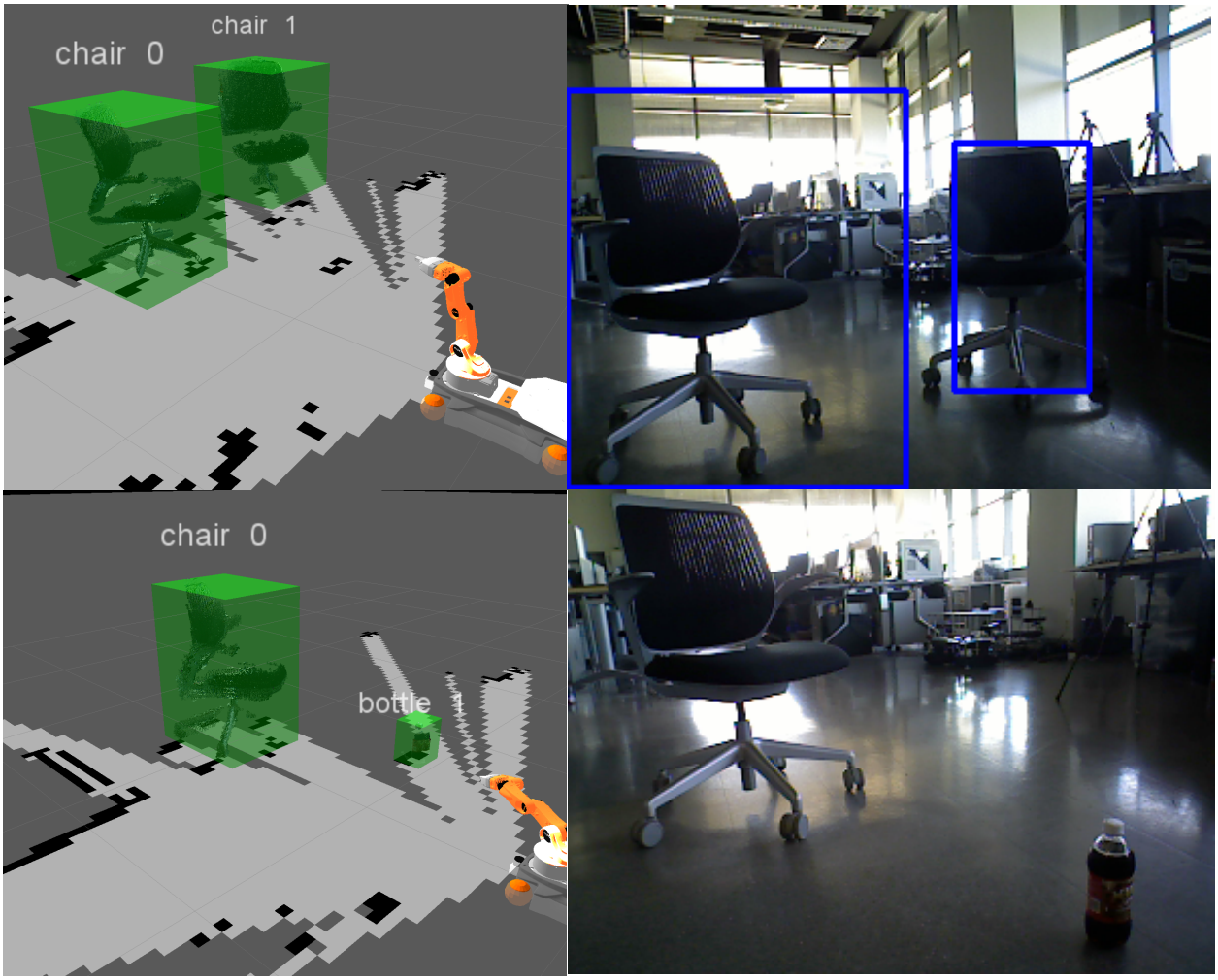

Figure 5.6: Example attributed object maps for scenes with multiple objects of one class (top) and multiple objects of different classes (bottom). A camera image from the robot's perspective provides spatial context.

or at least visual properties in the real world. The nature of such properties is inherently subjective: an object may have the attribute *shiny*, but properties such as this are not binary, and are subject to the interpretation of the observer. The object may have one part that is reflective, but otherwise is not shiny. The continuous nature of these properties are part of the motivation in using attribute probability, rather than a binary label, as input to the object attribute descriptor. The consequence of this is that the OAD average values for each attribute can better describe the degree to which the object appears to have the attribute.

We modeled two instances of each nonparametric object class that we consider. In each case, the robot was maneuvered so that it observed the object from many perspectives and built a reasonably complete model of it. The final object attribute descriptor is shown with the accompanying object model and a camera image of it in Figures 5.7 and 5.8. We discuss these results in more detail below, with the first example in each case being on the left in the figures, and the second example on the right. In general, OAD values in the 0.4 to 0.6 range indicate relative ambiguity according to the attribute models. Values above 0.6 are strongly positive in showing the attribute, and values below 0.4 indicate strong absence of the attribute.

bottle

The two bottle examples show reasonable agreement between their estimated attributes and a subjective interpretation of their real world properties. Both have a relatively strong display of the *glass* and *plastic* attributes, while having a lower response to the *wool*, *cloth*, *horizontal cylinder*, and furniture part attributes. Despite some attributes that the two have in common, a few properties could help to distinguish them. Although its value for the *clear* attribute is close to neutral, the green, semi-transparent water bottle on the left has a much higher value than that of the bottle on the right that contains opaque root beer. The *vertical cylinder* attribute does not appear as strongly as we would expect, perhaps due to the small size of these cylinders relative to the cylindrical

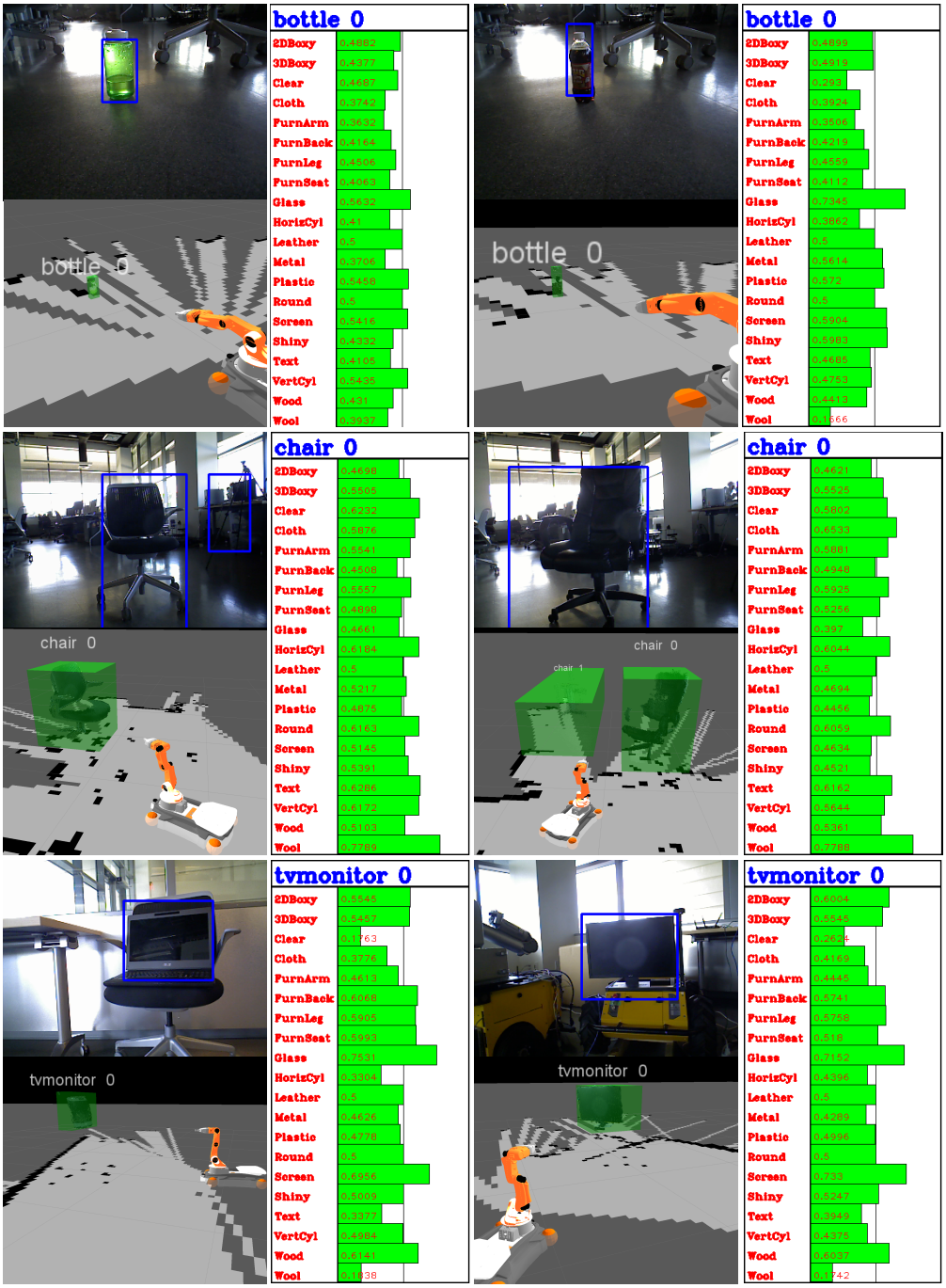

Figure 5.7: Example object attribute descriptors for several object instances of each class. Top to bottom, they are: *bottle*, *chair*, *tvmonitor*. Also shown are the object model and a view from the robot's perspective for visual context of the object's true properties. Note: the *chair* example on the right also contains a second model instance (*chair_1*) that has only received an initial model fitting and would need to be further observed in order to refine it.

objects that the model was trained on.

chair

Most of the attributes for the chair class examples are close in value, indicating good class-level agreement. Although many of the attributes are somewhat ambiguous, a few prominent values are worthy of note. Both chairs show a stronger response in the *3D Boxy* attribute than the *2D Boxy* attribute, but both are near neutral. The chair as a whole occupied 3D space, but it has several planar sections, and may appear somewhat 2D in its features. Similarly, the *horizontal cylinder*, *vertical cylinder*, and *round* attributes have large values, likely due to localized parts that have rounded shape, and not the overall object structure. Both chairs show a strong response to the material attributes *cloth* and *wool*, although the *leather* attribute is decidedly neutral for even the black leather chair on the right. However, as we would expect, the furniture part attributes (*FurnArm*, etc.) are well represented here compared to the examples of the *bottle* class.

tvmonitor

The laptop screen and flat screen monitor we use for the *tvmonitor* class examples do look very similar, and their attributes reflect this similarity of appearance. This also indicates that objects that look similar to humans may also "look" similar when described with attributes in this way. Both examples exhibit strong features of *2D Boxy*-ness, *glass*, and *screen*, and low values for *clear*, *text*, and *wool*, which is in good agreement with reality. However, the furniture part attributes and the *wood* attribute have erroneously large values.

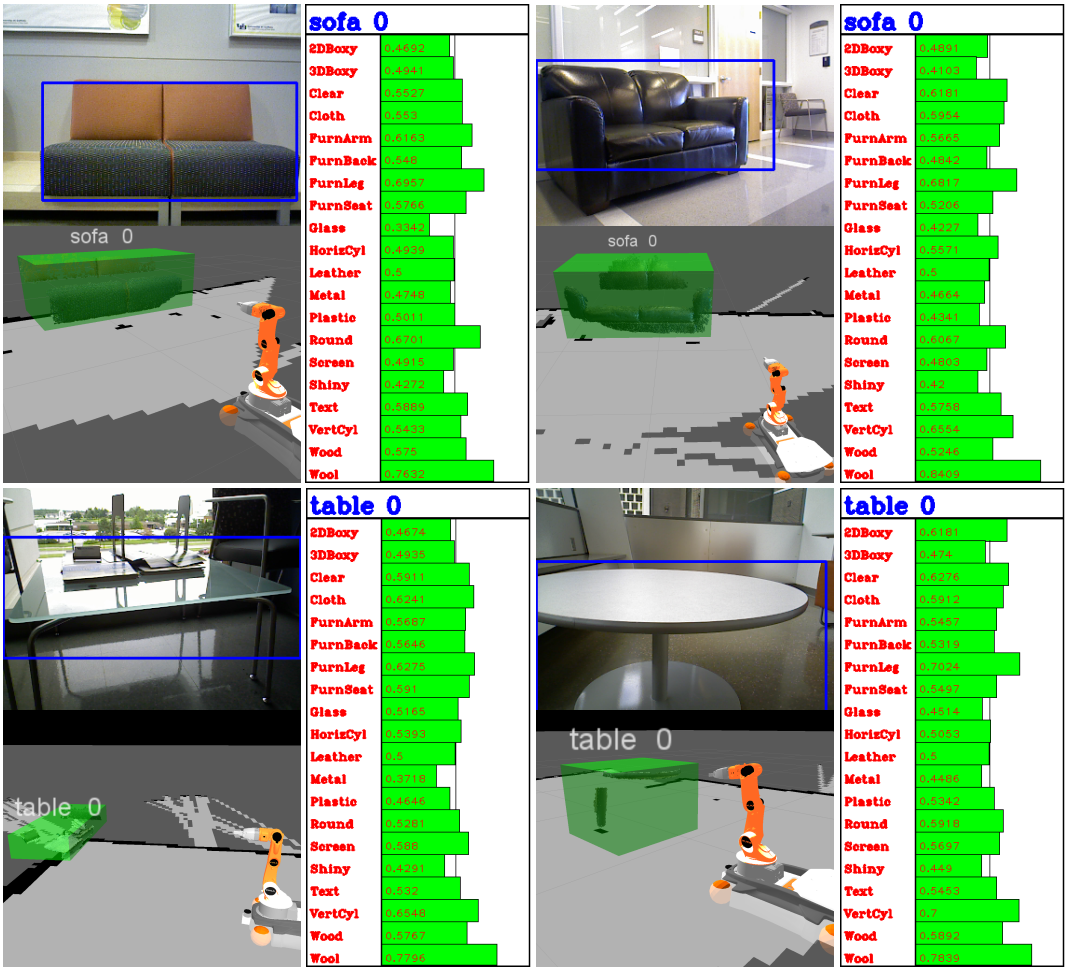

Figure 5.8: Example object attribute descriptors for several object instances of each class. The *sofa* class is on top and the *table* class is below. Also shown are the object model and a view from the robot's perspective for visual context of the object's true properties.

sofa

The sofa class examples we show in Figure 5.8 are both two-seater models, but otherwise show a number of differences in appearance. Both show large responses for the furniture part attributes, although the *furniture arm* attribute should be higher for the second example than the first. There are rounded parts to both, and the arms of the second example may look like vertical or horizontal cylinders from the front and side, respectively.

The material attributes for the first example are relatively accurate, with *cloth*, *wool*, and *wood* (the tables adjacent to the sofa) showing strong responses. However, for the second example, they are in poor agreement with reality: the *cloth* and *wool* attributes should be low, and the *leather* attribute should be very high. A low response for *glass* is appropriate for both, but the second example registers somewhat higher, perhaps due to the frosted glass window behind the sofa. This glass pane may also explain some of the high value of *clear* for example two. Both examples should also show low or ambiguous responses for *text*, and the glossy leather on example two should exhibit more of the *shiny* attribute as well.

table

Tables present some failure modes of our system, at least when deployed on the youBot platform. When the arm is fully extended upward, but the wrist is turned to face forward, the camera is mounted at a height of approximately $0.6m$. Most of the images of tables in the PASCAL VOC [13] data used to train our object detectors are taken from the perspective of a standing human. Consequently, full-sized tables are not detected because the robot is looking up at them instead of down. Attempts to model many different conference tables and desks were fruitless.

The two table examples we do present are shorter, coffee/end table style pieces that the robot's camera is tall enough to observe. However, these two cases still highlight some shortcomings of

our approach even on tables that trigger the detector. With the first example, the tabletop is made of frosted glass, and the transparency of the surface resulted in depth data that was either invalid or representative of the opaque objects behind it. In this case, the resultant point cloud model is just the points for the objects that are resting on top of the table.

The second table example is opaque, but because the tabletop is at almost the height of the camera, the points on the top are relatively sparse for each detection due to foreshortening effects. The density of the points on the surface of the tabletop is thus too low to survive the outlier removal modeling step. The result is that only the central leg of the table and the front edge of the tabletop are modeled well.

Like the other object classes, the attributes are somewhat consistent with the real-world objects: some attributes show strong agreement, and others do not. The furniture part attributes, especially *furniture leg*, have high responses. Detection bounding boxes for the second example are dominated by the tabletop surface, and this large plane in the scene is reflected in the large value for *2D Boxy*. Similarly, round and *vertical cylinder* are accurate descriptions of the shape of this example. But the value for the *clear* attribute ought to be larger for the first example, since it is made of frosted glass, and the *wood* and *wool* attributes should show weak responses for both of these objects.

Summary

These examples indicate a relatively accurate agreement with real world object properties, when considering the continuous nature of some of them (e.g. the chairs are somewhat round, but also somewhat boxy). The object attribute descriptors for these objects show potential for describing particular objects or differentiating objects within one class. Even if the attributes don't align exactly with reality, the object attribute descriptor structure can provide valuable descriptive information. In particular, the OADs for object instances that do have relatively different appearances

(the *bottles* and *tables*) are more different from one another than the OADs for the *chair* and *tvmonitor* instances, which do have similar appearances within their classes.

Several of the attribute classifiers do not respond much, if at all, to the inputs from our system. In particular, the *leather* attribute has produced neutral (0.5) output for all object detections over all the trials we conducted. This is due to the feature selection process that occurs in L1-regularization, where only a sparse subset of the features have non-zero weights in the model. In the case of *leather*, whatever features have been selected in the model must not be appearing in our input data. One way to attempt to address this problem would be to gather data and train the attribute models from real-world object detections, rather than the ground-truth annotations of the PASCAL VOC [13] dataset. Improving the utility of the object attribute descriptor by removing irrelevant attributes could be done by restricting the set of attribute models to a class-specific set. We have considered the set as a whole for all object classes here, in order to illustrate the response of the irrelevant attributes as well. One possible approach to distilling more salient features from these semantic attributes would be to perform principal component analysis, in order to extract the dominant components in some underlying subspace. PCA may produce more relevant semantic meta-features, but they would no longer be paired with a natural language description.

5.2.3 Attribute Stability

To demonstrate the stability of the values in an object attribute descriptor over time, and their convergence toward a single estimate for the object, we plot the attribute values for several of the trials in Section 5.2.2. We selected one example object each from the *bottle*, *chair*, and *tvmonitor* classes and plotted each of the attribute probabilities in the OAD against the number of observations (Figure 5.9). The data points making up these lines are sparse because the OAD is only updated during model refitting, which in the case of these experiments is every 10 observations.

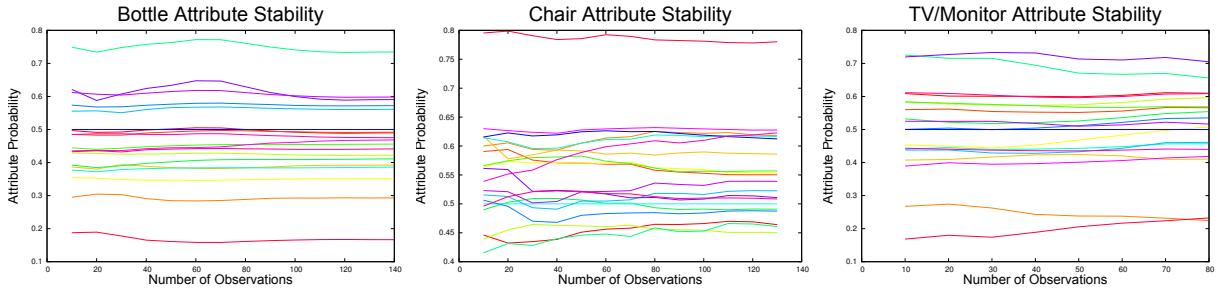

Figure 5.9: Attribute stability and convergence for an example object of each class. Attribute probability from the model's *object attribute descriptor* are plotted against observation count. Attributes are color coded, but names have been omitted for clarity. Left to right: *bottle*, *chair*, and *tvmonitor*.

Ignoring the identity of particular attributes in these plots, it is clear that they all exhibit stability over time. As the number of observations increases, the probabilities in the OAD tend to level off, and for those that are still evolving, the trend is gradual. The convergence of the attributes toward a stable model estimate is especially noteworthy with the *chair* example. Many of the attributes fluctuate somewhat rapidly for the first few model fittings, but gradually settle in toward a horizontal asymptote. Stability is one of the properties we desire from our attribute estimates, so that attributes do not fluctuate as the number of observations goes to infinity.

5.2.4 Attribute Repeatability

Lastly, we show the repeatability of our bottom-up attribute description approach. One instance of the *chair* class was modeled in 5 separate trials. In each trial, both the object and the robot (and camera) were fixed, and the object was modeled for 100 observations. We compare the final OAD values for each attribute in Table 5.1.

The results of this set of control trials indicate a great deal of stability in the model attributes, with a maximum standard deviation of about 2.5%, and most below 1%. In real world situations, differences in viewpoint, lighting, object location, and many other uncontrolled variables, may affect the repeatability of the object attribute descriptor. However, the data presented here indicates

Table 5.1: Table showing attribute repeatability, with attribute values for five modeling trials of a *chair* instance, along with mean and standard deviation.

Attribute	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Mean**	**Std. Dev.**
2D Boxy	0.4876	0.4854	0.506	0.539	0.5023	0.5041	0.0215
3D Boxy	0.5361	0.5332	0.5264	0.5305	0.5181	0.5289	0.0070
Clear	0.5784	0.571	0.5726	0.5906	0.5894	0.5804	0.0092
Cloth	0.6127	0.615	0.6195	0.6127	0.6038	0.6127	0.0057
Furn. Arm	0.578	0.5582	0.56	0.5694	0.5563	0.5644	0.0091
Furn. Back	0.5347	0.5317	0.5406	0.5694	0.5349	0.5423	0.0155
Furn. Leg	0.6289	0.6264	0.639	0.6785	0.6176	0.6381	0.0238
Furn. Seat	0.5528	0.559	0.5476	0.5665	0.5489	0.5550	0.0078
Glass	0.4074	0.4016	0.3823	0.3966	0.4066	0.3989	0.0102
Horiz. Cyl.	0.5776	0.5598	0.5493	0.5493	0.566	0.5604	0.0120
Leather	0.5	0.5	0.5	0.5	0.5	0.5	0.0
Metal	0.4253	0.4322	0.393	0.3748	0.4054	0.4061	0.0235
Plastic	0.4615	0.4747	0.4601	0.4583	0.4493	0.4608	0.0091
Round	0.6048	0.5808	0.5865	0.5892	0.5813	0.5885	0.0098
Screen	0.5158	0.5201	0.5317	0.5263	0.5085	0.5205	0.0090
Shiny	0.4733	0.4896	0.4752	0.4727	0.4778	0.4777	0.0069
Text	0.5774	0.5652	0.5537	0.5381	0.5607	0.5590	0.0145
Vert. Cyl.	0.6008	0.6187	0.6249	0.6674	0.6419	0.6307	0.0252
Wood	0.5822	0.5693	0.5852	0.6184	0.5886	0.5887	0.0181
Wool	0.7975	0.8036	0.8005	0.8087	0.7981	0.8017	0.0046

that the attributes are repeatable on their own, all other things being equal. Like attribute stability, repeatability is desirable, so that objects that are modeled on separate trials will have OADs that are closely similar to one another.

Chapter 6

Conclusions

6.1 Overview of Results

6.1.1 Parametric Object Modeling

We presented quantitative results for performing bottom-up modeling of stairways by extracting step edges from depth imagery and fitting these observations to a generative model. One of the major motivations for approaching the problem of stairway modeling is to assess traversability. We demonstrate that our approach is able to estimate the parameters of a stairway model, and therefore its physical dimensions, accurately enough to enable a deployed system to determine whether it is capable of ascending those stairs.

Many other objects and structural elements of the environment could be modeled in a similar way: combining object detections and fitting a simple generative model to them. For example, environmental barriers that robots are likely to encounter in most human-centric settings are doors. Assuming that a particular platform could interact with the door, it would be possible to model the door class and then use the estimates in the model to assess whether the robot can open the door. Doors typically have only a few parameters, and could be modeled from part-based object

detections. Aside from the physical size (width and height) of the door, most instances of the class can be described well if one knows the type of hardware on it (knob, handle, bar, or plate), the side that the hinges are on, and whether the door opens in or out. If the robot could piece together detections of the whole door with detections of its parts (hardware, hinges, etc.), it could determine how to interact with the door (turn the knob or pull the handle, push or pull to open). The position we take in advocating bottom-up modeling is that by observing the door object repeatedly, the robot would be able to estimate such a model as well as its confidence about the presence of those parts and how to perform the action of opening it. Waiting to interact with the door until it reaches a threshold in that confidence could prevent failure in the task.

The door opening problem has been well studied, and some recent approaches, including [112], demonstrate good performance in familiar environments. However, they make some assumptions about the appearance of doors (can't be flush with the surrounding wall) and handle (detector trained only on handles in their building) that would restrict its effectiveness in unknown environments. In the constrained environment they consider, they do require repeated observations of the handle (7 times) before confirming its detection. But in the unconstrained environments of other buildings, more observations of the door and its component parts would likely be necessary for effective and confident assessment of the door and its opening mechanism. As with our demonstrated examples, this passive approach to door modeling and mapping would be compatible with active tactile or vision-based perception or more fine-grained approaches to modeling a door that the robot is to interact with.

6.1.2 Foreground Object Segmentation

In performing object modeling, we developed an approach for automatic foreground object segmentation in order to refine the observations we get from bounding box object detectors. We desire 3D points of the detected object only, and not the background or any occlusions, which in many

object detections can occupy a significant portion of the box. By using a model of the object class' scale, we demonstrate that we are able to effectively produce cleaner segmentations of the object than taking the full bounding box. Additionally, when working with real-world object detectors, many false positive detections can be eliminated if that part of the image only "looks" like the object class, but is not of the correct scale.

This segmentation approach is potentially useful for refining feature computations on bounding box object detections by removing background and occlusions before features are computed. In our case, when measuring the features of a detected object for semantic attribute classification, the pixels on the background or any occlusions may pollute the histograms of feature values for the pixels that actually lie on the object. However, some feature types, like HOG, utilize the gradients that are found at object boundaries, and actually require that the background be present in order to correctly assess the properties of the object. We implemented this idea as an option in the *object_segmenter* node of the attributed object maps ROS stack for the pixel features (color and texture) but not for HOG. In testing, though, the results were not an improvement over using all of the data in a bounding box, likely because the training data for the attribute classifiers comes from full bounding boxes from the PASCAL VOC dataset [13]. However, there is potential for this approach to help purify feature computations if a dataset of segmented objects were used for training the attribute models instead of full bounding boxes.

6.1.3 Nonparametric Object Modeling

We demonstrate the ability to model objects of a variety of classes, including those with large variance in their shape or appearance, using minimal assumptions (a model of the visual appearance of the class, and a model of the object class' scale), and explicitly avoiding any templates or top-down information from the scene. Additionally, our bottom-up approach enables us to generate stable, repeatable, descriptors of objects with semantic attributes computed at the frame level.

We limit ourselves to a small number of generally indoor object classes here, but it would be possible to model many other object classes, given trained detectors and models of scale. There is potential for modeling even large, outdoor structural objects. With a long-range 3D sensor like a Velodyne registered with a camera, it would be possible to segment and aggregate a model point cloud for objects such as buildings or cars, albeit more sparsely than with RGB-D input.

6.1.4 Semantic Attribute Classification

We draw on the strong representation of object description in the literature when performing attribute classification. Due to the availability of the code and training data, and the compatibility of its approach to ours, we chose to utilize the method from [10] in computing semantic attributes. However, we adapted their approach to an online, robotically-deployed variant, re-implemented in C++ and parallelized with OpenMP.

Due to the popularity of the object description problem in the computer vision community, and very recently in robotics research, there are many other approaches to attribute assessment that could be applied in our system, either in place of, or in addition to, the Farhadi et al. approach. In particular, the recent work by Sun et al. [60] could be more appropriate for frame-level attributes because they implement classifiers that make use of depth features as well as RGB. However, it is not clear that their approach could operate in real time as our situation demands. Top-down attributes for color, shape, and size could also be computed from the model at the time of each refitting, in the same manner as in [37]. We avoided implementing this approach thus far, in order to focus on frame-level attributes, but for a deployed, real-world system, any and all useful attributes could be incorporated.

6.1.5 Semantic Mapping

Throughout the development of all aspects of this work, we have endeavoured to make design decisions such that our system would be complementary, and not contradictory, to other approaches in semantic mapping. Some existing object modeling and mapping systems may perform better quantitatively in some circumstances, but we approach the problem from a passive, bottom-up perspective in order to present an approach that may be useful in addition to these methods, or that may work in situations where the others do not. In particular, we acknowledge from the outset that our models may not be as precise as those that do explicit alignment or utilize an object template, but we aim to avoid assumptions and offer a system that develops approximate models and descriptions. In the case of both parametric and nonparametic objects, should a more precise model be needed than our system can provide, another approach can always use our model as a starting point and extend it or replace it with a better one.

6.2 Possible Applications

Our system design is geared toward building a map of localized, descriptive, object models, that in combination with a metric map can provide context for the environment, an inventory of potential objects of interest, or high-level features to use in some inference problem. We discuss a few potential applications here, including the application of stairway modeling to multi-floor mapping, direct search for an object based on its description, and place recognition or categorization.

6.2.1 Multi-floor Mapping

In Chapter 3, we discuss the prospective use of stairway modeling and mapping for autonomous exploration of multi-floor buildings. Indeed, this problem was a major motivation for investigating

the modeling of parametric objects to begin with. Given an appropriate climbing-enabled platform, the object mapping approach we present, coupled with other existing methods from the literature, could enable a system to perform autonomous exploration beyond a single level in a building. We discuss one barrier to this approach in Section 6.3.1, namely the difficulty in performing path planning in a way that observes stairways in the environment adequately for modeling. Adapting from teleoperation to autonomous behavior presents other challenges, but the robot's high-level controller must guide the robot's perception in a way that observes any stairways that exist in the environment that the robot seeks to explore, and this is a problem that we do not yet address in this work. However, with a strategy for overcoming this autonomous guidance problem, the remaining components of the system could be put in place rapidly.

6.2.2 Place Recognition and Categorization

As humans, we derive much of our context for the spaces we inhabit from the objects we find in them. The problem of place categorization (summarized with the question "what kind of room am I in?"), is a fundamental one in autonomous exploration of unknown environments, and one that can similarly be informed by objects for the robotic agent. One approach to inferring the topological structure of an indoor environment and labeling the component spaces with location classes is to construct a probabilistic graphical model from the map that incorporates both the structure of the space and features from it. One such approach is a Voronoi Random Field, as presented in [113]. This special case of a conditional random field (CRF) exploits the Voronoi decomposition of the explored space in the map to build a graph of nodes, and features extracted from the map are used to train the CRF. This approach could be extended to our problem, wherein the feature set for training could include the class, spatial arrangement, and attributes of the objects nearby to each node in order to learn a model that incorporates local spatial information from the map, as well as object descriptions to infer place and topological decomposition. The work by Rogers and Christensen

[36] uses a probabilistic approach to place categorization that incorporates the presence of object instances from various categories, in the context of object search (discussed in more detail below). With a map of object models generated by our system, prepositional relationships between the objects could be inferred to represent the spatial arrangement of those objects relative to locations within the map. We can then reason about place in a similar way, using the object models, their attributes, and their relationships, as elements of a probabilistic graphical model. We present a very preliminary investigation into the feasibility of using object class and attributes in performing place categorization in Appendix A.

6.2.3 Direct Object Search

The representation we advance with this work, an inventory of object models and descriptions, could be applied to the problem of direct object search. In this scenario, we wish to be able to provide the robot with a query object that we want it to find in the environment. This problem is addressed in [36] and [37], using either an object from a fixed database of models or a small set of top-down attributes on objects without class labels, respectively. The system we present could enable more natural human-robot interaction by providing context so the robot can search for an object based on natural language descriptions. So instead of a query for "medium-sized things in the cafeteria", our representation could permit a query for a "plastic chair" or a "shiny, clear, bottle that looks like a vertical cylinder".

One of the major simplifying assumptions of [37] is that the map of the environment is known *a priori*. If we consider the problem of object search in unknown environments, then contextual information derived *from* the objects can guide the robot and improve the results, and the problem of object search in an unknown environment is strongly tied to the place categorization problem addressed above. The approach in [36] utilizes a probabilistic model of object and place co-occurrence to simultaneously perform place categorization and refine the object search using cues

(presence of objects from its database) that it extracts online from the environment as it explores. The limiting factor in this work is the constrained nature of an object database. Our *attributed object map* representation could bridge the gap between these two approaches, providing a means of observing previously unknown objects, but then using those objects and their descriptions to guide the robot in its search.

6.3 Future Work

6.3.1 Methods

For the *attributed object map* representation to be useful to an autonomous robot, the gathering of data and modeling of objects would need to be performed effectively either in a fully passive way, or using a guided approach with active observations of the object. In the work presented here, we only address the modeling, and not the guidance problem, and thus provide our modeling system with at least the viewpoints from which we desire detections. In this way, a passive object detection and modeling system can operate effectively, since the teleoperator of the robot is able to ensure that the robot observes the desired objects. However, some preliminary experiments in autonomous object modeling and mapping indicate that a fully passive approach is not sufficient for modeling objects in real human environments.

We performed some trials with an iRobot Packbot running our deployed stairway mapping system in three modes: passive, semi-active, and active modes. Passive mode does not modify the path planning algorithm, and explores the space as it would with its primary goal of building an occupancy grid map. Semi-active mode operates in passive mode until an object of interest is detected, at which point the robot's path planning goals are set aside in favor of observation of the object from several perspectives, until the model converges. Active mode forces the robot to observe all

of the obstacles in the environment using its camera sensor, which has a much smaller field of view than the laser scanner it uses for mapping. But similarly to the semi-active mode, an observation of an object triggers a transition to a state in which the robot moves to multiple viewpoints to further observe the detected object.

Anecdotally, the only approach that worked effectively for the stairway object class was the fully active mode. We do not present experimental results here, and much work remains to be done in developing this approach, but our preliminary results indicate that the locations of stairways in many man-made environments are such that they are unlikely to be discovered unless the robot purposefully observes them. Often, stairways in indoor environments are found off of hallways, behind doors, or occluded by walls, so although the robot may clear all of the open space in the map, unless any stairways passed through the field of view of the the camera, they will remain unobserved. Our active approach seeks to maximize the detector's ability to observe the object by at least placing all of the obstacles in the map in the field of view of the camera as the robot explores. In order to enable our system to be of use in autonomous robotics scenarios, we intend to further investigate this problem.

6.3.2 Software

The discussion of possible applications and autonomous controllers indicates the potential for further work in both improving and using the system presented here. We intend to publicly release this software package as a ROS stack, so that other researchers can apply this representation to their own problems.

Among the challenges in engineering a software package that is useful for the community is making it robust, flexible, and easy to use. Although care has been taken in designing our system such that it provides these qualities, it is decidedly optimized for our purposes, and not set up to work

out of the box for other applications. The ROS computing framework allows for some straightforward modifications that could enable some of this desired flexibility, in its use of plugins and nodelets. In particular, refactoring our code to use plugins would allow modules such as foreground segmentation, and class models to be loaded or unloaded dynamically depending on the situation. Currently, the system needs to know *a priori* the classes of objects it should be looking for, and that set remains fixed during the life of the process. Doing this dynamically could allow for more detailed investigation with more object types when the context required it, but more computationally efficient search with a smaller set of objects in other situations. Use of plugins/nodelets could also streamline some of the computation by enabling more intra-process sharing of data, and a reduction of redundant computations.

Additionally, many existing, publicly released software packages could benefit from the information extracted from the environment by our system. Integration with some of these existing systems for representation (e.g. KnowRob [28]) or application (e.g. [36]) could immediately enable many people in the community to make use of the proposed system in their own research.

Appendix A

ImageCLEF Robot Vision Challenge

We performed an initial proof of concept experiment in the use of objects and attributes for place categorization that was motivated by the ImageCLEF 2012 Robot Vision Challenge. This semi-annual performance benchmark competition requires entrants to perform visual place classification on a sequence of images captured by a mobile robot that has been navigated around an indoor office environment. ImageCLEF provided several sequences of labeled training data, and this challenge presented a natural opportunity to investigate some of the preliminary problems in designing an attributed object map system.

We experimented with the same general workflow proposed here: detect objects, perform attribute classification on them, and then use these data to infer the location class of the image. Since there are multiple instances of some of the room types, this problem is one of visual place *categorization*. In this challenge, due to the nature of the data (camera and depth images with place labels), we were restricted to an image-wise place classification problem, but in our overall system, inference about place will be possible with objects detected and modeled over multiple observations instead of single images. In addition to improving on the sparse object detections that occur in individual images, using multiple observations permits the object models to be more robust and consistent in their attributes, and allows the system to capture spatial relationships between the objects as well. As the following results indicate, image-wise object detection presents some significant limitations

to the proposed approach. However, the extension to object modeling for attributed object maps addresses all of these shortcomings.

A.1 Overview

Instead of approaching this competition with an interest in producing the highest performing system possible, we were interested in using the structured problem and provided data as a testbed for the ideas we intended to extend to a more general representation with this proposed work. Most of the existing methods that have been proposed for the problem of place categorization utilize low-level features [2, 16], and in fact the ImageCLEF organizers provide code for extracting Pyramid Histogram of Gradients (PHOG) features and Normally Aligned Radial Features (NARF). In testing the high level idea we propose here—using attributed objects for place categorization—we took a very different approach of extracting semantically meaningful meta-features from the input data, and using those features for classification.

Since this experiment was not intended to contribute directly to the final system, and since the performance of this approach in the competition was not our primary concern, we took an unoptimized approach that used as many "out of the box" components as possible, and we did not tailor our approach to the data or the parameters of the competition. To that end, we used existing modules for object detection—Deformable Parts Models (DPM) from Histogram of Oriented Gradients (HOG) features [69, 114]—and for object attribute classification [10]. Rather than design a new classification technique for this particular problem, we used a Bag of Words model (*Bag of Attributes*) and a simple naive Bayes classifier for place labeling. All of these components were used "off the shelf", with no performance tweaking, but they allowed us to test the feasibility of using attributed objects almost immediately, without having to collect or annotate data.

However, the downside to this "out of the box" approach is that the available object detectors

Table A.1: List of attributes from [10]

2D Boxy	3D Boxy	Round	Vert Cyl	Horiz Cyl	Occluded
Tail	Beak	Head	Ear	Snout	Nose
Mouth	Hair	Face	Eye	Torso	Hand
Arm	Leg	Foot/Shoe	Wing	Propeller	Jet engine
Window	Row Wind	Wheel	Door	Headlight	Taillight
Side mirror	Exhaust	Pedal	Handlebars	Engine	Sail
Mast	Text	Label	Furn. Leg	Furn. Back	Furn. Seat
Furn. Arm	Horn	Rein	Saddle	Leaf	Flower
Stem/Trunk	Pot	Screen	Skin	Metal	Plastic
Wood	Cloth	Furry	Glass	Feather	Wool
	Clear	Shiny	Vegetation	Leather	

and attribute classifiers were not designed for this task. In particular, the attribute classifiers we obtained from Farhadi et al. [10] relied on a Pascal Visual Object Challenge (VOC) dataset, and consequently were restricted to the VOC object class set. Although these classes include several objects that could be found in the ImageCLEF data, including *bottle*, *chair*, *diningtable*, *pottedplant*, *sofa*, and *tvmonitor*, the majority of the classes are vehicles and animals, which do not appear. As a result, we restricted our object detectors to this set of six relevant classes for which we had "off the shelf" attribute classifiers. Similarly, this set of attributes (see Table A.1) was designed to describe the parts, shape, and material of all of the VOC classes, and thus many of these attributes are also not very applicable to the indoor object classes. However, even the attribute classifiers that produce erroneous labels (e.g. *engine* for furniture classes) do so consistently, so whatever properties of the objects these classifiers are actually identifying can also be detected consistently, even if the semantic label for the attribute seems misguided. The limitations of this approach restrict the generalization that is possible from the results, but our trial-wise comparisons allow for some insight on the discriminating power that is provided by using attributes as meta-features.

A.2 Methods

We performed object detection for the six indoor object classes from the VOC set—*bottle*, *chair*, *diningtable*, *pottedplant*, *sofa*, and *tvmonitor*—using the discriminatively trained, deformable parts models of [114]. The trained model for each class includes a positive detection threshold, t, that was learned from the training data. We found these learned values to be too restrictive for the ImageCLEF data—they produced very few detections in the often cluttered office environment—so after a small amount of heuristic investigation, we chose to use a detection threshold of $0.75t$ for each model, as it produced consistent detections for many of the true positives in the dataset, with a tolerable false positive rate. We performed multi-scale object detection with these models and recorded the object class and bounding box for each detected object in the dataset. See figure A.1 for images with several examples of these object detections. Figure A.2 shows the overall distribution of the ~ 2900 detected objects across the ~ 7500 image training dataset; due to the restricted set of object classes and the nature of the environment captured in the dataset, the overwhelming majority of the detected objects were *chairs* and *tvmonitors*. However, the few *bottles* and *sofas* present in the environment were detected consistently and at approximately the same true positive rate: there were only 2 *sofa* instances in the visited rooms, but ~ 100 chairs.

Another important aspect of the data besides the unequal distribution of object classes is that the distribution of object instances in the environment across the 9 location classes is significantly skewed towards the *LoungeArea, ProfessorOffice* and *StudentOffice* classes (see Fig. A.3). In the environment, there are few objects at all in some of the place classes (e.g. *ElevatorArea, Corridor, PrinterRoom, Toilet*), and these objects did not fall into the small set of object classes we've used. Consequently, the vast majority of the images with detected objects were from the "office" type rooms, which were additionally more present in the image set, resulting in a disproportional representation in the data used to train our models. Again, a more tailored approach could have

Figure A.1: Object detection examples from the ImageCLEF dataset. Bounding boxes are color coded for object class: *bottle*: yellow, *chair*: blue, *sofa*: green, *tvmonitor*: red.

compensated for this imbalance in data, but it was important to maintain the "out of the box" approach.

The detected object bounding boxes were used to extract the region of interest (ROI) for each object, and these ROIs were then labeled with attributes. The attribute classifiers of [10] are SVMs learned from features selected by L1-regularized logistic regression for distinguishing object classes with and without a particular attribute. We use the default threshold of 0 to classify each attribute to form a binary vector of length 64 for each described object.

In addition to the attribute set from [10], we also added some heuristically determined attributes from HSV color space. In the interest of introducing some color attributes at about the same level

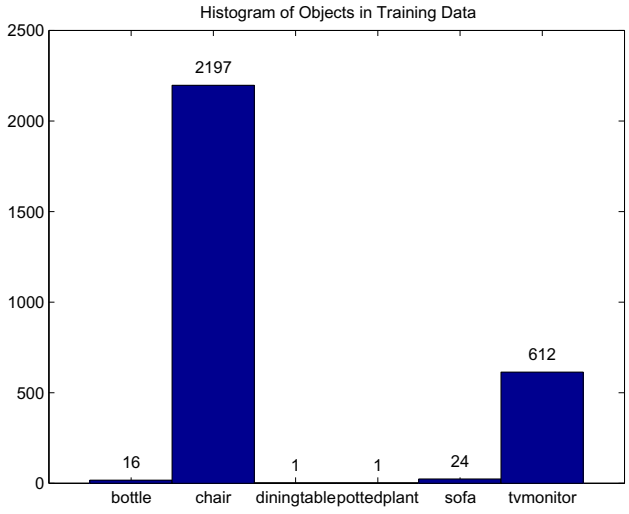

Figure A.2: Most of the objects detected in the ImageCLEF training dataset were of the *chair* and *tvmonitor* classes.

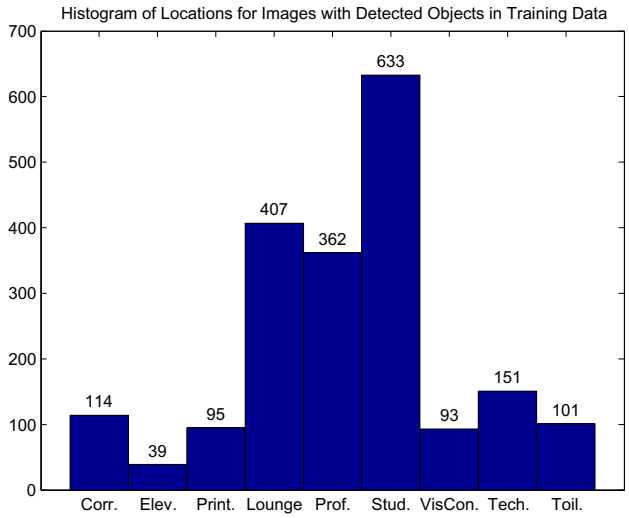

Figure A.3: Histogram of location classes for images with detected objects in the ImageCLEF training data. Although the 9 locations are not equally represented in the training data as a whole, the sparsity of detectable objects in some of the place classes (*ElevatorArea, Corridor, Toilet*) further skew the distribution of image classes toward the other classes.

of semantic abstraction as the regular attributes, but without doing any proper training, we use histogram thresholding to produce a labeling of the rough color properties of a detected object. To compute these, the object ROI is converted to HSV color space and coarse histograms are generated for the hue and value channels. The hue channel is binned into approximate regions corresponding to *red, yellow, green, cyan, blue, magenta* and the value channel is binned into *black* $[0-0.25)$, *gray* $[0.25-0.5)$, *light color* $[0.5-0.75)$, and *bright color* $[0.75-1]$. Each attribute was labeled positively if the histogram count for that bin was greater than a threshold percentage of the total pixels in the ROI. For the hue attributes, we required $> 25\%$ of the total ROI, and for the value attributes, $> 30\%$. In other words, a particular bin would have to be disproportionally represented to receive the corresponding label, but it would be possible for an object that is predominantly blue and green to be labeled with both attributes. This attribute vector of length 10 was then concatenated to the regular attribute vector, for a total of 74 semantic attributes, or meta-features, describing each object. Since we are performing image-wise classification for the ImageCLEF challenge, we pool the words for all of the objects in each image, for both training and testing, so each datum represents the attributes that describe an image.

We used a Bag of Words model for place classification, and we trained a naive Bayes classifier for each of several different variations of attributes used as the "words" in the model. The conditional independence assumption of this model is not appropriate for all attributes since many are highly correlated (although many are in fact independent), but we discard these relations in favor of a simpler model. In order to evaluate the discriminating power of each class of attributes (regular and color) and the benefits of attributes in general, we trained classifiers for the following "dictionaries"/trials:

1. Object class only - 6 words
2. All attributes without objects - 74 words
3. Regular attributes without objects - 64 words

4. Color attributes without objects - 10 words

5. All object-attribute pairs - 444 words

6. Regular object-attribute pairs - 384 words

7. Color object-attribute pairs - 60 words

For the attributes without objects, the attributes of all objects in a scene were pooled, discarding their object class. This word representation essentially describes the whole image in terms of the attributes of all of the detected objects in it. For the object-attribute pairs, each object-attribute combination is a distinct word: "red chair" is different than "red sofa" and than "3DBoxy chair". Place labeling was performed by processing the input image in the same manner as the training data—object detection, attribute classification, and image-wise attribute pooling—and then classifying it based on which place class achieves the maximum posterior for producing that observation of objects and/or attributes. We performed each of these trials using the classifiers trained on the three training runs, and tested for accuracy on a manually labeled test run. The results are summarized in Table A.2 and discussed in detail throughout the following section, with a quantitative comparison following the presentation of all of the trials.

A.3 Experimental Results

In our first trial, we used only the object classes as our feature set. Figure A.4 shows histograms of detected object classes for each location. In general, each place exhibits marginally different relative frequencies of *chairs* and *tvmonitors*, with a few places showing consistent detections of one of the less frequent classes (e.g. *sofas* in the *LoungeArea*). With only 6 words and really only two of those object classes detected with high frequency, we would expect this trial to perform poorly, and indeed it does. Figure A.5 shows that almost all of the test images have been labeled

as *StudentOffice* because it is over-represented in the training data relative to the other classes, and because the detections within that place are also of higher frequency than the other places for the two common object classes. Consequently, for images with sparse object detections, or with only *chairs* and *tvmonitors*, the *StudentOffice* class maximizes the posterior. The handful of test images that are classified as *LoungeArea* or *VisioConference* almost all contain *sofa* or *bottle* detections, respectively. For the *LoungeArea*, because *sofas* are found exclusively in this place, consistent with the training data, the few *sofa* detections are correctly classified. However, for the *VisioConference* class, *bottle* detections were not exclusive to this place in the training data, and their detections in the test data are even less biased towards this location class. Consequently, the naive Bayes model classifies most images with bottle detections into *VisioConference* erroneously. These results indicate that the class of detected objects alone does not provide sufficient discriminating power for this task, and they motivate the use of other high level features.

Our next set of trials considered the use of attributes alone, independent of the objects they describe. The attributes still require object detections in the image, but by pooling the attributes over all of the objects detected in the scene, they behave more as scene-level descriptors. Semantically, this approach does not necessarily provide the right context for the attributes, since the object class they refer to is important for the interpretation of each descriptor (e.g. Furniture Back is a meaningful and informative attribute for a *chair* object, but for a *bottle* would be a false positive). However, the histograms in Fig. A.6 show the relative distributions of attributes for the four object classes with multiple detections in the training data, and they illustrate that each object class exhibits a different distribution of attributes. Therefore, the distinguishing characteristics of each object class may come through in the attributes alone.

The place class-conditional histograms in Fig. A.7 illustrate that although many of the locations exhibit similar relative frequencies of some of the attributes, some of the subtle differences help to distinguish the classes. For example, the *ProfessorOffice* location exhibits much higher frequency

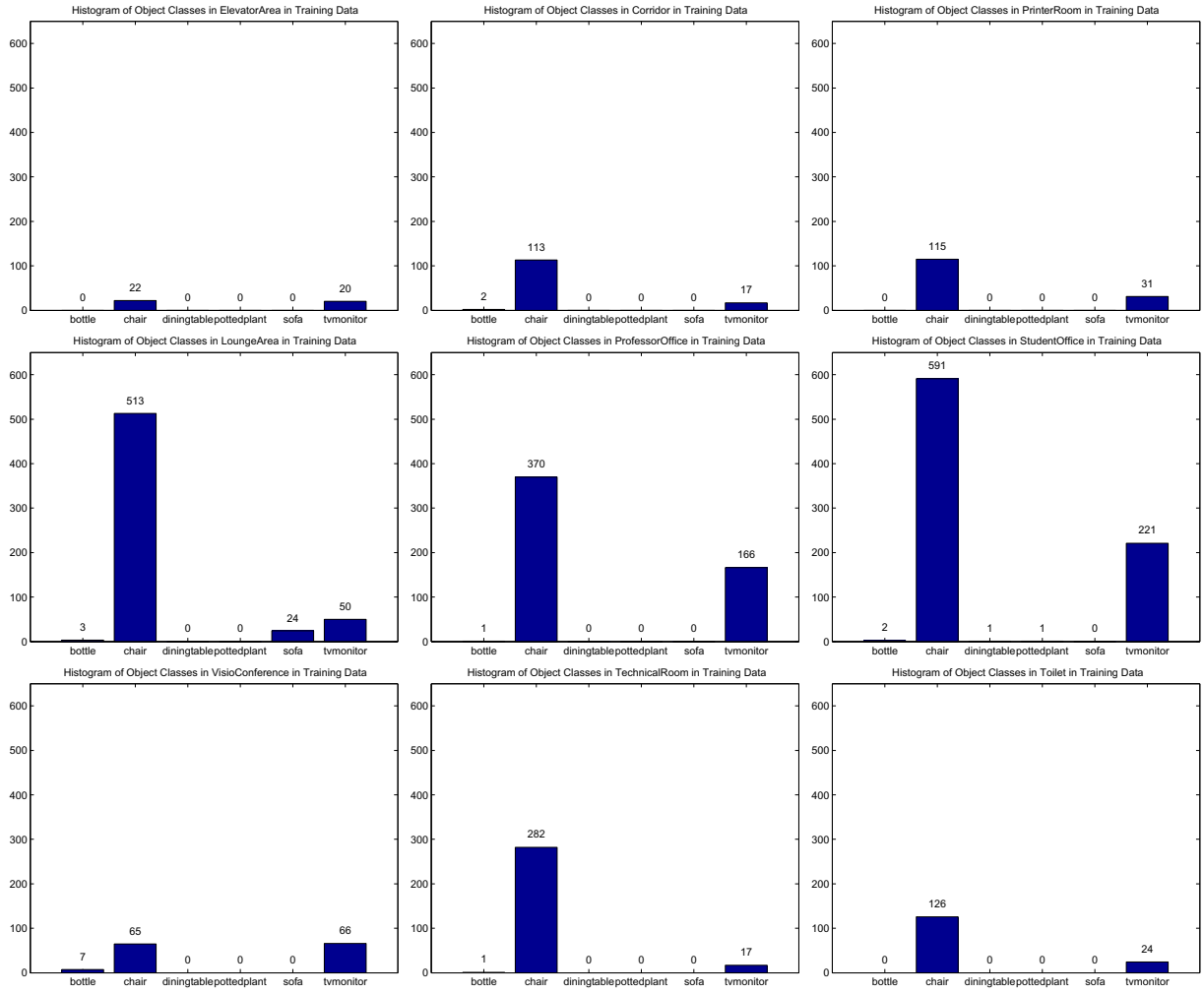

Figure A.4: Class-conditional distributions of object classes in the ImageCLEF training dataset.

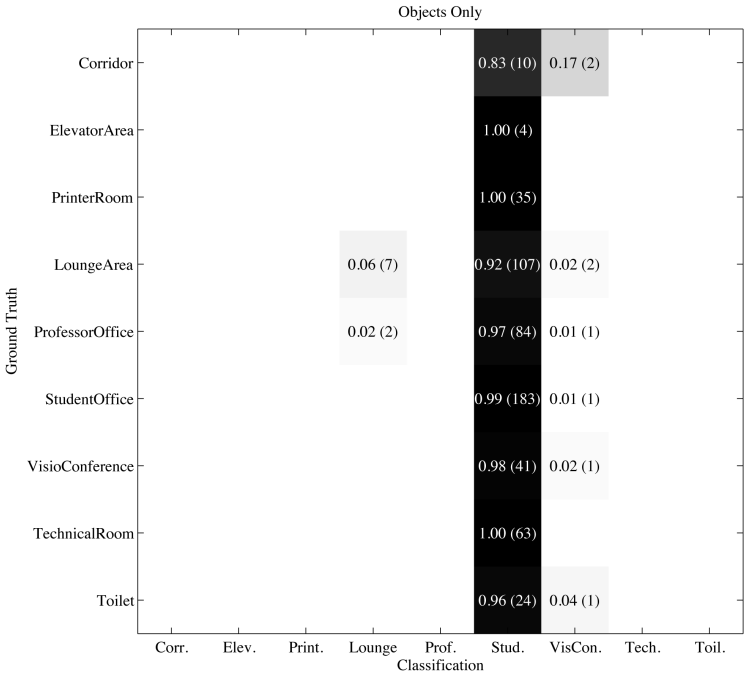

Figure A.5: Confusion matrix for Trial # 1: Objects only.

of the attribute *wood* relative to *plastic*, compared to the *StudentOffice* class. This is consistent with the furniture in the *StudentOffice* rooms being primarily plastic and more of the furniture in the *ProfessorOffice* rooms being made of wood. Although with object class included, we might be able to distinguish *plastic chairs* from *plastic tvmonitors*, the attributes alone do provide some discriminating power. The confusion matrices in Fig. A.8 illustrate that the combination of both regular and color features at the scene level are capable of distinguishing the well represented place classes reasonably well.

Our final set of trials considered both the object class and attributes by implementing object-attribute pairs as the features used for training the classifiers. Here, the attributes describe only their associated object, and there is no pooling. The plots in Fig. A.6 illustrate that the object classes exhibit different detectable properties. However, we want to exploit the inter-place-class variance of these attributes, so we must consider them on a place class-conditional basis, as in Fig. A.9. Although subtle, the distributions of object-attribute pairs exhibit different relative frequen-

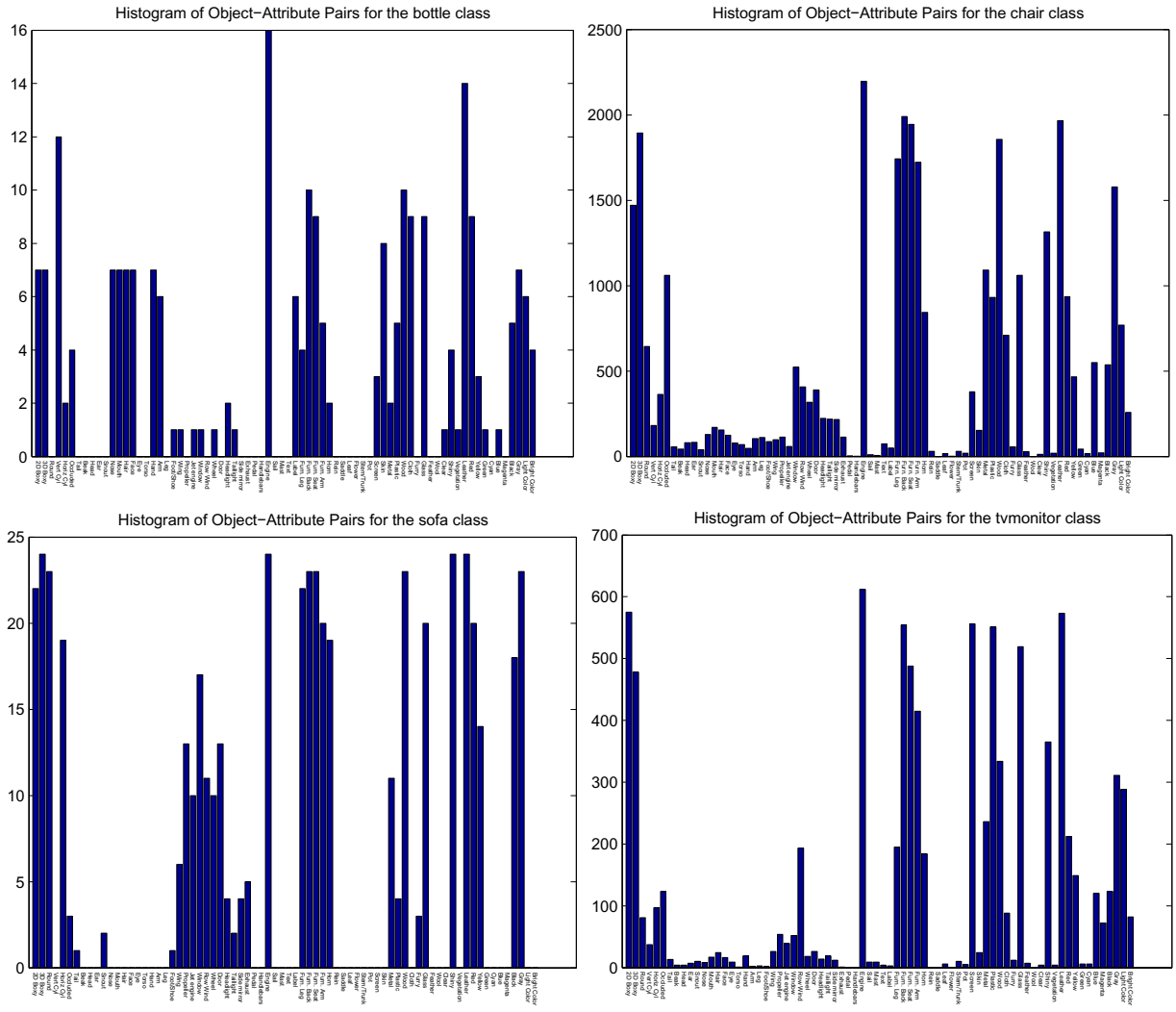

Figure A.6: Object class-conditional histograms of object-attribute pairs in the ImageCLEF training dataset. Note: plots for the *diningtable* and *pottedplant* classes have been omitted because they reflect only a single detection.

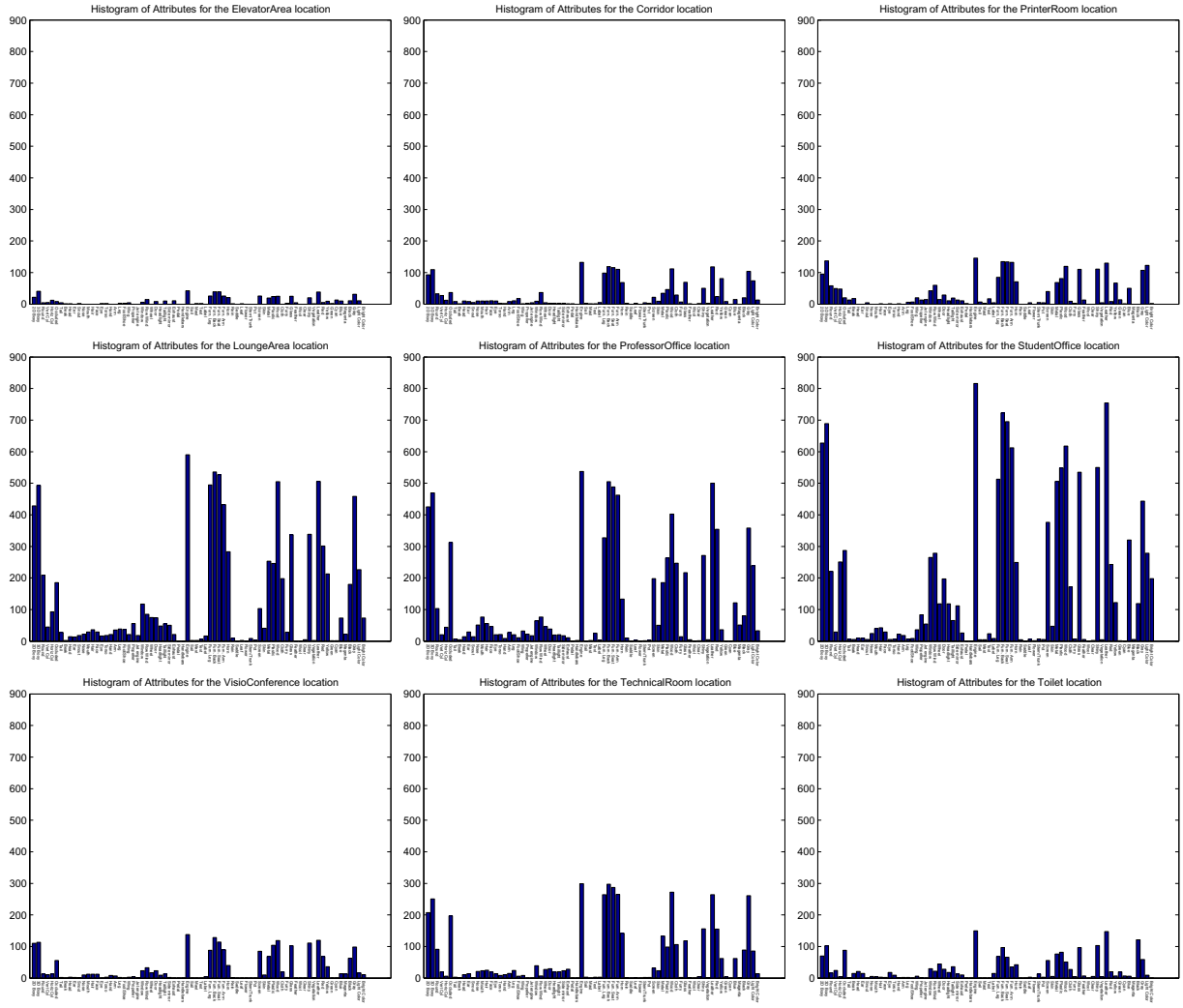

Figure A.7: Class-conditional histograms of all object attributes in the ImageCLEF training dataset.

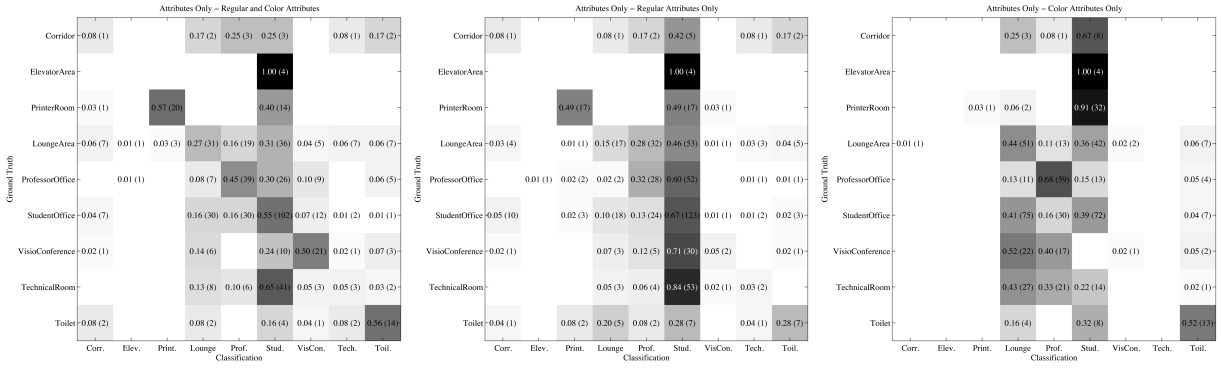

Figure A.8: Confusion matrices for Trial # 2: All attributes without objects (left), #3: Regular attributes without objects (center), and #4: Color attributes without objects (right).

cies of some of these meta-features, allowing for objects of the same class in different locations to provide some discrimination. The confusion matrices in Fig. A.10 for trials with regular, color, and both regular and color object-attribute pairs. Color attributes, in particular, seem to help in distinguishing the three highly represented location classes from one another in Trial #7, compared with the performance on these classes in the other trials. Some of this discriminating power is subdued when combined with the regular attributes, but the overall performance is roughly comparable to the attributes-only trials, albeit with some differences in strengths and weaknesses among the individual location classes.

The quantitative accuracy measures for all 7 trials are collected in Table A.2. We have computed the average precision and average recall across all location classes, as well as the overall image-wise accuracy. We compare these statistics to the performance of random labeling and class-biased labeling (always choosing the most frequent location class from the training data). Although Trial #1 produces higher overall accuracy than some of the attribute-based trials, the classifier ends up essentially performing class-biased labeling (almost all images labeled as *StudentOffice*), and the resulting class-wise average precision is approximately the same as random labeling. All of the attribute based trials showed significantly better class-wise performance than random, although they do not always do better in overall accuracy than the class-biased approach. Trial #2, using both regular and color object-agnostic attributes, achieves the best class-wise precision on average at 33.7%. However, considering only the three well represented location classes (*LoungeArea, ProfessorOffice*, and *StudentOffice*), the highest performance comes from Trial #7 (Color object-attribute pairs) with 54%. On the other hand, Trial #5 (all object-attribute pairs) demonstrates the highest average class-wise recall. Here it seems that attributes alone offer the better predictive power, but incorporating the object class permits several of the location classes to achieve much higher recall rates (e.g. *PrinterRoom* and *ProfessorOffice*, at the expense of clustering most of the false positive detections in the *VisioConference* class.

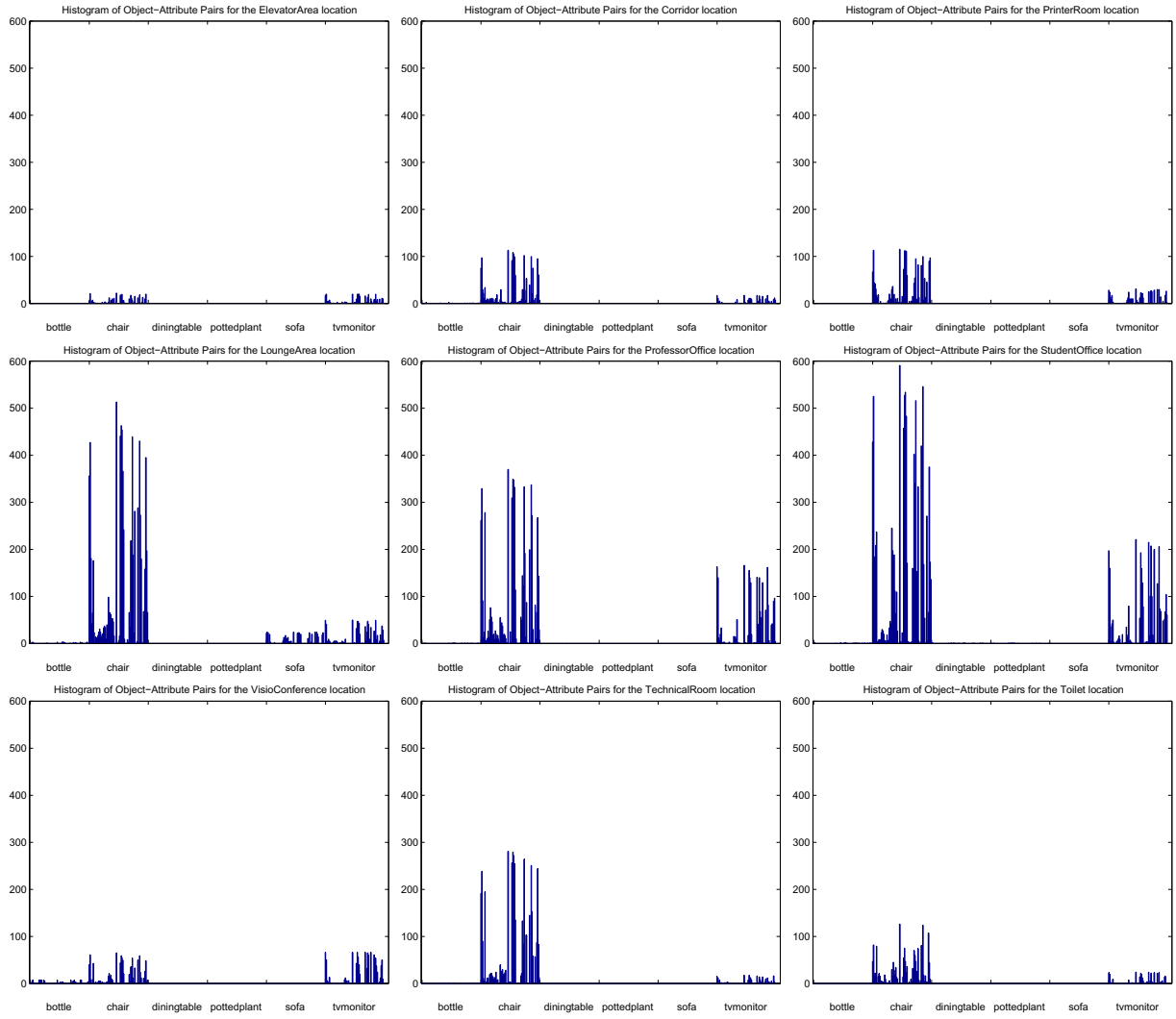

Figure A.9: Class-conditional histograms of object-attribute pairs in the ImageCLEF training dataset.

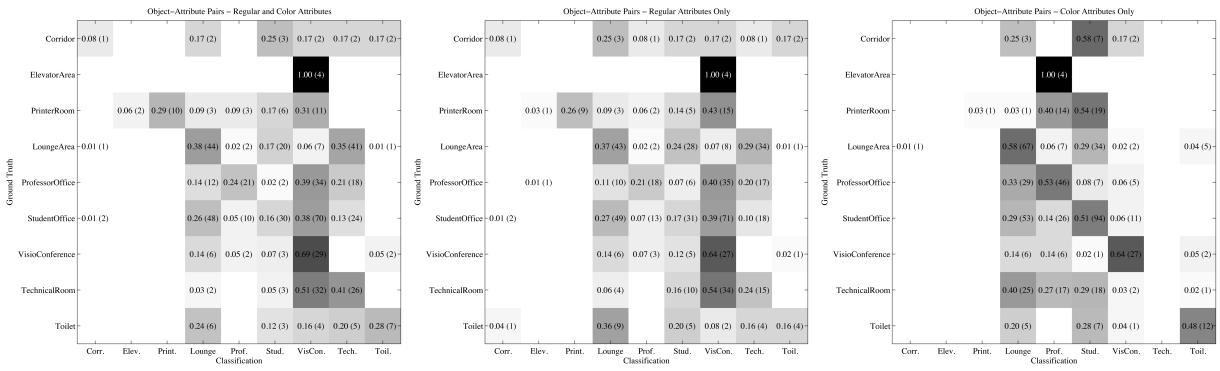

Figure A.10: Confusion matrices for Trial #5: All object-attribute pairs (left), #6: Regular object-attribute pairs (center), and #7: Color object-attribute pairs (right).

However, in terms of overall performance, Trial #7 achieves the second highest precision and the second highest recall, but due to superior performance on the three most common classes, performs better than the other trials by up to 60%. This performance may be more due to the nature of the dataset, and not the superiority of using only color and object class. The bias of the detected object distribution and the bias of the locations toward the *LoungeArea, ProfessorOffice*, and *StudentOffice* classes makes the color attributes more powerful in distinguishing those particular classes due to the object instances in those locations: the *chairs* and *tvmonitors* in those rooms have distinct class-wise colors. One other interesting observation from these results is that the use of both regular and color object-agnostic attributes performs better in all metrics than either regular or color attributes alone. Regular and color object-attribute pairs, however, do not appear to be more than the sum of their parts; color pairs alone outperform all pairs slightly in precision and significantly in overall accuracy. Although there are no other results yet released on the ImageCLEF 2012 data, and this collection of approaches has not yet been applied to any other datasets, the performance is approximately consistent with some other published results. Performance of several methods on the Visual Place Categorization dataset (5 or 11 room categories with data from 6 homes) is in the $35\% - 45\%$ range [115]. However, the use of more sophisticated, multi-modal systems enables much better performance than visual perception alone; low-level features plus room-level properties inferred from laser scan data, as well as object instance detection, are able to achieve results in

Table A.2: Table of Accuracy for Experimental Trials. Random labeling provides any image a $\frac{1}{9}$ chance of being correct labeled, whereas always choosing the label of the most frequent class from the training data yields higher overall accuracy. The highest performing trials for each metric are in **bold**.

Trial	Avg. Precision	Avg. Recall	Overall Acc.
1: Objects Only	0.1199	0.1372	0.3363
2: All Attributes w/o objects	**0.3369**	0.3468	0.4067
3: Regular Attributes w/o objects	0.2295	0.2926	0.3468
4: Color Attributes w/o objects	0.2313	0.3076	0.3468
5: All Object-Attribute Pairs	0.2818	**0.3941**	0.2958
6: Regular Object-Attribute Pairs	0.2363	0.3547	0.2606
7: Color Object-Attribute Pairs	0.3076	0.3756	**0.4349**
Random	0.1111		
Class-biased			0.3239

the $80-85\%$ range on place classification indoor office environments [16].

A.4 Conclusions

All of these results must be taken with a grain of salt. The image-wise classification problem is different than our intended problem, which includes continuity over multiple observations, and with the use of RGB-D data, we will really be approaching this from a multi-modal perspective. Although we approached the problem without any assumptions about the nature of the data (other than the presence of *some* objects), the generality of these methods have not been robustly established due to the limitations imposed by our "out of the box" approach. This restriction resulted in a minimal set of object classes and a set of general attributes, only some of which were relevant for those classes, as well as an overly simple classification model. These factors limited the overall effectiveness of these methods for visual place classification.

However, even in this restricted setting, we can extract some insights into the use of objects and attributes for higher level understanding of the robot's environment. Objects alone don't provide

enough discriminating power in this scenario—they achieve decent overall accuracy because of the class bias, but average precision is approximately random—but the addition of attributes (paired with objects or not) improves the class-wise accuracy, even if the improvement in overall accuracy is marginal. Even with an unoptimized approach on this fairly biased data, the results suggest that attributes provide semantically meaningful high-level information about the scene that is also useful in discriminating between location classes. The hope is that with a more tailored approach (more problem-specific classes and attributes), continuity of observations (localized models rather than single detections), and more spatially-aware inference methods, these principles can be made much more effective.

Bibliography

[1] A. Pronobis, O. Martinez Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 298–320, 2010.

[2] F. Ramos, B. Upcroft, S. Kumar, and H. Durrant-Whyte, "A Bayesian approach for place recognition," *Robotics and Autonomous Systems*, vol. 60, no. 4, pp. 487 – 497, 2012.

[3] E. L. Newman, J. B. Caplan, M. P. Kirschen, I. O. Korolev, R. Sekuler, and M. Kahana, "Learning your way around town: How virtual taxicab drivers learn to use both layout and landmark information," *Cognition*, vol. 104, no. 2, pp. 231–253, 2007.

[4] J. Bullens, M. Nardini, C. F. Doeller, O. Braddick, A. Postma, and N. Burgess, "The role of landmarks and boundaries in the development of spatial memory," *Developmental Science*, vol. 13, no. 1, pp. 170–180, 2010.

[5] X. Han, P. Byrne, M. Kahana, and S. Becker, "When do objects become landmarks? A VR study of the effect of task relevance on spatial memory," *PLoS ONE*, vol. 7, no. 5, May 2012.

[6] A. L. Shelton and T. P. McNamara, "Systems of spatial reference in human memory," *Cognitive Psychology*, vol. 43, no. 4, pp. 274–310, 2001.

[7] A. Collet, M. Martinez, and S. S. Srinivasa, "The MOPED Framework: Object recognition and pose estimation for manipulation," *The International Journal of Robotics Research*, vol. 30, no. 10, pp. 1284–1306, 2011.

[8] M. Sun, S. Kumar, G. Bradski, and S. Savarese, "Toward automatic 3D generic object modeling from one single image," in *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, May 2011, pp. 9 –16.

[9] V. Ferrari and A. Zisserman, "Learning visual attributes," in *Neural Information Processing Systems (NIPS)*, 2008.

[10] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, "Describing objects by their attributes," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2009.

[11] C. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 951 –958.

[12] J. Wu, H. Christensen, and J. Rehg, "Visual place categorization: Problem, dataset, and algorithm," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 4763 –4770.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results," http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[14] O. M. Mozos, R. Triebel, P. Jensfelt, A. Rottmann, and W. Burgard, "Supervised semantic labeling of places using information extracted from sensor data," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 391 – 402, 2007.

[15] A. Nuchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915 – 926, 2008.

[16] A. Pronobis and P. Jensfelt, "Understanding the real world: Combining objects, appearance, geometry and topology for semantic mapping," KTH Royal Institute of Technology, CVAP/CAS, Stockholm, Sweden, Tech. Rep. TRITA-CSC-CV 2011:1 CVAP319, May 2011.

[17] A. Ranganathan and J. Lim, "Visual place categorization in maps," in *IEEE/RSJ International Conference onIntelligent Robots and Systems (IROS)*, Sept. 2011, pp. 3982 –3989.

[18] C. Nieto-Granda, J. Rogers, A. Trevor, and H. Christensen, "Semantic map partitioning in indoor environments using regional analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 1451 –1456.

[19] D. Anguelov, R. Biswas, D. Koller, B. Limketkai, and S. Thrun, "Learning hierarchical object maps of non-stationary environments with mobile robots," in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 10–17.

[20] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernandez-Madrigal, and J. Gonzalez, "Multi-hierarchical semantic maps for mobile robotics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[21] S. Vasudevan and R. Siegwart, "Bayesian space conceptualization and place classification for semantic maps in mobile robotics," *Robotics and Autonomous Systems*, 2008.

[22] H. Zender, O. Martínez Mozos, P. Jensfelt, G.-J. M. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, 2008.

[23] S. Vasudevan, S. Gächter, V. Nguyen, and R. Siegwart, "Cognitive maps for mobile robots: An object based approach," *Robotics and Autonomous Systems*, vol. 55, no. 5, pp. 359–371, 2007.

[24] S. Ekvall, D. Kragic, and P. Jensfelt, "Object detection and mapping for service robot tasks," *Robotica*, vol. 25, no. 02, pp. 175–187, 2007.

[25] N. Blodow, D. Jain, Z. Marton, and M. Beetz, "Perception and probabilistic anchoring for dynamic world state logging," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2010, pp. 160–166.

[26] Óscar Martínez Mozos, P. Jensfelt, H. Zender, G.-J. M. Kruijff, and W. Burgard, "From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots," in *Proceedings of the ICRA 2007 Workshop on Semantic Information in Robotics (SIR)*, Rome, Italy, April 2007, pp. 33–40.

[27] M. Beetz, L. Mösenlechner, and M. Tenorth, "CRAM - A Cognitive Robot Abstract Machine for everyday manipulation in human environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2010, pp. 1012 –1017.

[28] M. Tenorth, L. Kunze, D. Jain, and M. Beetz, "KNOWROB-MAP - Knowledge-linked semantic object maps," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Dec. 2010, pp. 430 –435.

[29] R. Polastro, F. Corrêa, F. Cozman, and J. Okamoto, Jr., "Semantic mapping with a probabilistic description logic," in *Proceedings of the 20th Brazilian Conference on Advances in Artificial Intelligence*, 2010, pp. 62–71.

[30] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927 – 941, 2008.

[31] R. Rusu, A. Holzbach, M. Beetz, and G. Bradski, "Detecting and segmenting objects for mobile manipulation," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct. 2009, pp. 47 –54.

[32] R. Rusu, Z. Marton, N. Blodow, A. Holzbach, and M. Beetz, "Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 3601 –3608.

[33] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, "Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 1–6.

[34] O. Mozos, Z. Marton, and M. Beetz, "Furniture models learned from the WWW," *Robotics Automation Magazine*, vol. 18, no. 2, pp. 22 –32, June 2011.

[35] T. Morwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "BLORT - The Blocks World Robotic Vision Toolbox," in *Proceedings of the ICRA Workshop on Best Practices in 3D Perception and Modeling for Mobile Manipulation*, 2010.

[36] J. G. Rogers and H. I. Christensen, "Robot planning with a semantic map," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[37] J. Mason and B. Marthi, "An object-based semantic world model for long-term change detection and semantic querying," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 3851–3858.

[38] A. J. Trevor, J. G. Rogers, C. Nieto-Granda, and H. I. Christensen, "Tables, counters, and shelves: Semantic mapping of surfaces in 3d," in *IROS Workshop on Semantic Mapping*. Georgia Institute of Technology, 2010.

[39] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, 2008.

[40] S. Y. Bao, M. Sun, and S. Savarese, "Toward coherent object detection and scene layout understanding," *Image and Vision Computing*, vol. 29, no. 9, pp. 569 – 579, 2011.

[41] D. Meger and J. J. Little, "Mobile 3D object detection in clutter," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 4885 –4892.

[42] D. Lee, A. Gupta, M. Hebert, and T. Kanade, "Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces," in *Neural Information Processing Systems (NIPS)*, 2010.

[43] Y. Xiang and S. Savarese, "Estimating the aspect layout of object categories," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[44] B. Limketkai, L. Liao, and D. Fox, "Relational object maps for mobile robots," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005, pp. 1471–1476.

[45] K. Saenko, S. Karayev, Y. Jia, A. Shyr, A. Janoch, J. Long, M. Fritz, and T. Darrell, "Practical 3-D object detection using category and instance-level appearance models," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 793 – 800.

[46] J. Zhang, J. Xiao, J. Zhang, H. Zhang, and S. Chen, "Integrate multi-modal cues for category-independent object detection and localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 801 –806.

[47] Z.-C. Marton, D. Pangercic, N. Blodow, and M. Beetz, "Combined 2D-3D categorization and classification for multimodal perception systems," *International Journal of Robotics Research*, vol. 30, no. 11, pp. 1378–1402, Sept. 2011.

[48] Z. Marton, D. Pangercic, N. Blodow, J. Kleinehellefort, and M. Beetz, "General 3D modelling of novel objects from a single view," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, Oct. 2010, pp. 3700 –3705.

[49] K. Lai, L. Bo, X. Ren, and D. Fox, "Sparse distance learning for object recognition combining RGB and depth information," in *IEEE International Conference on Robotics and Automation*, 2011.

[50] ——, "A large-scale hierarchical multi-view RGB-D object dataset," in *IEEE International Conferenc Robotics and Automation (ICRA)*, May 2011, pp. 1817 –1824.

[51] ——, "A scalable tree-based approach for joint object and pose recognition," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[52] ——, "Detection-based object labeling in 3D scenes," in *Proceedings of the International Conference on Robotics and Automation*, 2012.

[53] M. Sun, G. Bradski, B.-X. Xu, and S. Savarese, "Depth-encoded hough voting for joint object detection and shape recovery," in *European Conference on Computer Vision*. Springer, 2010, pp. 658–671.

[54] A. Farhadi, I. Endres, and D. Hoiem, "Attribute-centric recognition for cross-category generalization," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2010, pp. 2352 –2359.

[55] D. Parikh and K. Grauman, "Relative attributes," in *IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, pp. 503 –510.

[56] D. Mahajan, S. Sellamanickam, and V. Nair, "A joint learning framework for attribute models and object descriptions," in *IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, pp. 1227 –1234.

[57] Z. Zhou, J. Bu, D. Tao, L. Zhang, M. Song, and C. Chen, "Describing human identity using attributes," in *Neural Information Processing*. Springer, 2011, pp. 86–94.

[58] K. Duan, D. Parikh, D. Crandall, and K. Grauman, "Discovering localized attributes for fine-grained recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3474–3481.

[59] J. G. Rogers, A. J. B. Trevor, C. Nieto-Granda, and H. I. Christensen, "Simultaneous localization and mapping with learned object recognition and semantic data association," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2011, pp. 1264 –1270.

[60] Y. Sun, L. Bo, and D. Fox, "Attribute based object identification," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[61] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox, "A joint model of language and perception for grounded attribute learning," in *International Conference on Machine Learning*, 2012.

[62] L.-J. Li, H. Su, Y. Lim, and L. Fei-Fei, "Objects as attributes for scene classification," in *ECCV First International Workshop on Parts and Attributes*, 2010.

[63] I. Nwogu, Y. Zhou, and C. Brown, "DISCO: Describing images using scene contexts and objects," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[64] A. Torralba, K. Murphy, W. Freeman, and M. Rubin, "Context-based vision system for place and object recognition," in *IEEE International Conference on Computer Vision*, 2003.

[65] P. Viswanathan, D. Meger, T. Southey, J. Little, and A. Mackworth, "Automated spatial-semantic modeling with applications to place labeling and informed search," in *Canadian Conference on Computer and Robot Vision (CRV)*, May 2009, pp. 284 –291.

[66] P. Viswanathan, T. Southey, J. Little, and A. Mackworth, "Automated place classification using object detection," in *Canadian Conference on Computer and Robot Vision (CRV)*, June 2010, pp. 324 –330.

[67] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte, "Classification and semantic mapping of urban environments," *International Journal of Robotics Research*, vol. 30, no. 1, pp. 5–32, Jan. 2011.

[68] G. Singh and J. Kosecka, "Acquiring semantics induced topology in urban environments," in *IEEE International Conference on Robotics and Automation*, 2012.

[69] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, 2010.

[70] H. Kang, A. Efros, M. Hebert, and T. Kanade, "Image composition for object pop-out," in *Workshops of the IEEE International Conference on Computer Vision*, 2009, pp. 681–688.

[71] N. Payet and S. Todorovic, "From contours to 3d object detection and pose estimation," in *IEEE International Conference on Computer Vision*, 2011, pp. 983–990.

[72] M. Ulrich, C. Wiedemann, and C. Steger, "Combining scale-space and similarity-based aspect graphs for fast 3D object recognition," *IEEE Transactions on Pattern and Machine Intelligence*, vol. 34, no. 10, pp. 1902–1914, 2012.

[73] A. Ückermann, C. Elbrechter, R. Haschke, and H. Ritter, "3D scene segmentation for autonomous robot grasping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[74] J. H. Joung, M. S. Ryoo, S. Choi, and S.-R. Kim, "Reliable object detection and segmentation using inpainting," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[75] M. Krainin, P. Henry, X. Ren, and D. Fox, "Manipulator and object tracking for in-hand 3d object modeling," *The International Journal of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[76] A. Janoch, S. Karayev, Y. Jia, J. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3-d object dataset: Putting the kinect to work," in *IEEE International Conference on Computer Vision*, 2011, pp. 1168–1174.

[77] E. Herbst, X. Ren, and D. Fox, "RGB-D object discovery via multi-scene analysis," in *IEEE/RJS International Conference on Intelligent Robots and Systems*, 2011, pp. 4850–4856.

[78] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, 2004.

[79] F. Endres, J. Hess, and N. Engelhard, "RGBDSLAM, ROS software package, GPL licensed," http://www.ros.org/wiki/rgbdslam, 2010.

[80] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision*, 2012.

[81] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM Transactions on Graphics*, vol. 23, no. 3, 2004, pp. 309–314.

[82] S. Oßwald, A. Gorog, A. Hornung, and M. Bennewitz, "Autonomous climbing of spiral staircases with humanoids," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 4844–4849.

[83] J.-J. Chou and L.-S. Yang, "Innovative design of a claw-wheel transformable robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[84] M. Brunner, B. Bruggemann, and D. Schulz, "Towards autonomously traversing complex obstacles with mobile robots with adjustable chassis," in *International Carpathian Control Conference (ICCC)*, 2012, pp. 63–68.

[85] K. Choi, H. Jeong, K. Hyun, H. Do Choi, and Y. Kwak, "Obstacle negotiation for the rescue robot with variable single-tracked mechanism," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2007, pp. 1–6.

[86] A. M. Johnson, M. T. Hale, G. C. Haynes, and D. E. Koditschek, "Autonomous legged hill and stairwell ascent," in *IEEE International Workshop on Safety, Security, & Rescue Robotics (SSRR)*, Nov. 2011, pp. 134–142.

[87] E. Mihankhah, A. Kalantari, E. Aboosaeedan, H. Taghirad, S. Ali, and A. Moosavian, "Autonomous staircase detection and stair climbing for a tracked mobile robot using fuzzy controller," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics*, 2008, pp. 1980–1985.

[88] R. Ray, B. Bepari, and S. Bhaumik, "On design and development of an intelligent mobile robotic vehicle for stair-case navigation," in *Intelligent Autonomous Systems*. Springer, 2010, pp. 87–122.

[89] D. Hernandez and K. Jo, "Outdoor stairway segmentation using vertical vanishing point and directional filter," in *International Forum on Strategic Technology (IFOST)*. IEEE, 2010, pp. 82–86.

[90] D. Hernández, T. Kim, and K. Jo, "Stairway detection based on single camera by motion stereo," *Modern Approaches in Applied Intelligence*, pp. 338–347, 2011.

[91] S. Oßwald, J.-S. Gutmann, A. Hornung, and M. Bennewitz, "From 3D point clouds to climbing stairs: A comparison of plane segmentation approaches for humanoids," in *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2011, pp. 93–98.

[92] V. Pradeep, G. Medioni, and J. Weiland, "Piecewise planar modeling for step detection using stereo vision," in *Workshop on Computer Vision Applications for the Visually Impaired*, 2008.

[93] X. Lu and R. Manduchi, "Detection and localization of curbs and stairways using stereo vision," in *IEEE International Conference on Robotics and Automation*, April 2005, pp. 4648 – 4654.

[94] Y. Cong, X. Li, J. Liu, and Y. Tang, "A stairway detection algorithm based on vision for ugv stair climbing," in *IEEE International Conference on Networking, Sensing and Control*, 2008, pp. 1806–1811.

[95] A. Mourikis, N. Trawny, S. Roumeliotis, D. Helmick, and L. Matthies, "Autonomous stair climbing for tracked vehicles," *The International Journal of Robotics Research*, vol. 26, no. 7, p. 737, 2007.

[96] S. Wang and H. Wang, "2D staircase detection using real adaboost," in *International Conference on Information, Communications and Signal Processing*. IEEE, 2009, pp. 1–5.

[97] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 20–25.

[98] A. Ozkil, Z. Fan, J. Xiao, S. Dawids, J. Kristensen, and K. Christensen, "Mapping of multi-floor buildings: A barometric approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 847–852.

[99] J. Hesch, G. Mariottini, and S. Roumeliotis, "Descending-stair detection, approach, and traversal with an autonomous tracked vehicle," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 5525–5531.

[100] J. Delmerico, J. Corso, D. Baran, P. David, and J. Ryde, "Toward autonomous multi-floor exploration: Ascending stairway localization and modeling," U.S. Army Research Laboratory, Tech. Rep. ARL-TR, 2012.

[101] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[102] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[103] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proceedings of the International Conference on Machine Learning (ICML)*. ACM, 2004, p. 78.

[104] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: Design of dictionaries for sparse representation," *Proceedings of Signal Processing with Adaptive Sparse Structured Representations*, vol. 5, pp. 9–12, 2005.

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

[106] OpenMP Architecture Review Board, "OpenMP Application Program Interface Version 3.0," May 2008. [Online]. Available: http://www.openmp.org/mp-documents/spec30.pdf

[107] M. Varma and A. Zisserman, "A statistical approach to texture classification from single images," *International Journal of Computer Vision*, vol. 62, no. 1-2, pp. 61–81, 2005.

[108] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005.

[109] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[110] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2, 2009.

[111] C. Dubout and F. Fleuret, "Exact acceleration of linear object detectors," in *European Conference on Computer Vision*. Springer, 2012, pp. 301–311.

[112] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote *et al.*, "Autonomous door opening and plugging in with a personal robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010, pp. 729–736.

[113] S. Friedman, H. Pasula, and D. Fox, "Voronoi random fields: extracting the topological structure of indoor environments via place labeling," in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 2007, pp. 2109–2114.

[114] P. Felzenszwalb, R. Girshick, and D. McAllester, "Discriminatively Trained Deformable Part Models, Release 4," http://people.cs.uchicago.edu/ pff/latent-release4/.

[115] E. Fazl-Ersi and J. K. Tsotsos, "Histogram of oriented uniform patterns for robust place recognition and categorization," *International Journal of Robotics Research*, vol. 31, no. 7, 2012.