Topological Mapping and Navigation in Real-World Environments

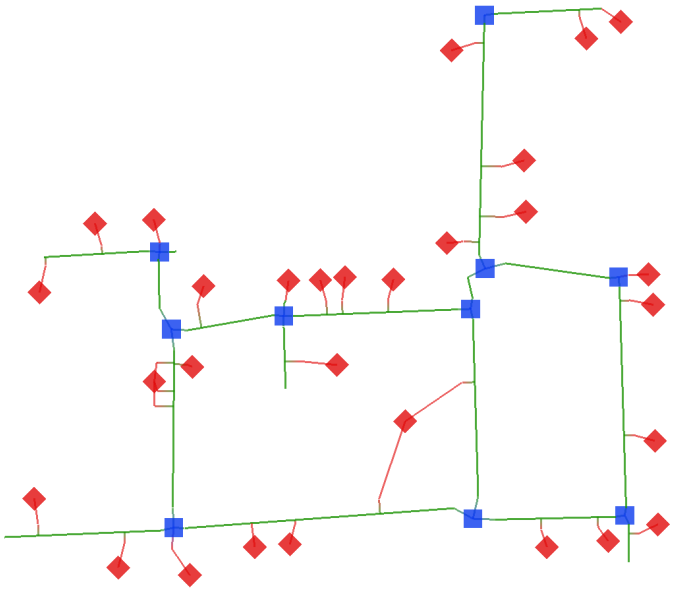
by

Collin Eugene Johnson

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering and Computer Science)
in the University of Michigan
2018

Doctoral Committee:

Professor Benjamin Kuipers, Chair
Associate Professor Odest Chadwicke Jenkins
Assistant Professor Matthew Johnson-Roberson
Associate Professor Edwin Olson



Collin Eugene Johnson

collinej@umich.edu

ORCID iD: 0000-0002-0606-8423

Dedication

To my family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

ABSTRACT

Topological Mapping and Navigation in Real-World Environments

by

Collin Eugene Johnson

Chair: Benjamin Kuipers

We introduce the Hierarchical Hybrid Spatial Semantic Hierarchy (H^2SSH), a hybrid topological-metric map representation. The H^2SSH provides a more scalable representation of both small and large structures in the world than existing topological map representations, providing natural descriptions of a hallway lined with offices as well as a cluster of buildings on a college campus. By considering the affordances in the environment, we identify a division of space into three distinct classes: path segments afford travel between places at their ends, decision points present a choice amongst incident path segments, and destinations typically exist at the start and end of routes.

Constructing an H^2SSH map of the environment requires understanding both its local and global structure. We present a place detection and classification algorithm to create a semantic map representation that parses the free space in the local environment into a set of discrete areas representing features like corridors, intersections, and offices. Using these areas, we introduce a new probabilistic topological simultaneous localization and mapping algorithm based on lazy evaluation to estimate a probability distribution over possible topological maps of the global environment. After construction, an H^2SSH map provides the necessary representations for navigation through large-scale environments. The local semantic map provides a high-fidelity metric map suitable for motion planning in dynamic environments, while the global topological map is a graph-like map that allows for route planning using simple graph search algorithms.

For navigation, we have integrated the H^2SSH with Model Predictive Equilibrium Point Control (MPEPC) to provide safe and efficient motion planning for our robotic wheelchair, Vulcan. However, navigation in human environments entails more than safety and efficiency, as human behavior is further influenced by complex cultural and social norms. We show how social norms for moving along corridors and through intersections can be learned by observing how pedestrians around the robot behave. We then integrate these learned norms with MPEPC to create a socially-aware navigation algorithm, SA-MPEPC. Through real-world experiments, we show how SA-MPEPC improves not only Vulcan's adherence to social norms, but the adherence of pedestrians interacting with Vulcan as well.

CHAPTER 1

Introduction

Unlike the well-engineered and well-mapped road networks, indoor environments, like office buildings, hospital complexes, or college campuses, contain a vast variety of geometric structures. These indoor environments are complex, easily spanning dozens of floors and containing hundreds of rooms, intersections, and hallways. Yet an environment's size only partially captures its complexity. These environments, along with smaller environments like individual homes and apartments, contain a variety of situations that vary over time or with the robot's perspective, including densely-crowded hallways, corridors with doors frequently opening and closing, cluttered rooms, furniture that occasionally moves, and specular surfaces, like glass walls and metal chair legs, that are often invisible to the robot's sensors. Addressing the scale and complexity of these environments is essential for enabling service robots to operate robustly in everyday life.

1.1 Supporting Mapping and Navigation

Recent work on simultaneous localization and mapping (SLAM) has made considerable progress towards the goal of robust robotic mapping of large-scale environments. Using a novel sensing solution, a full SLAM system has been deployed on a consumer floor-cleaning robot [1]. New algorithms for graph-based metrical mapping are robust to substantial data association errors [2, 3, 4, 5]. Appearance-based approaches can successfully detect loop closures after driving hundreds of kilometers [6] and can successfully localize across extreme environmental changes [7]. Probabilistic techniques have been applied to topological mapping [8, 9, 10] that allow mapping large environments by avoiding the exponential growth inherent in the topological mapping problem.

Despite these successes, considerable work remains to achieve the goal of long-term, large-scale mapping and navigation. Though additional problems will arise, we identify four capabilities any mobile robot navigation system must have to support long-term,

large-scale mapping and navigation. First, the SLAM algorithm used for constructing the robot's representation of the environment must be scalable over large distances. Second, this SLAM algorithm must also be scalable across long time durations. Third, the map representations and algorithms must support robust localization and mapping, both locally and globally, even in the presence of limited sensing, dynamic objects, and quasi-static changes in the environment. Fourth, the constructed map must contain enough information for the robot to plan and navigate a route from its current location to any other reachable location in the map.

This thesis introduces a new hybrid topological-metrical map representation, the Hierarchical Hybrid Spatial Semantic Hierarchy (H^2SSH), that extends the Hybrid Spatial Semantic Hierarchy (HSSH) [11], which represents the robot's environment using four distinct layers that provide metrical and topological representations of both small-scale and large-scale space. The H^2SSH improves on the HSSH by providing hierarhical representations of both local and global space that improve the scalability of the topological mapping problem. The improved representation also provides a more human-like description of the environment to be facilitate human-robot interaction.

Constructing an H^2SSH map requires a new approach to topological place detection. Our approach extends previous work in semantic place classification [12, 13] to allow identification of a set of place labels for an environment that are consistent with an underlying knowledge representation, in our case, the H^2SSH.

Once places can be detected, we show how lazy evaluation can be used for topological SLAM to avoid the expontential growth inherent in the topological mapping problem. This lazy evaluation algorithm allows scalable topological mapping of real-world environments.

In the remainder of this chapter, we first discuss the benefits and drawbacks of existing work for topological mapping that form the foundation of the contributions of this thesis. We then describe in detail the contributions outlined in the preceding paragraphs.

1.2 Topological Mapping

Topological mapping is the process of abstracting a continuous spatial environment into discrete areas, discovering the connectivity among these areas, and thus, identifying the underlying decision structure of the environment. Topological mapping abstracts the continuous experience of the robot into a discrete sequence of topological events. These events correspond to the robot's arrival at or departure from an area, which might be a hallway intersection or the region around a visually distinct landmark. A constructed topological map is a graph-like abstraction of large-scale space that represents the world as two types of

areas: places, where qualitatively distinct decisions are presented to the robot, and paths, which are connections between places. This graph-like abstraction provides a number of benefits for mapping.

Parsing the environment into small, discrete areas, as in a topological map, separates local structure from global structure. This distinction between the local and global structure of the environment allows the mapping problem to be factored into smaller, more easily solvable parts. For example, high-precision metrical mapping is only required within the robot's sensory horizon. Similarly, a topological map provides a useful representation for planning by factoring the motion planning problem into: (a) a simple graph search through large-scale space, and (b) metric motion planning in small-scale space.

A topological map is a symbolic representation whose places and paths describe the affordances (opportunities for action) in the environment. In this way, the size of a topological map is a function of the structure of environment itself and not of the trajectory the robot takes through the environment or the time spent in the environment, as is the case for the pose graphs typically used in metrical graph-based SLAM. As a result, long-term topological mapping does not require special operations to prune information from the map. Instead, as the robot repeatedly travels through an environment, the model for each area in the map can be improved as additional observations are made. For example, the improved models can allow the robot to recognize the movement of furniture or doors opening and closing.

Finally, a topological map is a human-like representation of the environment. The similarity in representation can ease human-robot interaction, as it provides natural interfaces for commanding the robot and describing the environment.

Topological maps are not suited to all mobile robot tasks, but a wide variety of robots, like telepresence robots, smart wheelchairs, and indoor delivery robots, are emerging that can benefit from a topological map abstraction. Common among these robots are operating in structured human environments and interacting with or receiving commands from a human operator. Additionally, the primary purpose of these robots is to move reliably from one place to another, as opposed to producing a metrically accurate map for further use by human operators, e.g a reconnaissance robot.

Existing topological SLAM algorithms suffer problems with scalability and robustness to change and sensing errors. These problems are not unique to topological SLAM and, as mentioned above, scalability and robustness to errors are major focuses of current metrical SLAM research [3]. However, given the difference between metrical and topological SLAM, problems with scalability and robustness manifest themselves in different ways. As a result, solutions to these problems will differ.

A topological map is a sparse representation of the environment that abstracts even large environments into a relatively small number of areas. However when performing topological SLAM, the number of possible topological maps can grow hyper-exponentially with the number of areas visited by the robot [14]. Even though a topological map is a much sparser representation of the environment than a metrical map, the number of areas visited by a robot while exploring a typical environment, like a single floor of a building, will make explicit enumeration of all possible topological maps intractable.

Avoiding the explosion of possible map hypotheses has been a major focus of research in topological mapping. Some approaches use an active exploration strategy to bound the number of potential map hypotheses by limiting the size of loops closed and using some variant of a breadth-first search through the environment [15, 16]. However, these active exploration approaches cannot be readily applied to service robots under the direction of a human because the routes taken by the robot are specified by the human user.

Recently, probabilistic algorithms for topological mapping have emerged [8, 17]. Probabilistic topological mapping algorithms incorporate metric and appearance information to focus the search for the correct map on the most probable hypotheses or to assert a loop closure only when the posterior is extremely peaked around a single place. These approaches rely on pruning the space of possible topological maps, either using a particle filtering approach [8] or explicitly discarding low-probability map hypotheses [17]. These pruning-based approaches are able to map large environments, but always risk discarding (or never generating) the correct topological map. Specifically, if an erroneous measurement causes the correct hypothesis to be removed from the hypothesis space, then the correct map can never be found.

Another problem for the use of existing topological SLAM algorithms is their assumption that the underlying topological structure of the environment is static. A door opening or closing, for example, will be detected as an inconsistency in the topological map, which results in the map being discarded by the SLAM algorithm. Consequently, topological maps become brittle in real-world environments where the state of doors is continually changing, a potentially severe limitation. A similar situation occurs when there is a false positive or false negative place detection. In both cases, the correct topological map of the environment cannot be recovered. As a result, successful topological mapping requires perfect place detection, which experience has shown to be impractical in many situations. This thesis does not solve this problem, but the presented lazy evaluation algorithm can be extended to detect and recover from place detection failures, as discussed in our future work Section 8.2.1.

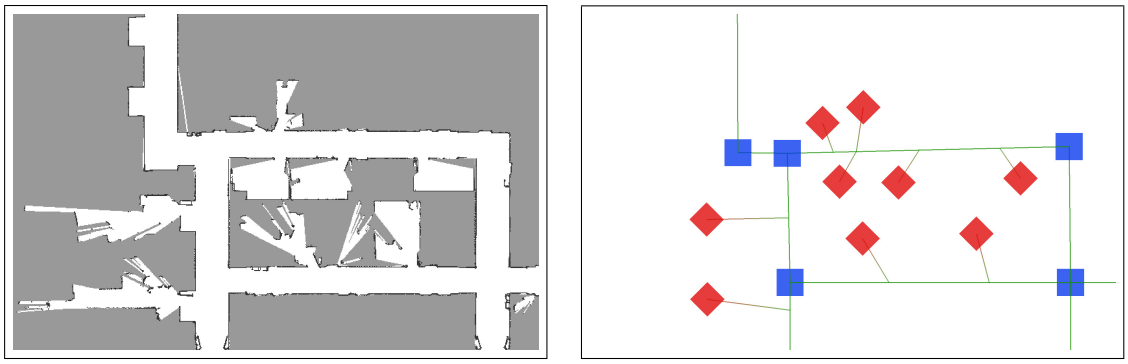
1.3 Contributions

This thesis presents solutions to the scalability problems for topological mapping. We present both representational and algorithmic approaches for dealing with the inherent complexity of the real world. Our primary contributions to topological mapping are two-fold. First, we introduce a new topological map representation better able to explain everyday environments. This representation allows us to relax many of the assumptions necessary for reliable place detection in the HSSH. To evaluate this method, we demonstrate place detection and topological SLAM in a variety of standard and newly-collected datasets. Second, we adopt an entirely probabilistic approach to both topological place detection and the global topological SLAM problem. We show how probabilistic techniques allow for robust and scalable construction of topological maps in large, non-static environments spanning multiple connected buildings.

The H^2SSH extends the HSSH representation in three significant ways. First, by considering the types of actions supported by a topological map – traveling along paths and transitioning between places – we identify a distinction between two types of places, decision points and destinations, which allows the H^2SSH to represent an environment in more detail without affecting the size of environments that can be feasibly mapped. Second, we represent path segments as more than simple graph edges connecting two places. In the H^2SSH, each path segment now contains a sequence of zero or more destinations along each side. Whereas before, each room along a corridor resulted in a decision point outside its door with a zero-length path segment between the room and decision point, we now represent this situation more naturally as a simple transition between a path segment and destination (see Figure 1.1). Third, the H^2SSH supports a hierarchy of topological maps, such that a place in one topological map can itself be a topological map. These nested topological maps provide a scalable description of common environments like multifloor and multi-building office and college campuses.

1.3.1 Place Detection and Classification

In the HSSH, Beeson et al.[11] detect and define places in small-scale space using *gateways*. A gateway specifies either the boundary of a place neighborhood along an edge of the extended Voronoi graph or the direction of unexplored space along a path segment. Gateways and obstacles are combined to define the boundary of a place. Places and paths are distinguished using a simple approach: a region containing exactly two well-aligned gateways is a path segment, and any other region, i.e. intersections and dead ends, is a place neighborhood. Furthermore, place detection occurs within a small scrolling metric



(a) (b)

Figure 1.1: (a) shows the occupancy grid of a typical office environment. The corresponding H^{2}SSH-style topological map of the building is shown in (b). The lines are path segments, the squares are decision points, and the diamonds are destinations, each office is a single destination connected to its adjacent hallway.

map centered on the robot.

The H^{2}SSH extends the HSSH representation which requires the place detection and classification algorithms to also change. Most importantly, we must now distinguish between decision points, destinations, and path segments. We use a two-step probabilistic algorithm to parse a metric map into a discrete set of non-overlapping areas. First, we identify potential gateway boundaries between areas in the environment. Second, we search the hypothesis space of areas defined by these boundaries to find a high-probability set of labeled areas consistent with the H^{2}SSH representation, for example a decision point must be at the intersection of two or more path segments.

The output of our place classification algorithm is a set of non-overlapping labeled areas that define the topological structure of the local environment. This set of areas is converted into a local topological graph, which consists of nodes that represent areas and edges that correspond to gateway boundaries between areas. As the robot drives through the environment, it moves from one node in the local topological graph to another. The transition from one node to another node occurs whenever the robot crosses a gateway boundary separating two areas.

The sequence of nodes of the graph visited by the robot as it explores the environment provides a topological description of the robot's motion through the environment. We can then use this sequence to construct a global topological map of the large-scale environment.

1.3.2 Large-Scale Robotic Mapping

We present a lazy evaluation algorithm for probabilistic topological mapping that incrementally builds and searches a tree of map hypotheses to provide a usable map hypothesis online, while still guaranteeing the correct map can eventually be found. Our algorithm annotates each leaf of the tree with a posterior probability. When a new place is encountered, new map hypotheses are expanded based on this posterior probability along with a proposal distribution drawn from local sensor data, which means that only the most probable hypotheses are expanded. By focusing on the most probable hypotheses, we dramatically reduce the number of hypotheses evaluated, allowing real-time operation and tractable mapping of large environments. Additionally, our approach never prunes consistent hypotheses from the tree, which means the correct hypothesis always exists somewhere in the tree and can eventually be recovered.

1.3.3 Socially-Aware Navigation in Dynamic Environments

Navigation through a topological map consists of route planning through the graph defined by the map followed by local planning through each place to follow the prescribed route. The ease of route planning through even massive environments is an advantage of topological maps over metric maps. Additionally, this thesis shows how topological information can improve a robot's ability to plan and navigate locally in metric maps.

The benefits to metric planning can be seen by considering the semantic information encoded in a topological map. Each place in the map defines a small, discrete set of possible actions used for navigating through the global environment. These actions are inspired by the human cognitive map of the environment [18, 19], and therefore can be used to describe how a human is moving through the global environment. Consequently, predicting the goal for a person becomes a simple estimation problem. We present a recursive Bayes solution to pedestrian goal prediction, along with improved trajectory estimation.

Similarly, the correspondence between the topological map and the human cognitive map allows the robot to understand how to move through the environment as a human would. In particular, the robot can gain an understanding of social norms associated with navigation through indoor environments – stay to the right and don't make tight left turns around corners – which allows it to move through the environment and interact with humans more naturally. We show hows these social norms can be learned from observing pedestrians around the robot and integrate them with an MPEPC-based planner [20] to create a new socially-aware navigation algorithm, SA-MPEPC. Using real-world experiments, we then show how SA-MPEPC improve the robot's adherence to social norms while nav-

igating through the environment. Our experiments also show that the robot's improved behavior also improves the adherence to social norms of pedestrians interacting with the robot.

CHAPTER 2

Background and Related Work

2.1 Topological Mapping

A topological map is a symbolic representation of space built by abstracting the robot's continuous experience of the environment into a discrete sequence of events, such as entering a place, exiting a place, or turning around on a path. Similarly, a planned route through a topological map is a sequence of actions that describe the topological decisions that must be made to carry the robot from one location in the map to another. Thus, both construction of and navigation within a topological map depend on its definition of place, or more broadly, on the way the robot understands the qualitative structure of its local surround within the environment.

Central to this qualitative understanding of the environment is the idea that the world can be represented as a set of interconnected distinctive places, a concept similar to the cognitive maps people construct of their environment [18]. Places in the topological map occur at distinctive states in the environment, while paths indicate that a navigable route exists between the places at either end.

The path representations used in topological maps are largely the same across all approaches. A path has a place at each end and indicates that traveling along the path with take the robot between these places. Some approaches, especially more recent probabilistic methods [17], annotate paths with metrical information, such as the length, whereas others, like the SSH, define a control law to use for navigation along the path. Path segments in the SSH [21] are wall-following control laws to guide the robot from one distinctive state to another. Similarly, Choset and Nakatani [22] create paths along the edges of the Voronoi skeleton equidistant to exactly two obstacles. In the HSSH, Beeson et al. [11] define path segments as parts of the world where exactly two aligned gateways exist.

The SSH [21] and the HSSH [11] extend the concept of a path to a higher-level construct in the environment. Instead of a path connecting the places at either end, a *path* consists

of a sequence of places with each sequential pair of places along the path connected by a *path segment*. Thus, a path segment in the SSH and HSSH is the same as a path in other approaches.

Unlike paths, numerous representations have been used for the place abstraction in the topological map. These representations can be broadly divided into two categories: structural places and landmark places.

A structural place is a location where the geometric structure of the robot's local environment changes in such a way that new actions are available to the robot. These locations occur at the intersection of hallways, near office doors, and in other similar situations. A key feature for structural place is that the description of the place defines the actions available to the robot at that place. The description may include the angle of an incident path [23], the branch of the Voronoi skeleton to follow [22], the control law to follow [21], or the gateway to cross [11].

A landmark place is a description of the robot's perception of the environment at a particular location, which depends on the robot's sensors. For example, the richness of a visual sensor may result in landmarks occurring in parts of the environment that are unremarkable to a laser rangefinder. The features used to represent the place varies from a bag-of-words description of visual features [6], a bubble space representation of the spherical distribution of relevant features [24], or a simple occupancy grid [8]. Two common strategies for creating landmark places are to create them at fixed intervals based on time or distance traveled [6], to find locations where sensor measurements jump unexpectedly [25], where co-visibility of features changes [26], or where a qualitative change in the feature-based description of the environment is found [27].

Based on the place representation – structural or landmark – we can classify a topological map as being either a structural topological map or a landmark topological map. The most important difference between structural and landmark topological maps is in the semantics of their places. Structural places directly encode the topological actions available in the environment, whereas landmarks, which may occur in similar locations, provide only a feature-based description of the location's appearance. A structural map, therefore, provides the robot with information on both where it is in the environment and how to plan a route and travel to a new place in the map. In contrast, a landmark map can only tell the robot where it is, leaving some other process to plan and travel routes through the map.

While the semantics of structural and landmarks maps differ, they both use the same fundamental abstraction of places and paths. We argue that this simple division of the environment into two categories is insufficient for building topological maps of large-scale environments.

Topological approaches create a symbolic description of the environment that represents vertices in the environment as discrete places corresponding to spaces like intersections and rooms. These places are connected by paths, like hallways, which form the edges in the graph. A topological map supports wayfinding using simple graph search algorithms like A*. The route found through the topological map is well-defined sequence of actions to take at each vertex.

2.2 Place Detection and Labeling

The H^2SSH introduces two categories of places, decision points and destinations. As a result, topological map construction requires not only detecting where a place is, but also what type of place it is. This problem is generally described as place classification or labeling.

Mozos et al. [12] learn a classifier for individual laser scans using AdaBoost. As the robot drives through the environment, it labels each scan the robot takes based on the classifier. Areas are then defined by continuous regions that receive the same classification. Recent work by Goeddel et al. [28] applies a convolutional neural network to Mozos' data and shows how it improves on the hand-selected features that have predominantly used to date.

One difficulty with classifying individual laser scans or local map patches is a lack of consistency between adjacent scans because information about the adjacency relations in the environment are lost. To address these inconsistency issues, Friedman et al. [13] incorporated connectivity features based on the Voronoi skeleton, like the minimum-length loop in the skeleton, with a conditional random field to improve the consistency of place classification. Shi et al. [29] introduced additional graph centrality measures of the Voronoi skeleton to further improve labeling accuracy, and Liao et al. [30] use a hierarchical voting scheme based on the branching structure of the Voronoi skeleton of an environment. An alternative approach by Pronobis et al. [31] integrates spatio-temporal information about the labels associated along the robot's trajectory through the environment to improve the consistency of the semantic labels.

Friedman et al. [13] introduced Voronoi random fields. The Voronoi random field adds information about the connectivity of the environment to the classifier based on the Voronoi graph. The connectivity information helps create a more consistent labeling of locations in the environment. Shi et al. [29] further the work of Friedman et al. by including additional features about the structure of the environment, including centrality metrics and a variety of laser-based features. We also use centrality features in our work, though we additionally

consider centrality within the visibility graph of the environment. Finally, Liao et al. [30] use the Voronoi skeleton at multiple levels of detail in conjunction with a voting scheme and automatic feature learning using deep network to achieve the highest accuracy place classification to date.

While we leverage the Voronoi skeleton for our approach, we use the connectivity information in the environment in a different way. In the above approaches, the way places are connected in the environment is an implicit part of the feature space for classification. In contrast, we have an explicit representation that specifies how places can be connected in the environment. This explicit representation ensures that strange results like corridors in the middle of rooms or vice versa cannot occur. Therefore, the final set of labeled places in the environment has a logical structure, even when it diverges from human-labeled ground truth.

Similar to our approach of creating an initial segmentation of the environment and then classifying the segments, Brunskill et al. [32] use spectral clustering of a simple graph representation of the environment to produce a segmentation of the environment. They then apply a learned classifier based on [12] to label the segmented regions. Liu et al. [33] discuss known problems with spectral clustering, like inconsistent results due to parameters, and propose an alternate segmentation approach using a mutual information graph, which results in a more robust and consistent segmentation of the environment. However, their approach can still generate noisy, meaningless areas that they must later filter out.

Markov chain Monte Carlo (MCMC) sampling for place classification has also been used by Liu and von Wichert [34]. They use a dramatically different approach, whereby they create a generative model of possible worlds and sample from this distribution to create a semantic map of the environment. Beginning from simple rules (a room is rectangular with four walls, a room has at least one door, each cell belongs to one room), they use extracted line segments and free space to evaluate hypotheses for where the walls and doors exist in the environment.

Like many methods, our place classification relies solely on laser-based features or those extracted from occupancy grids. However, a variety of other modalities have been employed for semantic place classification. Pronobis et al. [31] combine visual and laser models of the environment, along with door information to label an environment. Later work [35] introduced object information to infer more categories for large-scale semantic mapping. Another approach [36] has a human specify the location of places, which are then fit to Gaussian models to create a segmentation of the environment.

Finally, Rituerto et al. [37] create semantic labels for a sequence of images capture by an omnidirectional camera. Like our approach, they recognize the importance of labeling

both places and transitions in the construction of a topological map. However, they treat transitions as a different class, rather than a simple line segment boundary. As such, their constructed map defines stairs, elevators, and doorways as types of transitions, whereas doorways are simply gateways in our approach and stairs and elevators are different types of paths.

2.3 Probabilistic Topological Mapping

Early work by Kuipers and Byun [16] decided among a set of possible map hypotheses by attempting to navigate a route planned within a map hypothesis. If the robot was unable to follow the route, the map hypothesis was discarded. Choset and Nagatani [22] use a similar strategy to determine whether a new place was previously visited. They create a set of possible places at which the robot could have arrived, select a path to follow, and rule out inconsistent matches based upon the next visited place. They continue this process until a single match remains. Dudek et al. [23] describe an exploration strategy to find the correct topological map by maintaining a set of distinct markers. The robot leaves these markers at places with unexplored paths. When the robot traverses a new path, it can then determine if the place at the end was previously visited or is new. In later work, Marinakis and Dudek [15] discuss the tradeoff an exploration strategy must make between full exploration of the environment and keeping the number of map hypotheses tractable.

These exploration-based approaches to finding the correct map are ultimately unsatisfactory because they depend on complete control of the robot's actions, which robots that serve humans (such as a robotic wheelchair) or that interact with other agents likely will not have. They may also require a large number of traversals through the environment, which is again impractical on a robot operating in the human environment or under temporal constraints. In this paper, we assume the robot has no control over the order in which places in the environment are explored.

While probabilistic methods have been the focus of metric mapping since the seminal work by Smith, Self, and Cheeseman [38], only recently have probabilistic algorithms for topological mapping emerged. Probabilistic topological mapping algorithms incorporate metric and appearance information to focus the search for the correct map on the most probable hypotheses or to assert a loop closure only when the posterior is extremely peaked around a single place.

Ranganathan and Dellaert [8] use a Rao-Blackwellized particle filter where each sample represents a topological map hypothesis. The algorithm maintains the N most probable map hypotheses. If N is small enough, the algorithm is able to run in real-time and, based

on the presented results, discover the correct map. However, the particle filtering approach inherently contains the risk of discarding or never generating the correct map. Specifically, if an erroneous measurement causes the correct hypothesis to be removed from the sample set, the correct map can never be found because the history of map hypotheses is not maintained.

An alternative probabilistic approach is presented by Tully et al. in [17] based on the tree of maps described by Dudek et al. in [39]. In [17], the nodes of the tree are annotated with a posterior probability using a recursive Bayes formulation. To avoid the exponential growth of the tree, hypotheses with a low measurement likelihood or posterior probability are pruned after each update. This pruning step still leaves a substantial number of hypotheses in the tree, making real-time operation unlikely. Furthermore, pruning the tree means no guarantee can be made that the correct map will be found. In an extension of their work Tully et al. [9] introduce a garbage-collector hypothesis that detects when the correct map hypothesis is likely to have been discarded, but they provide no description of how the correct map can be recovered.

Lazy evaluation of a hypothesis tree using map likelihoods has been previously applied in the context of robotic mapping. Hähnel et al. [40] proposed a lazy evaluation approach to determining data associations in a feature-based metric map. Similar to our heuristic search for loop closures, they maintain a tree of data association decisions for all time steps. Each node is labeled with the log-likelihood of the measurements given the data association. The node with the highest log-likelihood is considered until the maximum log-likelihood node is a leaf in the tree. Our approach differs by using a heuristic to estimate the log-likelihood of a node at a future time. By doing so, our tree search potentially considers fewer hypotheses and contains a smaller set of leaf nodes.

More recent work in metric mapping has made progress towards robust metric SLAM by incorporating a metric for the quality of a loop closure directly in the pose graph optimization process, rather than through search like Hähnel et al. [40]. These algorithms address problems created by outlier constraints in the back-end optimization that result from data association errors in metric SLAM front-end. Olson and Agarwal [2] use a Gaussian mixture model to describe the distribution across possible loop closures while also including a high-covariance null hypothesis to allow completely disregarding a particular solution. The switchable constraints approach from Sunderhauf et al. [4] uses a similar strategy to weight the cost of different loop closure constraints in the graph. Dynamic covariance scaling introduced by Agarwal et al. [5] improves on switchable constraints by removing the additional switching variables, and thus improving the system runtime. Pfingsthorn et al. [3] create a generalization of the previous robust graph SLAM approaches

by introducing an additional step between the front-end and back-end of the SLAM system the pre-filters outlier loop closure constraints.

In all of these approaches, the outlier constraints are the result of an incorrect loop closure assertion resulting from an incorrect match between the robot's current location and a previous location. In the case where the robot is actually revisiting a previous location, topological SLAM avoids incorrect loop closures through explicit enumeration of the loop closure possibilities. However, the density of pose graphs makes an explicit search for loop closures infeasible. However, current topological SLAM approaches will fail when a place detection error (false positive or false negative) creates an inconsistency in the topological map. The robust metric SLAM approaches, however, can often detect such an outlier and ignore it, i.e. by accepting the null hypothesis [2].

2.4 Socially-Aware Navigation in Human Environments

Amongst many challenges including subtle body language cues, knowledge of cultural and social norms, and interactions with inattentive people, the everyday world is a complex and highly dynamic environment for a robot to navigate. Faced with such environments, some robots are able to use simple navigation strategies like waiting for agents to move out of the robot's way [41]. Many robots though – especially service robots like intelligent wheelchairs or telepresence systems – cannot rely on such strategies because efficient motion is needed and waiting for crowds to disperse before moving is not an option. A variety of approaches are used for navigating in these dynamic environments, including treating the world as quasi-static and replanning quickly using an algorithm like D*-Lite [42], forward-predicting objects using a constant velocity model [43, 44], or optimizing the robot's trajectory to minimize probability of collision with moving agents [45, 46, 20, 47].

Within the broad scope of motion planning research, the improved abilities of robots to navigate safely in highly dynamic environments has led to increasing interest in *socially-aware navigation*, whereby a robot considers additional factors beyond safety and efficiency in deciding how to move through the world. As noted by Kruse et al. in their review of the field [48], most of these additional performance measures are drawn from social science research into proxemics (measures of interpersonal distance) [49] and social norms (accepted cultural behaviors for a variety of situations) [50]. A recent study by Zanlungo et al.[51] provides experimental evidence for the commonly acknowledged social norm of moving along the right (or left) of a corridor, even when not interacting with other pedestrians.

The importance of social awareness for improving human comfort in the proximity of

robots has been established by a number of studies [52, 53, 54, 55]. However, because social norms are so loosely-specified and culture-dependent, the most successful approach for making a robot understand and follow social norms is to learn them from observing people's behavior in a variety of social situations.

Techniques for learning normative behavior focus on three overlapping themes, as identified by Kruse et al. [48]: comfort, naturalness, and sociability. *Comfort* focuses on reducing discomfort people feel when interacting with a robot [56, 55], *naturalness* focuses on low-level behaviors that make a robot seem more human-like [57, 52], and *sociability* focuses on robots exhibiting explicit behaviors, like waiting in line [58]. Okal and Arras [59] attempt to generalize the concept of normative behavior for a robot by formalizing a state representation that included additional social attributes along with low-level motion primitives.

Methods for augmenting classical motion planning with social cues fall into three main categories: cost maps, potential fields, and minimizing the probability of a negative interaction. Sisbot et al.[60] constructs cost fields around humans accounting for safety and visibility, whereby the robot prefers to be visible to people to avoid surprise. Kruse et al.[61] built on this work to include ContextCosts, which are direction-dependent costs that adjust the social costs based on the strength of the robot's interaction with a pedestrian. If the robot detects is not interacting strongly with a person, e.g. they are crossing its path in an intersection, the social costs are reduced to the degree that the robot follows the shortest path route and simply slows to wait for them to pass. A later user study [57] showed that this slowing behavior was preferable to path-changing behaviors because the robot's intentions were clearer.

Potential fields methods focus on the application of the Social Forces model [62] to robot navigation and pedestrian trajectory prediction. Social forces originated as a method for predicting human interactions and trajectories when moving through the world. The original model was extended to include explicit collision prediction (CP-SFM) by Zanlungo et al.[63], which better matches observed patterns of human motion. CP-SFM was shown by Shiomi et al.[52] to produce more predictable robot behavior, allowing for safer interactions. Ferrer et al.[64] extend the social forces model by differentiating person-person interactions from person-robot interactions, and then integrate their social forces into a potential field planner for a large-scale deployment.

Chung et al. [65, 58] present the Spatial Behavior Cognition Model (SBCM) that divides the navigation problem into General (GSE) and Specific Spatial Effects (SSE). GSEs define a similar set of metrics to MPEPC, whereby costs are associated with static and dynamic obstacles, as well as actions. SSEs are environment-specific features that influence

pedestrian behaviors, learned using a histogram of pedestrian motion in the environment. Unlike SSEs however, the responses learned by our approach generalize to any environment with a structure that can be represented as a topological map.

Kim et al. [66] use Inverse Reinforcement Learning to learn a cost function associated with navigation actions to navigate a smart wheelchair around dynamic obstacles. However, their approach is limited by choice of sensors and thus unable to generalize to fully autonomous motion. Additionally, they only learn passing behaviors and default to the shortest path to the goal otherwise. An alternate approach by Dondrup et al. [67] creates a qualitative description of a passing interaction and learns state transition probabilities for how the passing action should be performed. Their approach focus on passing behaviors between a single person and robot, which limits the applicability of their method to more complex scenarios. More complex passing behaviors are learned by Chen et al. [68] using deep reinforcement learning. They achieve impressive performance in learning norms for passing pedestrians in a busy real-world environment, but they also focus on solely on passing behaviors and otherwise optimize for time efficiency.

Given the inherent uncertainty in understanding the behaviors of people, probabilistic approaches hold great appeal. Closely related to our probabilistic formulation is work by Rios-Martinez et al. [45] who introduce social awareness in an RRT-based approach by incorporating a probability of disruption, which occurs when the robot violates social conventions associated with groups of people interacting. This additional probability fits naturally within our approach, where progress is weighted against the probability of collisions. Park et al. [69] focus on identifying a single negative interaction, humans intentionally blocking the robot, using Gaussian Processes.

More recent work by Trautmann et al.[47] attempts to model the joint interactions amongst all agents in the environment, as opposed to our simpler approach where agents are treated independently when predicting their actions. They apply Gaussian Processes to estimate both pedestrian and robot positions and select an action based on that result. However, their approach requires full knowledge of nearby pedestrians and the robot is limited to a slow speed of 0.3m/s, whereas our approach is more limited in prediction ability, but allows safe operation at 2.5m/s in general environments. An alternate approach, Multi-Policy Decision Making (MPDM) [70], samples a joint space of agents and policies, forward simulates these states, accounting for agent interactions using a Social Forces model, and selects the robot policy that minimizes a cost metric based on progress to the goal and the social force exerted on other agents. A followup for this approach [71] focuses on finding the worst-case outcomes to select a safe policy to execute.

CHAPTER 3

The Hierarchical Hybrid Spatial Semantic Hierarchy

This chapter describes the Hierarchical Hybrid Spatial Semantic Hierarchy (H^2SSH), an extension of the Hybrid Spatial Semantic Hierarchy (HSSH) [11], a hybrid metric-topological map representation that provides metrical and topological representations of small-scale (local) and large-scale (global) space. We begin with a brief overview of the HSSH. We then discuss an alternate interpretation for a topological map as a collection of navigation affordances, which describes qualitative properties about how a robot can move through the environment. We then provide an overview of the core concepts of the H^2SSH and the intuition behind the representation. Finally, we formalize the H^2SSH by defining its local and global topological representations.

3.1 The Hybrid Spatial Semantic Hierarchy

The Hybrid Spatial Semantic Hierarchy (HSSH) is a map representation that uses metric and topological representations of small-scale space, the area within the robot's immediate sensory horizon, to build topological and metrical maps of the large-scale environment. The HSSH is a hierarchy of ontologies, where each layer in the hierarchy provides a different abstraction of space. The Local Metrical layer uses a local simultaneous localization and mapping (SLAM) algorithm to build a Local Perceptual Map (LPM) of the small-scale space around the robot. The Local Topological layer performs place detection within the LPM, determining the extent and local topology of each place. The Global Topological layer uses the places detected by the Local Topological layer to build a global topological map of large-scale space. The Global Metric layer uses the global topology to construct a metric map of large-scale space. The Global Metrical layer is described in detail by Beeson et al. [11] and is not discussed further in this thesis. The flow of data within the HSSH is

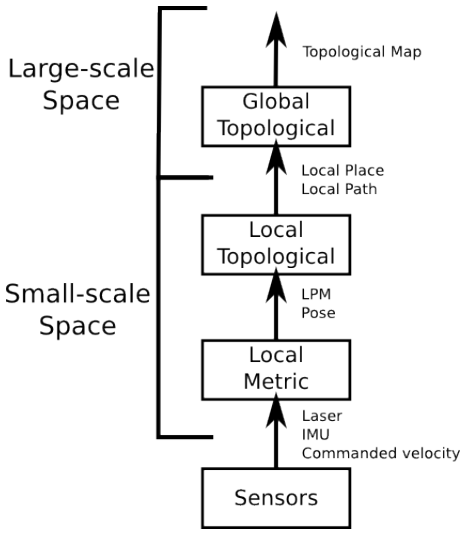


Figure 3.1: The flow of data through the HSSH. Raw sensor data is processed by the Local Metrical layer using a local SLAM algorithm to produce the LPM of small-scale space and the robot's pose within the LPM. The Local Topological layer searches for decision points in the LPM and produces local place and local path information. The Global Topological layer uses a topological SLAM algorithm to produce a topological map for large-scale space. The Global Metrical layer is not shown in this figure.

shown in Figure 3.1. The following sections provide a brief summary of the layers of the HSSH. For a more thorough discussion, see [11].

3.1.1 Local Metrical Layer

The Local Metrical layer of the HSSH represents the local surround of the robot – the portion of the environment within the robot's sensory horizon – using an occupancy-grid-based representation called the Local Perceptual Map (LPM). The LPM is a fixed-size scrolling metrical map of the environment, where the robot stays approximately centered in the map at all times. As the robot drives through the environment, portions of the map that fall outside the fixed boundary around the robot are discarded.

Using a small, fixed-size map ensures that no large-scale loop closures occur within the LPM, so an explicit search for loop closures need not be performed. We assume the robot is always well-localized within the LPM. However, the reference frame of the LPM with respect to the large-scale environment is expected to slowly drift over time.

3.1.2 Local Topological Layer

The Local Topological layer of the HSSH creates a symbolic description of small-scale space by detecting places in the LPM. Places in the HSSH are located at decision points

Figure 3.2: Example LPMs constructed during navigation of an office building.

in the world, which are locations where the robot is presented with qualitatively distinct options about the next action to take. For an indoor environment, a decision point is typically located at a hallway intersection. Each place is described by a metric map of the place neighborhood and by its local topology, or *small-scale star*.

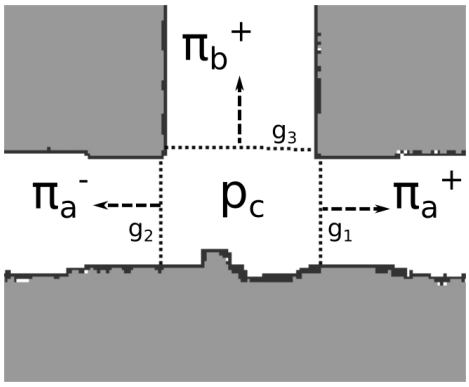
A *place neighborhood* is a metric map of the portion of the LPM within the boundary of the place, as defined by static obstacles and *gateways*.

A *gateway* represents the boundary between a place neighborhood and the rest of the environment [11]. A gateway's endpoints are located on static obstacles. The gateway provides two headings, inbound and outbound, that define a reference heading for control laws that guide the robot into or out of the place neighborhood associated with the particular gateway.

The local topology of a place is determined using the set of paths incident to the place. Unlike other topological map representations, a path is not an edge between two places, but is rather a larger ordered sequence of places that either terminates or passes through a place. Each incident path is represented by two local path fragments, which represent the portion of the larger path contained within the LPM. A *local path fragment* describes the direction of travel along the associated path when departing from the place, as well as whether or not the local path fragment can be navigated. If a path terminates at a place, one of the two associated local path fragments will be marked as unnavigable.

The *small-scale star* is a symbolic representation of the local topology of a place and is a circularly ordered list of the directed local path fragments for the paths incident to the place. For example, a T-intersection consists of two paths, three navigable path fragments, and one unnavigable fragment. Figure 3.3 shows the local topological representation of a T-intersection.

The formal description of the Local Topological representation is contained in Table 3.1.



(a)

s_{p_c}:

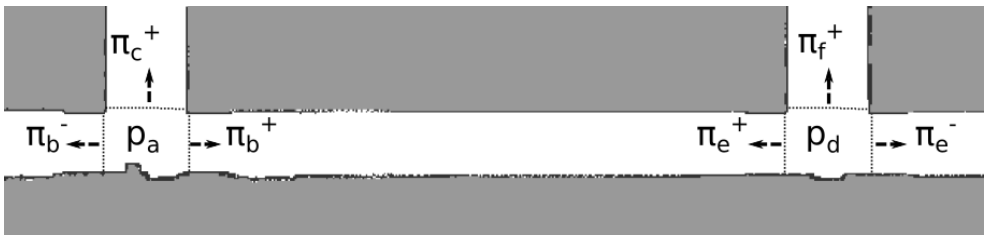
$[\langle \pi_a^-, 1 \rangle,$
$\langle \pi_b^+, 1 \rangle,$
$\langle \pi_a^+, 1 \rangle,$
$\langle \pi_b^-, 0 \rangle]$

(b)

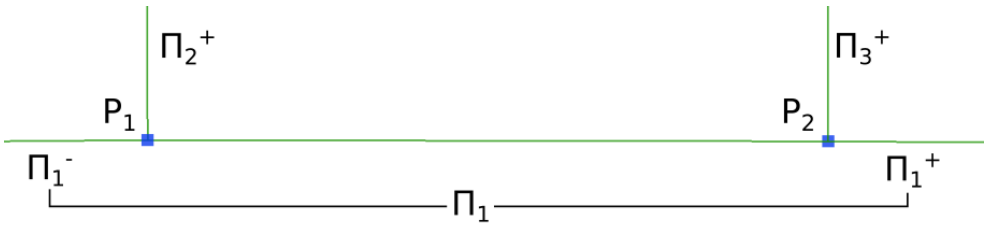
Figure 3.3: A place neighborhood within the Local Topological layer of the HSSH in shown in (a). The extent of a place is the portion of free space within the gateway boundaries. The dashed arrow associated with each gateway shows the outbound heading, which is the normal of the gateway boundary. $\pi_a^-, \pi_a^+, \pi_b^+$ are directed local path fragments. Each directed local path fragment describes the portion of a local path visible from the place neighborhood along with the direction of motion along the local path. π_a^- and π_a^+ are part of the same local path, as indicated by the shared subscript a. π_b^- is an unnavigable local path fragment and is not shown in the figure. The corresponding small-scale star is shown in (b). Each directed local path fragment is associated with a $\{0, 1\}$ to indicate whether or not it is navigable.

gateway:		
	boundary:	pair of endpoints in the LPM
	heading:	heading of gateway normal in the LPM pointing away from the place
place:		
	map:	local metric map
	gateways:	set of all gateways bounding the area
	small-scale star:	cyclic order of local path fragments (counter-clockwise)
path segment:		
	g^-:	gateway leading in the minus direction
	g^+:	gateway leading in the plus direction

Table 3.1: The Local Topological representation of the HSSH. The local metric map associated with a place is the portion of the LPM contained within the place boundary defined by static obstacles and gateways.



(a)



(b)

Figure 3.4: A path in the HSSH is a one-dimensional sequence of places and the path segments between them. In (a), we see two places, p_a and p_d, as represented in the Local Topological layer (see Figure 3.3). p_a and p_d are associated with directed path fragments ($\{\pi_b^-, \pi_b^+\}$ and $\{\pi_e^-, \pi_e^+\}$) that describe the direction of motion along some larger path. In the corresponding Global Topological representation of this same portion of the environment in (b), each of the places and paths is associated now with a global place or path, which is indicated by the change in subscript from letters to numbers. The directed local path fragments $\{\pi_a^-, \pi_a^+\}$ and $\{\pi_e^-, \pi_e^+\}$ become associated with the same global path Π_1, while the other path fragments are part of unique paths Π_2 and Π_3.

3.1.3 Global Topological Layer

The Global Topological layer represents large-scale space using a global topological map, a structure consisting of places, paths, and path segments. The places and paths in the global topological map are detected in the Local Topological layer.

A *place* in the global topological map is represented by a large-scale star, which is the global analogue of the small-scale star. A *large-scale star* is a circularly ordered sequence of directed global paths. A bijective mapping exists between a large-scale star and its corresponding small-scale star. Each directed local path fragment in the small-scale star corresponds to a directed global path in the large-scale star. Note that directed local path fragments from different places may correspond to the same directed global path. For example, using the example in Figure 3.4, we see that both π_e^+ and π_b^+ are mapped to Π_1^+.

A *path* is a linearly ordered sequence of path segments. Each *path segment* connects two places in the map and contains an estimate of the displacement, $\lambda = (\Delta x, \Delta y, \Delta\theta, \Sigma)$, between the reference frames of these two places. Figure 3.4 shows an example of the representation of a path in the Global Topological layer, along with its relation with the Local Topological layer.

path segment:		
	p^-:	place at the upstream end of the path segment
	p^+:	place at the downstream end of the path segment
	λ:	$\langle \delta x, \delta y, \delta\theta, \Sigma \rangle$
	path:	path to which the path segment belongs
path:		
	places:	ordered sequence of places
	segments:	ordered sequence of path segments
place:		
	large-scale star:	circularly ordered set of adjacent path segments (counter-clockwise)

Table 3.2: The Global Topological representation of the HSSH.

3.2 H^{2}SSH Overview

For a robot to successfully travel through an environment, the robot needs to have a representation of every relevant part of the environment that it encounters. For a metric map, like an occupancy grid, the representation simply encodes whether a portion of the environment is occupied or free. Thus, the robot only knows whether something can be traversed.

A topological map represents the environment symbolically as a graph-like structure of nodes and edges. The topological map representation requires the robot to create symbolic abstraction of each relevant part of its environment. This abstraction defines the robot's understanding of the navigation semantics of its environment. To create the abstraction of the environment necessary for building a topological map, we define an ontology of space using the concepts of *areas* and *navigation affordances*.

Affordances were originally defined by Gibson [72]. We describe them as the opportunities for action between an agent and its environment. Thus, affordances depend on the agent's abilities as well as the environment's properties. For mobile robots, we define the *navigation affordances* in the environment as those affordances related to the robot's motion within the environment.

One affordance previously studied by Uğur et al. [73] is *traversibility*, which specifies whether or not a robot can reach and pass through some pose in the environment. If the robot is in collision with an object, then a particular pose is not traversable. Thus, traversibility defines *where* a robot can be in an environment. Though simple, the idea of the traversibility of an environment is implicitly used by all motion planning algorithms that calculate collision-free trajectories for a robot.

Following the work of Tsai et al. [74], we extend the concept of navigation affordances

beyond traversibility by examining how the shape of the environment provides information on *how* to move through the environment. We identify two affordances sufficient for describing how a mobile robot can move through the environment.

First, we note that a mobile robot spends much of its time traversing hallways, footpaths, sidewalks, and roads. All of these structures have an elongated shape with a dominant axis along which motion takes place. This axis informs the robot where a hallway leads and how to navigate there, even if the end of the hallway is out-of-sight. We say that when moving along a hallway or similar structure, the robot *travels* along the dominant axis. In other words, the hallway provides the affordance of traveling along its axis.

Second, most human-built environments are structured. Indoor environments most often consist of rooms, corridors, stairs, elevators, and large open halls. Outdoor environments consist of sidewalks and plazas or road networks. In each case, the environment consists of a set of distinct entities, like individual offices, corridors, or roads. A route through these environments, therefore, requires moving through a sequence of these entities. The boundary between adjacent entities thereby provides the ability to *transition* from one entity to the next.

Viewing the environment from the perspective of navigation affordances allows a topological map to be interpreted as a map describing the opportunities for moving provided by the environment. In the H^{2}SSH, we define an ontology of space to represent the world using travel and transition affordances. The ontology used in the H^{2}SSH is shown in Figure 3.5 and is described in Section 3.4.

Using this interpretation, the H^{2}SSH representation divides the environment into non-overlapping spaces, which we call *areas*. An *area* is the basic entity in the environment. Each area is bounded by static obstacles, like walls and drop-offs, and by gateways [11], which define the transition affordances between adjacent areas. Our ontology of space defines three types of areas, path segments, decision points, and destinations, that each serve a functionally different purpose for a mobile robot.

A *path segment* is an area that provides a travel affordance between places. We distinguish between the places at the ends of the path segment from the places along either side of the path segment. A path segment can be unambiguously mapped by the robot, and except in rare or pathological situations, like a circular building, the robot enters the path segment from one place and exits the path segment at another. While traveling along the path segment, the robot may observe places, corresponding to offices and other rooms, which we call *destinations*. These destinations are added to the path segment's topological description. Note that decisions are not only made at decision points. In the H^{2}SSH, they are also made when traveling along a path segment, as the robot must decide whether or

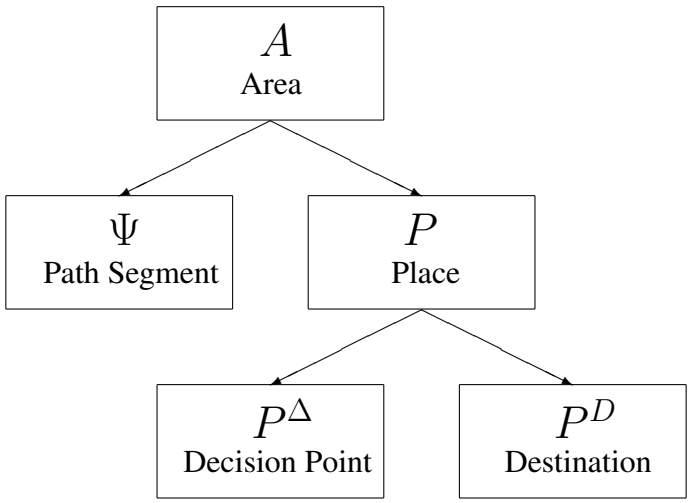


Figure 3.5: The ontology of space used to represent the world in the H^{2}SSH. A *path segment* (Ψ) has a major axis of travel, providing the option to travel forward or backward along this axis. The path segment may also have destinations on the right and left sides. A *decision point* (P^{Δ}) is at the end of two or more unaligned path segments and affords the robot a choice of which path segment to follow. The decision point may also have one or more destinations adjacent to it. Finally, a *destination* (P^{D}) is an area to which the robot can travel that doesn't meet the criteria for a path segment or decision point. Destinations are often (but not always) areas that are only accessible from a single other area, like offices and conference rooms.

not to enter one of the destinations along the sides of the path segment.

A *decision point* is an area located at the end of two or more unaligned path segments. Decision points correspond to the places used in most topological map representations. Like path segments, a decision point can be adjacent to one or more destinations.

A *destination* is an area that doesn't meet either of the above criteria for path segments or decision points. Destinations correspond to locations like offices and conference rooms that are often at the start or end of a route through the environment, but are typically not otherwise visited during travel through the environment. In the hierarchical map (Section 3.5.2), destinations can also represent large-scale spaces that correspond to individual floors in a building or the whole building itself.

Our representation differs from existing topological and semantic map representations by distinguishing between the types of places in the environment based on the navigation affordances available to the agent. A decision point serves a functionally different role in navigation than a destination. In general, destinations are the goal locations where the robot wishes to travel and the decision points are the places where turns will be made while following a route to a destination.

The H^{2}SSH creates a topological abstraction for both small-scale and large-scale space, corresponding to the Local Topological and Global Topological layers. The Local Topological level segments the Local Perceptual Map into distinct area and determines when the

robot has transitioned to a new area. The Global Topological level establishes the global connectivity of paths and places and the overall structure of the environment. In the following sections, we formalize the topological representations in the H^{2}SSH.

3.3 H^{2}SSH Local Metrical Layer

Constructing a topological map requires first detecting the places in the map, a process typically called *place detection*. Place detection occurs within small-scale space, which is defined in the H^{2}SSH as the union of three regions: the portion of the environment contained within the robot's current area, the portion of the environment within all areas adjacent to the current area, and the portion of the environment within the robot's sensory horizon. In the H^{2}SSH and HSSH, we represent small-scale metrical space using a Local Perceptual Map (LPM).

The LPM is a high-precision occupancy grid of the local surround that scrolls as the robot moves through the environment. In the HSSH, the LPM is a fixed-size region centered around the robot. In the H^{2}SSH, the LPM varies in size depending on the configuration of the areas in the environment. This change means areas larger than the previous fixed-size LPM can be created within the H^{2}SSH. With our new definition, as the robot moves, the LPM will stretch to contain newly visible portions of the environment. When a new area is entered, the LPM boundary will shift to remove any areas no longer adjacent to the current area.

3.4 H^{2}SSH Local Topological Layer

The Local Topological layer of the H^{2}SSH provides a symbolic description of the LPM as a set of discrete, non-overlapping areas, which we call the *local topological map*.

A *gateway* is a line segment, whose endpoints are static obstacles, that separates two areas. Each area in the LPM is bounded by static obstacles, like walls, and gateways. The heading of a gateway is the normal to the gateway boundary that points away from the area. The heading is used for determining if gateways are aligned for constructing the transition cycle defined below. Additionally, the heading can be used to select a target pose when planning a route to leave an area.

A *transition* describes the topological action that carries the robot from one area to an adjacent area. In small-scale space, transitioning between areas corresponds to crossing a gateway separating the areas. A transition $\langle g, a_a, a_b \rangle$ is represented by its associated gateway g and the areas it bounds a_a and a_b.

A *path segment* is an area that provides a travel affordance between places. A path segment is adjacent to a place at each end, (p^-, p^+). Along each side of a path segment are zero or more adjacent destinations. The destinations on the left and right side, where left and right are determined when facing towards p^+ from p^-, are represented by a *transition sequence*.

A *transition sequence* τ is a linear order of the transitions available along the side of a path segment. τ is a sequence of tuples $\langle r_n, d^-, l, \sigma_{d^-} \rangle$, where $r_n n$ is the transition, d^- is the distance along the path segment starting from p^-, l is the length of the transition's gateway boundary, and $\sigma^2_{d^-}$ is the uncertainty of d^-.

A value, $\lambda^-_+ = \langle \delta x, \delta y, \delta\theta, \Sigma \rangle$, stores the transform from the reference frame of p^+ to the reference frame of p^-. Using the composition operator $\oplus$ from Self et al. [38], a pose $\mathbf{x}^+$ in the p^+ reference frame can be described in the reference frame of p^- with (3.1):

$$\mathbf{x}^- = \lambda^-_+ \oplus \mathbf{x}^+ = \begin{bmatrix} x^+ cos(\delta\theta^-_+) - y^+ sin(\delta\theta^-_+) + \delta x^-_+ \\ x^+ sin(\delta\theta^-_+) + y^+ cos(\delta\theta^-_+) + \delta y^-_+ \\ \theta^+ + \delta\theta^-_+ \end{bmatrix} \tag{3.1}$$

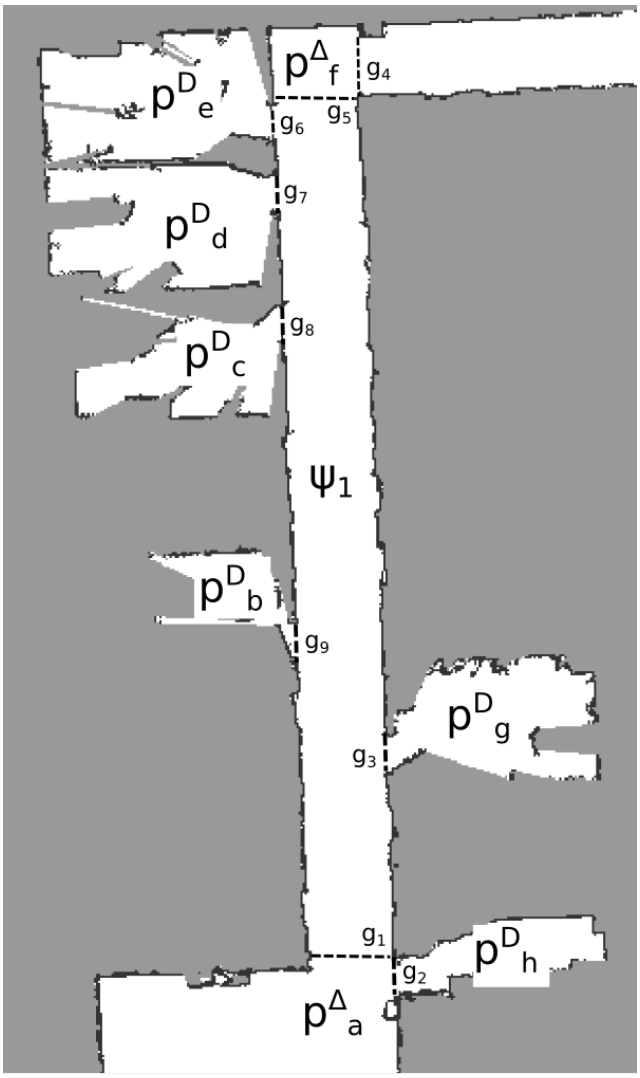
While not strictly necessary, this metric distance and length information allows for transitions in the sequence to be found even when previous transitions in the sequence are not detected, i.e. a door has closed, or additional transitions are detected, i.e. a door has opened. Given a perfect transition detector, this information wouldn't be necessary. However, such a detector does not exist, so the metric information improves the robustness of the map representation.

Whereas a path segment is represented by a linearly ordered sequence of places, a *place* is represented by a circularly ordered sequence of areas, which we call a *transition cycle*. There are two types of places in the H^{2}SSH, decision points and destinations, which are defined based upon the type of their adjacent areas. A *decision point* is located at the ends of two or more unaligned path segments. Decision points typically occur at intersections, or in large plazas, atria, or lobbies. A *destination* is a place that does not meet the criterion for being a decision point. A destination can be adjacent to zero or one path segment ends and zero or more places (decision points or other destinations). Only destinations can exist along the sides of a path segment.

A *transition cycle* c is a circularly-ordered set of the transitions available at a place. The transitions in a cycle are organized into pairs of aligned transitions that divide the cycle into two equal-sized sets. Thus, the cycle should always contains an even number of transitions. Since not every place contains an even number of transitions and not every transition is naturally aligned with another (see Figure 3.7), we pair each such transition

gateway:		
	boundary:	pair of endpoints in the LPM
	heading:	orientation of boundary normal pointing away from area
transition:		
	gateway:	gateway grounding transition in the LPM
	areas:	areas bounded by the transition
area:		
	id:	unique identifier
	map:	local metric map
place, **decision point**, **destination**:		
	cycle:	cyclic order of transitions to adjacent areas
path segment:		
	p^-:	place at minus end
	p^+:	place at plus end
	λ_+^-:	$\langle \delta x, \delta y, \delta\theta, \Sigma \rangle$
	left:	linearly ordered sequence of destinations
	right:	linearly ordered sequence of destinations

Table 3.3: The Local Topological representation in the H^{2}SSH (compare with Table 3.1 on page 21). Transitions are the boundaries between adjacent areas, grounded in the metrical map by a gateway. All areas have a local metric map representation, along with a set of transitions leading to adjacent areas. Places – destinations and decision points – are represented symbolically as a cyclic order of transitions to adjacent areas. Path segments are terminated by places (p^-, p^+), and contain a linear sequence of destinations along each side. The λ_+^- defines the change in the local reference frame from p^+ to p^-.



ψ_1:

p^-: p_a^Δ

p^+: p_f^Δ

λ_+^-: λ_a^f

left: $[\, p_b^D, p_c^D, p_d^D, p_e^D \,]$

right: $[\, p_g^D \,]$

Figure 3.6: An example of how a path segment is represented in the H^{2}SSH. Note that p_h^D is adjacent to p_a^Δ, not ψ_1. For brevity, we use only the area a transition leads to when describing the transition sequences along the sides of ψ_1.

with a *null transition*, $r_\varnothing$, which is a non-navigable transition that leads nowhere and serves as a placeholder in the cyclic order to allow a purely qualitative description of left and right, as discussed below.

Given a transition cycle, $c = [\, r_1, r_2, ..., r_N \,]$, we can describe the available actions using three functions:

$next(r_n, c)$:	the next transition to r_n in counter-clockwise order.
$prev(r_n, c)$:	the previous transition to r_n in counter-clockwise order
$aligned(r_n, c)$:	the transition aligned to r_n

Each function takes as an argument a transition, r_n. When entering an area via r_n, the transitions to the right are defined as those from r_n to $aligned(r_n, c)$ in the counter-clockwise order of c, and those to the left are those from $aligned(r_n, c)$ to r_n in the same counter-clockwise order.

Using these functions, we can easily define three qualitative types of actions. A straight action corresponds to transitioning from some r_n to $aligned(r_n, c)$, the available left turns consist of all transitions in the sequence beginning with $left_0 = prev(r_n, c)$ and ending when $aligned(r_n, c) = prev(left_m, c)$, and the available right turns are analogously the sequence of transitions beginning with $right_0 = next(r_n, c)$ and ending when $aligned(r_n, c) = next(right_m, c)$.

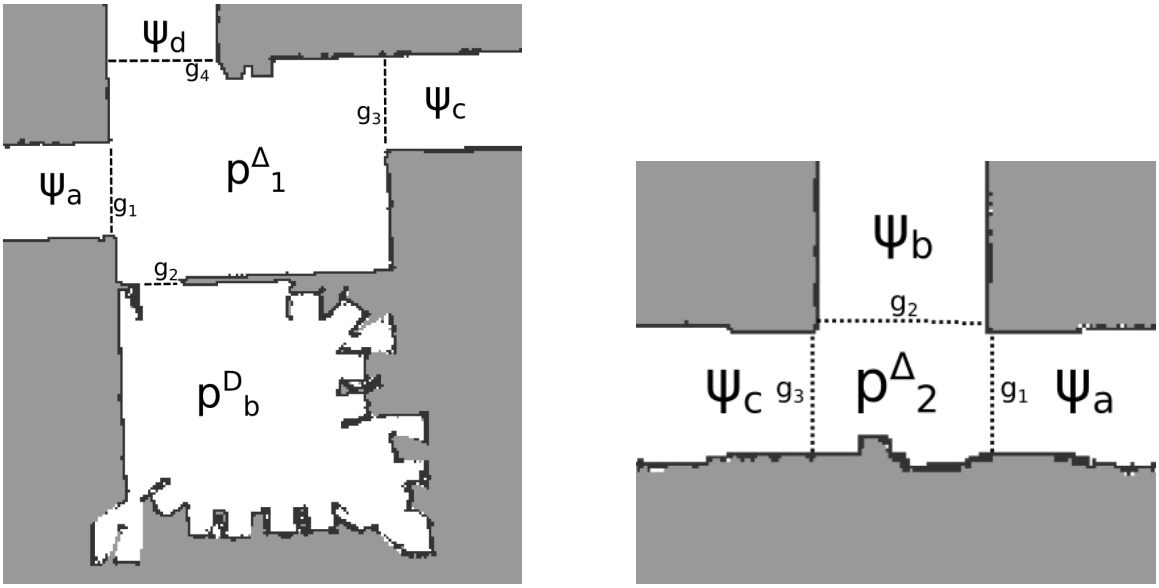
3.4.1 Local Topological Location

We call the robot's state in the local topological map its *local location* l_k. The subscript k indicates the location with respect to the discrete events that define the robot's topological motion through the environment. Thus, l_k is its location after topological event k has occurred.

In the H^{2}SSH, the robot is always located at a place or on a path segment. When at a place p_k, the robot state is the tuple $l_k = \langle p_k, r_{entry} \rangle$, where r_{entry} is the transition through which the place was entered.

When on a path segment, the robot state is the tuple $l_k = \langle \psi_k, r_{entry}, dir \rangle$, where ψ_k is the current path segment, r_{entry} is the transition through which the path segment was entered, and dir is the direction the robot is traveling along the path. The direction along the path takes one of three values:

$dir = \varnothing$ occurs when a robot enters a path segment from a destination in its left or right transition sequence. In this case, the robot does not have a defined direction. Once the robot begins moving towards an endpoint, then the direction will change to $dir \in \{+, -\}$.



(a) (b)

$c_{p_1^\Delta}$:

r_1: $\langle g_1, \psi_a, r_3 \rangle$

r_2: $\langle g_2, p_b^D, r_4 \rangle$

r_3: $\langle g_3, \psi_c, r_1 \rangle$

r_4: $\langle g_4, \psi_d, r_2 \rangle$

(c)

$c_{p_2^\Delta}$:

r_1: $\langle g_1, \psi_a, r_3 \rangle$

r_2: $\langle g_2, \psi_b, r_4 \rangle$

r_3: $\langle g_3, \psi_c, r_1 \rangle$

r_4: $\langle \varnothing, \varnothing, r_2 \rangle$

(d)

Figure 3.7: This figure shows two common decision points encountered in an office environment and their associated transition cycles. Each transition shows the associated gateway, the area it leads to, and the transition it is aligned with in the transition cycle. (a) shows a complex decision point with a transition cycle $c_{p_1^\Delta}$. In (c), we see that the slightly offset transitions leading to path segments ψ_a and ψ_c are determined to be aligned. As a result of being aligned transition, ψ_a and ψ_c will be part of the same global path when the global topological map is constructed. Additionally, the transitions leading to ψ_d and p_b^D are also aligned, even though they lead to different types of areas. Allowing different types of transitions to be aligned ensures that a qualitative description of left and right always exists, regardless of how a place is entered. (b) shows a T-intersection without an adjacent destination. As a result, we see in (d) that $c_{p_2^\Delta}$ contains a null transition, r_4, aligned with r_2. The null transition allows our definition of left and right to hold regardless of the configuration of transitions in a place.

Possible Path Directions	
$+$	the robot is moving towards p^+
$-$	the robot is moving towards p^-
$\varnothing$	the robot has not selected a direction yet

The entry transition r_{entry} is determined based on the robot's motion through the environment. However, at the robot's initial location l_0, the robot has yet to move. Therefore, $r_{entry} = \varnothing$ for l_0.

3.5 H^{2}SSH Global Topological Layer

The Local Topological layer of the H^{2}SSH defines the robot's motion through areas within the local environment defined by the LPM. The robot's knowledge of the areas in the LPM is temporary. As the robot travels through the environment, areas will shift outside the bounds of the LPM and be forgotten. Consequently, when the robot revisits an area that had fallen out of the LPM, a new area will be detected in the Local Topological layer.

The goal of the Global Topological layer of the H^{2}SSH is to create and maintain a globally consistent topological map of the areas in the environment. When the robot visits an area, the Global Topological layer is responsible for finding possible matches between the new area and a previously visited area, and for adding new areas to the map. The details of this mapping process are described in Chapter 6. The remainder of this section describes the map representations constructed by the Global Topological layer.

The Global Topological representation has two responsibilities. First, it establishes a 1-1 correspondence between local areas detected by the Local Topological layer and global areas in the global topological map. Second, it identifies topological *regions* in the environment, which are parts of the environment, like buildings along a sidewalk, that appear as single destinations along a path segment, but are themselves large-scale structures. An extensive example of topological regions is presented in Section 3.5.4.

3.5.1 From Local Areas to Global Areas

At the Local Topological level, the local environment consists of one or more areas bounded by gateways. Each gateway provides the means by which the robot can transition between adjacent areas, and each area is described by the configuration of its transitions, i.e. path segments have transitions at each end and along their sides, while places have a circular ordering of transitions.

When moving from the Local Topological to Global Topological representation, the concept of a transition as affording motion between adjacent areas is unchanged. Whereas a local transition r is associated with a particular gateway, a *global transition* R simply defines an adjacency relation between two areas.[1] The type of each area (ψ, p^D, p^Δ) associated with a local transition r is the same as the type of each area associated with the corresponding global transition R. Additionally, a unique identifier is associated with each global transition to account for areas with multiple transitions between them. Thus, a local transition $r_1 = \langle 3, g_3, a_b \rangle$, where 3 is the locally unique id for the transition, g_3 is a gateway, and a_b the local areas, becomes $R_1 = \langle 1, A_1, A_2 \rangle$, where 1 is the assigned id and A_1 and A_2 are global areas in the topological map, with $type(a_a) = type(A_1) \wedge type(a_b) = type(A_2)$.

Given the relation between a local and global transition, there exists a bijective mapping between a *global transition cycle* C and its corresponding local transition cycle c. The cyclic order of local transitions in the local transition cycle is identical to the cyclic order of global transitions in the global transition cycle. That is, for each local transition in c, $r_n = \langle g_n, a_a, a_b \rangle$, there exists a global transition $R_n = \langle n, A_i, A_j \rangle$, where $type(a_a) = type(A_i) \wedge type(a_b) = type(A_j)$.

As with a transition cycle, there exists a bijective mapping between a *global transition sequence* $\mathcal{T}$ and its corresponding local transition sequence τ. For each transition to a local destination in τ, there is a transition to a global destination in $\mathcal{T}$.

Using the global transition cycle and global transition sequence, we can define global path segments, global destinations, and global decision points similarly to their Local Topological analogues:

- An *area* is a non-overlapping, unique portion of the environment. Each area is described by a local metric map and is associated with a parent region, which will be explained in Section 3.5.2.
- A *path segment* is an area that affords travel between places adjacent to each end. Along the left and right sides of a path segment are zero or more adjacent destinations. The destinations on the left and right sides are each represented by a global transition sequence.
- A *decision point* is adjacent to the end of two or more unaligned path segments and is represented by a global transition cycle.
- A *destination* is any place that does not meet the criterion for being a decision point and is represented by a global transition cycle.

[1] Our convention is to use lower-case letters for all local topological symbols and upper-case letters for all global topological symbols.

In addition to the global versions of local areas, the Global Topological layer also contains paths. A *path* is a linearly ordered sequence of path segments and places. The places in the path are the endpoints of the path segments. Adjacent path segments in a path have aligned transitions at the place they share as an endpoint.

3.5.2 Hierarchical Regions

A robot's location in the environment can be described at many levels of abstraction. For example, a robot that operates on a university campus may describe the environment in multiple ways. At the lowest level is the robot's pose (x, y, θ) in its local environment, which could be location of its charging station. The next level would be the lab (a destination) where its charging station is located. The robot's lab can then be on a floor of a larger building, which itself is one building on the university campus. The H^2SSH supports these different types of locations by representing the global environment as a hierarchy of *regions*.

A *region* is a structure in large-scale space represented by a global topological map. Each region contains zero or more child regions and can be a child of another region. The parent-child relationships between regions form a containment hierarchy, whereby a region can be wholly contained within one region and can wholly contain other regions. Within a region, each child region is a destination within the global topological map for that region.

Motion into and out of a region occurs via its *exits*. Each *exit* is a global transition that carries the robot from an area within the region to an area within one of its parent regions or an adjacent region at the same level of the hierarchy. Exits can be shared by multiple regions in the hierarchy. For example, a door leading out of a building is an exit for the region representing the specific floor of the building and is also an exit for the building's region.

Three important cases exist for the region hierarchy. First, the root of the hierarchy is the only region with no parent. If a new region is created at the same level of the hierarchy, e.g, the robot moves from one building to another via a shared hallway, then a new root region is created that becomes the parent of the previous root and the newly created region.

Second, at the bottom of the hierarchy are regions with no children. A childless region corresponds to a global topological map that contains no destinations that are associated with a child region.

Finally, not every destination in the environment is associated with a region. Many destinations, like offices or conference rooms, are represented by a local metrical map, like the other types of global areas. Furthermore, these non-hierarchical destinations can exist

transition:		
	id:	globally unique identifier
	areas:	adjacent areas
region:		
	parent:	containing region
	children:	contained regions
	topological:	global topological map
	exits:	transitions out of the region
area:		
	id:	globally unique identifier
	parent:	containing region
	map:	local metric map
destination:		
	region:	contained region
	cycle:	global transition cycle of adjacent areas
	origin:	pose of place origin in global metrical map
path segment:		
	p^-:	ending place (upstream w.r.t order along the path)
	p^+:	ending place (downstream w.r.t order along the path)
	λ:	$\langle \delta x, \delta y, \delta\theta, \Sigma \rangle$
	left:	transition sequence of destinations
	right:	transition sequence of destinations
	path:	parent path
path:		
	places:	ordered sequence of places
	segments:	ordered sequence of path segments
decision point:		
	cycle:	global transition cycle of adjacent areas
	origin:	pose of place origin in global metrical map

Table 3.4: The Global Topological representation in the H^{2}SSH (compare with Table 3.2 on page 23). The map associated with a destination is now a global topological map, which allows for creating a recursive structure of global topological maps. The segments of a path are interleaved with the places, adjacent places have a path segment running between them. A fully-explored path will have places at each end, while a partially-explored path will have a place at zero or one ends.

in the global topological map of any region in the hierarchy. Therefore, a region with a hierarchical destination associated with a child region does not require all destinations to also have child regions.

3.5.3 Global Topological Location

Similar to the Local Topological location, we call the robot's state in the Global Topological layer its *global location* L_k, where the subscript k indicates the location after topological event k has occurred.

As with a local topological map, in a global topological map, the robot is always at a place or on a path segment. When at a place, the robot state is the tuple $\langle P_i, R_{entry} \rangle$, where P_i is the current place and R_{entry} is the transition through which the place was entered. When on a path segment, the robot state is the tuple $\langle \Psi_i, R_{entry}, dir \rangle$, where Ψ_i is the current path segment, R_{entry} is the transition through which the path segment was entered, and dir is the direction of motion along the path, as defined in Section 3.4.1.

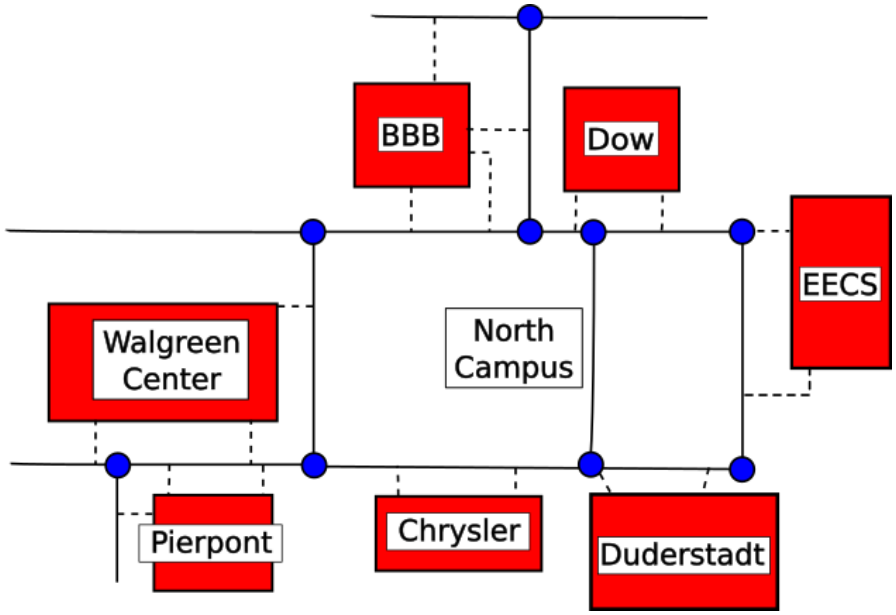
When the robot enters a path segment from a destination in one of its transition sequences, $dir = \varnothing$ as in the Local Topological layer. The direction of motion in the global topological map will change to $dir \in \{+, -\}$ once the direction of motion in the local topological map is determined.

In addition to its location within a single global topological map, the hierarchical structure of the Global Topological layer requires the robot to always be located in one or more regions. Using a superscript to indicate a particular region, we can then define the robot's global location as the sequence containing its location within each region's global topological map, $L_k = (L_k^0, L_k^1, \ldots, L_k^\rho)$.

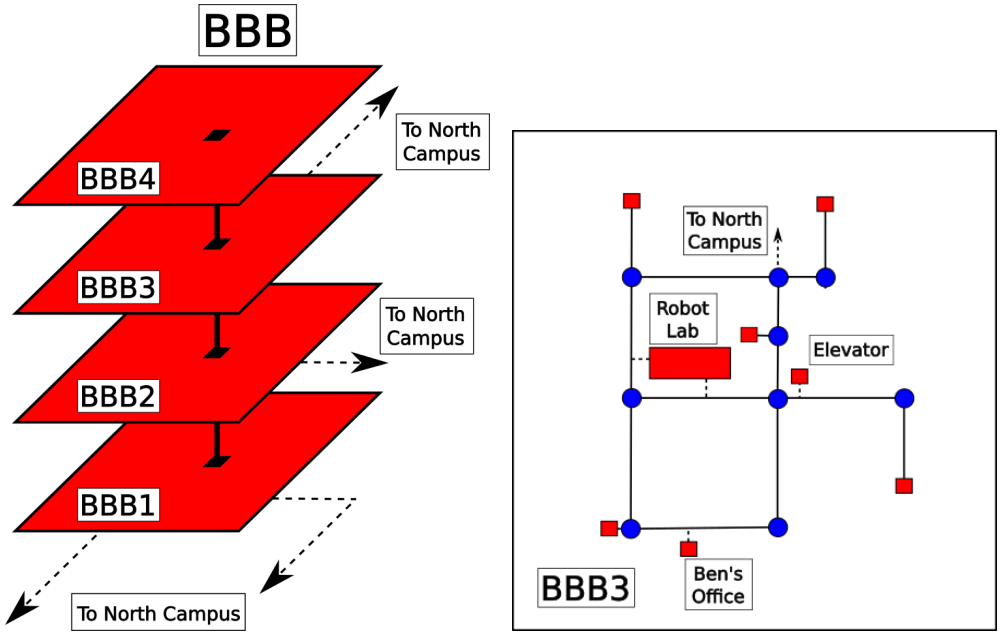
3.5.4 Global Topological Example

To illustrate the representation of the region hierarchy, we use an example illustrating the representation of North Campus at the University of Michigan. Figure 3.8 shows the topological map for three levels of the region hierarchy. Figure 3.9 shows the containment hierarchy formed by the regions.

Figure 3.8a shows the largest region, which encompasses all of North Campus. In this region, path segments correspond to the sidewalk networks that lead from one building to another, and the decision points are sidewalk intersections. Each destination in this map is a building accessible from the sidewalks, and therefore contains a new region. For brevity, we show only the region maps associated with the Beyster building and a single floor of the Beyster building.



(a)



(b) (c)

Figure 3.8: This example shows topological maps of three different regions on North Campus. For each map, solid lines correspond to path segments, blue circles to decision points, and red rectangles to destinations. The dashed lines indicate transitions leading to destinations or different regions in the hierarchy. (a) is the global topological map for all of North Campus. Each of the labeled buildings is a destination within this map. (b) is the global topological map of the Beyster building (BBB). The map contains a single path segment representing an elevator connecting four floors, each of which is a destination. (c) is the global topological map of the third-floor of BBB.

We reiterate that even though in the North Campus map each destination is a region, this property is not required by the H^{2}SSH representation. A single region can contain both destinations that are themselves regions and destinations that correspond to a single destination in the local topological map.

Moving down the hierarchy, Figure 3.8b shows the global topological map of the Beyster building. This topological map is typical of regions associated with buildings. In particular, the building is represented as a single path segment, corresponding to an elevator, with each floor accessible from this elevator being a destination containing a global topological map.

Though for simplicity this example only shows a single elevator, multiple elevators can exist between the floors of a building. In the case of multiple elevators between floors, each elevator will be a path segment in the map. These path segments will be adjacent to the destination representing the particular floor in the building. Moving between the elevators will require planning in the topological map of a floor of the building. Similarly, if ramps or staircases allow travel between floors, they will appear as individual path segments connecting the floors. In general, the global topological map of a building will contain only the path segments that lead from one floor to another, with each floor being its own destination.

The bottom of the hierarchy is shown in Figure 3.8c. This topological map corresponds to a single floor of the Beyster building. In this map, the path segments are hallways and the decision points are hallway intersections. Destinations at the end of path segments correspond to dead ends in the map. Many destinations exist on this floor of the map, one for each office, conference room, and restroom, but they are left out of the figure. We show a few important locations in the environment, namely a faculty member's office, a laboratory, and the elevator used for exiting the region.

3.6 Discussion

The H^{2}SSH representation adds destinations and regions to the previous HSSH representation. The differences between the two representations have important consequences of the scalability of the topological SLAM algorithm. In particular, the complexity of the topological mapping problem often grows exponentially with the number of places visited, so limits on the number of places improves the overall scalability of the representation.

To illustrate the benefits of the H^{2}SSH representation, we use a common scenario that a mobile robot can expect to encounter during daily use, navigation along an office corridor. We show how the hierarchical structure in the H^{2}SSH representation – namely destina-

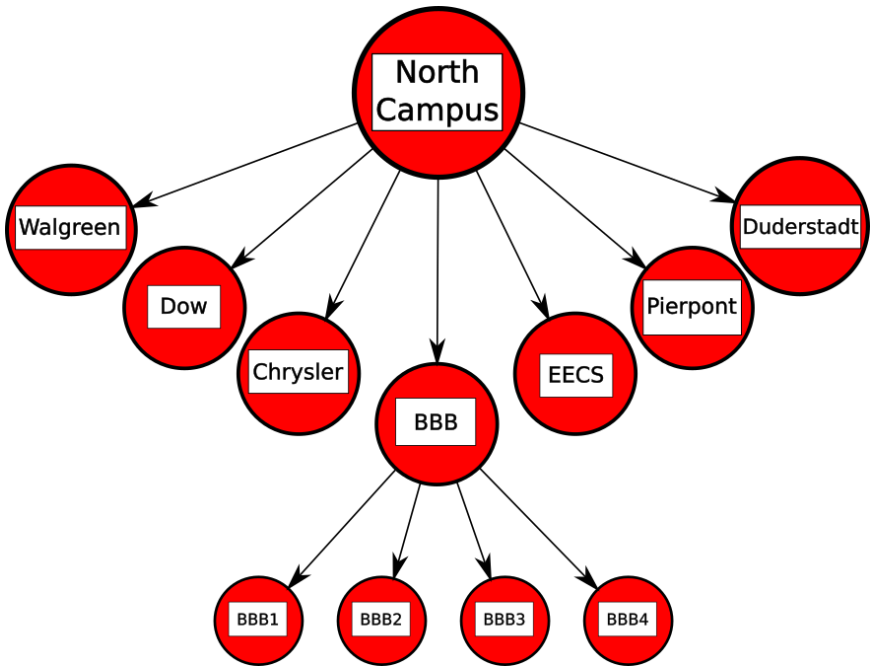


Figure 3.9: The region hierarchy for North Campus in H^{2}SSH. An arrow pointing from one circle to another indicates the region at the tail of the arrow contains the region at the head. Thus, North Campus contains every building, and BBB contains every floor of BBB. All other buildings contain multiple floors, but they are omitted from this figure for clarity.

tions along path segments – can be exploited to allow for more scalable mapping of large environments.

Consider a typical map of part of an office building, as shown in Figure 3.10a. The map consists of interconnected hallways, each of which is lined with offices. At the ends of each hallway is an intersection with another hallway.

We contrast how a robot would represent this situation using the HSSH versus the H^{2}SSH.

In the HSSH, the environment consists of places and paths, and travel through the environment is always an alternating sequence of places and paths. In this office environment, the intersection at the end of each hallway is a place, and each of the offices is a place. Additionally, because each place must be connected to adjacent places via a path, the space in front of each office will be an intersection connected by a zero-length path to the office.

In the H^{2}SSH, the environment consists of path segments, decision points, and destinations. Travel through the environment consists of moving between adjacent areas. In this office environment, the intersection at the end of each hallway is a decision point, and each of the offices is a destination along a path segment.

In the HSSH and other topological maps, each office creates a new place in front of the office along the hallway, which creates a new place for possible loop closures. De-

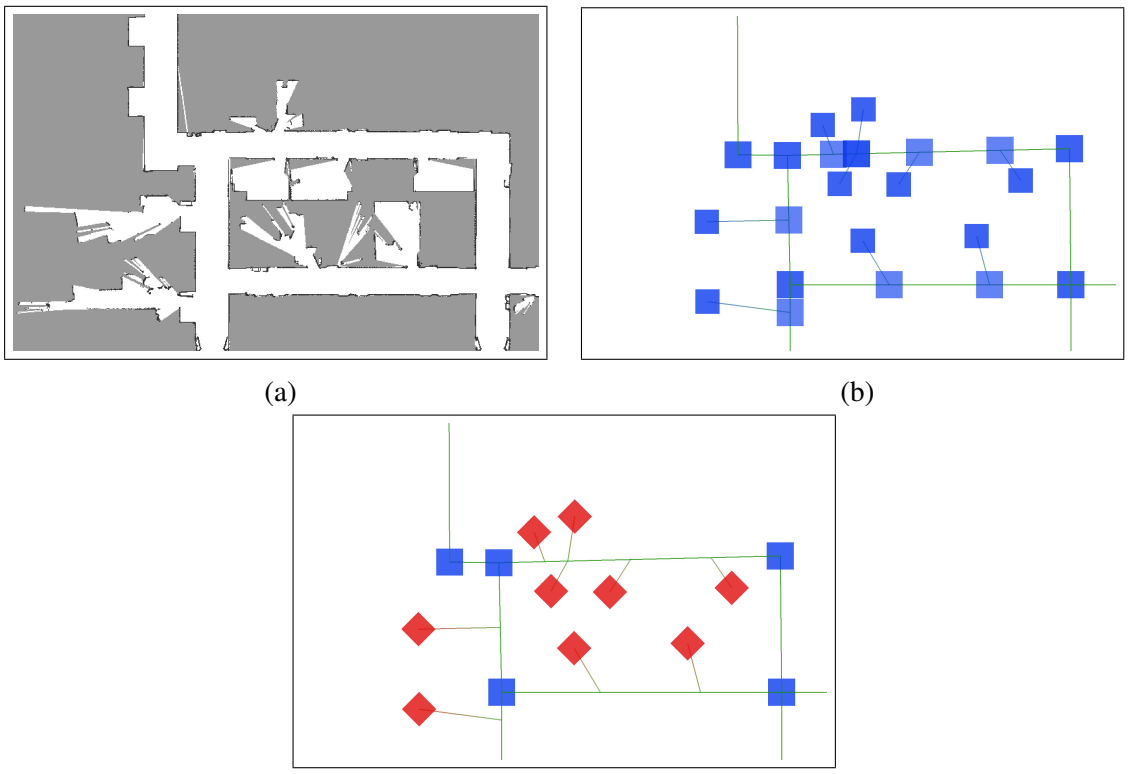


Figure 3.10: A comparison of the topological maps of an environment in the HSSH and H^{2}SSH. (a) shows an occupancy grid of a portion of a typical office building built by a robot navigating through the hallways of the building. (b) is the HSSH-style topological map of the building. The lines are paths and the squares are places. Notice how each office generates two places in the topological map, one place representing the office itself and a second place located in the hallway adjacent to the office. In comparison, (c) is the H^{2}SSH-style topological map of the building. The lines are path segments, the squares are decision points, and the diamonds are destinations. Notice how each office creates a single destination in the map connected to the adjacent hallway. The HSSH map requires 11 more places than the H^{2}SSH map to represent the same environment, each of which create potential loop closures as the robot explores the larger environment.

pending on the density of offices, a robot may have difficulty mapping even a simple office environment in real-time, and mapping larger environments may be completely intractable.

In contrast, the H^2SSH distinguishes between types of places. As the robot drives along a hallway, it identifies destinations and records their location along the current path segment – no loop closure hypotheses are created. As a result, all offices along the hallways in the building can be represented without increasing the number of possible map hypotheses.

CHAPTER 4

Place Detection and Labeling Using Affordances

This chapter describes a probabilistic algorithm for detecting and labeling places and paths in an environment. We use the symbolic representation of space in the Local Topological layer of the H^2SSH that partitions the environment into discrete areas. To create this partitioning, we first locate the potential transitions in an environment by finding a set of gateway hypotheses, which then define an initial set of area hypotheses. Given these potential gateways and areas, we learn a probabilistic model and use constraints defined by the H^2SSH representation to find a set of gateways and areas consistent with the H^2SSH. Using these labeled areas, we then transform the robot's trajectory through the environment into a sequence of topological events describing the sequence of areas visited by the robot. Our algorithm can be used for incremental place detection in small-scale space and can also be used to scalably partition large-scale maps, allowing it to be used for both topological SLAM and extracting a topological map directly from a global metrical map.

4.1 Place Detection for Topological Mapping

A topological map is appealing due to the simplicity of route-finding within the map and its correspondence with the human cognitive map. However, topological mapping is difficult because it requires the robot to create a higher-level abstraction of the environment when compared to a metrical map. In particular, topological navigation requires the robot to reliably detect a sparse sequence of places and paths as it travels through the environment.

As described in Chapter 3, the H^2SSH defines the topological map in terms of two navigation affordances. A *transition* affordance defines how the robot can change its topological location by crossing a gateway boundary between adjacent areas. A *travel* affordance defines how a robot can move along a path to reach the place at the other end, even when that place is currently outside the robot's sensory horizon. Given this affordance-based representation, place detection for the H^2SSH consists of finding the transition affordances

(represented by gateways) in the environment, which define the boundaries between areas, and assigning a label – path segment, decision point, or destination – to each area.

Potential gateways in the environment are found by identifying change regions and then associating a gateway with each region. These change regions are found using a classifier trained with hand-labeled gateways. Unlike the HSSH though, we do not assume that all detected gateways are meaningful boundaries in the local environment, but instead acknowledge that many of the gateway hypotheses may be false positives. Finding the true positive gateways is left to the area labeling algorithm.

The area labeling algorithm uses the gateway hypotheses to partition the environment into an initial set of area hypotheses. Together, these area and gateway hypotheses define a dynamic constraint satisfaction problem (CSP) that specifies the combinations of gateway and area labels that are consistent with the H^2SSH representation. The size of state space defined by the dynamic CSP grows exponentially with the number of gateway hypotheses, so exact inference is computationally expensive. Instead, we combine loopy belief propagation with a Markov chain Monte Carlo (MCMC) algorithm to perform variational inference in this state space. Our algorithm identifies inconsistent regions in the current assignment of labels to areas and samples transformations to find high-probability consistent solutions. This sampling process is repeated until a consistent labeling of the entire map is found. The probabilistic model that is sampled incorporates the change region probabilities for gateways, an area hypothesis probability learned from training data, and a boundary relation probability also learned from training data.

The solution found by the MCMC algorithm is a set of labeled areas and boundaries for a single map. During topological mapping in the H^2SSH, the labeled areas change incrementally as the LPM expands and contracts to contain the visible portion of the environment. When labeling these incremental maps, we initialize the areas in the environment using the solution from the previous map to help ensure consistency between two iterations of the sampling algorithm.

Finally, the goal of the place detection algorithm is the abstraction of the robot's continuous trajectory into a sequence of topological events. These events correspond to changes in the topological state of the robot: entering or exiting areas, or turning around on a path. The metric trajectory of the robot is overlaid on the labeled areas and boundaries to detect when these transitions occur.

In this chapter, Section 4.2 describes the algorithm used to locate the set of possible gateways. Section 4.3 defines our probabilistic model of the labeling problem. Section 4.4 describes the MCMC algorithm used to find a solution to labeling problem. We then describe how to adapt the MCMC approach to support incremental place detection within

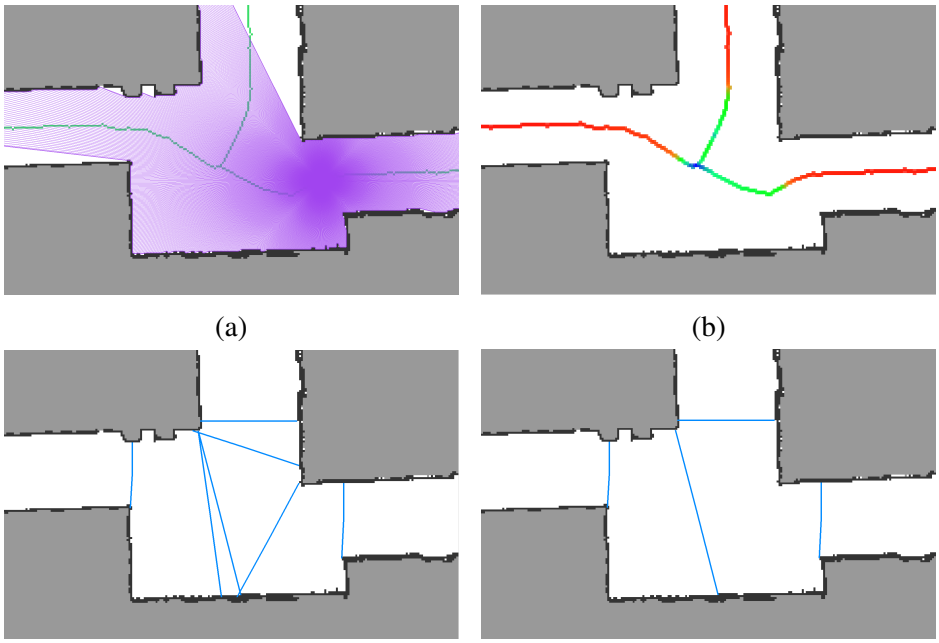


Figure 4.1: An illustration of the algorithm for finding gateways. (a) shows an example of an isovist calculated within the map. (b) shows the eccentricity isovist field for the Voronoi skeleton. (c) shows the initial set of gateways before intersecting gateways are removed. The line segments in (d) are the final set of gateways found in the map.

an LPM in Section 4.5. Finally, we describe how topological events are detected in Section 4.6.

4.2 Locating Potential Gateways

When dividing the environment into distinct areas, we first find potential gateways, or boundaries, between areas. In determining the initial set of gateways, we make the simplistic assumption that the robot's view of the environment within a single area is likely to be uniform. Thus, boundaries will exist where the change in the visible portion of the environment is greatest.

To quantify the robot's view of an environment, we use the concept of *isovists* [75]. An isovist is a polygon that represents the portion of the environment visible from a single (x, y) position and is analogous to a 360°scan from a laser rangefinder. Various properties of the isovist, such as the minimum distance to an obstacle or its eccentricity, allow the isovist to be abstracted into a scalar value. Finding the isovists throughout an environment allows for the construction of an *isovist field* F, which associates one or more scalar values with each isovist position in the environment.

Using the isovist field, we can partition an environment into regions by placing boundaries between areas where significant changes in the isovist field occur. In other words, the boundaries between areas occur where the change in the view of the environment is greatest.

Algorithm 4.1 Gateway Location Algorithm Outline

- Construct the reduced Voronoi skeleton of the map.
- Construct the isovist field F of the Voronoi skeleton.
- For each edge in the Voronoi skeleton:
  - Compute $p(g_c|F)$ for each cell c.
  - Create a gateway for each region with $p(g_c|F) > \gamma$.

Our algorithm for finding gateways, outlined in Algorithm 4.1, identifies these regions of visual change. We begin by computing the isovist field for the environment. We then identify where significant changes in the isovist field occur and create gateway in each of these regions. This approach generates false positives due to clutter, discretization errors, or incomplete knowledge of the environment. However, these false positives will be handled in the second phase of the classification process. The only requirement is to avoid false negatives because they cannot be discovered by our MCMC algorithm and result in distinct areas always being considered as one.

4.2.1 Calculating the Isovist Field

Within an occupancy grid, the isovist field can be approximated by calculating an isovist for each cell located in free space. In an occupancy grid, an isovist can be calculated by tracing some number of rays, N_{rays}, along angles in the range $\theta \in [0, 2\pi)$ from the isovist position (x, y) until they hit either an occupied cell, the edge of the map, or some maximum ray length r_{max}. The isovist polygon is created from these rays by connecting the endpoints of adjacent rays. Note that an isovist polygon contains the same data as if a 360°laser scan was gathered at (x, y), as can be seen in Figure 4.1a.

To find gateways, we compute the isovist field for the Voronoi skeleton of the environment. The Voronoi skeleton is a discretized approximation of the Voronoi graph of the environment, consisting of junctions, which are cells equidistant to three or more obstacles, and edges, which are a sequences of eight-way connected cells in the occupancy grid

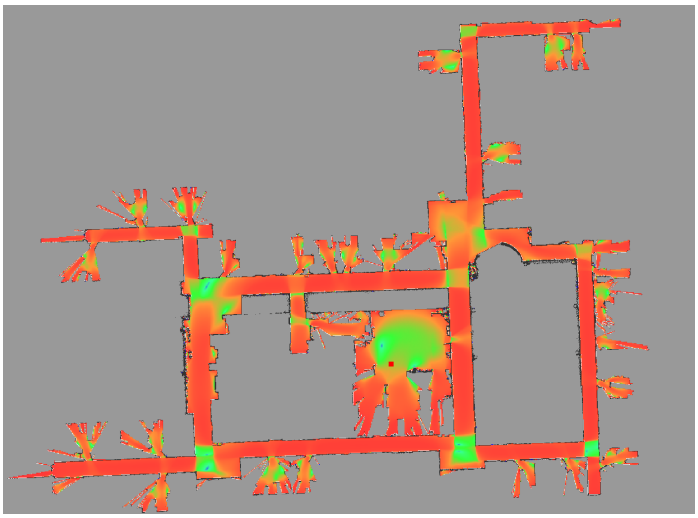


Figure 4.2: The eccentricity isovist field of the third floor of the Beyster building. The colors associated with the eccentricity area scaled between blue = 0, green = 0.5, and red = 1. Corridors have an eccentricity approaching 1, and thus appear as shades of red. Intersections between corridors have eccentricities that depend on the relative length of the incident corridors, so their color varies between orange, green, and blue. Notice the strong gradient that forms around the boundary between hallway intersections and office doors.

that are adjacent to two obstacles and run between junctions. We construct to the Voronoi skeleton using the algorithm described by Lau et al. [76].

In addition to reducing the computation needed to detect gateways, the Voronoi skeleton serves two additional purposes. First, isovists near the walls and other obstacles in the map can change dramatically based on clutter or noise in the occupancy grid. Using the Voronoi skeleton reduces the noise in the isovist field by only computing isovists that are maximally far from obstacles for a given part of the environment. Second, the Voronoi skeleton simplifies the creation of area hypotheses after the gateways have been computed by providing connectivity information between gateways, which is discussed in Section 4.4.

4.2.2 Finding Potential Gateways

Gateways in the environment occur where the visible area undergoes a substantial change. To measure the change in the environment, we compute three types of gradients in the Voronoi isovist field. The *max gradient* is the gradient of the isovist field at a cell in the skeleton. The *sum gradient* is the integral of the gradient within a small radius r of a cell. The *boundary gradient* splits the isovist in two halves, one on each side of the shortest line segment running through the isovist center, as shown in Figure 4.3. The gradient is the change in the isovist scalar computed separately for each half-isovist.

Using the three types of changes, we compute a feature vector for each cell in the

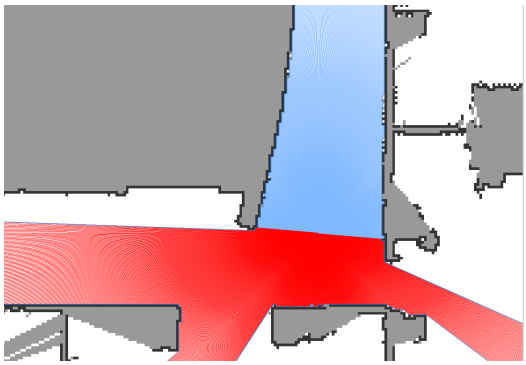


Figure 4.3: A split isovist divides the isovist in two halves. The split is along the shortest isovist line running through the center of the isovist. The half of the isovist that a particular ray belongs to is indicated by its color. The boundary gradient is the change in an isovist scalar computed for the half-isovists.

Voronoi skeleton that contains the max, sum, and boundary gradients for each scalar listed in Table 4.1. The max-gradient is computed using a quadratic Savitzky-Golay filter [77], which helps smooth noise in scalar field caused by discretization effects. For the sum-gradient, we use a radius of $r = 1$.

We find gateway hypotheses by computing the likelihood of a gateway being located at each cell in the Voronoi skeleton using:

$$p(g_c|F) = \eta p(F|g_c)p(g_c) \tag{4.1}$$

where $p(F|g_c)$ is learned from hand-labeled maps using an AdaBoost classifier with decision stumps and $p(g_c)$ is a uniform prior. We calibrate the distribution scores to more accurately reflect the underlying probability distribution using logistic correction [78].

Next, we find all cells along the edge where $p(g_c|F) > \gamma$. We create one gateway for each connected cluster of cells with probability above γ. We empirically found a value of $\gamma = 0.15$ to yield a very low number false negatives while keeping the number of false positives manageable.

We create a gateway g_c by connecting the two closest obstacles to c with a line segment. This definition can generate gateways that intersect or share endpoints. We filter these gateways using a simple check. If two gateways intersect or both endpoints are within a small amount ($0.25m$), we keep the more probable gateway and discard the other.

4.3 Classification Problem Formalization

The set of gateway hypotheses represents the potential boundaries between areas in the environment, thereby defining a set of area hypotheses in the map. Together, these gateway and area hypotheses define the variables in a graph, which we call the hypothesis graph, $H = (V, E)$. In the hypothesis graph, $V = \{A, G\}$, where A is the set of areas, and G is the set of gateways. Each gateway is connected to the two areas it bounds, and each area is connected to all bounding gateways.

Each area node $a \in A$ represents an area hypothesis and has a domain consisting of three possible labels $L(a) \in \{\Psi, P^{\Delta}, P^{D}\}$. Each gateway node $g \in G$ represents a gateway hypothesis and is a simple Boolean variable $B(g) \in \{0, 1\}$, which indicates whether or not the gateway is an active boundary, that is whether the area hypotheses on the two sides of the gateway should be considered one area ($B(g) = 0$) or two distinct areas ($B(g) = 1$).

The goal of the place classification algorithm is to find the most probable assignment of labels to A and active gateways in G that satisfy a set of constraints within the map M. The probability of an assignment of values to H is:

$$\begin{aligned} p(H|M) &= p(G, A|M) \\ &= p(G|A, M)p(A|M) \\ &\approx \prod_{i=1}^{|G|} p(g_i|A, M) \prod_{j=1}^{|A|} p(a_j|M) \end{aligned} \tag{4.2}$$

In this model (4.2), we make the Naïve Bayes assumption and treat the probability of each gateway as independent of other gateways, and the probability of each area as independent of other areas.

Note however, that a gateway's probability is conditioned on the areas it bounds in addition to the map. The intuition is that certain adjacency relations are more common than others. For example, destinations occur more frequently along path segments than at decision points, so gateways separating path segments and destinations are more probable than those separating decision points and destinations.

4.3.1 Area Label Probability

In most cases, the constraints on possible labels produce many consistent labelings for the areas in the environment. To address the ambiguity amongst labelings for an area a, we calculate the probability of a particular label $L(a)$, given a feature vector z_a for an area

within the map M:

$$\begin{aligned} p(L(a)|M) &= p(L(a)|z_a) \\ &= p(z_a|L(a))p(L(a))/p(z_a) \\ &\approx p(z_a|L(a)) \end{aligned} \tag{4.3}$$

where $L(a) \in \{\Psi, P^{\Delta}, P^{D}\}$, and we assume a uniform prior $p(L(a))$ across all possible labels and a uniform prior $p(z_a)$ across all possible feature vectors.

An essential feature of the area label probability is the ability to express degrees of confidence in the assignment of a particular label to an area. By examining the relationship between the probability for all possible labels, we can quantify how ambiguous a portion of the environment is. Being able to quantify is essential to the direct the MCMC sampling (Section 4.4) to high-probability portions of the search space.

The features z_a used to compute the likelihood function $p(z_a|L(a))$ for an area (Table 4.1) are a combination of isovist-based features that describe the local environment and visibility graph-based features that describe the area's relationship with the global environment. These features have been previously used to analyze environments [79, 80, 12, 13, 29]. Our approach hypothesizes the boundary of a place independently of its classification. Thus, we are also able to calculate features of the free space contained within the area as well as statistics about the isovist field instead of relying only on individual laser scans or isovists, as in other approaches [12, 13, 32, 30].

Individual Isovist Features
Area
Perimeter
Eccentricity
Circularity
Waviness
Compactness
Ray compactness (μ_d/max_d)
Local minimum angle[1]
$\{\mu, \sigma, \gamma\}$ ray distance
$\{\mu, \sigma\}$ distance to center of polygon
$\{\mu, \sigma\}$ difference in length of adjacent rays

Table 4.1: Individual isovist features used to calculate appropriateness.

[1] Angle between local minima of isovist ray distances [12]

Area Features
Number of gateways
Average gateway length
Maximum gateway length
Mean distance between gateways
Shape eccentricity
Shape circularity
Ratio of gateway length to shape perimeter
Minimum loop in distance Voronoi skeleton [13]
Eccentricity of Voronoi skeleton
$\{\mu, \sigma\}$ of the eccentricity isovist field
$\{\mu, \sigma\}$ of the circularity isovist field
$\{\mu, \sigma\}$ of the waviness isovist field
$\{\mu, \sigma\}$ of the compactness isovist field
$\{\mu, \sigma\}$ of the ray compactness isovist field
$\{\mu, \sigma\}$ of the local minimum angle isovist field
σ/μ of isovist area
σ/μ of isovist perimeter
σ/μ of isovist ray distance
σ/μ of isovist distance from polygon center
σ/μ of the distance ratio of adjacent isovist rays
$\{\mu, \sigma\}$ of the clustering coefficients of the visibility graph [79]
$\{max, \mu, \sigma\}$ of the betweenness centrality of the visibility graph [81]
$\{\mu, \sigma\}$ of the page rank of the visibility graph [82]
$\{\mu, \sigma, max\}$ of the betweenness centrality of the Voronoi skeleton [81]

Table 4.2: Area-based features used to calculate appropriateness.

Using hand-labeled maps, we trained two classifiers to distinguish between path segments, destinations, and decision points. Like Mozos et al. [12], one isovist classifier was trained to classify a single isovist using the individual isovist features 4.1. The other classifier was trained using to classify an area hypothesis using the area features 4.2. Both classifiers were trained using AdaBoost with decision stumps with a one-versus-all approach.

The probability α_a^L of a label L applying to an area a is calculated by combining the results of the area and isovist classifiers:

$$\alpha_a^L = \frac{p_{area}(z_a|L(a))p_{iso}(z_a|L(a))}{\sum_{l\in\{\Psi,P^\Delta,P^D\}} p_{area}(z_a|l)p_{iso}(z_a|l)} \tag{4.4}$$

where p_{area} is the area feature classifier and p_{iso} is the isovist feature classifier. p_{iso} is

computed by taking the mean of the probability the isovists contained in the area.

4.3.2 Gateway Boundary Probability

The probability of a gateway vertex $p(g|A, M)$ from (4.2) is a distribution over the values $B(g) = \{0, 1\}$. We compute this distribution using (4.5):

$$p(g|A, M) \approx p(g|M)p(g|A) \tag{4.5}$$

$p(g|M)$ is the probability computed by the gateway classifier in (4.1).

To define $p(g|A)$, we assume a gateway is conditioned only on its adjacent areas $p(g|N(g))$. We compute a categorical distribution all thirty-two possible tuples of the gateway active flag and labels $\langle B(g), a_1, a_2\rangle$. Thirty-two pairs are possible because we distinguish between adjacency at a path segment's endpoint versus along its side.

We estimate the distribution by simply counting the instances of each label combination in the hand-labeled maps used for training. A uniform Dirichlet prior is used to avoid zero probability singularities for pairs with no instances.

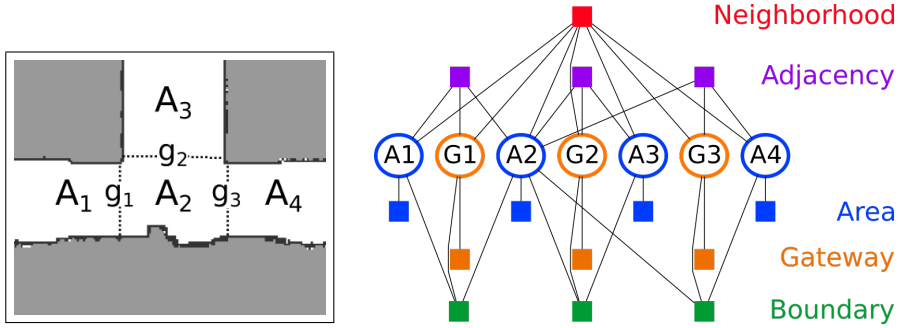
4.3.3 Area Label Constraints

Many assignments of values to H are not valid representations of the environment as described in Table 3.3, which imposes restrictions on the valid configurations of areas for an environment. These restrictions can be formulated as constraints on the possible labels that can be assigned to the areas in A. These constraints are as follows.

- A path segment must terminate at a place.
- A decision point must be adjacent to at least two unaligned path segments.
- An area on the left or right side of a path segment must be a destination.
- A decision point is only adjacent to path segments or destinations.
- The graph induced by the path segments and decision points is connected.

4.3.4 Factor Graph Formalization

The place classification problem described in this section can be formulated as a factor graph that describes the probabilistic model associated with areas, gateways, and the constraints between them. Each area and gateway is represented by a variable in the factor graph.



(a) (b)

Figure 4.4: The factor graph representation of a T-intersection. Note how the unaligned constraint must connect with every area and gateway hypothesis.

Five types of factors exist in the graph:

- The probability distribution over labels for each area.
- The probability distribution for a gateway, given the change in the isovist field.
- The probability distribution for a gateway, given its boundary areas.
- The adjacency constraint between areas that share a gateway boundary.
- The neighborhood constraint between an area and its adjacent gateways and area hypotheses.

The distributions associated with (1), (2), and (3) are detailed in Sections 4.3.1 and 4.3.2. The constraint factors are factors that take a value of 1 or 0 depending on whether or not the constraint is satisfied. Thus, any area a conditioned on a failing constraint has $p(a) = 0$.

Figure 4.4 shows an example of the factor graph for a map containing a single T-intersection. The area and gateway factors correspond closely to the CRF model from Friedman et al.[13], but the constraint factors are unique to our approach.

Unfortunately, the structure of this factor graph makes approximate inference using loopy belief propagation infeasible. The neighborhood constraint must be connected to all gateways and areas adjacent to an area. For the T-intersection, $3^4 2^3$ states exist, and more complex and cluttered environments have many more potential gateway boundaries.

Worse still, the neighborhood constraint needs to account for gateways that can potentially be adjacent to an area in the final classification of the map, which means that not only gateways directly adjacent to an area, but theoretically all gateways and areas in the map have to be connected for the neighborhood constraint to account for all possible gateway configurations.

4.4 Place Classification with MCMC

Formally, the goal of the place classification algorithm is to find an assignment of area labels $\{\forall a \in A : L(a) \in \{\Psi, P^{\Delta}, P^{D}\}\}$ and boundary active flags $\{\forall g \in G : B(g) \in \{0, 1\}\}$ that satisfies all constraints described in Section 4.3.3. However, exponential growth in the number of possible states in the factor graph for the constrained place classification problem means a solution cannot be found via exact (or approximate) inference using belief propagation because the size of messages being passed around the network grows exponentially with the number of gateways in the map.

We address the exponential growth by using a two-step approach for place classification. First, we find the approximate marginals for all areas and gateways using loopy belief propagation on a simpler factor graph without constraint factors. Note that even though the constraints are not explicit, the boundary probability $p(g|N(g))$ provides a soft constraint on the valid relations between adjacent areas because no invalid relations appear in the training data. Next, we use Markov chain Monte Carlo (MCMC) sampling [83] to efficiently search for a complete assignment of values to variables. The basic algorithm is listed in Algorithm 4.2 and described in more detail below.

Algorithm 4.2 PlaceClassificationMCMC(H)

Input: H : The hypothesis graph of an environment.

```
H' ← LoopyBeliefProp(H)
while IsInconsistent(H) do
    F ← InconsistentAreas(H)
    T ← ∅ // A collection of possible transformations to H.
    for n = 1 to N_samples do
        f ← Sample(F)
        T_G ← SampleTransformationSequence(f, H)
        T.Add(T_G)
    end for
    T_n ← Sample(T)
    for t in T_n do
        H' ← ApplyTransformation(t, H')
    end for
end while
return H'
```

Each iteration of MCMC sampling transforms the hypothesis graph to a new state: $H \rightarrow H'$. An iteration begins by finding the set of all inconsistent areas F (areas with at least one failing constraint) in H. The inner loop creates $N_{samples}$ possible transformation sequences of H. Each transformation sequence is a sequence of transformations that as-

Algorithm 4.3 SampleTransformationSequence(f, H)

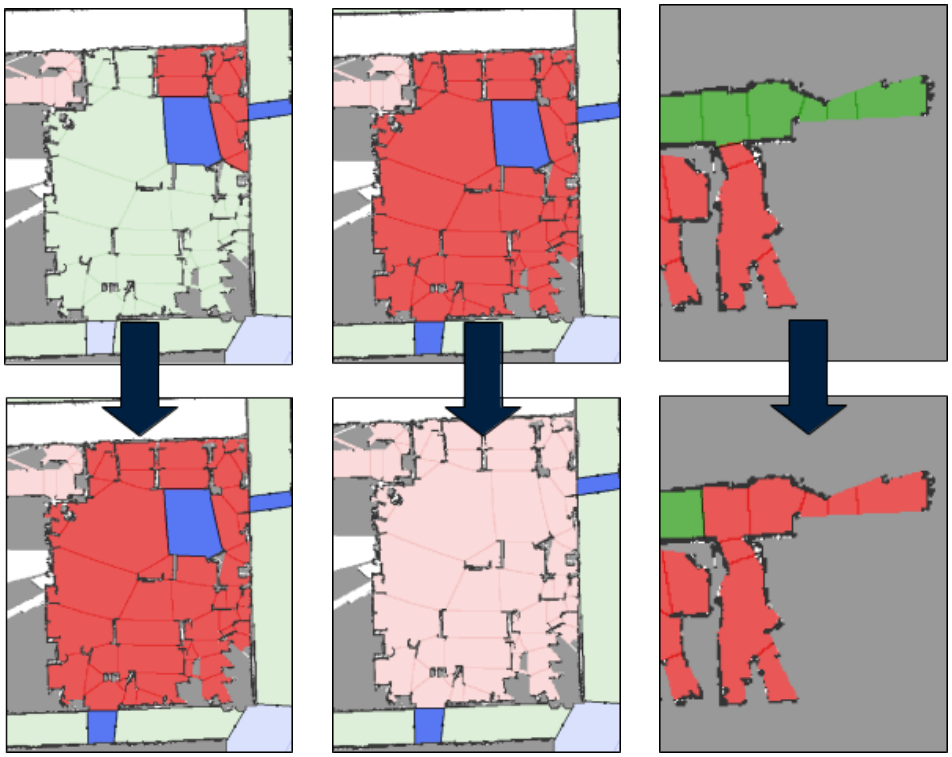
Input: f : An inconsistent area in H.
Input: H : The hypothesis graph containing f.
1: $G \leftarrow f \bigcup N(f)$ // Add neighbors of f to sampling set.
2: $H' \leftarrow H$
3: $T \leftarrow \emptyset$
4: $N_{trans} \leftarrow |G|$
5: $n \leftarrow 0$
6: **while** IsInconsistent(f) **and** $n < N_{trans}$ **do**
7: $a' \leftarrow$ UniformSample(G)
8: $t_{a'} \leftarrow$ GenerateTransformations(a', H)
9: $t' \leftarrow$ Sample($t_{a'}$)
10: T.Add(t')
11: $H' \leftarrow$ ApplyTransformation(t', H')
12: $n \leftarrow n + 1$
13: **end while**
14: **return** T

signs new values to a subset of nodes in the neighborhood of $f \in F$. After creation of the transformation sequence, the sequence to apply to H is sampled from T. The probability of sampling $t' \in T$ is $p(H_{t'}|M)/\sum_{t \in T} p(H_t|M)$. Using these transformations and the probabilistic model, H quickly converges to a consistent solution.

When computing $p(H_{t'}|M)$ during sampling, we add a penalty $p_f = \omega|F|$ computed using the number of failing constraints for a hypothesis, where F is the set of failing constraints. We initialize $\omega = -3$. Then if the number of constraints $|F|$ does not decrease between iterations, we increase ω by 5%. In this way, the constraint penalty will come to dominate the overall probability of a hypothesis, forcing the search to out of an inconsistent local minima and to other portions of the search space.

Algorithm 4.3 attempts to find a transformation sequence to H that makes an inconsistent area f consistent. To solve the failing constraints associated with f, we consider transformations to its local neighborhood in the graph $N(f)$ because f's constraints directly depend on these neighboring areas.

The goal of sampling a transformation sequence is to make an area consistent. Consequently, we attempt to sample transformations until the area is consistent. However, the sampling can't be guaranteed to find a solution on any given iteration. To ensure that we don't waste excessive resources attempting to solve a particularly difficult problem, we limit the total length of a sequence to the square of the size of the neighborhood, $|N(f)|^2$. Doing so ensures that samples are spread widely enough through the search space to eventually find a reasonable solution.



(a) Change label (b) Merge areas (c) Split area

Figure 4.5: Examples of the transformations sampled by our MCMC algorithm. The top row of maps are the initial maps and the bottom row are the maps after the transformation was applied. Dark colors indicate areas with failing constraints, while light areas have all constraints satisfied. As can be seen, some transformations cause new constraints to fail, while others solve constraints.

To find the next transformation in the sequence, we sample an area $a' \in G$, generate all possible changes to a', and then sample one of the transformations t' and add it to the sequence T. An area a' is a connected subgraph of H containing a subset of A joined only by inactive gateways ($B(g) = 0$).

For an area a', we generate a set of possible transformations that create a new graph H'. These changes are described below and shown in Figure 4.5.

- Change the label for a'.
- Merging a' with adjacent areas with the same label.
- Removing and relabeling an area from a' and merging it with any neighbors with the new label.
- Splitting a' into multiple areas by changing an inactive gateway to be active. The split areas are merged with any adjacent areas with the same label.

After applying the transformation t' to H', the areas in G change. Some transformations create new areas or remove them. As areas are created or removed, G is updated accordingly so the areas sampled are not the same on each iteration.

We run the MCMC sampling until a configuration H^* is found that satisfies all constraints. Once all constraints are found, we further improve the labels by performing a deterministic local search starting from H^*. If a change is found that increases the probability of H^*, we apply it. This process iterates until a local maximum is found.

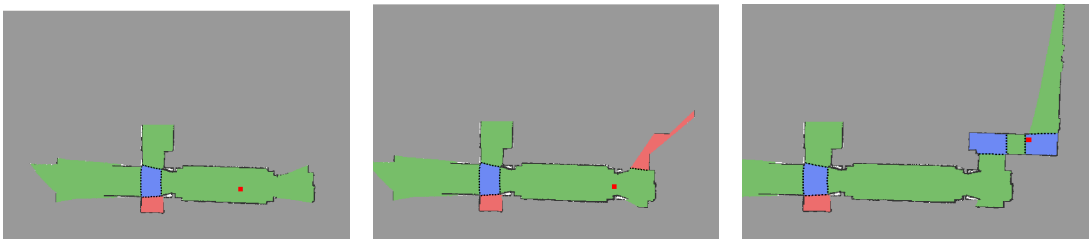
4.5 Incremental Place Detection

The place classification algorithm described so far is suitable for extracting a topological map from an existing global metrical map. However, global topological mapping requires places and their associated topological events to be detected incrementally within the LPM as the robot explores the environment. This problem is more difficult because less information is available about the overall structure of the environment and because the robot's knowledge of the environment changes as the robot explores. Furthermore, the classified places must be stable across time to avoid false positive or false negative place detections, which can cause an incorrect topological map to be built.

The most straightforward approach for place detection is to run the above place classification algorithm on each LPM constructed during exploration of an environment and to use the resulting places and topological events for global topological mapping. However, as the robot travels through and explores the environment, its knowledge of the structure and shape of the environment changes. The changing structure of the map will change the visibility-based features that we rely on for locating gateways and calculating the probability of the area labels. These changing features can cause the solution found by the labeling algorithm to change, as shown in Figure 4.6.

A label changing while a robot is in that area amounts to the robot changing its mind about the topological structure of the environment. As a result, portions of the global topological map must be re-evaluated whenever the labels change, which can be computationally expensive. Also, if following a series of turn-based directions, an error can result in the robot becoming lost or not reaching its intended goal.

We make three modifications to our place classification algorithm to improve the stability of the estimated labels. First, we track the final gateway boundaries over time to ensure the gateways that defined the place boundaries from the previous solution are included in the set of potential gateways for the current iteration. Second, we defer the detection of transitions to new areas when they are sufficiently uncertain. Finally, we separate previ-



(a) (b) (c)

Figure 4.6: The area labels for the environment will change as the robot explores more of the environment. In (a), the path segment the robot (red square) is currently on appears to end. In (b) the robot has explored more of the environment and now detects what it believes to be a destination on the left side of the path segment. In (c), the robot has moved through the zig-zag in the environment and sees a long path segment. Upon seeing this path segment, the previously-visited environment is determined to be a sequence of path segments and decision points, rather than a single destination as was initially found in (b).

ously visited areas from areas currently being explored and consider only the latter during place classification.

Gateway Consistency The potential gateways in the environment are found based on the changes in the visibility of the environment. For a static global metrical map, this visibility information does not change over time. However, within the map of small-scale space, the visible portion of the environment changes as the map stretches and scrolls with the robot's motion. As a result, gateways that formed boundaries between labeled places can disappear due to a change in the map causing previously labeled places to change. Our straightforward solution to this problem is ensure all gateways from the places found on the previous update exist on the current update.

Deferring Area Transitions During incremental place detection, the boundary of the map is continually adjusted as new parts of the environment come into view. However, during exploration the map will sometimes simply not contain enough information to definitively establish the area labels, as illustrated in Figure 4.6.

Fortunately, the robot knows what it doesn't know and can easily identify the unexplored frontiers in the map. Using these frontiers, we define an uncertain transition as a transition from a path segment or decision point to a destination that contains a frontier large enough for the robot to travel through. Though more sophisticated measures of uncertainty could be used, we find this simple check dramatically reduces the number of false place detections.

Figure 4.6 shows how a false transition is avoided. What initially appears to be a des-

tination is actually some other configuration of areas, as the robot quickly discovers so no effort is wasted undoing the results of transitioning to the incorrectly-classified destination.

Classifying Only Unexplored Territory As defined in Section 3.3, small-scale space contains the robot's current area, as well as all neighboring areas. With the exception of the initial robot location, the robot was previously located at one of these neighbor areas. During incremental place detection, we fix the boundaries and label of this previously visited area. By fixing this area, the search for labels is focused on the boundaries and label of the robot's current area. Furthermore, changes in the visibility do not alter previously detected places. We implement this fixed-label area by adding a new constraint to the search that simply returns false if the label or boundary of the area changes.

4.6 Detecting Topological Events

When traveling through a metric map, the robot's motion can be described by a sequence of poses $x_{0...T}$. Correspondingly, the robot's motion through a topological map can be described by a sequence of locations $l_{0...K}$ (Section 3.4.1).

Whereas the robot's pose is typically estimated at a fixed time interval, the robot's topological location changes only when specific topological events occur: transitioning to a new area or turning around on a path segment. Otherwise, the robot's motion through the world leaves the topological location unchanged. We explain how each of these events are detected below.

Area Transitions

An *area transition event* describes the motion from one area to another. In the H^{2}SSH, moving between two areas is accomplished by crossing a gateway separating them. To detect area transition events, we store the sequence of robot poses since the time the last transition occurred, $x_{t_k...T}$, where t_k is the time at which event k occurred. After each update of the area labels, we check each consecutive pair of poses in the sequence to see if a line segment connecting them intersects a gateway boundary. If an intersection with a gateway boundary is detected, then the robot has transitioned to a new area. When a transition is detected, all poses before the time at which the transition occurred t_{k+1} are erased from the pose sequence.

The above approach works well given perfect pose information. However, real-world localization estimates have some uncertainty associated with them. As a result, the robot's pose estimate may jump back and forth across a gateway boundary in some situations.

To avoid the creation of spurious area transition events, we add a small hysteresis to the detection of consecutive area transitions involving the same gateway. Instead of immediately generating an area transition event when the robot is detected to cross the gateway, we require the robot to travel a fixed amount ($0.5m$ for our implementation) before area transition events can occur for the same gateway.

Turning Around

A *turn around event* describes the topological action that occurs when the robot changes the direction it is traveling along a path segment. When the robot enters a path segment, it is moving from the place at one end to the place at other end. Depending on which place the path is entered from, the robot is either moving in the $+$ or $-$ direction. To move in either of these directions, the robot's heading must be approximately aligned with the axis of the path segment pointing away from the entry place. If the robot's heading changes sufficiently such that it is now pointing towards the entry place, the robot can be said to have turned around.

To detect turn around events, we compare the robot's current heading θ_t with the heading of the path segment's axis θ_ψ. If $\theta_t - \theta_{psi} > \frac{\pi}{2}$ when the robot is driving forwards or $\theta_t - \theta_{psi} < \frac{\pi}{2}$ when the robot is driving backwards, then the robot is longer facing the place in its current direction. Thus, the direction should be reversed. Like with area detection events, we require the robot to move a short distance ($1m$) before confirming it has turned around to avoid generating many events when simply turning in place.

4.7 Discussion

In this chapter, we described an algorithm for detecting and labeling places and paths in an environment using navigation affordances. Central to our algorithm was the use of isovists and their associated isovist field for understanding the structure of the environment.

Compared to other place detection algorithms that also rely on learned classifiers, our approach is unique in guaranteeing the consistency of the solution with respect to an underlying knowledge representation. Whereas other approaches [32, 13, 12, 33] learn a classifier and naively apply it to maps, we enforce the constraint that place labels must be meaningful for navigation. Consequently, while the results of our labeling algorithm occasionally create some strange places, a well-defined symbolic route can be planned to every place.

Additionally, our method supports incremental place detection as a robot explores an environment, which no other classification approach has used. Previous approaches to place

detection have either focused solely on discovering decision points [11] or landmarks [25, 84]. However none of these methods provide both place detection for a structure-based topological map and semantic classification of the environment.

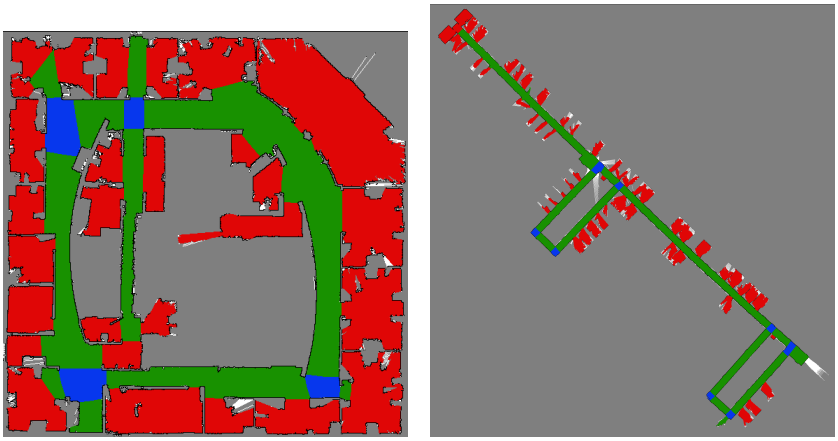
CHAPTER 5

Evaluating Topological and Semantic Abstractions

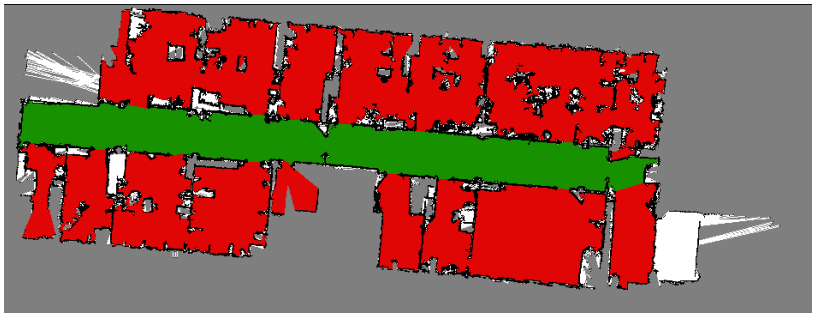
This chapter evaluates the place labeling and detection algorithm detailed in Chapter 4 using 17 datasets gathered on the University of Michigan campus and standard datasets from the Radish repository [85]. We trained a classifier for each building included in our results using a leave-one-out-approach. For multi-floor buildings, we also left all floors of the building out of the training data. Different floors of a single building are likely to have many of the same architectural properties and would thus bias our training data to improve those particular results.

The output of the place labeling algorithm is a semantic map of the environment that classifies locations in the environment based on their structure and navigation affordances. Appendix A contains the complete set of maps used, including the metric map that was labeled, our ground-truth semantic map, and the semantic map created MCMC algorithm.

We note that ground truth for place labels is subjective, and likely to vary between different people and researchers. Therefore an exact comparison requires using the same labeled test data, which is not generally available. Furthermore, no standard set of labels exists, so they also vary between different approaches. For example, some approaches consider doorways as one possible label [12, 13], whereas doorways in our approach are represented by gateways, rather than a separate label. As a result where we make comparisons, we compare directly against results presented in the cited paper, so the ground truth being compared against is different. However, to help understand this variability, we recruited a small number of people to hand-label a set of maps. These results are presented in Section 5.3. In other results, ground-truth was created by the author's own interpretation of the environment. In all cases though, the overall results presented here show our approach to be comparable to other state-of-the-art approaches.



(a) Intel (b) ABuilding



(c) Fr79

Figure 5.1: The best results for our MCMC sampling algorithm on standard maps used for comparison with prior approaches in Table 5.1 and Table 5.5. The colors are red for destinations, green for path segments, and blue for decision points.

5.1 Cell-by-Cell Evaluation

Cell-by-cell comparison is the standard metric in place classification and is computed by comparing the ground-truth label with the assigned label for each free cell in the occupancy grid of the map being evaluated. The metric is the percent of correctly labeled cells.

Using standard maps, Intel, Fr79, and ABuilding, we compare our MCMC algorithm to existing approaches, ranging from Mozos' pioneering work [12] to very recent results using deep learning [30, 28]. Other approaches exist [86, 33], but do not provide comparable metrics. From these papers, we selected results where training data and testing data came from separate environments, which is most comparable to our approach.

As can be seen, our results are on-par with the state-of-the-art across all three maps. Most place classification algorithms are able to achieve similar results of classification rates in the 90s for office-corridor-type environments and 80s for less rectilinear environments, namely Intel.

Interestingly, the earliest work by Mozos et al. [12] achieves classification rates similar

Table 5.1: Comparison of cell-by-cell accuracy of place classification approaches. (%)

Method	Fr79	Intel	ABuilding
MCMC	95.3	84.01	**95.99**
Ada[12, 13]	93.94	82.23	88.6
VRF$_D$[13]	91.50	86.40	93.60
VRF$_M$[13]	91.30	**88.20**	94.20
SVM[29]	92.23	85.47	
CRFoGVG[29]	**99.38**	76.78	
SPCoGVG[29]	92.04	86.89	
CNN-Liao[30]	89.76	82.40	
CNN-Goeddel[28]	92.40		

to more recent methods. The main improvement has been in been the spatial consistency of approaches that integrate spatial connectivity information via the Voronoi skeleton, including [13, 29, 30] and our work. In addition to the improved spatial consistency of Voronoi-skeleton-based methods, our MCMC algorithm also constructs a knowledge representation of the environment suitable for topological navigation and mapping.

Our results in the larger set of maps (see Appendix A) show similar performance to the standard maps. In corridor-like environments like the Infinite Corridor, EECS, or BBB, we have strong results. Our algorithm struggles more in environments with larger open spaces, like GGB or CSAIL, where large open spaces with substructure are interpreted as large destinations rather than a sequence of path segments and decision points.

In the worst case, significant clutter causes oversegmentation of map, which is readily apparent in Figure 5.3. Here we see a great deal of confusion in the Pierpont map, where dining tables, couches, and pillars block the visibility of the larger environment. These occlusions create sharp gradients in the isovist field, which then create high probability gateways. With these types of environmental features, the Voronoi-skeleton-based algorithms are likely to yield higher performance because they are not subject to the constraints our algorithm is. Alternately, if one is trying to describe how to navigate through the clutter, the resulting H^{2}SSH map provides a rigorous way to describe a route through the various tables.

Another difficulty in our evaluation environments occurs when large rooms like lecture

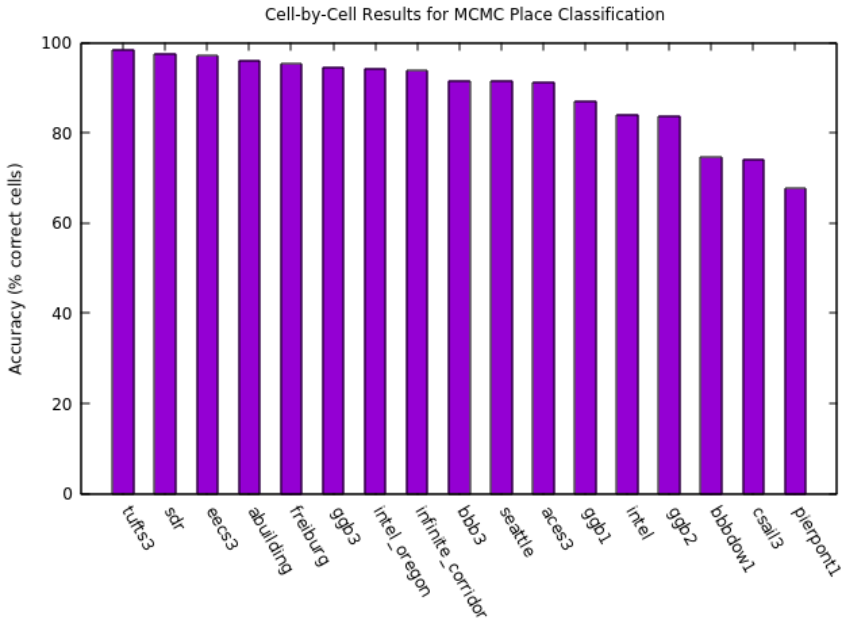


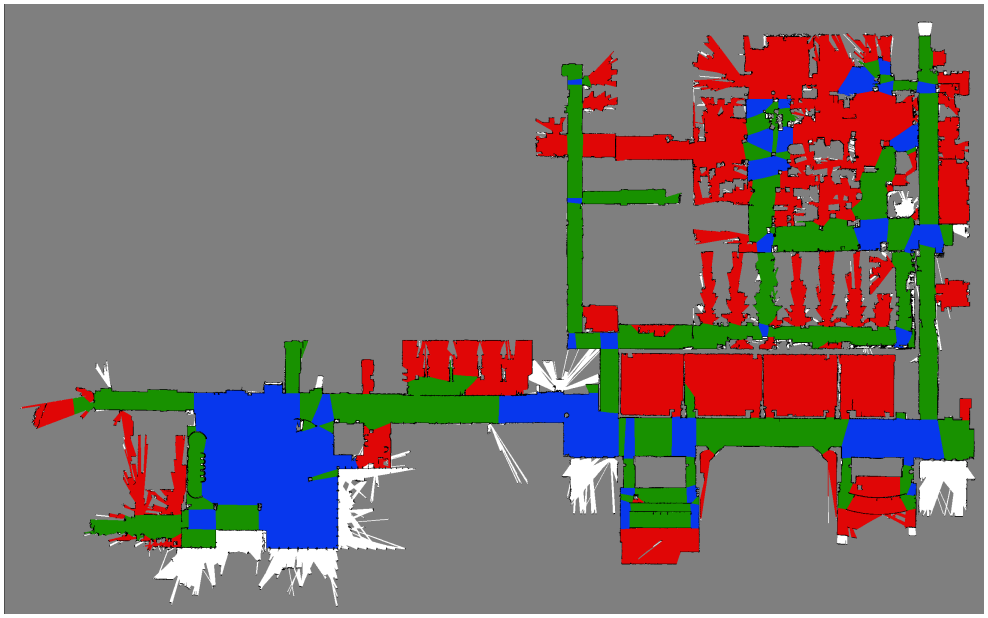
Figure 5.2: Results for cell-by-cell evaluation for all test maps. Our performance across additional environments yields similar performance to standard datasets for corridor-type environments, but more cluttered or unusual environments cause difficulty, as explained in the text.

Table 5.2: Statistics for cell-by-cell accuracy across all test maps.

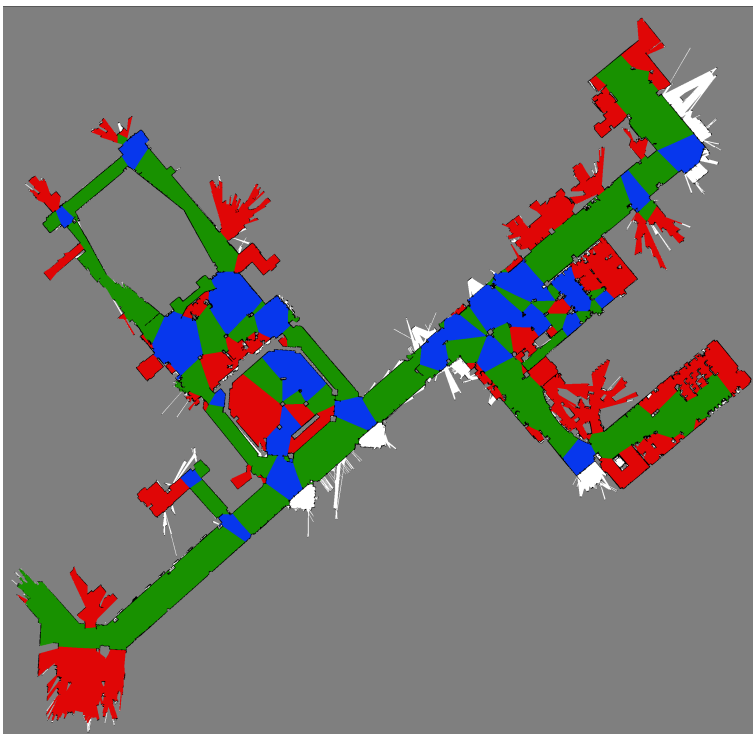
Min(%)	Max(%)	Mean(%)	Std(%)	Median(%)
68.1	98.5	85.3	9.9	87.2

halls contain significant internal structure. An appropriate representation for these rooms would be a hierarchical structure that views them has rooms from the hallway, but small topological maps internally. Our approach is currently unable to create this structure, as a result features like the lecture halls along the west side of the bbbdow1 environment force unexpected decision points to appear in the middle of the main corridor.

One final note: When analyzing the results of prior works, we noticed discrepancies in the Fr79 and Intel maps between different approaches, which we highlight in Figure 5.4. In Figure 5.4a, the bottom right region is absent in ground-truth data provided by [12], which we believe is the reason it is absent from later papers. In analyzing the map further, our glass detection and mapping algorithm [87], detected glass along the wall separating the orange and white regions, which is a possible cause for its removal from the dataset. The other regions removed by [29, 30] in Figure 5.4a and Figure 5.4b have no easily explained reason.



(a) The first floor of the Beyster and Dow buildings



(b) The first floor of the Pierpont and Chrysler buildings

Figure 5.3: Labeled maps of difficult environments at the University of Michigan.

(a) Fr79 (b) Intel

Figure 5.4: In the Fr79, regions labeled in orange are missing from [12, 29, 30, 28]. Additionally, [29, 30] are missing the blue region. In Intel, [29, 30] remove the orange regions from the map.

5.2 Topological Error Evaluation

The primary aim of our place classification algorithm is to construct a semantic map representation of the environment grounded in the affordances available to a mobile robot. However, a cell-by-cell comparison does not capture the usefulness of the semantic abstraction for navigation. In particular, cell-by-cell comparison does not evaluate the spatial consistency of the semantic labels. For example, mislabeling a rarely visited alcove has less impact than mislabeling a decision point at the intersection of the most ¡used hallways in a building. However, the cell-by-cell comparison does not capture topological errors in the place labels, e.g. placing corridors in the middle of rooms or rooms in the middle of corridors. To more directly measure topological errors in the place labels, topological edit distance was introduced by Friedman et al. [13].

Topological edit distance (TED) compares the sequence of places along routes through the two different labelings of an environment. Each route is converted to a string with a different character representing each type of label and then the edit distance (with insertion and deletion costs of 1) between the two values is calculated. Friedman et al. normalize the topological edit distance by the length of the route, which means the range of values is 0 (for an exact match) to 200 (for every label being wrong). Thus, $TED/2$ is approximately the percent of the length of a route spent in an area with the wrong label while navigating from a starting location to a goal.

Though we present results for the topological edit distance of our approach (Section 5.2.2), we do so only to allow a comparison with [13]. Instead we use a metric inspired by topological edit distance, Route Edit Distance (RED), which better captures error that occur during topological navigation. Whereas topological edit distance samples the route along the Voronoi skeleton at a fixed interval, route edit distance treats navigating through an area

Table 5.3: Summary of the route edit distance for all test maps.

Best	Worst	Mean	Std	Median
0.111	3.510	1.163	1.065	0.700

as a single action, regardless of its size. This distinction is important because in topological navigation to the end of a path segment, for example, is treated as a single topological action during route planning and execution, and not as a sequence of actions that depends on the path's size. Thus, route edit distance is proportional to the number of incorrect actions that would be executed by a robot following a route described using the ground-truth map of the environment.

5.2.1 Route Edit Distance

To evaluate our MCMC algorithm using route edit distance, we sampled 1000 random pairs of positions in the map and computed the minimum length path between them following the Voronoi skeleton. The string representation of the route adds a single character per area visited. The route edit distance is computed by comparing the string with the string generated in the ground-truth map using an edit distance with insertion, deletion, and modification costs of 1. We use a modification cost of 1 so that we get a one-to-one correspondence between action errors, i.e. visiting a decision point instead of a path segment, rather than a wrong label having an edit distance of 2 like with topological edit distance.

Table 5.3 shows a summary of the route edit distance for all test maps. On average our approach generates about 1 error per route through the environment. However, exactly how these errors manifest themselves in real-world navigation is difficult to determine because a variety of error-recovery techniques are possible in topological navigation. If we consider a robot navigating under the direction of a human operator, this error rate means the robot would require assistance in reaching its goal on average once per route.

The error rate only provides an average number of interventions, which can be affected by outliers or small portions of the environment that are poorly labeled. Table 5.4 shows the zero error rate for our test maps. The zero error rate is the percentage of routes with a route edit distance of 0, meaning the robot would successfully reach its goal with no interventions. In over half the maps, the zero error rate is over 50% with the highest error rate having a failure on only one-in-ten routes.

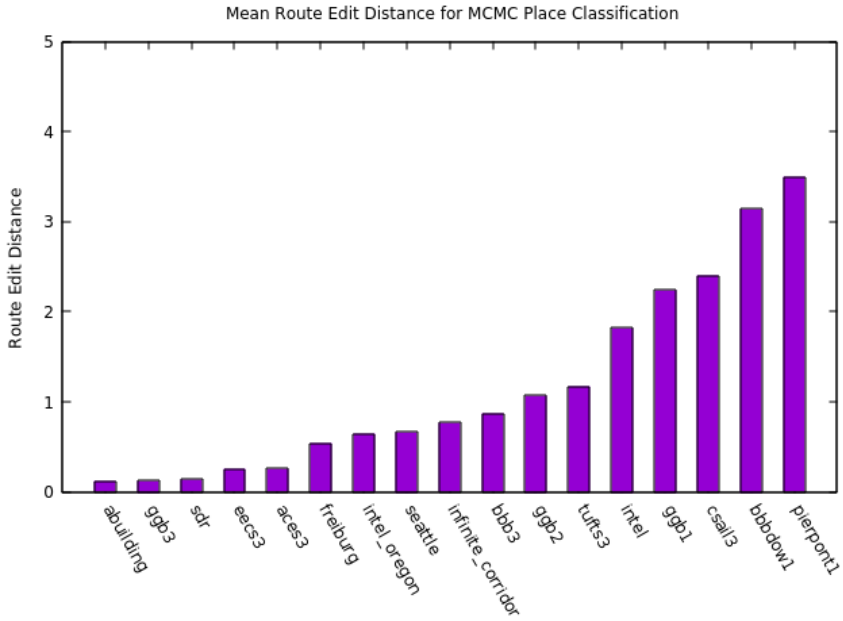


Figure 5.5: The route edit distances are shown here for all test maps. Maps with an edit distance less than 1 average less than one error per route through the map. Notice though that a high cell-by-cell accuracy does not necessarily correspond to a low route edit distance. The tufts3 map has the highest cell-by-cell accuracy (98.5%), but has an average route edit distance of 1.17 because the one labeling failure is a decision point along the one corridor that runs through the entire environment. Conversely, ggb2 has an accuracy of 83.8%, but a lower average route edit distance (1.07) because most of the errors are in boundaries being in different locations relative to ground truth, rather than the underlying topological structure being different.

Table 5.4: Zero error rate for all test maps.

Map	Zero Error Rate (%)	Map	Zero Error Rate (%)
abuilding	89.4	bbb3	56.5
ggb3	88.5	ggb2	48.7
sdr	87.3	tufts3	37.3
eecs3	85.0	ggb1	32.4
aces3	82.9	intel	19.6
seattle	69.5	bbbdow1	13.1
intel_oregon	69.2	csail3	12.8
freiburg	65.6	pierpont1	12.6
infinite_corridor	62.4		

Table 5.5: Comparison of topological edit distance.

Method	Fr79	Intel	ABuilding
MCMC	**5.85**	38.3	**12.1**
Ada_S[13]	76.6	62.6	79.4
Ada_{SC}[13]	74.2	59.8	35.7
VRF_D[13]	23.7	25.7	18.2
VRF_M[13]	21.0	**22.2**	14.3

5.2.2 Topological Edit Distance

We directly compare against Friedman et al. [13], who calculate a label at fixed distances along the Voronoi skeleton, by using the same approach for computing the topological edit distance. The exact distance between nodes was not indicated in their paper, and we estimate it to be $0.5m$. For the following results, we generated 1000 random pairs of positions in the map and computed the minimum length path between them following the Voronoi skeleton.

A comparison of topological edit distance shows our approach produces a comparable description of the environment than both variations of Voronoi random fields [13]. For Fr79 and ABuilding, our excellent classification accuracy and greater spatial consistency results in lower error with respect to ground-truth.

However, our performance on the Intel map is inferior. We believe this to be caused by the different in types of labels. The VRF uses single nodes for intersections, which are easily defined, even in curved hallways like Intel. Our decision point makes it less clear where a decision point might be located and so our algorithm creates a curved path segment instead. This path segment is along the shortest path between many areas in the map, so it causes the edit distance to spike. Comparing the labeled maps visually (Figure 5.6 vs. Figure 1 [13]) though, the overall topological structure of our map better classifies the paths and decision points. Our description of the path running through the center of the map is correct (though not the decision point at the end), whereas the VRF creates a disconnected path segment surrounded by two rooms.

5.3 Human Labeling Variability

In all evaluation so far, we have compared maps against hand-labeled ground truth maps created by the author. However, interpretations of an environment will vary from person to

Figure 5.6: The labeled Intel map produced by our MCMC algorithm. (Repeated from Figure A.1 for convenience.)

person. This variability can have a potentially significant impact on the evaluation of the place labeling algorithm.

To better understand the variability in a person's interpretation of an environment, we asked eight robotics graduate students to hand-label six occupancy grid maps. For each map, they were asked to draw gateway boundaries and label each of the areas as destination, decision point, or path segment. The maps were then converted into digital maps by the author.

Using these human-labeled maps, we perform two analyses. First, we evaluate our algorithm against each of these alternate ground-truth maps, as well as our ground-truth used for other results. Second, we evaluate each human-labeled map against all other human-labeled maps. We also create a visualizations of the per-pixel agreement between the human-labeled maps (an example (Figure 5.8) is copied from the complete set of results in Appendix A.2).

Table 5.6 and Figure 5.7 show the variability in the semantic interpretation of the environments amongst the participants labeling maps. There is strong agreement about the rectilinear corridor-and-office environments (eecs3 and bbb3) – note that all students work in BBB and attend classes in EECS, so they are very familiar with the environments. Once the environment is less orthogonal and contains larger open spaces though, we see much more

Table 5.6: Variability of MCMC and human labels compared against human labelings.

Method	MCMC	Human	MCMC	Human
Map	Cell-by-cell	Cell-by-cell	Route edit distance	Route edit distance
bbb3	90.8 ± 0.03	92.9 ± 0.03	1.00 ± 1.34	1.41 ± 1.56
bbbdow1	64.5 ± 0.05	71.7 ± 0.09	4.05 ± 1.30	3.34 ± 1.72
csail	69.6 ± 0.03	71.3 ± 0.07	2.89 ± 0.57	2.64 ± 1.00
eecs3	94.8 ± 0.03	94.6 ± 0.03	0.94 ± 1.09	1.30 ± 1.44
intel	85.8 ± 0.03	89.7 ± 0.04	1.84 ± 1.58	1.72 ± 1.56
pierpont1	65.5 ± 0.02	76.9 ± 0.07	4.20 ± 0.76	1.80 ± 0.79

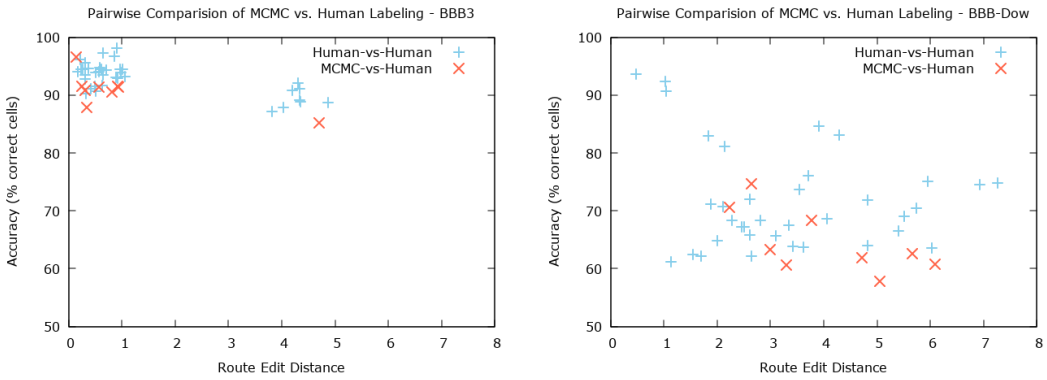
disagreement. For example in Intel, we found participants were split evenly on whether a single path segment runs along the right side of the map or whether it is multiple path segments intersecting at decision points along the curve (Figure 5.8). We see similar disagreement about the precise location of the bottom left decision point.

For most of the experiment maps, our MCMC algorithm produces results in line with the variability observed amongst the labels created by experiment participants. However, the oversegmentation problem in pierpont1 that breaks up the large exhibit hall and sitting areas has no analogue in the human-labeled maps, which all easily separate the clutter from the large-scale topological structure.

5.4 Discussion

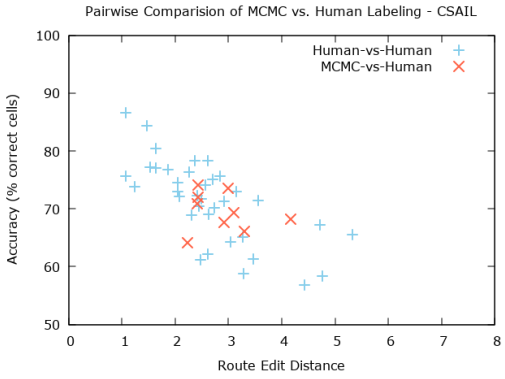
We have evaluated our MCMC place classification using an extensive set of maps, created at the University of Michigan and retrieved from the Radish repository [85]. Compared to other approaches using standard datasets, our algorithm exhibits state-of-the-art performance. We achieve the highest cell-by-cell accuracy of any method on the ABuilding dataset and the second highest accuracy on the Fr79 dataset. However, as shown in Figure 5.4, the results from Shi et al.[29] have some rooms removed from their maps, so a direct comparison based on their published results is less meaningful.

To evaluate the quality of the topological abstraction created by our algorithm, we introduce a metric, Route Edit Distance, based on the Topological Edit Distance [13] to compute an estimate of the number of errors the robot would make navigating to a goal using the topological abstraction. On average, the abstraction generated by our MCMC algorithm has only one error per route.

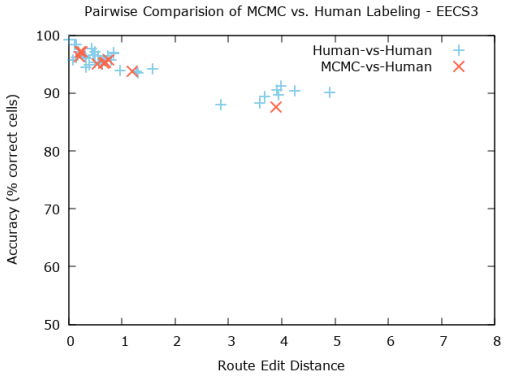


(a) bbb3

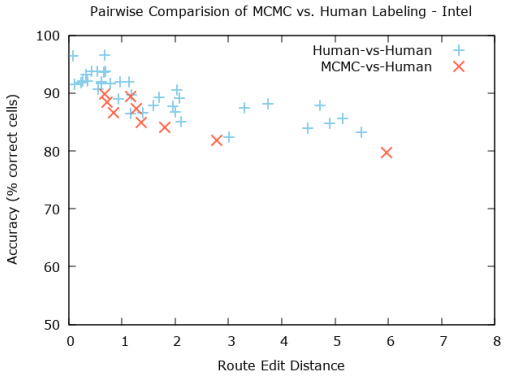
(b) bbbdow1



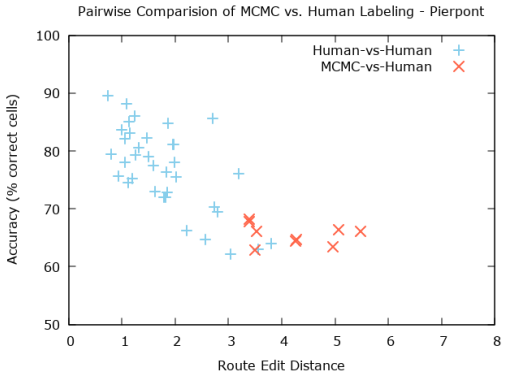
(c) csail



(d) eecs3

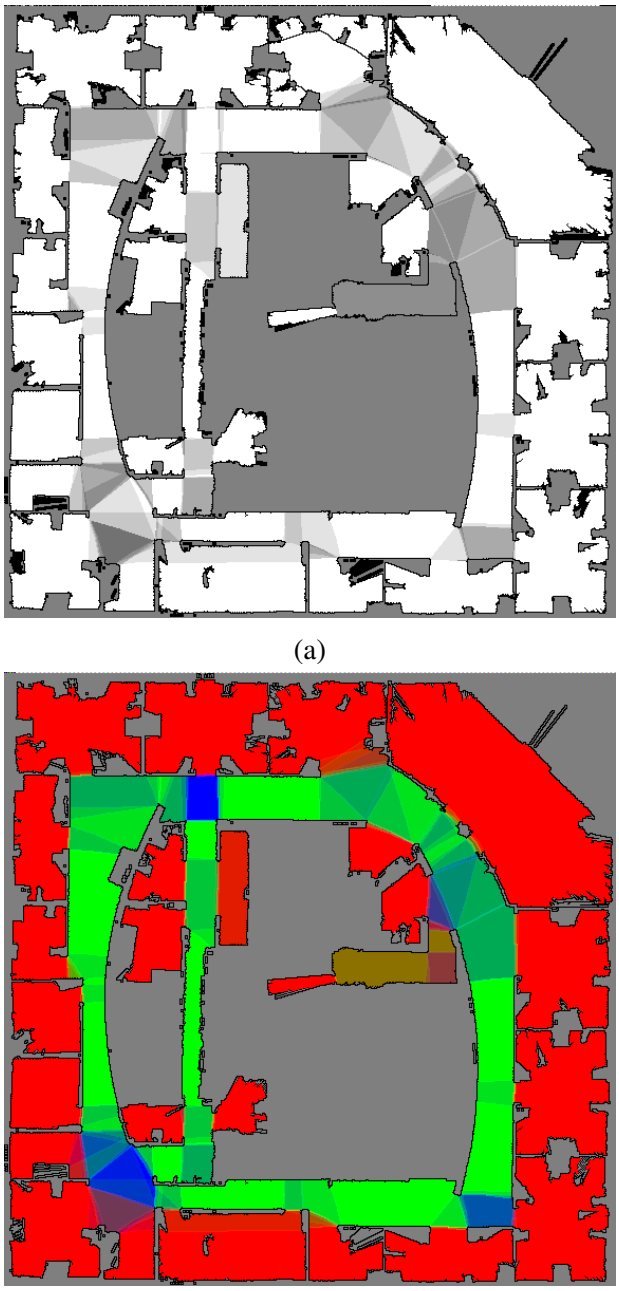


(e) intel



(f) pierpont1

Figure 5.7: These plots show the raw comparison data used to generate the numbers in Table 5.6. Each '+' is shows the result of comparing one human labeling against another. Each 'x' is a comparison of our MCMC labeling against a human labeling. Note that for most maps, our MCMC results fall within the cluster of human-vs-human results. In the bbb3 and eecs3 maps, an outlier in the human labels exists. This person placed a decision point in front of each office door, which causes the much higher route edit distance between their map and all other labels, including our MCMC labels.



(a)

(b)

Figure 5.8: Human variability results for intel. From Section A.2: (a) shows the consensus amongst the different labelings. White indicates complete agreement between all maps, while light gray is maximum disagreement (all classes with the same number of votes), the grayscale value is set using $L_{max} - L_{min}$. Black areas in the map are free space cells without a semantic label because they correspond to areas on the other side of glass walls or errant beams during mapping. (b) shows what labels were assigned by the human labelers. Each labeled map votes green (path segment), red (destination), or blue (decision point) for each cell in the combined map. Thus, path segment/decision point ambiguity is teal, decision point/destination ambiguity is purple, and path segment/destination ambiguity is brownish.

In our evaluation, we have introduced many new maps not previously used in the place classification literature. Some of these environments are similar to previous hallway-and-office maps, but others are considerably larger and more varied than any previously used maps. Our MCMC algorithm fails to attain the same performance on these more complex maps than it does on the simpler standard maps.

Some of the difficulty with these environments is due to our algorithm, however we hypothesized that some errors were due to natural variation in the semantic interpretation of an environment. We performed an initial experiment to test this hypothesis by having eight robotics graduate students hand-label maps of six environments of varying complexity.

Our human map-labeling experiment shows that ground-truth for semantic maps is extremely subjective. For simple environments, like a building with mostly offices along hallways, we see little variation. Beyond these environments though, there can be little expectation that two people will agree on exactly how to describe many environments. Consequently, future research in semantic mapping should incorporate ground-truth from multiple sources to better evaluate the quality of the generated abstractions.

While variability in interpretation (Table 5.6) can account for much of the below-expected performance, bbbdow1 and pierpont1 present challenges that our current approach cannot handle. In particular, densely cluttered environments, like tables in a communal dining space, create high-confidence gateways that require a probabilistic model that considers more than just the distribution across the initial area and gateway hypotheses to dismiss as insignificant.

CHAPTER 6

Scalable Topological Mapping Using Lazy Evaluation

This chapter presents a scalable approach to topological mapping. We describe a topological mapping algorithm that uses lazy evaluation to search the space of topological map hypotheses. By performing lazy evaluation on the map hypotheses in the tree of maps, our algorithm avoids the need to prune hypotheses to maintain computational feasibility. Furthermore, the heuristic search focuses expansion of the tree on the most likely hypotheses, allowing a small number of hypotheses to be evaluated for each visited area, thereby allowing real-time updates. Thus, the algorithm operates online and in real-time, while ensuring the correct map is not pruned from the hypothesis space, which, to the best of our knowledge, no existing topological mapping algorithm achieves.

6.1 The Topological Mapping Problem

A topological mapping algorithm builds a graph-like map that identifies the connectivity of places within a robot's environment. The input to the algorithm is a sequence of topological events that describe the places visited by the robot, $Z^K = \{z_0, \ldots, z_K\}$, and the actions taken at each place, $U^K = \{u_1 \ldots u_K\}$. The goal is to find the correct topological map of the environment, M_K^*, consistent with the detected places and actions taken by the robot.

In most cases, multiple consistent topological maps exist for a given sequence of places and some ambiguity exists as to which topological map is correct. As a result, most topological mapping algorithms maintain many hypotheses about the structure of the enviroment. For each place in the sequence, a number of potential map hypotheses are created, each asserting a potential loop closure with an existing place or the discovery of a new place. The benefit of maintaining multiple map hypotheses is that it allows ambiguity about a loop closure to be resolved when disambiguating information becomes available in the future. The basic topological mapping algorithm is outlined in Algorithm 6.1.

Algorithm 6.1 BuildTopologicalMap(Z^K, U^K)

Input: Z^K : A sequence of detected places
Input: U^K : A sequence of actions taken at the corresponding place
Output: H_K : A set of consistent topological maps $\{H_K = \{h_K^1, \ldots, h_K^n, \ldots, h_k^N\}\}$

```
H_0 ← InitialMap(z_0)
for k = 1 to K do
    for h_{k-1}^n ∈ H_{k-1} do
        UpdateRobotLocation(h_{k-1}^n, u_k)
        if IsConsistentHypothesis(h_{k-1}^n, z_k) then
            CreateChildHypotheses(h_{k-1}^n, z_k, H_k)
        else
            DeleteHypothesis(h_{k-1}^n)
        end if
    end for
end for
return H_K
```

While the topological mapping approach outlined in Algorithm 6.1 can be used naively, the number of map hypotheses can hyper-exponentially [8], thereby making the space of possible topological maps intractably large. To combat this exponential explosion of map hypotheses, a number of approaches have been developed. First, the robot can explore the environment using a strategy that minimizes the number of consistent map hypotheses [16, 22, 23, 15, 88]. Second, analytic constraints, like graph planarity [89], can eliminate some map hypotheses. Third, metric information, like the distance between places, and appearance models, like occupancy grids of places or a collection of visual words, can be used to calculate a probability distribution across the consistent map hypotheses. We will focus on this third approach, probabilistic topological mapping, along with the construction of topological maps using the H^{2}SSH representation, for the remainder of this chapter.

Our algorithm builds on the tree of maps by developed Dudek et al. [39] to represent the space of topological map hypotheses. The tree of maps is a data structure where each node represents a consistent topological map hypothesis, which is a tuple containing a topological map and the robot's location in that map. The leaves of the tree are the map hypotheses consistent with all observed places, and the depth of the tree is equal to the total number of places visited. When a new place is entered, a new set of hypotheses are generated for each leaf in the tree of maps by asserting every consistent loop closure within each map hypothesis as well as the hypothesis that the robot has entered a new place.

Like [17], we extend the tree of maps representation by annotating each leaf of the tree

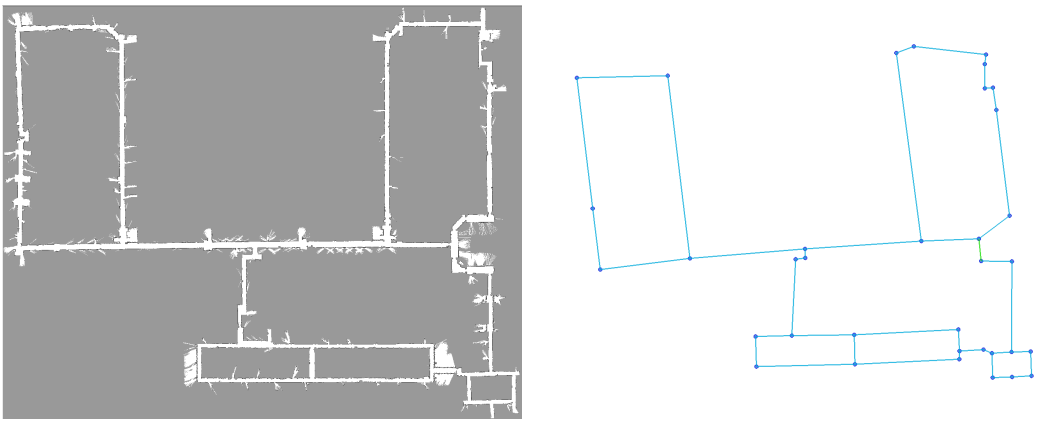


Figure 6.1: The topological map constructed of the standard Infinite Corridor dataset. Despite drift in the local reference frame of the robot, the correct topological map is still the most probable map hypothesis. Unlike a metric map, the distortion caused by local reference drift does not affect the navigability of the map, as the robot simply navigates from place to place and the metric information only serves as a heuristic for route planning.

with a posterior probability, which is used as a heuristic for determining the order in which to evaluate nodes. When the robot arrives at a place, the leaf nodes are expanded in order of decreasing posterior probability. To limit the overall size of the tree, on each update we expand a subset of all leaf hypotheses, considering only the most probable hypotheses.

Expanding a node generates a new set of consistent map hypotheses that incorporate the new place. The size of the tree of maps can be further reduced by lazily expanding the child of a map hypothesis. We accomplish this reduction using the robot's position uncertainty. Using this approach, the robot can close small loops with little ambiguity, while closing large loops naturally increases the robot's uncertainty about the environment, thereby generating more hypotheses about its possible structure.

In addition to our lazy evaluation algorithm, the H^2SSH representation itself improves the scalability of the topological mapping problem. By categorizing different types of areas and recognizing the difference between them, the possible set of loop closures generated when entering a new place is reduced because only a subset of places will be of the same type. For example, the mapping algorithm can distinguish between entering an intersection and entering an office, meaning those two actions generate disjoint sets of child hypotheses. In contrast, with other representations with a single type of place, like the HSSH [11] or the GVG [22], each door encountered along a corridor generates potential loop closures, rapidly growing the hypothesis space. Though many of these possible loop closures will quickly be eliminated, they create additional work that is completely avoided by treating these destinations as separate from the decision points in the environment.

While the use of place when discussing topological mapping is standard, the remain-

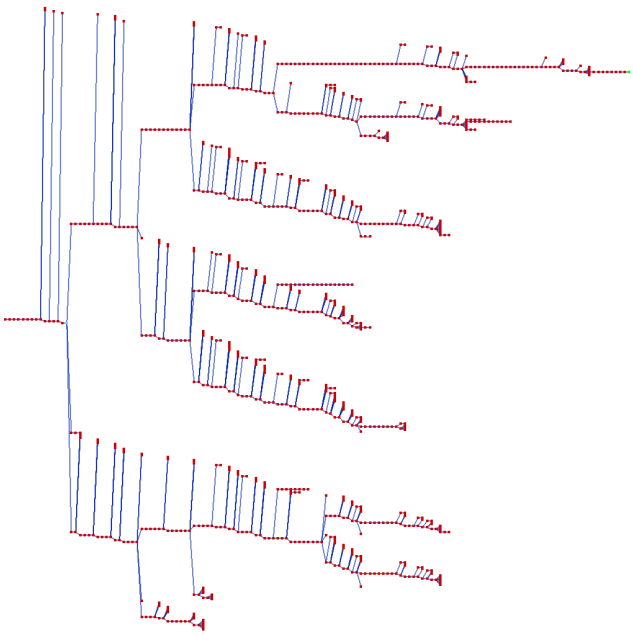


Figure 6.2: The probabilistic tree of maps built by our lazy evaluation algorithm when mapping the Infinite Corridor dataset. Nodes in the tree are red and parent-child relations are blue lines. Notice the many leaves of the tree that are never expanded due to a low probability.

der of this chapter will now discuss topological mapping in terms of areas. This change emphasizes that loop closures can be detected amongst any entities in the topological map (Section 6.5).

6.2 Probabilistic Tree of Maps

In the tree of maps, the set of maps at each depth k of the tree, $H_k = h_k^0, h_k^1, \ldots, h_k^N$, represents the space of all topological map hypotheses consistent with all areas visited by the robot up to event k, where the subscript k indicates the last topological event included in the hypothesis, and the superscript n enumerates a specific map hypothesis. We extend the tree of maps representation by annotating each leaf of the tree with a posterior probability, $p(h_k^n|Z^K, U^K)$, thereby creating a probabilistic tree of maps. Figure 6.2 shows the tree of maps constructed for the Infinite Corridor dataset.

Because each depth k within H_K represents the entire space of consistent topological maps including all events update to depth k, the following holds:

$$\sum_{n=1}^{N} p(h_k^n|Z^K, U^K) = 1 \tag{6.1}$$

Table 6.1: Definition of symbols used to describe the probabilistic tree of maps.

Symbol	Description
$U^K = [u_0, u_1, \ldots, u_K]$	The sequence of actions up to event K.
$Z^K = [z_1, z_2, \ldots, z_K]$	The sequence of observations up to event K.
$z_k = \langle m_k, c_k, \lambda_{k-1,k} \rangle$	The observations at event k.
m_k	The LPM of the observed place at event k.
c_k	The local transition cycle observed at the area k.
$\lambda_{k-1,k}$	The displacement of reference frame of area k from the frame of area $k-1$.
Λ^K	The observed displacements between places in Z^K.
$h_k^n = \langle M^n, X^n, A^n, \chi^n \rangle$	A map hypothesis in the tree of maps incorporating events up to depth k
$\langle a, m_a, c_a \rangle : a \in A_k^n$	The set of areas in a map hypothesis h_k^n.
M_k^n	The topological map for h_k^n.
X_k^n	The robot's location in M_k^n.
χ_k^n	The planar embedding of M_k^n.

Problematically though, the size of H_K has the potential for hyper-exponential growth as a function of K in many navigation scenarios [8]. Consequently, evaluation of the full distribution in (6.1) is intractable, even for relatively small values of K, as we will show in Section 6.6. Before describing our algorithm for combating the exponential growth of the probabilistic tree of maps in Section 6.4, we first detail how the probability of a topological map hypothesis is calculated.

6.3 Map Hypothesis Probability

We now describe how we calculate the probability of a map hypothesis in the probabilistic tree of maps. As shown in Table 6.1, each map hypothesis h_k^n contains a topological map M_k^n, a set of areas A_k^n, and a planar embedding of the map χ_k^n. Using these values, applying Bayes' Rule, and dropping irrelevant conditionals, we arrive at the following equation for the probability of a map hypothesis. The distributions in (6.4) will be described in the

following sections.

$$p(h_K^n|Z^K,U^K) = \eta p(Z^K|h_K^n,U^K)p(h_K^n|U^K) \tag{6.2}$$
$$= \eta p(c^K,\Lambda^K,m^K|M_K^n,X_K^n,A_K^n,\chi_K^n,U^K)p(M_K^n,X_K^n|U^K) \tag{6.3}$$
$$\approx \eta p(c^K|A_K^n,M_K^n)p(\Lambda^K|\chi_K^n,M_K^n)p(m^K|\chi_K^n,M_K^n)p(M_K^n,X_K^n|U^K) \tag{6.4}$$

6.3.1 Local Transition Cycle Likelihood, $p(c^K|A_K^n, M_K^n)$

Given a sequence of observed local transition cycles, $c^K = c_{0:k}$, we calculate $p(c^K|A_K^n, M_K^n)$, the likelihood of observing this sequence of local transition cycles given the topology of the map. Since we currently assume that the local topology is correctly detected, the likelihood for a local transition cycle is either 0 or 1, as defined in (6.5), where C_K^n is the global transition cycle of the place visited at event k in M^n.

$$p(c_k|C_k^n,M_K^n) = \begin{cases} 1 & \text{if } c_k = C_k^n \\ 0 & \text{if } c_k \neq C_k^n \end{cases} \tag{6.5}$$

$$p(c^K|A_K^n,M_K^n) = \prod_{i=0}^{K} p(c_i|C_i^n,M_K^n) \tag{6.6}$$

6.3.2 Place Layout Likelihood, $p(\Lambda^K|\chi_K^n, M_K^n)$

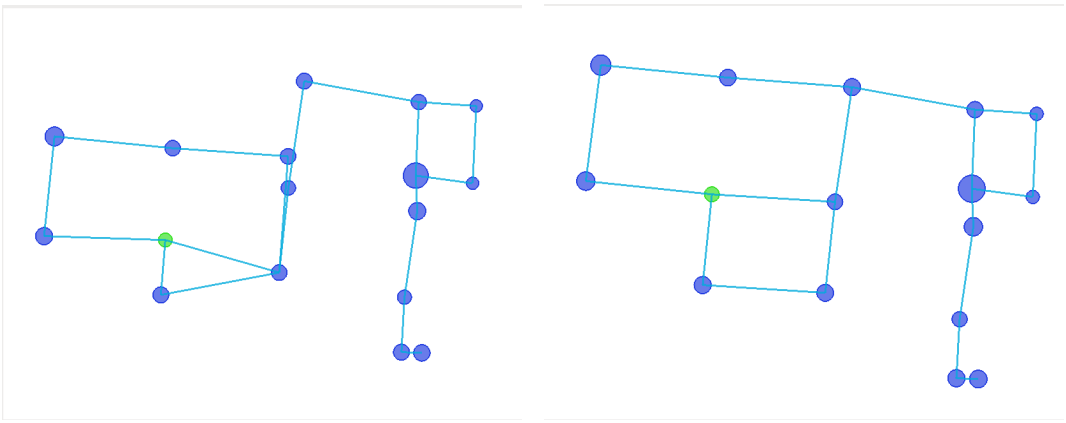
Given a map M_K^n and a set Λ^K of lambdas $\lambda_{k-1,k}$ describing the measured displacement between each successive pair of visited places, we can calculate the maximum posterior place layout χ_K^n. Each $\lambda_{k-1,k} \in \Lambda^K$ is a Gaussian distribution representing the transformation from the origin, $(0,0,0)_k$, of area a_k to the origin, $[(0,0,0)_k]_{k-1}$, of area a_{k-1}.

$$\mu_{\lambda_{k-1,k}} = (\Delta x, \Delta y, \Delta\theta) \tag{6.7}$$

$$\Sigma_{\lambda_{k-1,k}} = \begin{bmatrix} \sigma_{\Delta x}^2 & 0 & 0 \\ 0 & \sigma_{\Delta y}^2 & 0 \\ 0 & 0 & \sigma_{\Delta\theta}^2 \end{bmatrix} \tag{6.8}$$

The area layout χ minimizes the cost function:

$$\begin{aligned} E_\chi &= (\Lambda_\chi - \Lambda^K)^T \Sigma_{\Lambda^K}^{-1} (\Lambda_\chi - \Lambda^K) \\ &\propto -\log p(\Lambda^K|\chi_K^n, M_K^n) \end{aligned} \tag{6.9}$$



(a) (b)

Figure 6.3: (a) When an incorrect loop closure is made, the final layout takes on unusual shapes. This map is a portion of the EECS Building, where no triangular loops actually exist. (b) For reference, this is the correct map of the environment given the topological events that have occurred up to this time.

Thus, we minimize the log-likelihood $-\log p(\Lambda^K|\chi_K^n, M_K^n)$ to obtain:

$$\chi^n = \arg\max_{\chi} p(\Lambda^K|\chi_K^n, M_K^n) \tag{6.10}$$

We then use this value $p(\Lambda^K|\chi_K^n, M_K^n)$ as the likelihood of a given area layout. Figure 6.3 shows the optimized χ_K^n for two topological map hypotheses.

Note that the cost function in this minimization is the same cost function used in a variety of pose-graph-based SLAM approaches. The key difference between the area layout and a pose graph lies in the nodes in the graph. In our topological SLAM, each node represents an area in the environment, whereas each node in the pose graph represents a pose along the robot's trajectory. This topological graph is typically a much coarser representation than the corresponding pose graph of the same trajectory through the environment.

6.3.3 Area Compatibility Likelihood, $p(m^K|\chi_K^n, M_K^n)$

As defined in the H^{2}SSH, each area $a \in A^n$ represents a non-overlapping and distinct portion of the environment. Therefore, a map hypothesis in which distinct areas physically overlap is less likely than a map hypothesis with no overlap. Using χ^n, the extent of each LPM can be mapped into a single global reference frame, which allows for straightforward calculation of the overlap between two areas.

For each $a \in A^n$, χ^n specifies the pose of the area center, χ_a^n. The associated LPM, m_a, can be transformed to be centered at χ_a^n, giving m_a^χ. The compatibility between two areas

is defined in (6.11) and is based on the ratio of the overlapping extent between the areas to the minimum of the area extents.

$$c(m_i, m_j) = \frac{\text{extent}(\text{intersection}(m_i^{\chi}, m_j^{\chi}))}{\min(\text{extent}(m_i), \text{extent}(m_j))} \tag{6.11}$$

Using (6.11), the overall area compatibility, $p(m^K|\chi_K^n, M_K^n)$, of a map with N_a distinct areas is the product of the compatibility between each pair of areas in M^n.

$$p(m^K|\chi_K^n, M_K^n) = \prod_{i=0}^{N_a} \prod_{j=i+1}^{N_a} exp(-\beta c(m_i, m_j)) \tag{6.12}$$

6.3.4 Map Hypothesis Prior

The prior, $p(M^n, X^n|U^K)$, estimates a prior probability for a map hypothesis considering only the actions taken by the robot – no sensor information is included. Thus, the distribution represents the probability that the robot is in an environment whose map has the structure of M^n and that the robot happens to be at X^n after K events. As discussed by Tully et al [17], we have no way to know this distribution.

Given that we can't know the true distribution $p(M^n, X^n|U^K)$, the simplest option is to assume a uniform prior across all possible maps. If no assumptions are made about the nature of the environment being mapped, then the uniform prior is the best choice. However under weak assumptions, namely that the environment is bounded in size and the robot is likely to revisit areas, two alternate distributions have been previously explored.

The Bayesian information criterion [90] was used to define a prior in Tully et al. [17]. This prior favors map hypotheses with fewer areas and is shown in (6.13) where $|M^n|$ is the number of areas in a map and K is the total number of topological events. The Bayesian information criterion prefers simpler explanations for the presented data. An alternate way to view this prior is that the size of the environment has an upper bound, and over time new areas are less likely to be visited.

$$p(M^n, X^n|U^K) \propto \exp(|M^n| \log K) \tag{6.13}$$

6.4 Lazy Evaluation of Map Hypotheses

The goal of our lazy evaluation algorithm is to find the most probable topological map hypothesis for the environment that is consistent with all observations made by the robot

Z^K and all actions taken by the robot U^K. Our approach is outlined in Algorithm 6.2, and a detailed description follows.

Algorithm 6.2 LazyEvaluation(H_k, Z^{k+1}, U^{k+1})

Input: H_k : A valid hypothesis tree
Input: Z^{k+1} : The sequence of topological events that have occurred.
Input: U^{k+1} : The sequence of actions taken at the corresponding place
Output: H_{k+1} : A new hypothesis tree with at least one leaf consistent with all $k+1$ observations.

```
1:  L_k ← Leaves(H_k)
2:  Q ← PriorityQueue
3:  for all h^n ∈ L_k do
4:     Push(Q, h^n)
5:  end for
6:  h^max ← ∅
7:  while HasNext(Q) and Posterior(h^max) < Posterior(Top(Q)) do
8:     h^n = Pop(Q)
9:     d = Depth(h^n)
10:    X^n_{d+1} ← UpdateRobotLocation(h^n, u_{d+1})
11:    if IsConsistentHypothesis(h^n, X^n_{d+1}, z_{d+1}) then
12:       C ← Expand(h^n, X^n_{d+1}, z_{d+1})
13:       for all c ∈ C do
14:          Insert(H_{k+1}, c)
15:          if Depth(c) < k + 1 then // Child is not complete, so put it on the queue.
16:             Push(Q, c)
17:          else if Posterior(c) > Posterior(h^max) then // Is child new best hypothesis?
18:             h^max ← c
19:          end if
20:       end for
21:    else
22:       PruneHypothesis(h^n)
23:    end if
24: end while
25: return H_{k+1}
```

When topological event z_{k+1} occurs, Algorithm 6.2 performs an update to the tree of maps to find a new map hypothesis consistent with all topological events Z^{k+1}. To generate new map hypotheses for the tree H_{k+1}, we begin by considering the set of leaf hypotheses, $L_k \subseteq H_k$. A leaf hypothesis $h^n_d \in L_k$ is a map hypothesis without child hypotheses in the tree of maps that is consistent with all topological events up to its depth in the tree d where $d < k+1$.

As opposed to previous approaches that exhaustively expand every hypothesis in the

Algorithm 6.3 Expand(h^n, x^n_{d+1}, z_{d+1})

Input: h^n : A valid topological map hypothesis up to event k.
Input: X^n_{d+1} : The robot's new location after taking action u_{d+1}.
Input: z_{d+1} : The next event in the topological event sequence.
Output: C : The set of consistent children of h^n.

$C \leftarrow \emptyset$
if $X^n_{d+1} \in M^n$ **then** // New location is a known area
  $c \leftarrow \langle M^n, X^n_{d+1}, A^n, \chi^n \rangle$
  Insert(C, c)
else // Arrived a frontier area
  $h^{new} \leftarrow$ AddArea(h^n, z_{d+1}) // Hypothesis that the robot is at an unvisited area.
  Insert(C, h^{new})
  $F \leftarrow$ Frontiers(h^n)
  // Search all areas with a frontier transition and close loops with compatible areas.
  for all $f \in F$ **do**
    if IsPossibleLocation(f, z_{d+1}) **then**
      $c \leftarrow$ CloseLoop(h^n, f, z_{d+1})
      Insert(C, c)
    end if
  end for
end if
return C

tree of maps [17, 39], we evaluate only a subset of hypotheses and consider them in order of decreasing probability. We determine the evaluation order by assigning a value e^n_k to every hypothesis in L_k. This value is an estimate of the unnormalized posterior probability of the map hypothesis h^n_d after incorporating all events up to k. For those hypotheses with $d = k$, $e^n_k = p(Z^K|h^n_d, U^K)p(h^n_d|U^K)$. For those hypotheses with $d < k$, we use a heuristic (described below) to approximate how the likelihood and prior of the map will change after incorporating events $[z_{d+1}, \ldots, z_k]$.

While the calculated posterior for these incomplete leaves ($d < k$) could be used when determining the evaluation order for hypotheses, this value overestimates the posterior of the hypothesis' children after incorporating future events because the likelihood of a map hypothesis monotonically decreases as new events are incorporated. As an alternative to the calculated posterior, we use an approximation of the posterior for these hypotheses.

The intuition behind the heuristics used is that the most probable maps are expanded first when searching. Thus, the change in probability of already evaluated maps is likely to be an optimistic estimate of the change in probability for maps that were initially less probable. We use a heuristic for both the possible likelihood and possible prior of a hypothesis.

The likelihood heuristic, $h(l_d)$, where d is the depth of the tree at which the heuristic

applies, uses the minimum change in measurement likelihood from a parent hypothesis to any its children. This quantity is equivalent to the maximum ratio of child likelihood to parent likelihood:

$$h(l_d) = \max\left(\frac{p(Z^d|h_d^n, U^d)}{p(Z^{d-1}|parent(h_d^n), U^{d-1})}\right) \tag{6.14}$$

The heuristic gives an estimate of how we expect the likelihood of a map hypothesis to change when incorporating the next topological event.

Our prior heuristic, $h(p_d)$, where d is the depth of the tree at which the heuristic applies, uses the maximum value of the prior among the hypotheses at the maximum depth of the tree K:

$$h(p_d) = \max(p(h_K^n|U^K)) \tag{6.15}$$

Combining (6.14) and (6.15), we calculate e_d^n for a map hypothesis:

$$e_d^n = p(Z^i|h_d^n, U^d)h(p_d)\prod_{i=d+1}^{k} h(l_i) \tag{6.16}$$

Given the formula for the priority value associated with a given leaf hypothesis, we now describe the lazy evaluation algorithm in full. At the start of an update, all leaf hypotheses L_k are placed in a max-priority queue, where the priority for a hypothesis is e_d^n. At the start of each iteration, the best hypothesis is popped off the priority queue.

Processing of this hypothesis begins by finding the new location of the robot in the map X_{d+1}^n, given the latest topological event u_{d+1} (Line 10). The topological action u_{d+1} corresponds to crossing a transition in the map. Therefore, X_{d+1}^n is simply assigned to be the area on the other side of this transition. This area can either be an existing area $a \in A^n$ or a frontier (unknown) area.

After finding the new robot location, the consistency of the map hypothesis is checked (Line 11). For a frontier area, the consistency check always succeeds, as no prior expectation on the area exits. When the robot revisits a previous area though, the consistency check confirms the following properties are consistent between the expected location X_{d+1}^n and the detected area z_{d+1}:

- The area types are the same.
- If at a place, the transition cycles are compatible (see Section 6.5)
- If on a path, the detected entry transition (from an endpoint or from a transition sequence) is the same as the expected entry transition.

For example, if the robot detects it is at a plus intersection, but expects to be at a T intersection, then that hypothesis is inconsistent.

The hypotheses that are found to be consistent are then expanded. When expanding a hypothesis h_d^n, we use topological event z_{d+1}. Each child hypothesis created during the map expansion process is added to the priority queue and might also be evaluated during the current update.

The algorithm continues until the queue is empty or a complete and consistent map hypothesis has been found that incorporates all topological events up to z_{k+1} whose posterior probability is greater than the posterior probability of the map hypothesis at the top of the queue:

$$p(Z^{k+1}|h_{k+1}^n, U^{k+1}) > p(Z^k|h_k^{top}, U^K) \tag{6.17}$$

By using this stopping condition, the algorithm evaluates only as many hypotheses as needed until no other hypothesis could be more probable. Thus, the quality of the measurements dictate how many hypotheses need to be evaluated. Furthermore, by evaluating the leaves in order of decreasing probability, we focus computation on the most likely portions of the hypothesis space.

The key step in Algorithm 6.2 is Line 12, which is described in Algorithm 6.3. When expanding a map hypothesis, the key question is: given a map hypothesis and a new event z_{k+1}, what are the possible loop closures that could have been made within the map (Section 6.5)?

6.5 Loop Closures in the H^{2}SSH

This section relies heavily on the detailed representations in the H^{2}SSH. We refer the reader to Chapter 3 for the complete specification of all terminolgy used hereafter.

Solving the topological mapping problem requires enumerating all possible loop closures (Line 10 in Algorithm 6.3). Searching for loop closures requires the robot to reason about which transitions in the map have been explored and which are still frontiers. Each transition in the map has a flag that indicates whether or not the robot has crossed it. Uncrossed transitions are considered topological frontiers and are the locations of potential loop closures.

There are five types of transitions can be performed in the H^{2}SSH, corresponding to the following possible actions:

(1) $\psi_{end} \rightarrow P$: Entering the place at the end of a path segment.
(2) $\psi_{seq} \rightarrow P^D$: Entering a destination along a path segment.
(3) $P \rightarrow P^D$: Entering a destination from a place.
(4) $P \rightarrow \psi_{end}$: Entering a path segment from one end.
(5) $P^D \rightarrow \psi_{seq}$: Entering a path segment from a destination along a side.

When the robot performs one of these actions and crosses a frontier transition, a loop closure has potentially occurred – the alternative is the robot has entered a previously unvisited area. These transitions can be divided into three possibilities: (a) entering a transition cycle (1, 2, 3), (b) entering a path segment from an endpoint (4), (c) and entering a transition sequence (5).

Transition Cycle: A loop closure can be made between two places P and P' if the transition cycles are equivalent when the entry transition $R_{P'}$ in P' corresponds to a frontier transition R_P in P:

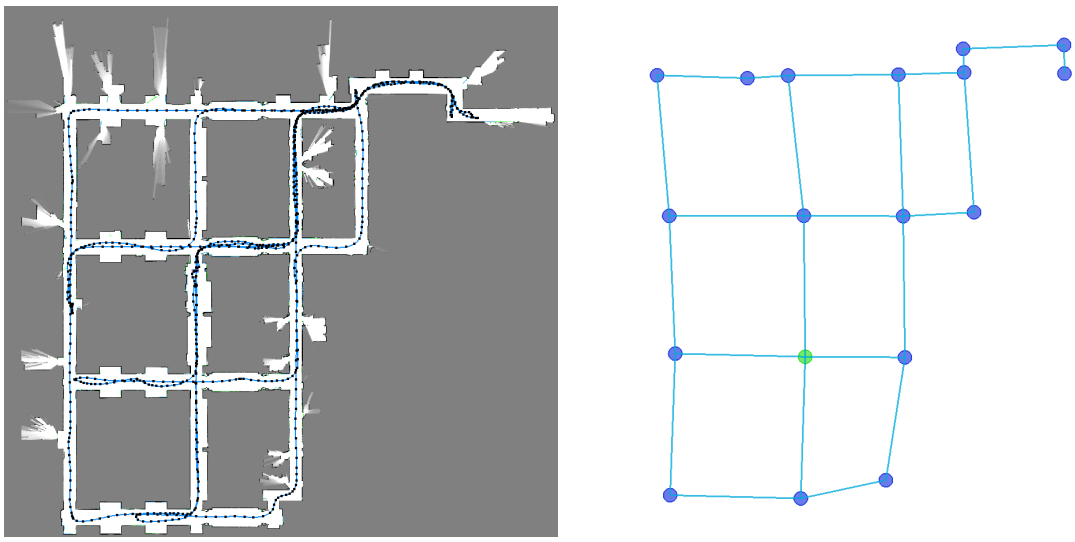
$$\begin{aligned}\text{PossibleLoop}(P, R_P, P', R_{P'}) = \text{Frontier}(R_P) \\ \wedge \text{Type}(R_P) = \text{Type}(R_{P'}) \\ \wedge (C_P = C_{P'}) \mid R_P \equiv R_{P'}\end{aligned}$$

Loop closures at places are the most common form of loop closure, as they include loop closures that occur at decision points. For some transition cycles (most commonly plus intersections), multiple possible loop closures can exist, whenever multiple rotations of the transition cycle yield satisfy the condition above. For other common intersections (L and T), only a single loop closure can possibly exist.

Path Endpoint: A loop closure can be made between the endpoint of two path segments Ψ and Ψ' if one of the end transitions R_{Ψ}^{end} in Ψ is a frontier and leads to the same type of place as Ψ':

$$\begin{aligned}\text{PossibleLoop}(\Psi, R_{\Psi}^{end}, \Psi', R_{\Psi'}^{end}) = \text{Frontier}(R_{\Psi}^{end}) \\ \wedge \text{Type}(R_{\Psi}^{end}) = \text{Type}(R_{\Psi'}^{end})\end{aligned}$$

Loop closures through path endpoints are rare events. In order for a frontier gateway to exist at the end of a path segment, the robot must drive far enough down a path segment to see the end transition, then turn around and exit through the entry transition. Generally, path segments are traversed in their entirely as the robot moves around the environment.



(a) (b)

Figure 6.4: The trajectory driven by the robot (a) while autonomously exploring the third floor of the EECS building at the University of Michigan and the constructed topological map (b). The robot drove autonomously between goals randomly selected from a pre-defined set until the full map was explored.

Transition Sequence: A loop closure can be made between two path segments Ψ and Ψ' along a transition sequence if a frontier transition R_{Ψ}^{seq} exists in the transition sequence $\mathcal{T}_{\Psi}$ for Ψ:

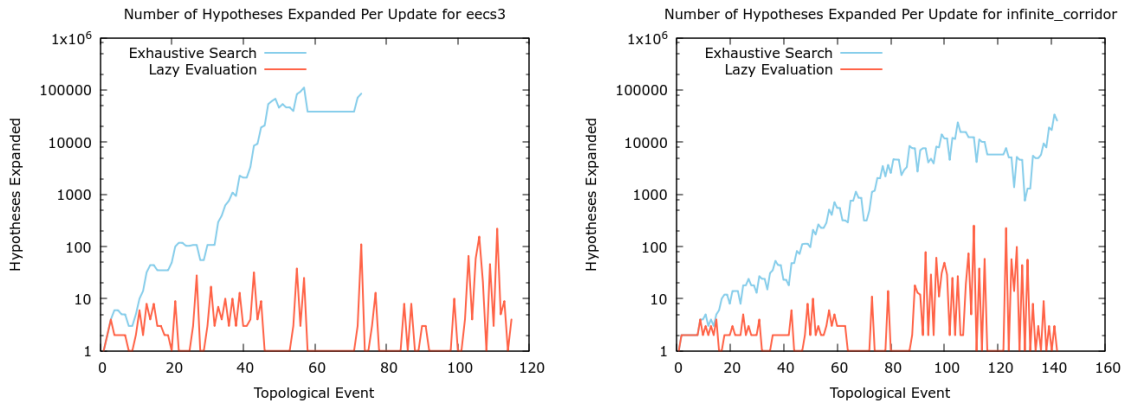
$$\text{PossibleLoop}(\Psi, R_{\Psi}^{seq}, \Psi', R_{\Psi'}^{seq}) = \text{Frontier}(R_{\Psi}^{seq})$$

For a given transition sequence $\mathcal{T}_{\Psi}$, a possible loop closure exists for every frontier transition. However, these transitions are rare. In most cases, a destination along a path segment has a single transition, so exiting the destination does not generate any loop closures. Possible loop closures occur only if the destination has multiple transitions, and the robot enters from one transition and exits through another.

6.6 Results

We evaluated our lazy evaluation algorithm in on datasets of environments at the University of Michigan as well as the standard Infinite Corridor dataset. To collect our datasets, we used our robotic wheelchair, Vulcan, equipped with two laser rangefinders, an inertial measurement unit, and wheel encoders. Figure 6.4 shows the eecs3 environment and its corresponding topological map found by our lazy evaluation algorithm.

The goal of our lazy evaluation algorithm is to enable topological mapping of large-



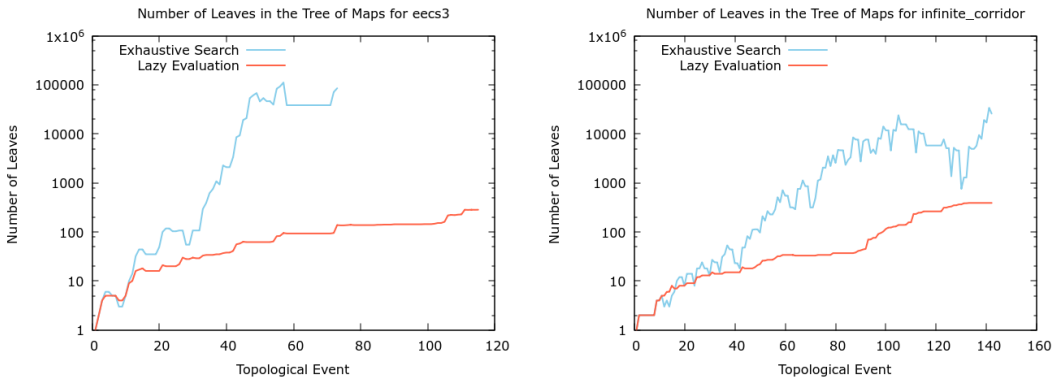
(a) eecs3 (b) infinite_corridor

Figure 6.5: These plots show the number of hypotheses expanded on each iteration of our lazy evaluation algorithm compared with the naive exhaustive algorithm. Note the log-scale on the y-axis. For both eecs3 and infinite_corridor, lazy evaluation avoids the exponential growth in the number of hypotheses being expanded. Whereas the exhaustive evaluation shows a continual increase in the hypotheses expanded, lazy evaluation often does little work, only spiking when potential loop closures are encountered by the most probable map. Flat portions in the eecs3 for both lazy evaluation and exhaustive search correspond to the robot backtracking along its trajectory. The exhaustive search for eecs3 stops after 74 events because the system exhausted its 16GB of available RAM.

scale environments while avoiding the exponential explosion of possible map hypotheses. We evaluate the success of our approach by considering to quantities related to the search. First, we look at the number of map hypotheses expanded on each iteration of the algorithm. The number of expanded hypotheses is the amount of work performed on a single iteration of Algorithm 6.2. Second, we consider the total number of leaves in the tree of maps after completing an iteration of Algorithm 6.2. The number of leaves represents the size of the search space being considered during lazy evaluation. As can be seen in Figure 6.5 and Figure 6.6, our lazy evaluation algorithm avoids the exponential growth in the number of map hypotheses expanded when a new event occurs and exponential growth the number of hypotheses in the portion of the hypothesis space being searched (leaf hypotheses).

The difference in the number of hypotheses expanded (Figure 6.5) in eecs3 versus infinite_corridor shows how our algorithm responds to increased motion uncertainty. The Infinite Corridor dataset contains significantly longer loops and has worse odometry and laser scans than is available on Vulcan. As a result, the estimates in $Lambda^K$ have much more uncertainty. As a result, the number of leaves grows more steadily than in the EECS environment, which makes larger, less frequent jumps, but is more confident in its map hypotheses.

The growth in the number of map hypotheses versus the number of events matches the behavior of other probabilistic approaches [8, 17], where a small number of map hypotheses



(a) eecs3 (b) infinite_corridor

Figure 6.6: These plots show the number of leaves in the tree of maps after each iteration of our lazy evaluation algorithm compared with the naive exhaustive algorithm. Note the log-scale on the y-axis. For both eecs3 and infinite_corridor, the search space for lazy evaluation is typically orders of magnitude smaller than the exhaustive search. When lazy evaluation finds the correct map, the tree stops growing entirely until new frontiers begin to be explored again. In contrast, the exhaustive search tree continues growing because low-probability hypotheses continue to be evaluated. The exhaustive search for eecs3 stops after 74 events because the system exhausted its 16GB of available RAM.

eventually dominate the posterior distribution. While the overall behavior of each algorithm is similar, the performance differs in significant ways.

Our topological place representation uses the decision structure extracted from a metric representation of the place. This representation describes exactly the number of paths incident to a place, which limits the space of possible map hypotheses because a place with all paths connected to places will no longer be a candidate for loop closures. Additionally, the configuration of incident paths can identify inconsistent map hypotheses by assigning a probability of zero to hypotheses with different transition cycles 6.3.1. Only these hypotheses are pruned from the tree of maps.

Tully et al. [17] use a similar place representation to our own that is based on junction points in the Voronoi diagram. They also perform their search on a tree of maps annotated with posterior probabilities. However, their algorithm actively prunes low-probability hypotheses in addition to inconsistent hypotheses, which means the correct hypotheses could be pruned from the tree, making the correct map unrecoverable. By always expanding all hypotheses, the number of hypotheses can grow very large during times of ambiguity. As shown in their results, the maximum number of hypotheses grows into the thousands, whereas our approach never considers more than 250 hypotheses in our experiments.

The place representation used in [8] treats each place as a simple landmark with no information about the number of incident paths. As a result, each place can match any other

place. When combined with their data-driven proposal, which requires a per-particle optimization across all possible matching landmarks, this simple place representation causes significant slowdown. For the Infinite Corridor dataset, they required an average of $13.2s$ per observed landmark, which totals $805s$ for their 61 observed landmarks. Our lazy evaluation approach required $34.8s$ to process the entire map. While the difference in processing power makes a direct comparison difficult, our algorithm runs over *20 times* faster, which cannot be accounted for solely by increased computing power.

6.7 Discussion

We have a demonstrated a probabilistic topological mapping algorithm that effectively reduces the search space of potential map hypotheses to allow topological mapping in real-time while never eliminating the correct map from the search space. Our algorithm uses a lazy evaluation approach to only expand the most probable map hypotheses based on current sensor data. By focusing our search on only the most probable hypotheses, we have shown the size of the search space grows significantly slower than a naive exhaustive search, making topological mapping of large-scale environments feasible without enforcing planarity constraints [89], active exploration [16, 22, 15, 88], or formal reasoning [91, 11].

Like previous approaches [17, 8], lazy evaluation focuses on a small set of highly probable map hypotheses to make the topological mapping problem tractable. However, both of these approaches prune the hypothesis space by either sampling only a small number of map hypotheses [8] or explicitly discarding low-probability hypotheses [17]. As a result, the correct map hypothesis can be discarded from the search space and never be retrieved.

Tully et al. [9] address the problem of discarding by adding a garbage-collector hypothesis to their set of hypotheses that maintains an estimate of the total probability of discarded hypotheses. While they are able to detect kidnapped robot situations and trigger a global localization process, they provide no explanation for how the correct map hypothesis could be retrieved, given the same sensor inputs.

In our approach, consistent map hypotheses are never discarded from the search space. The lazy evaluation framework simply ignores low probability maps and allows different maps to be maintained at different depths of the tree of maps, thus we can guarantee the correct map is always in the search space. Of course, extremely poor measurements can make the correct map very improbable, but this problem is general to all mapping algorithms.

Maintaining map hypotheses at different depths of the tree requires additional storage, since all place detection events must be maintained. This required storage space is negligible though and consists of only an occupancy grid representation of the area, a λ-value,

and the symbolic description of the area (transition sequences or transition cycles).

Finally, we note interesting behavior in the exhaustive search results (Figure 6.6). While the hypothesis space for naive landmark maps grows hyper-exponentially as a function of the number of landmarks, a structural topological map, like the H^2SSH, exhibits much more complex behavior. While a general exponential increase is seen in both eecs3 and infinite_corridor, there are also periods where the number of hypotheses decreases significantly, particularly in the Infinite Corridor, where the number of consistent hypotheses drops briefly below 1000 after 130 events. We believe the cause of this drop to be the prevalence of T intersections in this environment. T intersections place more constraints on the possible topologies in the environment, as opposed to plus intersections, which provide a wealth of loop closures, as can be seen in the EECS environment where over 85,000 hypotheses exist after just 73 events.

CHAPTER 7

Socially-Aware Navigation Using Topological Maps and Social Norm Learning

In this chapter, we present a method for learning and following social norms using an H^2SSH representation of the environment. We show how the H^2SSH simplifies pedestrian intention estimation by providing a small set of possible action classes. Using the estimated intentions, we create a qualitative representation of the current navigation situation, which provides two distinct benefits. First, social norms for large-scale navigation through an environment can be easily learned by simply counting the observed behavior instances in each situation. Second, the behaviors we learn are conditioned only on the generic topological action taken, i.e. moving down a corridor or from one place to another. Thus, learned norms generalize to any environment with the same topological abstractions, allowing the robot to behave appropriately in previously unvisited environments. We have implemented the social norm learning on our intelligent robotic wheelchair, Vulcan [92], and have integrated preferences based on social norms into a stochastic motion planner for dynamic environments [93], allowing our wheelchair to move naturally through everyday environments.

7.1 Social Norms for Navigation

Robot navigation in real-world environments, typified by college campuses, shopping malls, hospital complexes, and office buildings, requires interaction with other moving agents, most often pedestrians. To navigate safely and naturally in such environments, a robot must reason about the behaviors of these pedestrians. In particular, the robot needs to estimate the current and future positions of pedestrians when planning and executing its own trajectory through the world to ensure safe motion.

While pedestrians have few physical constraints on their motion, pedestrian behavior is often dictated by social norms that prescribe *preferred* behaviors for an agent to perform

in a particular social situation [50]. For a robot navigating in an indoor environment, social norms define preferences for how the robot should move through its environment and interact with the agents it encounters, e.g. what side of a hallway to travel on, how far to follow behind someone, or what an appropriate passing distance is [51, 49, 55]. In contrast to standard approaches that optimize the safety and efficiency of the robot's motion, a more socially aware robot additionally considers adherence to social norms in deciding the best trajectory to follow.

By understanding social norms, the robot can better understand how pedestrians are likely to move through the environment, which enables more natural interactions. Over time, a robot can refine its own behavior to more closely align with how a person would be expected to act when faced with a similar situation.

In this paper, we focus on social norms for navigation in structured environments, where the robot has both a metric and topological map of its environment. For a robot moving through such environments, social norms prescribe behaviors for different situations the robot encounters. For example, while traveling along a corridor, the robot should generally stay to the right and pass on the left, or when turning at an intersection, the robot should not cut the corners when making left turns, which increases the chance of a collision with a pedestrian following the norm of moving on the right.

However, the space of possible situations the robot must react to is vast, and situations with different social norms can arise for myriad reasons. For example, the robot might be in a country that moves along the opposite side of corridors than the country in which it was programmed. Consequently, a robot must learn the social norms for its environment by observing other agents' behaviors.

Learning social norms for different situations requires an understanding of the following concepts: the robot must understand an agent's intention, the robot must be able to describe the situation the agent faces, and the robot must be able to describe the behavior taken by the agent. We present a solution to these inter-related problems by creating a qualitative representation of social interactions using metric and topological representations of an agent's intentions and behavior.

Our method describes a situation using the topological actions being taken by all agents in the environment, which creates a qualitative description of the situation around the robot. This qualitative description can be easily estimated using a laser- or camera-based pedestrian tracker. In a metric map, the action taken by a pedestrian can be described by its position and velocity. Using these state estimates, we perform Bayesian inference to estimate the topological action being taken by an agent. Along a corridor, these actions correspond to moving from one end to the other. At a place or intersection, an agent is

selecting amongst a small set of adjacent places to move to next. Having created this description, social norms can then be learned by accumulating evidence about the behavior of other agents in response to a situation.

We have implemented a socially-aware navigation algorithm by integrating knowledge of social norms and goals into our existing motion planning algorithm, MPEPC, that plans locally smooth and safe trajectories within a five-second horizon, balancing progress towards its goal against the probability of collisions with static or dynamic obstacles in the environment [20, 93]. We introduce the social norms as an additional cost layer for the robot to consider when computing the navigation function to reach its goal, making the robot prefer to follow them whenever it is safe to do so.

The remainder of this chapter proceeds as follows. Section 7.2 defines our navigation situation abstraction that is the basis for our approach to learning social norms. Section 7.3 describes our approach for estimating the probability distribution across all topological actions in an area. Section 7.4 presents our method for learning social norms. Section 7.5 describes our socially-aware motion planning algorithm based on MPEPC. Finally, we evaluate the full system implemented on a robotic wheelchair in Section 7.6. Appendix B contains a detailed description of the object tracker used by the methods presented in this chapter.

7.2 Situations for Topological Navigation

As previously discussed (Section 4.6), navigation through the H^{2}SSH can be described using two actions: traveling along a path or transitioning from one area to another. When traveling along a path, an agent moves towards the place at one of the ends, which we refer to as p^+ or p^-. When transitioning between areas, an agent crosses a gateway g separating them.

When executing these actions, an agent may encounter crowded intersections or empty hallways. It might need to give way to an oncoming pedestrian or follow a pedestrian through a corridor. We formalize these different interaction scenarios, including the possibility of an empty environment, using the concept of a *situation*.

A *situation*, $S_t = \langle M, X_t, O_t, \alpha_t \rangle$, describes the robot's navigation in terms of the environment's topological and metric map M, the robot's current pose and topological location X_t, the observed pedestrians O_t near the robot, and the current topological action being executed $\alpha_t \in \{\Psi, \Delta\}$, where Ψ is a path travel action and Δ is a transition action.

We describe each pedestrian's state $o \in O_t$ based on the set of possible actions for its topological location. For a motion along a path ($\alpha = \Psi$), a pedestrian is either moving

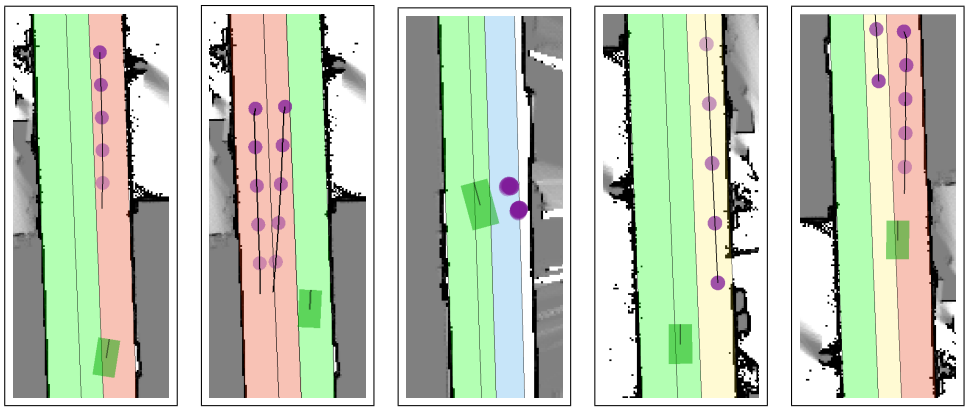


Figure 7.1: A variety of situations encountered by the robot (green rectangle). Pedestrians and their predicted trajectories are the purple circles, spaced $1s$ apart in time. The situation states are drawn as: green = empty, red = oncoming,yellow = away,blue = stationary.

towards the place at one of the ends, which we refer to as p^+ or p^-, depending on whether the pedestrian is moving in the same $(+)$ or opposite $(-)$ direction as the agent, or is stationary . When transitioning to a new area ($\alpha = \Delta$), the pedestrian is either crossing in the same (g^+) or opposite (g^-) direction as the agent.

7.3 Topological Intention Estimation

Predicting an agent's intentions presents a problem similar to predicting its future trajectory, namely there exists a vast space of possible intentions for an agent at any time. However, we can dramatically simplify the space of possible goals using a topological map. In the H^{2}SSH, performing a navigation action amounts to crossing one of the gateways in the agent's current area. Thus, intention estimation reduces to the task of estimating which gateway an agent will use to leave its current area. We show how the Bayesian Human Motion Intentionality Prediction (BHMIP) algorithm from Ferrer et al. [94] can be adapted to estimating which the current gateway goal of an agent.

Given the estimated state o_k of an agent and a set of goals $G_k = \{g_k^0, \ldots, g_k^N\}$, we estimate a probability for each possible goal $p(g_k^n | o_{0\ldots k}, M_k) : n \in \{1 \ldots N\}$. $o_{0\ldots k}$ is the set of all state estimates since the agent was detected in the area, and M_k is the topological representation of the area. For each goal $g \in G$, we estimate the probability $p(g_k | o_{0\ldots k}, M_k)$ by making the Markov assumption and using a recursive Bayes filter:

$$p(g_k | o_{0\ldots k}, M_k) \propto \frac{p(o_k | g_k, M_k) p(g_k | o_{k-1}, M_k)}{\sum_{n=0}^{N} p(o_k | g_k^n, M_k) p(g_k^n | o_{k-1}, M_k)} \tag{7.1}$$

We assume an uniform prior across all goals $p(g_k|o_0, M_k) = 1/(N+1)$ for the initial state. For a robot operating in a single environment, extended observation of pedestrian behaviors can be used to estimate a non-uniform prior based on the frequency with which gateways are traversed. We save such estimation for future work. G_k describes the complete set of topological actions in the environment, allowing for a simple sum of probabilities to be used for normalization.

When computing the likelihood of a particular goal, the primary consideration is the estimated heading of the agent because the heading is the best estimate of where the agent will be in the future. In this paper, we estimate heading from motion because our laser-based system can't perceive details of body positioning. Additional sensors, particularly cameras, could provide additional information to further improve the heading estimate.

Using motion estimates (v_x, v_y) from a tracking algorithm, we can estimate a probability distribution for the agent's heading:

$$p(\Theta) = \mathcal{N}(\theta; \mu_\theta, \sigma_\theta^2) \tag{7.2}$$

$$\mu_\theta = atan2(v_y, v_x) \tag{7.3}$$

$$\sigma_\theta^2 = J_\theta^T \begin{pmatrix} \sigma_{v_x}^2 \\ \sigma_{v_y}^2 \end{pmatrix} J_\theta \tag{7.4}$$

$$J_\theta = \left(\tfrac{-v_y}{v_x^2+v_y^2}, \tfrac{v_x}{v_x^2+v_y^2} \right)^T \tag{7.5}$$

To estimate the likelihood of the agent having a particular goal using this model, we compute the range of headings that would result in the robot crossing the gateway associated with a goal, as shown in Figure 7.2.

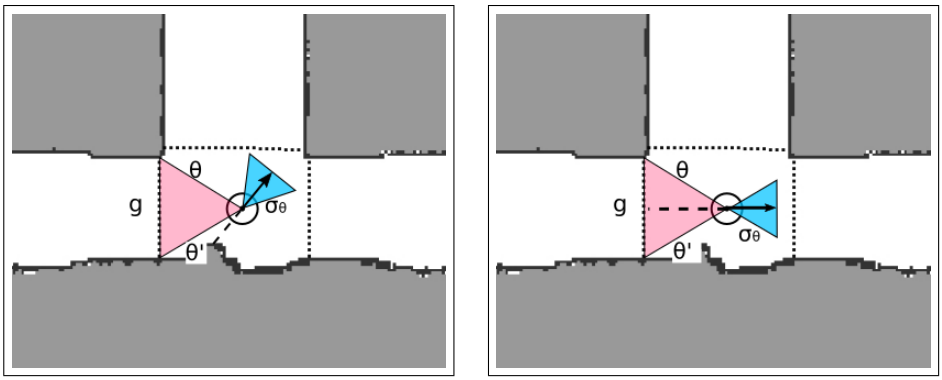
Each gateway can be described as a line segment with two endpoints (x_g, y_g) and (x'_g, y'_g). Using these endpoints, we can find the range of angles subtended by the gateway in the agent's frame, whose origin is (x_k, y_k, μ_θ):

$$\theta_g = \text{atan2}(y_g - y_k, x_g - x_k) - \mu_\theta \tag{7.6}$$

$$\theta'_g = \text{atan2}(y'_g - y_k, x'_g - x_k) - \mu_\theta \tag{7.7}$$

where both θ_g and θ'_g are wrapped to the range $[-\pi, \pi]$. Given the subtended angle $[\theta_g, \theta'_g]$, where $|\theta_g| < |\theta'_g|$:

$$\begin{aligned} p(x_k|g, M_k) &= P(\theta'_g < \Theta < \theta_g) \\ &= P(\Theta < \theta_g) - P(\Theta < \theta'_g) \end{aligned} \tag{7.8}$$



(a) (b)

Figure 7.2: (a) shows the angle subtended by gateway g relative to the the agent, drawn as a circle with the arrow indicating its heading. The blue cone on the right shows the estimated heading μ_θ and along with a 1σ bound. In this case, θ and θ' are on the same side of the dashed wraparound line. The other possible scenario is shown in (b), where θ and θ' are on opposite sides of the wraparound at π.

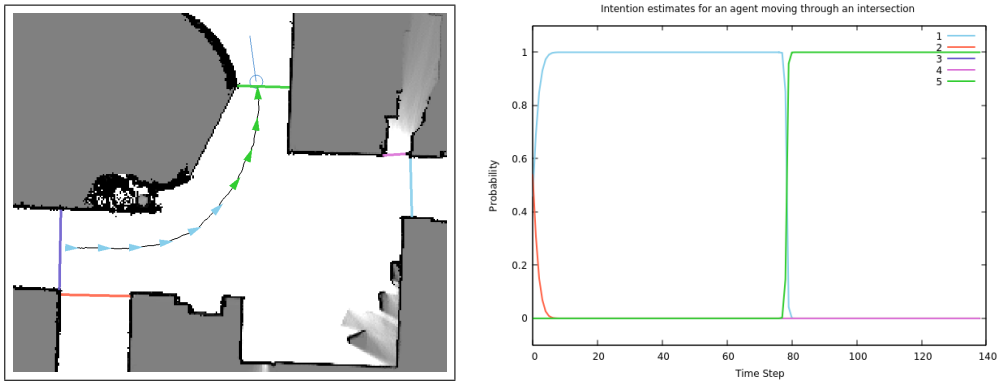
When computing (7.2), we assume the agent will turn the smallest amount needed to reach its goal. Thus, the magnitude of the rotation falls in the range $[-\pi, \pi]$, where negative angles are right turns and positive angles are left turns. Figure 7.2a shows the case when the range of angles subtended by the gateway is only positive or negative . However, in the case that $[\theta_g, \theta'_g]$ spans across the wraparound angle at π, the agent can turn right or left in order to reach g (Figure 7.2b). For this case, (7.8) becomes:

$$\begin{aligned} p(x_k|g, M_k) &= P(\pi < \Theta < \theta_g) + P(\pi < \Theta < \theta'_g) \\ &= P(\Theta < \theta_g) + P(\Theta < \theta'_g) - 2P(\Theta < \pi) \end{aligned} \tag{7.9}$$

7.4 Learning Navigation Social Norms

A social norm describes the expected behavior of an agent in the environment when faced with a social situation. However, social norms are loosely defined, so a variety of people following the same social norm, like moving along the right side of a corridor, are likely to exhibit a range of behaviors. To account for the variation in behaviors, we represent a social norm as a probability distribution over possible behaviors, which can be learned by observing how pedestrians behave when responding to various situations.

Formally, we learn probability distributions across possible robot poses $\bar{x}_t$, given a



(a) (b)

Figure 7.3: The evolution of goal probabilities using our intention estimation technique. (a) shows the path taken by an agent. The colored triangles match the most probable goal at that location. (b) is the plot of the goal probabilities for each goal as the agent moves through the intersection.

navigation situation S_t:

$$p(\bar{x}_t|S_t) = p(\bar{x}_t|M, X_t, O_t, \alpha_t) \tag{7.10}$$

$$\approx p(\bar{x}_t|O_t, \alpha_t) \tag{7.11}$$

We can approximate the complete distribution in (7.10), which depends on the robot's location X_t in a particular map M, using a more general situation description that depends only on agent states O_t and the current action α_t (7.11). By ignoring conditioning on the robot's specific state within a map, we learn a distribution that describes social norms for topological actions in any environment, even previously unexplored environments. However, if the robot continually operates within a single environment, (7.11) can be used as a prior for learning the more complex distribution in (7.10), which allows the robot's behavior within known environments to be refined over time to account for local variations in how people interact.

Our approach learns two related social norms. The first norm is a preference for the robot's lateral position while traveling longitudinally along a path. The other norm describes where the robot should cross a gateway when transitioning from one area to another. Therefore, $\bar{x}_t$ in (7.10) describes either the robot's position along the line orthogonal a path segment's axis or its position along a gateway boundary.

To ensure the learned model generalizes to new environments, we normalize $\bar{x}_t$ to the range $[0, 1]$. In this range, 0 is the location of the left wall relative to the nominal direction of motion. Therefore, the left side of a corridor or gateway relative to the center has distance in the range $[0, 0.5)$ and the right side range is $(0.5, 1]$.

We then divide $\bar{x}_t$ into N bins of equal size. By discretizing the normalized range for $\bar{x}_t$, (7.10) becomes a simple discrete distribution, whose entire state space can be easily enumerated. We estimate the parameters of this distribution by observing other agents' responses to situations in the environment, incrementing the bin that corresponds to their position, and then normalizing over the total number of observations.

7.4.1 Learning Norms for Path Segments

When navigating a path segment, $p(\bar{x}_t|O_t, \alpha_t = \Psi)$ is a distribution across the robot's lateral position along the path segment, where Ψ is the action that takes the robot from one end of the path segment to the other. We estimate $p(\bar{x}_t|O_t, \alpha_t = \Psi)$ by considering a simpler case first, where we ignore other agents in the environment, thus estimating $p(\bar{x}_t|\alpha_t = \Psi)$.

To estimate $p(\bar{x}_t|\alpha_t = \Psi)$, we observe other agents in the environment. For agents traveling through the environment, we can directly observe $\bar{x}_t$ and count the number of instances of their position being in a bin $n \in \bar{x}_t$. Dividing by the total number of observation yields the probability distribution $p(\bar{x}_t|\alpha_t = \Psi)$.

In the more general case, we must consider how agents interact with one another. While the state for an agent's action α_t is simple, interactions amongst agents can be complex. In our representation, each agent can be in one of N lateral positions with a state of $+$ or $-$, depending on their motion relative to the robot. For an environment with K objects, there are $(2N)^K$ possible states.

Rather than attempting to directly learn a distribution across this potentially huge state space, we create a simplified representation of the situation and learn a new distribution:

$$p(\bar{x}_t|O_t, \alpha_t = \Psi) \approx p(\bar{x}_t|L_t, \alpha_t = \Psi) \tag{7.12}$$

The simplified situation L_t divides the path segment laterally into L bins. Each bin has one of four states: $\{+, -, x, \emptyset\}$, which indicates if that bin is empty ($\emptyset$), is occupied by an agent moving in the direction ($\{+, -\}$), or contains a stationary agent (x). With this representation, a total of 4^L possible situations exist.

To compute L_t, each bin $l \in L$ is matched with the nearest agent occupying the lateral position of the bin in the direction the robot is heading. The agent is considered stationary if the velocity is under $0.25m/s$, which accounts for noise in the velocity estimate. Otherwise, the state is assigned to ($\{+, -\}$) based on the estimated goal of the agent, or ($\emptyset$) if no object occupies the bin.

When learning $p(\bar{x}_t|L_t, \alpha_t)$, we can use the same basic counting approach as when learning $p(\bar{x}_t|\alpha_t)$. We create a description of the situation L_t for each pedestrian. In this de-

scription, the robot is included as an agent that can occupy one of the bins in L_t. However, we must compute a different description of the situation L_t for each pedestrian because they each have a different perspective.

7.4.2 Learning Norms for Transitions

Motion through from one area to another occurs by crossing the gateway between the areas. The relevant position $\bar{x}_t$ for this action is where to cross the gateway boundary. Like the norm for path segments, we can learn this distribution by observing an agent's position whenever it crosses a gateway boundary.

As with path travel, we first learn the simplified distribution $p(\bar{x}_t|\alpha_t = \Delta)$ by ignoring other agents. We use a simple discrete distribution to represent $p(\bar{x}_t|\alpha_t = \Delta)$, which we estimate by measuring where each agent crosses a gateway in the environment and incrementing that the count of that bin, then normalizing over all measurements.

The distribution $p(\bar{x}_t|O_t, \alpha_t)$ represents more complex interactions amongst agents as they move from one area to another via a gateway. For this distribution, we consider only the subset of agents $P_t \subseteq O_t$ whose estimated action is to move across a gateway in the opposite of the robot:

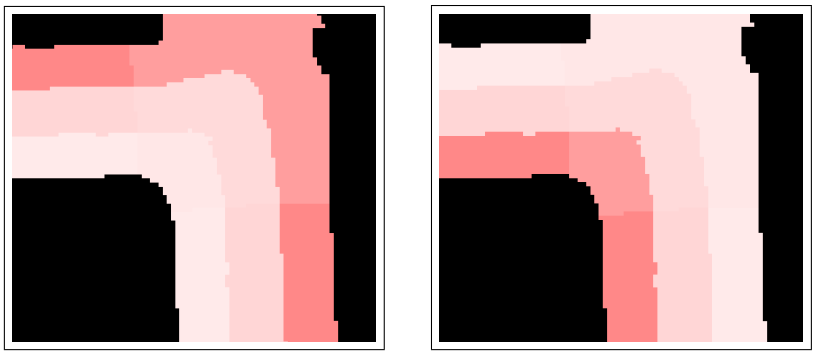
$$p(\bar{x}_t|O_t, \alpha_t = \Delta) \approx p(\bar{x}_t|P_t, \alpha_t = \Delta) \tag{7.13}$$

For example, if the robot is entering a place through some gateway g, we consider all agents who are leaving the place through that same gateway. We condition the social distribution on $|P_t|$. Thus, we estimate the distributions for each value of $|P_t|$ experienced by the robot during training using the same counting approach as with $p(\bar{x}_t|\alpha_t)$.

7.5 Socially-Aware MPEPC

MPEPC generates plans at a fixed time horizon T_H with an update rate of 10Hz. Each planning cycle computes the best robot trajectory for the next T_H seconds which requires estimating the future trajectory of all dynamic objects around the robot for the next T_H seconds as well. The best trajectory is computed by optimizing the progress towards the goal, as defined by a navigation function, weighted by the probability of safe motion.

We integrate the above goal prediction and social norm behaviors with MPEPC to create a new socially-aware MPEPC (SA-MPEPC) by defining a cost map for the learned norms from Section 7.4 to integrate them into the navigation function used for defining progress through the environment to bias the robot's decision making towards obeying social norms.



(a) (b)

Figure 7.4: Examples of cost maps used to define the social norm navigation function. The cost map in (a) is for the robot making a right turn from the horizontal corridor to the vertical corridor. Note that the cost is different when navigating through the intersection because the cost is based on the transition norm rather than the path norm. In (b), the cost map is for executing a left turn at the same interaction. The high-cost and low-cost regions are reversed since the learned norm is to stay to the right.

In the existing MPEPC implementation [93], the robot's navigation function is computed using the wavefront algorithm defined by Konolige [95]. The wavefront grows from the goal position outward using 8-way connectivity. Each cell in free space contains the distance to the goal, thereby defining a gradient that can be followed to reach the goal. Walls are marked as infinite distance.

The navigation function used by MPEPC is computed using a combination of the distance to the goal and a cost function. In many applications, including MPEPC, cost is a function of the distance to obstacles. By ensuring the navigation function is infinite in collision states for the robot, the gradient of the navigation function will always lead the robot on a collision-free path to the goal.

We integrate the learned social norms into MPEPC by introducing an additive *social cost* into the cost function for computing the navigation function for each free cell in the occupancy grid:

$$C_{cell} = C_{obst} + C_{social} \tag{7.14}$$

The cost associated with being near an obstacle C_{obst} is an exponential function of the distance to the obstacle:

$$C_{obst} = \begin{cases} 0 & \text{if } d_{obst} \geq D_{max} \\ \alpha \left(\frac{D_{max} - d_{obst}}{D_{max}} \right)^{\beta} & \text{if } d_{obst} < D_{max} \end{cases} \tag{7.15}$$

Both social norms in Section 7.4 are probability distributions over the robot's lateral

position when moving along a path segment or across a gateway, where the robot should prefer moving through high-probability regions. The cost of being in a particular position is proportional to the learned probability of not being in a particular position:

$$C_{social} = \begin{cases} k_3/(1 - p(d_{norm}|P_t, \alpha_t)) & \text{if } \alpha_t = G \\ k_3/(1 - p(d_{norm}|L_t, \alpha_t)) & \text{if } \alpha_t = \Psi \end{cases} \tag{7.16}$$

where σ is an adjustable weight.

In our implementation, we use the Voronoi skeleton to compute d_{norm}. There are typically multiple branches of the Voronoi skeleton. We select the branches of interest by finding the shortest path along the skeleton between the entry and exit gateways. Only the skeleton cells along this shortest path are used for computing d_{norm}.

The normalized distance relative to the left wall for a cell in the map is $d_{norm} = d_{obst}/(2d_{skel})$ when the skeleton is to the right of the cell. When the skeleton is to the left of the cell, $d_{norm} = 1 - (d_{obst}/2d_{skel})$. Here d_{obst} is the distance to the nearest obstacle and d_{skel} as the distance of the nearest Voronoi skeleton cell to an obstacle.

Examples of the cost maps generated by (7.14) are shown in Figure 7.4. Using such a cost map, a $2D$ navigation function can be computed using the wavefront algorithm. When computing the wavefront, we initialize the goal from which the wavefront emanates to be the entire gateway boundary, rather than a specific point. Doing so ensures that the gradient of the navigation function will lead the robot across the gateway boundary, regardless of where it reaches it, as opposed to forcing it across at a single point, which isn't necessary for topological navigation.

7.6 Experimental Methods and Results

The goal of our socially-aware navigation algorithm is to enable the robot to learn how to behave in different social navigation situations, thereby improving its interactions with pedestrians to allow more socially acceptable and safer motion through the environment.

To test the effectiveness of our algorithm, we have implemented the socially-aware MPEPC algorithm on our robotic wheelchair, Vulcan [92]. Vulcan is equipped with two Hokuyo URG-30LX lasers, an IMU, and wheel encoders. The software runs on a standard laptop with an Intel i7-4800MQ processor and 8GB of memory.

We learned the distributions in (7.12) and (7.13) using training data collected by Vulcan during autonomous explorations of campus buildings at the University of Michigan. While exploring, Vulcan was controlled using the MPEPC algorithm described in [93].

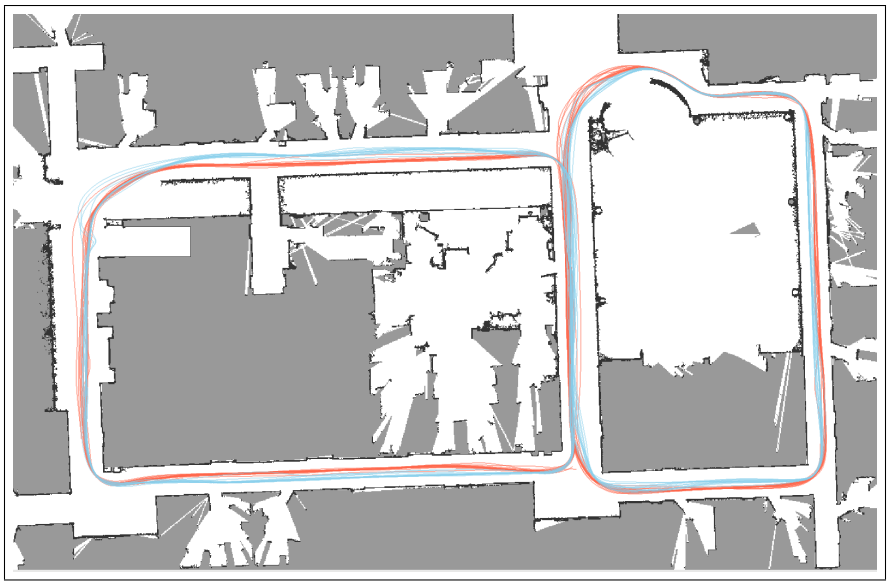


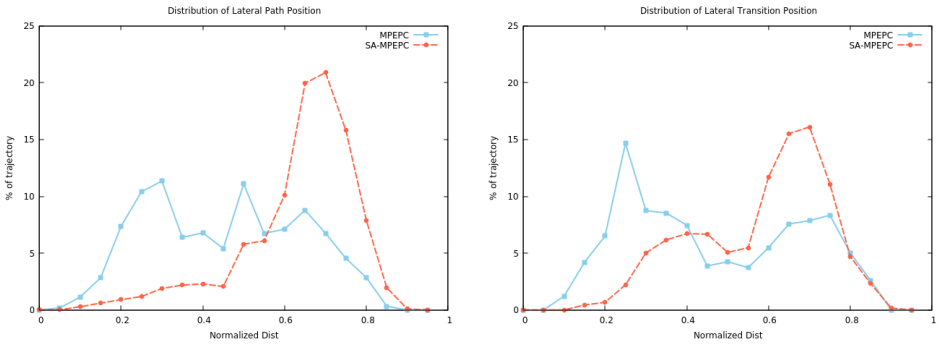
Figure 7.5: The test environment used for evaluation. The trajectories generated by SA-MPEPC are orange. The MPEPC trajectories are blue. For all trajectories, the robot was moving clockwise around the left loop and counter-clockwise around the right loop. During navigation, the robot was commanded to drive to the center of each intersection. Once within $10m$, it would switch to the next target. The right-bias of SA-MPEPC is clear along the hallways. The preference for taking wider left turns and tighter right turns can be seen in the top left and top center intersections. This figure is best viewed in color and magnified.

To evaluate the effectiveness of our new approach, we compare the performance of SA-MPEPC against the previous MPEPC algorithm. For this evaluation, we performed approximately 90 minutes of autonomous circuits with each algorithm around the figure-8 loop shown in Figure 7.5. These circuits were performed at varying times of day, so the robot would encounter a more varied set of social situations and pedestrians. During autonomous navigation, the robot's pose and velocity were estimated at 50Hz, and the position and velocity of pedestrians were estimated at 20Hz using only the onboard laser sensors.

No volunteers were used as pedestrians, so all robot-human interactions were with people carrying out their daily activities unaware of the experiment being performed. To ensure the safety of others, Vulcan was operated during all experiments by one of the authors who would manually intervene if unsafe conditions were detected.

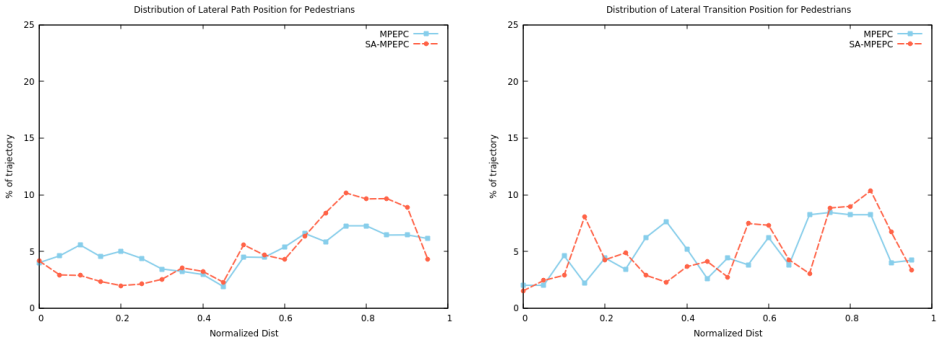
7.6.1 Lateral Position During Navigation

In our approach, preferences for behavior influenced by social norms are learned as probability distributions over the agent's lateral position. Lateral position represented using the



(a) (b)

Figure 7.6: The distributions over the normalized lateral position of the robot show a clear shift to the right for both paths (a) and transitions (b).



(a) (b)

Figure 7.7: The observed agent positions also see a shift to the right, though the effect is less dramatic.

normalized distance from the left wall, with a distance of 0 being the agent touching the left wall and a distance of 1 touching the right wall. Since the test was performed in the US, the robot successfully learning and following social norms will be demonstrated by a significant increase (closer to 1) in the normalized lateral position of the robot.

The results of our experiment demonstrate a clear rightward shift in the robot's position while traveling along a path and across transitions, as can be seen in Figure 7.6. We compared Gaussian distributions computed from the experimental data (Table 7.1) and found a significant difference between MPEPC and SA-MPEPC ($p < .001$, $t = 349$) supporting the hypothesis that a robot controlled by SA-MPEPC travels along paths closer to the right wall. Similarly, comparing Gaussian distributions for the lateral position when transitioning between areas, we found a significant difference between SA-MPEPC and MPEPC ($p < .001$, $t = 41.9$).

In our experiment, we also explored how the robot's adherence to social norms affects

Table 7.1: Distribution of normalized distance for the robot and observed agents.

	SA-MPEPC	MPEPC	p-value
Robot Path	0.66 ± 0.02	0.48 ± 0.03	$< .001$
Robot Transition	0.61 ± 0.03	0.49 ± 0.05	$< .001$
Pedestrian Path	0.62 ± 0.08	0.55 ± 0.09	$< .001$
Pedestrian Transition	0.58 ± 0.08	0.57 ± 0.07	$< .3$

the behavior of other agents in the environment.

We hypothesize that the robot's more normative behavior improves the adherence to norms of agents the robot interacts with. Table 7.1 shows a significant rightward shift in the lateral position of observed agents traveling along paths ($p < .001$, $t = 19.35$). The shift in the mean of 0.07 corresponds to $0.15 - 0.2m$ in the test environment, depending on the corridor. We do not, however, find a significant difference in the lateral position of transitions between the two approaches ($p < .3$, $t = 0.6745$).

7.6.2 Oncoming Pedestrian Avoidance Behavior

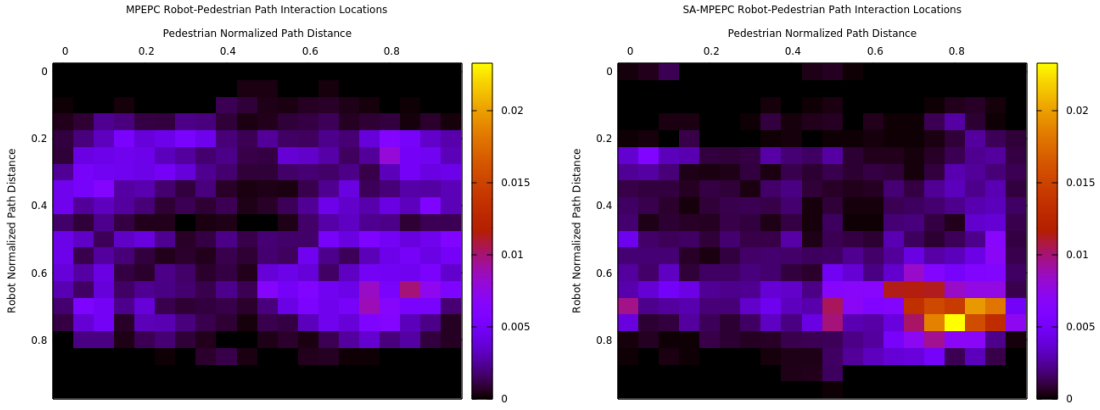
In addition to lateral positioning while navigating, we explored the behavior of SA-MPEPC when faced with another common scenario: passing an oncoming pedestrian moving the opposite direction along a corridor. During our experiment, we performed 42 oncoming passes with MPEPC and 61 oncoming passes with SA-MPEPC.

To assess the safety of the passing behavior, we looked at the passing distance between the robot and the oncoming pedestrians. We found no significant difference ($p < .4$, $t = 0.372$) between the average passing distance of $0.52m$ for SA-MPEPC versus $0.50m$ for MPEPC. This behavior is expected because MPEPC does not rely on the navigation function to ensure the safety of the vehicle because progress towards the goal is weighted the probability of collision with pedestrians. Therefore, safe distances are determined by the uncertainty of the robot's perception of the environment, along with encoded preferences on passing distances, which do not change between SA-MPEPC and MPEPC.

While the passing distance is similar, the qualitative behavior (pass on the left or the right) of SA-MPEPC does show improved conformance to passing norms. In the US, the expected behavior is for each agent to stay to the right, so an oncoming pedestrian will be to the left when passing. We found a significant increase ($p < .001$, $z = 3.421$) in the

Table 7.2: Comparison of oncoming passing behaviors.

	SA-MPEPC	MPEPC	p-value
Count	61	42	
Distance (m)	0.52 ± 0.22	0.50 ± 0.24	$< .4$
% Left	88.5	59.5	$< .001$



(a) MPEPC (b) SA-MPEPC

Figure 7.8: 2D histograms encoding the number of robot-pedestrian interactions at each combination of normalized lateral positions of the pedestrian (x) and robot (y). SA-MPEPC has a dominant peak in the lower right corner, corresponding to both the pedestrian and robot being on their own right sides of the corridor. In contrast, MPEPC has a much weaker peak in the bottom right corner and more interactions with a left-side robot and right-side pedestrian (top right) or a left-side robot and left-side pedestrian (top left).

percent of time SA-MPEPC passes an oncoming pedestrian on their left.

This improvement in passing behavior comes from the preference to stay to the right. Often, the robot is already moving to the right, so an oncoming pedestrian, also moving on their right, can simply pass the robot with no deviation from the trajectory. In more complex situations, where the robot is not moving along the right side of the corridor, perhaps due to clutter or other pedestrians, the learned norm encodes a preference to move right. As a result, the gradient in the navigation function will bias the robot to move right to avoid the collision.

7.7 Discussion

A robot operating as a part of society should adhere to the cultural norms and behaviors of that society. For mobile service robots, these social norms include how one should travel along corridors and through intersections – namely to bias motion to one side to avoid interfering with people moving the opposite direction and to allow room for faster moving people to safely pass. In this chapter, we have shown how our robotic wheelchair, Vulcan, uses the hybrid topological-metric representation of the H^2SSH to learn social norms for large-scale navigation in the environment.

Our topological situation abstraction allows potentially complex navigation scenarios to be tractably represented. The robot can then learn from observation how people in the environment react in these scenarios to learn preferences for its own behavior. In particular, we have shown that the robot is able to recognize biases in the lateral position of pedestrians in the environment and incorporate those biases into its own motion.

We integrated the learned social norms into our existing MPEPC algorithm [20, 93] to create socially-aware MPEPC (SA-MPEPC). The knowledge of social norms is implemented using an additional cost map when computing the navigation function that defines the optimal route through the environment. This approach ensures that social norms for navigation are followed even when the robot is moving through an otherwise empty environment.

Our results demonstrate that SA-MPEPC generates improved normative behavior compared to the baseline MPEPC algorithm, which optimizes only for safety and efficiency. The right-ward bias observed by Vulcan is integrated into its motion preferences, resulting in Vulcan staying to the right when traveling along paths and transitioning between areas Figure 7.5. By staying to the right, Vulcan has improved interactions with oncoming pedestrians, demonstrating the appropriate passing behavior with far greater frequency.

Our results show that Vulcan's improved adherence to social norms also increases the adherence of other pedestrians to those same norms, demonstrating the mutually-reinforcing nature of these norms. During testing, we observed many instances with SA-MPEPC where neither Vulcan nor an oncoming pedestrian was forced to slow down, since their preferred trajectories in the environment were collision-free. With MPEPC though, we observed many more instances of a collision avoidance dance, where Vulcan was on a collision course with a pedestrian and both Vulcan and the pedestrian attempted to avoid the collision in the same direction.

Finally, we note the importance of learning and following social norms that extend beyond the close human-robot interaction scenarios, like passing behaviors or approach

distances, that have been the focus of research (see [48] for an extensive review). Human-robot interaction occurs in other situations, as well. For example, a robot moving through the environment will be noticed even by people outside its interaction range, and more importantly, service robots like telepresence robots and intelligent wheelchairs are directly interacting with the person using them at all times. In these scenarios, adherence to social norms will directly impact people's comfort and trust with the robot. Thus, we believe widespread deployment of any robot will need to integrate knowledge of these norms to gain the acceptance of humans in their environment.

CHAPTER 8

Discussion

8.1 Conclusion

We presented the Hierarchical Hybrid Spatial Semantic Hierarchy (H^2SSH), a hybrid topological-metric map representation, capable of providing scalable representations of both small and large structures in the world, by providing natural descriptions of a hallway lined with offices as well as a cluster of buildings on a college campus. By considering the affordances in the environment, we identify a division of space into three distinct classes: path segments afford travel between places at their ends, decision points present a choice amongst incident path segments, and destinations typically exist at the start and end of routes.

Construction of the H^2SSH map required a new approach for place classification and detection. The approach used in the HSSH by Beeson et al.[11] only detected decision points in a small, scrolling LPM. Other place classification approaches provided no means to guarantee the resulting places and path segment satisfied the constraints of the H^2SSH required for construction of the global topological map. We solved this problem by employing a novel algorithm the combined loopy belief propagation with Markov chain Monte Carlo (MCMC) sampling to find consistent and highly probable place labels.

Because we evaluated our algorithm on a much wider variety of environments (17 different maps in total), we conducted an experiment where eight human participants created ground-truth maps of six different environments. While more extensive experimentation is needed, our initial results suggest that semantic understanding of the environment is varies, sometimes significantly, from person to person. For example, one participant placed a decision point in front of every office door along a corridor, like the HSSH, whereas all others treated them as adjacent to the path segment, like the H^2SSH. These preliminary findings point towards the need for a more comprehensive and rigorous means of evaluating approaches to semantic mapping.

Using the detected places, our probabilistic topological mapping algorithm uses lazy

evaluation to perform real-time topological mapping by efficiently maintaining and expanding a probabilistic tree of maps. Unlike previous approaches [17, 8, 9], our algorithm does not prune consistent map hypotheses to avoid exponential growth. The best guarantee that other approaches can make is that the algorithm can detect if the correct map was probably discarded [9]. Because we do not prune consistent hypotheses, ours is the only approach that can guarantee the correct map is always in the search space, though sufficiently poor sensor measurements can result in the correct map not being probable enough to be found.

Our evaluation shows that, in addition to not aggressively pruning the search space, lazy evaluation also outperforms other approaches. Even when subjected to large loop closures in the Infinite Corridor dataset, our algorithm expands at most 250 hypotheses on an update, whereas Tully et al. [17] generate thousands of hypotheses in uncertain situations. The Rao-Blackwellized particle filtering approach of Ranganathan and Dellaert [8] uses a simple landmark-based representation which results in many more possible loop closures, even when sampling from a particle filter. Our lazy evaluation algorithm runs an order of magnitude faster on the well-known Infinite Corridor dataset.

After creating an H^2SSH abstraction of the environment, we use the global topological map and the local semantic labels for navigation, planning routes in the topological map and performing motion planning in the LPM. We extend our previous work, MPEPC [20, 93], by integrating knowledge of social norms learned by observing pedestrians in the robot's environment.

We abstract the dynamic interactions with pedestrians around the robot into simple discrete representation, we call *situations*. Situations allow the potentially complex interactions encountered by the robot to be described qualitatively, like person approaching on the left or two people moving away on the right. Identifying these situations is simple and learning how people respond requires only counting instances of their observed behavior.

We implemented our improved, socially-aware motion planner SA-MPEPC on our robot wheelchair, Vulcan [92] using an additional cost map that to encodes the learned responses to the situations faced during navigation. These costs are integrated into the navigation function that defines the progress metric used by MPEPC. By integrating the social awareness into the underlying navigation function, the robot's behavior exhibits following social norms, like staying to the right, even in the absence of pedestrians.

Our results demonstrate that SA-MPEPC generates behavior that improves adheres to social norms compared to the baseline MPEPC algorithm, which optimizes only for safety and efficiency. The rightward bias observed by Vulcan is integrated into its motion preferences, resulting in Vulcan staying to the right when traveling along paths and transi-

tioning between areas Figure 7.5. By staying to the right, Vulcan has improved interactions with oncoming pedestrians, demonstrating the appropriate passing behavior with far greater frequency. Additionally, our results show that Vulcan's more normative behavior also increases the adherence of other pedestrians to those same norms, demonstrating the mutually-reinforcing nature of social norms.

8.2 Future Work

8.2.1 Robust Topological Mapping

Our lazy evaluation algorithm is capable of performing topological SLAM in large environments, finding the correct map amongst the thousands of possibilities. However, our algorithm currently requires the sequence of topological events to be perfect, meaning that all topologically-significant places are detected and only topologically-significant places are detected. That is, there are no false positives and no false negatives. While we can still use our approach to build topological maps, long-term autonomy will require a more robust approach that can handle the occasional and inevitable changes and failures of the place detection algorithm.

Our probabilistic framework is well-suited to addressing this problem. First, we can incorporate a data-driven proposal [8] to lazily expand child hypotheses in the tree of maps, as opposed to our current approach which always fully evaluates all child hypotheses. Once we have reduced the number of map hypotheses generated under nominal conditions, we can introduce new hypotheses that take into consideration the possibility that either false negative or false positive place detections occurred.

The potential for many more hypotheses will also require a more advanced likelihood model for map hypotheses. We plan to integrate laser- and vision-based place models to improve the ability of our lazy evaluation to focus on only high-probability portions of the hypothesis space.

8.2.2 Region-based Hierarchical Mapping

So far, we addressed the problem of scalability in topological mapping by reducing the size of the hypothesis space by using a hierarchical representation for paths (Chapter 4) and by efficiently searching only the most probable portions of the space of possible maps (Chapter 6). This approach works well for a multi-building environments connected on a single floor. However, campus-scale environments can easily span hundreds of floors and

tens of thousands of places [96].

The H^{2}SSH can represent these massive environments by constructing topological maps in which some destinations are themselves large-scale topological maps 3.5.2. This recursive representation allows for an arbitrarily deep hierarchy of regions to be mapped. Creating a hierarchy of regions allows the algorithm to scale using the classic approach of divide-and-conquer. Rather than mapping an environment with N places, we map independent regions with $N/2$ places and combine the results. For the topological mapping problem, whose complexity grows exponentially in the worst case, even reducing the size by half can be the difference between a tractable and intractable problem. Increasing the depth and breadth of the hierarchy further improves the scalability.

We plan to integrate these hierarchical regions to map a large environment with multiple buildings and floors of buildings connected via elevators. By detecting when the robot changes floors using an elevator, we can segment the environment into multiple regions and avoid the difficult problem of detecting regions based on their appearance. Since many environments have similar elevator connections, this approach will generalize to a large class of environments.

8.2.3 Predicting Pedestrian Collision Zones

Safe motion through the environment requires knowing where obstacles are and the behavior of other agents. The social norms learned in Section 7.4 improve safety by encoding probability distributions across possible behaviors for the robot to take, given a situation S. However, these models require the robot to have complete knowledge of S, which is often not possible when the robot's field-of-view is limited by static and dynamic objects.

We can use the learned norms to infer the probability of being in a particular situation given partial knowledge due to obstacles blocking the part of the robot's field-of-view. Specifically, while moving down a corridor, the robot can infer the probability of an oncoming pedestrian being in a blind spot by marginalizing over the state of observable pedestrians. This inference allows the robot to operate more safely in real-world environments, where the robot's knowledge of the world is always incomplete. And while this inference may possible with other approaches, our state space makes the computation extremely fast and easy.

Rather than optimistically assuming no agents exist in the robot's blind spots, we can use the learned social norms and gateways in the topological map to infer where pedestrians are likely to be. We will consider two cases analogous to the two classes of situations addressed by our current method: the probability that there's an oncoming pedestrian in the

robot's blind spot while moving along a path segment, and the probability of a pedestrian appearing in the robot's local environment by crossing a gateway.

When traveling along a hallway, the robot's view can be partially obscured by other agents moving through the environment. As a result, the robot may not be able to safely pass pedestrians moving slowly in front of it. Rather than being forced to move slowly by never passing or dangerously by always passing, we can use the learned norm in (7.12) and an observation of the current location of pedestrians to predict where oncoming pedestrians may be to make a more reasoned decision.

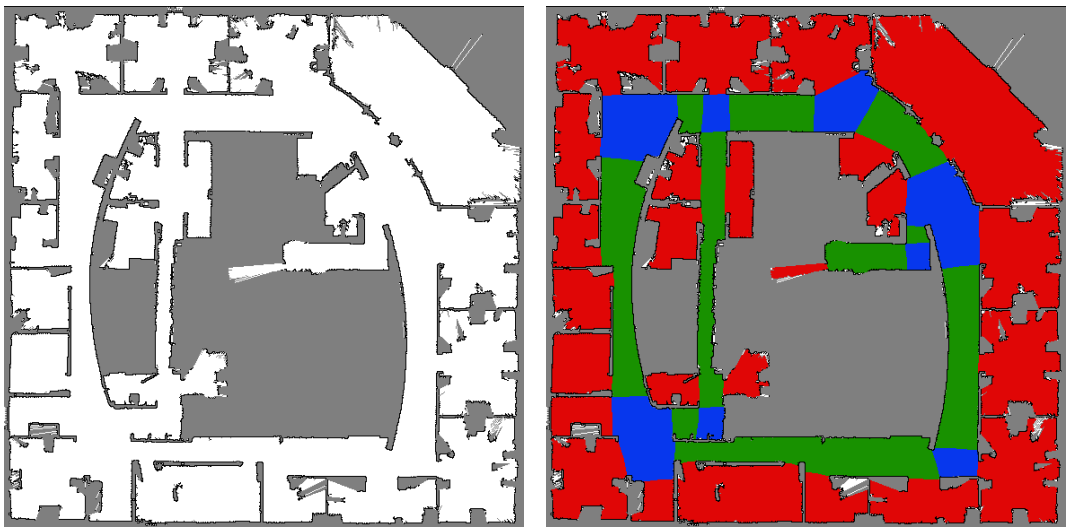
Within the topological map, agents move from one area to another via gateways. Thus, a new agent appears in the robot's field-of-view either because the robot moved such that the agent became visible or the agent moved into the field-of-view while moving towards or across a gateway. The norm learned in Section 7.4.2 defines a probability distribution over where pedestrians cross gateways. This distribution and an analysis of the visible portion of the environment therefore provides a straightforward means of estimating where new pedestrians might appear.

APPENDIX A

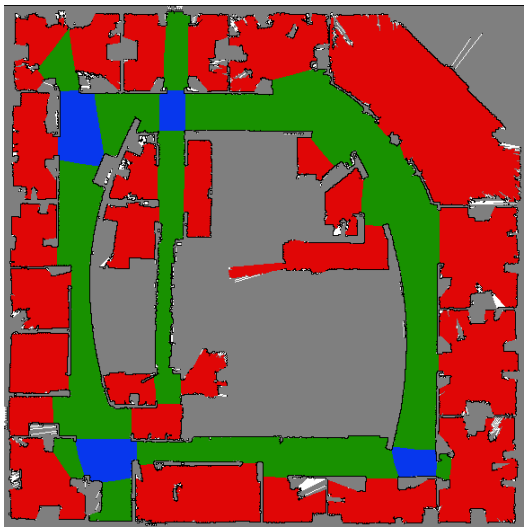
Ground-truth Labeled Maps

A.1 Place Labeling Results

In this section, we provide all ground-truth maps used for our evaluation. For each map, we provide: (a) the ground-truth metric map from which semantic labels were extracted, (b) the ground-truth labels used to generate our results, and (c) the semantic map produced by our MCMC algorithm.

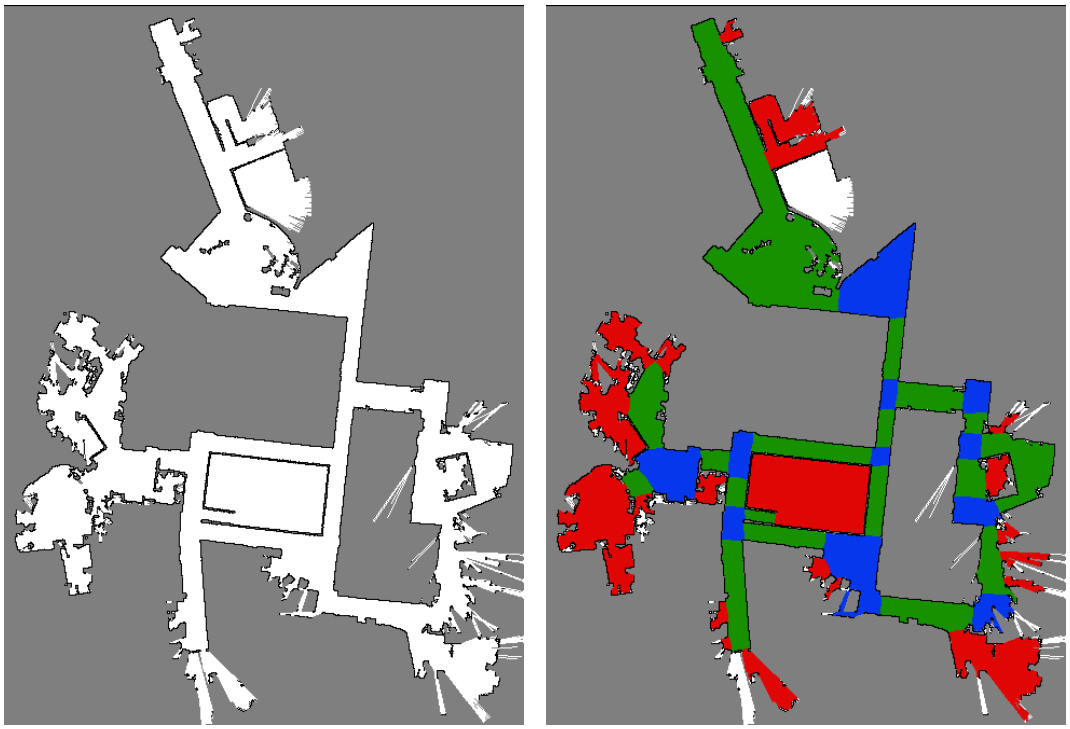


(a) Metric map (b) Ground-truth labels



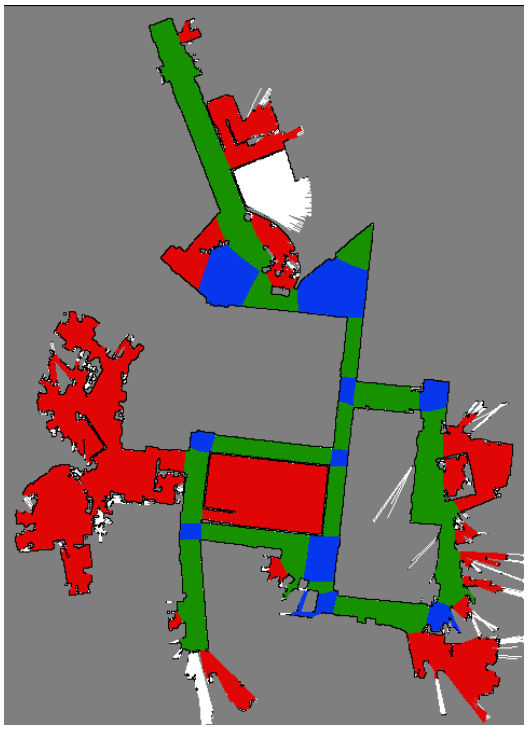
(c) MCMC labels

Figure A.1: Maps used for evaluation of intel.



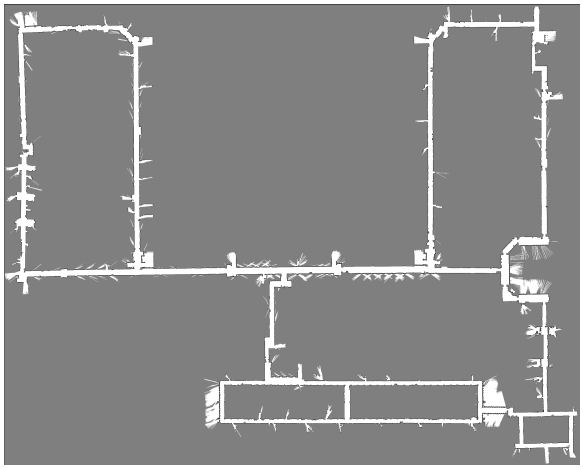
(a) Metric map

(b) Ground-truth labels

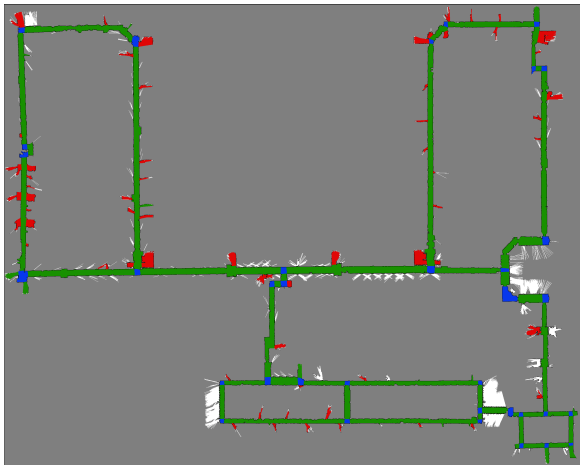


(c) MCMC labels

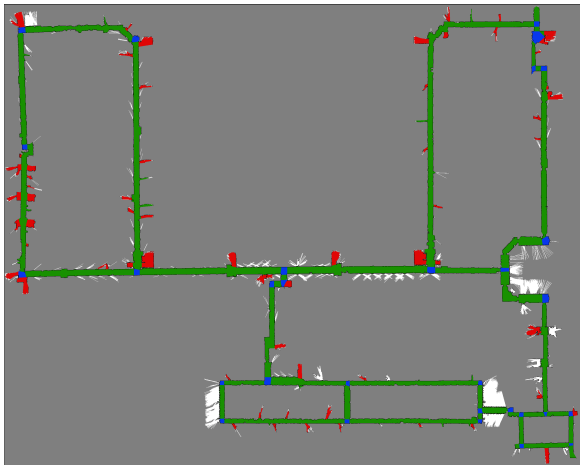
Figure A.2: Maps used for evaluation of csail.



(a) Metric map

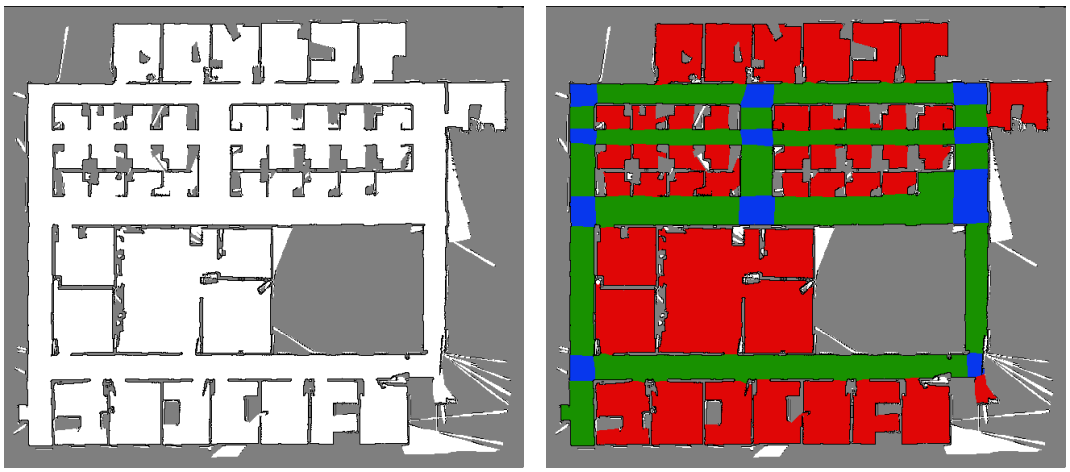


(b) Ground-truth labels



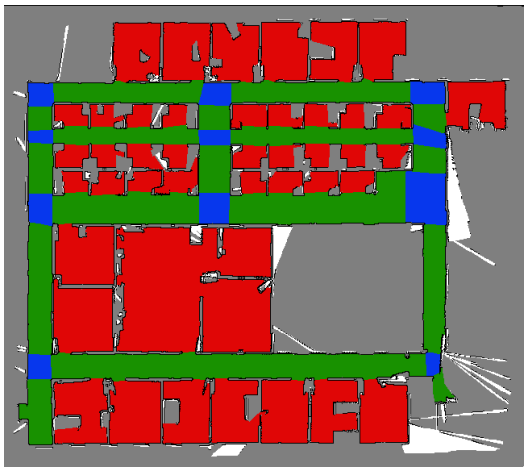
(c) MCMC labels

Figure A.3: Maps used for evaluation of infinite_corridor.



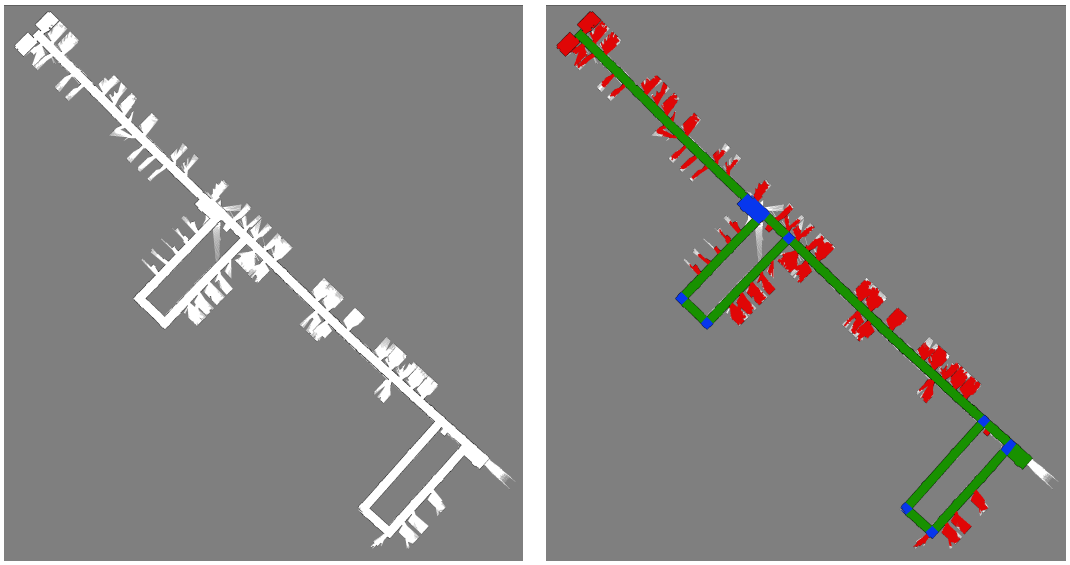
(a) Metric map

(b) Ground-truth labels



(c) MCMC labels

Figure A.4: Maps used for evaluation of sdr.

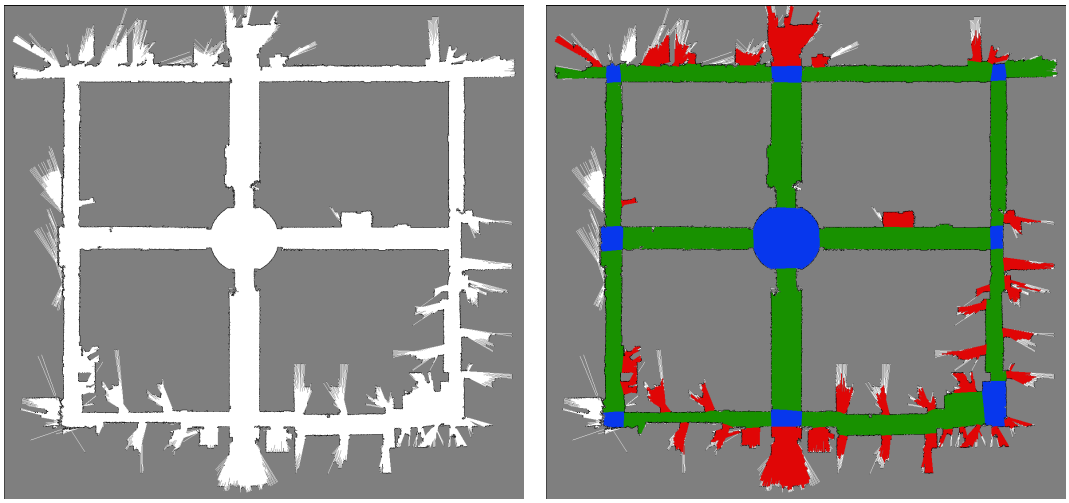


(a) Metric map

(b) Ground-truth labels

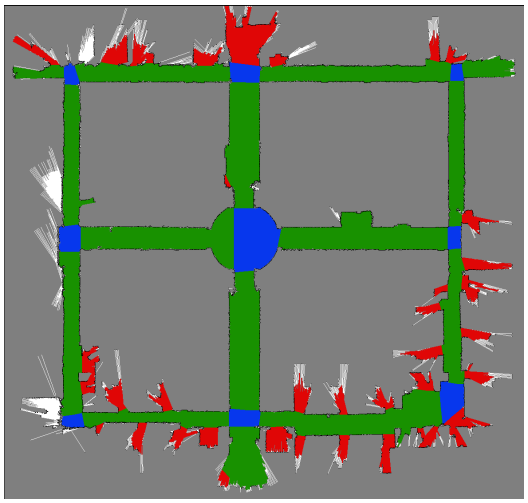
(c) MCMC labels

Figure A.5: Maps used for evaluation of abuilding.



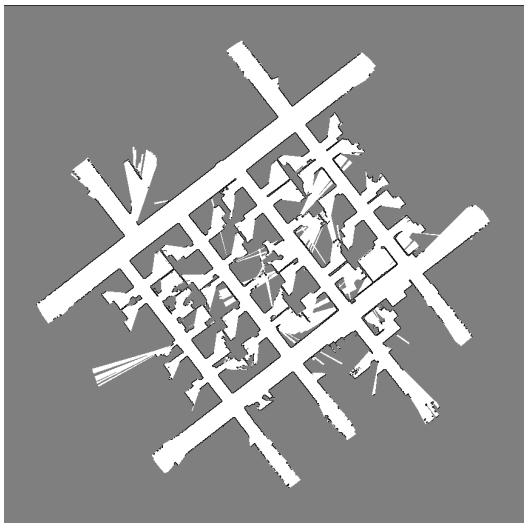
(a) Metric map

(b) Ground-truth labels

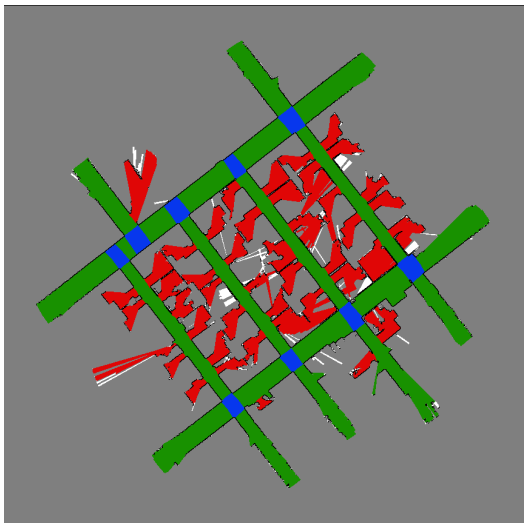


(c) MCMC labels

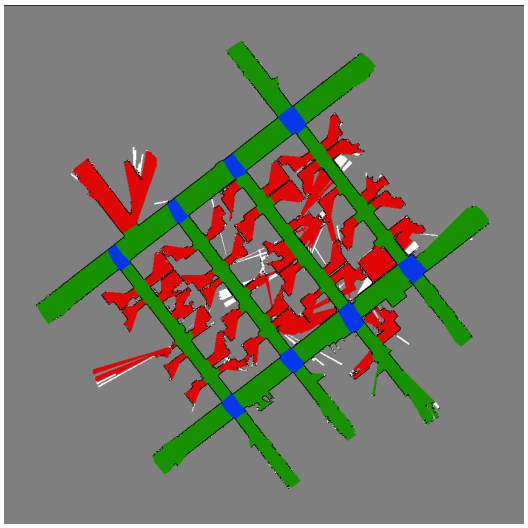
Figure A.6: Maps used for evaluation of aces3.



(a) Metric map

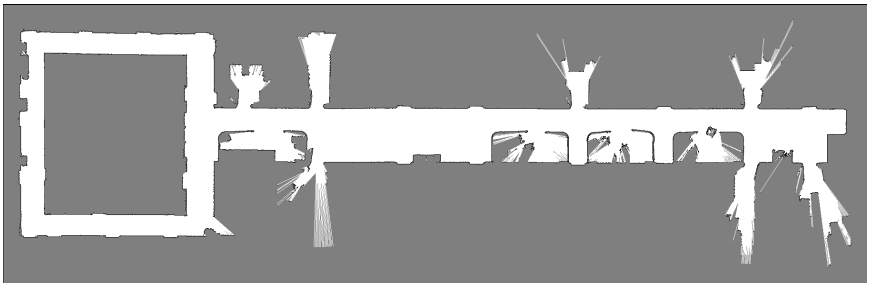


(b) Ground-truth labels

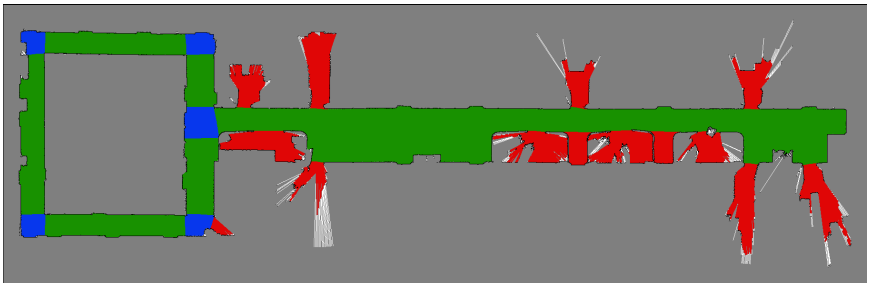


(c) MCMC labels

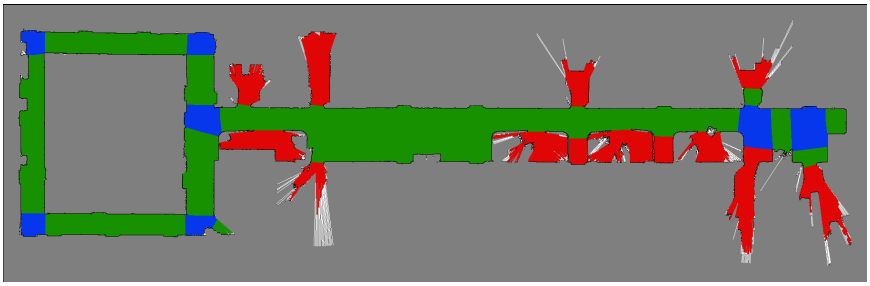
Figure A.7: Maps used for evaluation of oregon.



(a) Metric map



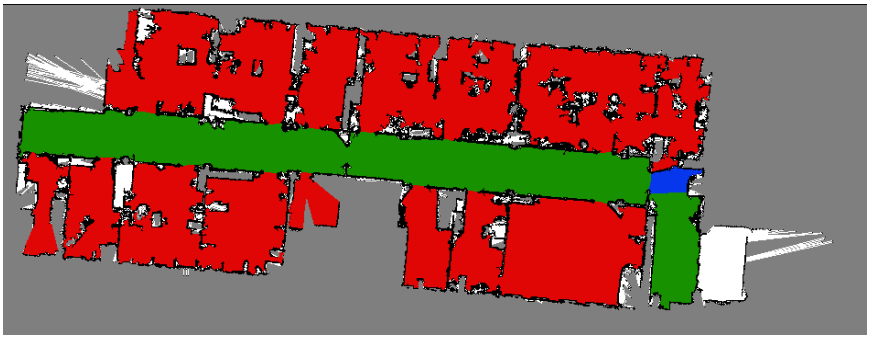
(b) Ground-truth labels



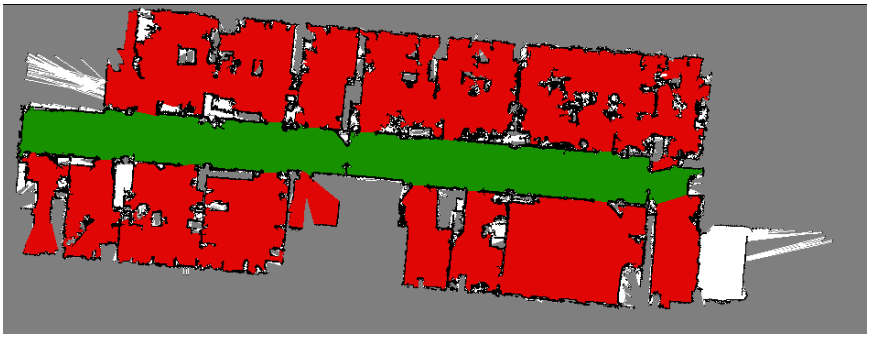
(c) MCMC labels

Figure A.8: Maps used for evaluation of seattle.

(a) Metric map

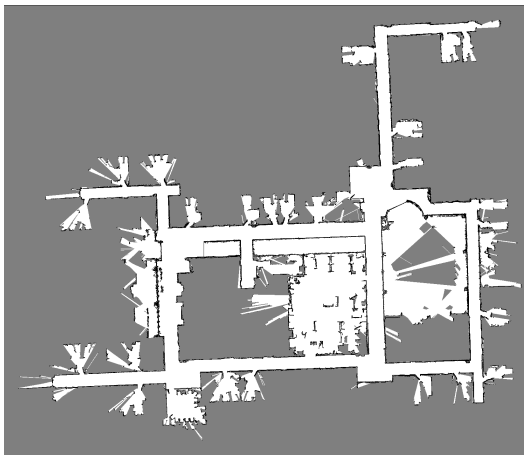


(b) Ground-truth labels

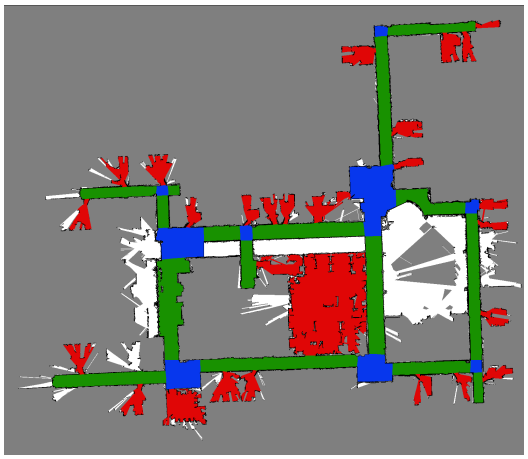


(c) MCMC labels

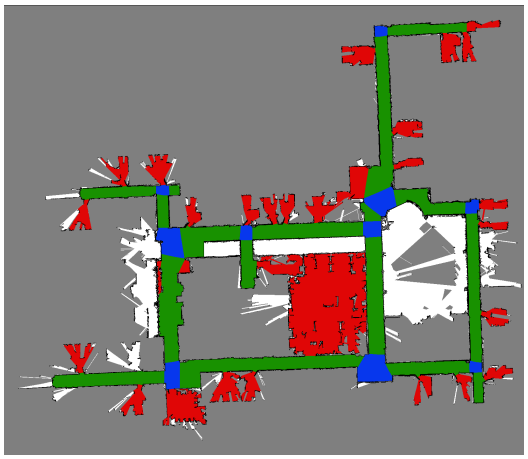
Figure A.9: Maps used for evaluation of fr79.



(a) Metric map

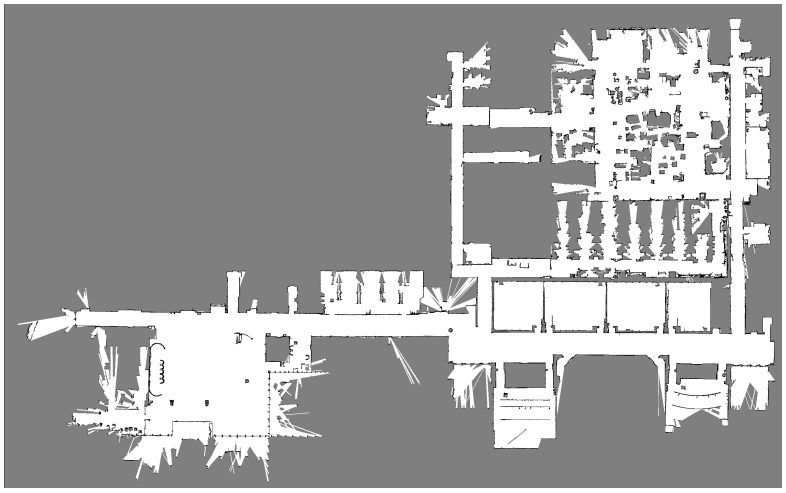


(b) Ground-truth labels

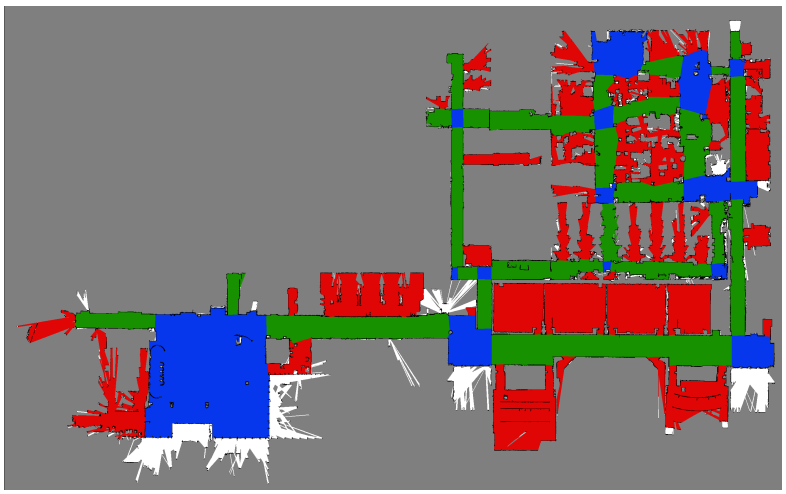


(c) MCMC labels

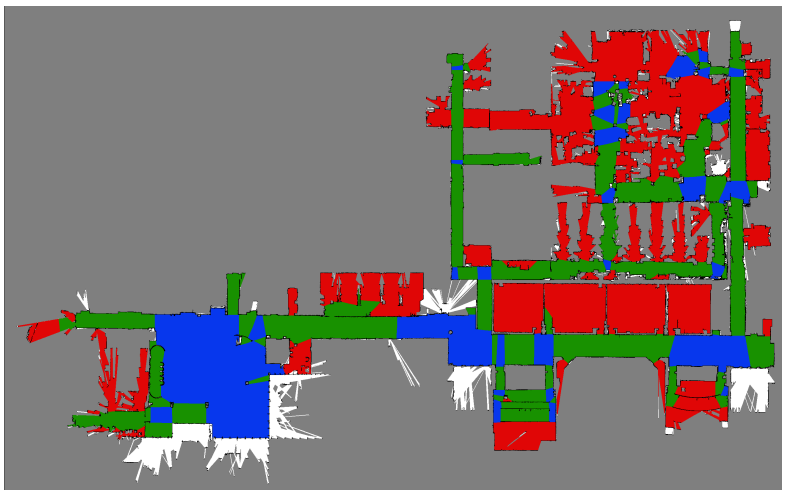
Figure A.10: Maps used for evaluation of bbb3.



(a) Metric map

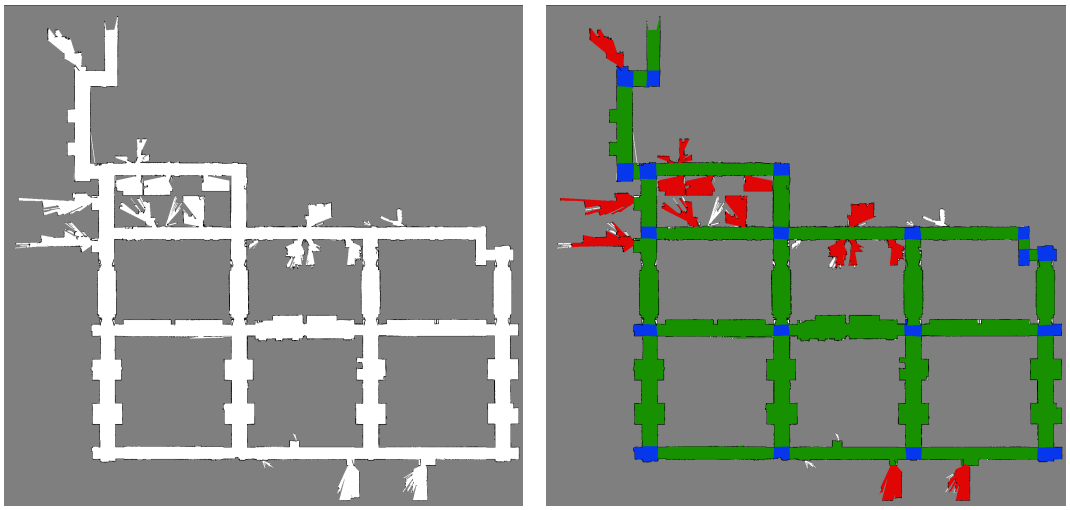


(b) Ground-truth labels

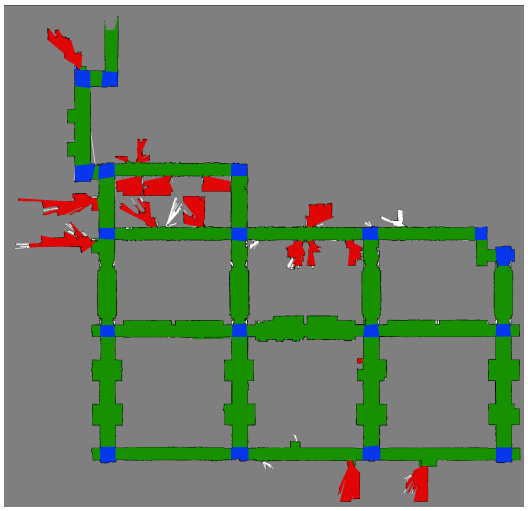


(c) MCMC labels

Figure A.11: Maps used for evaluation of bbbdow1.

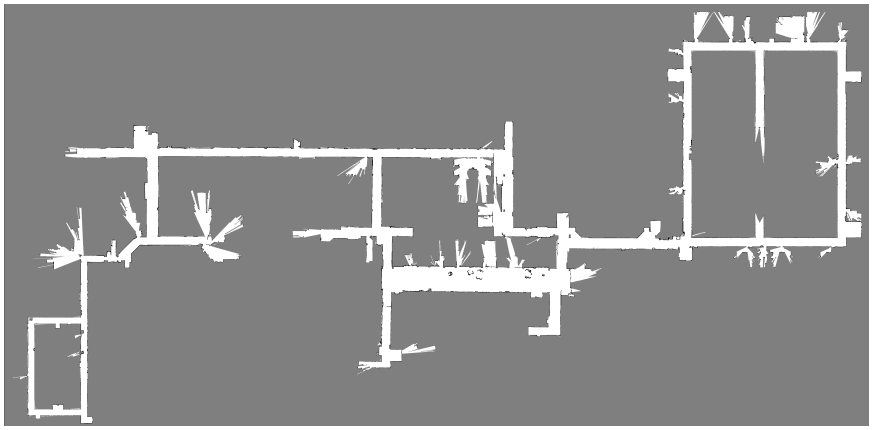


(a) Metric map (b) Ground-truth labels

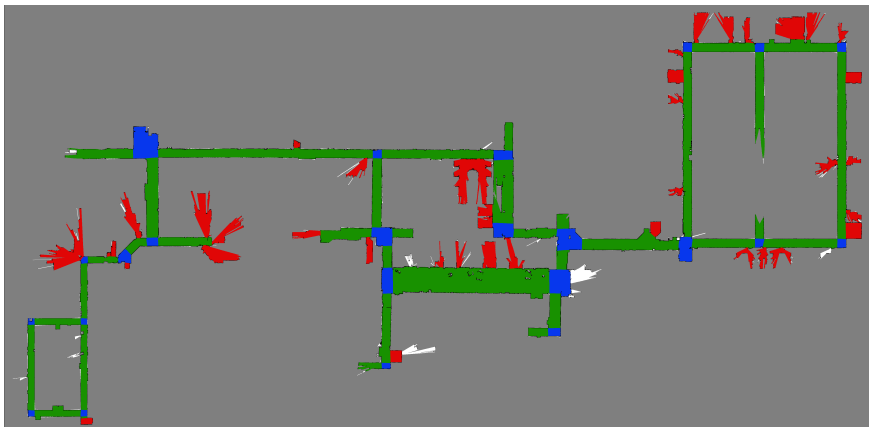


(c) MCMC labels

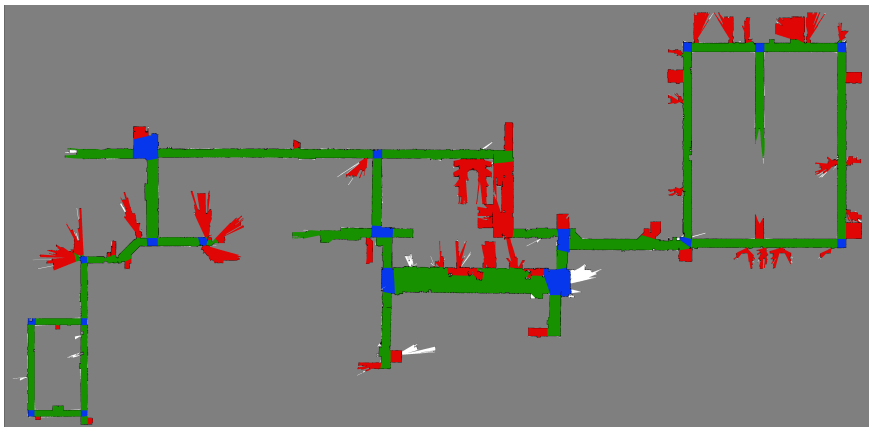
Figure A.12: Maps used for evaluation of eecs3.



(a) Metric map

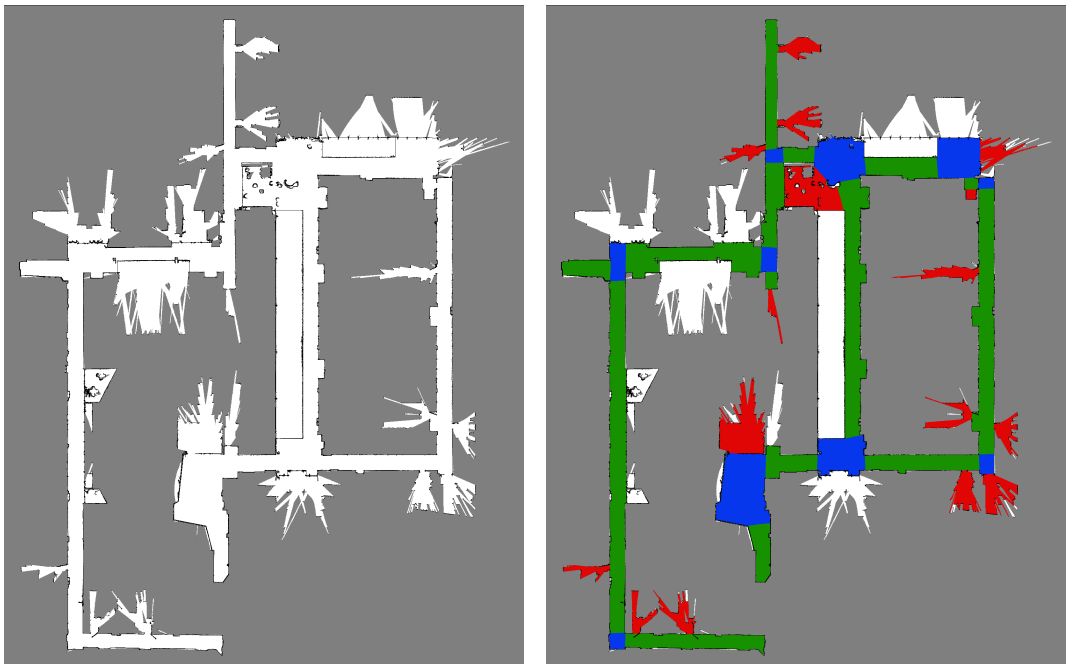


(b) Ground-truth labels

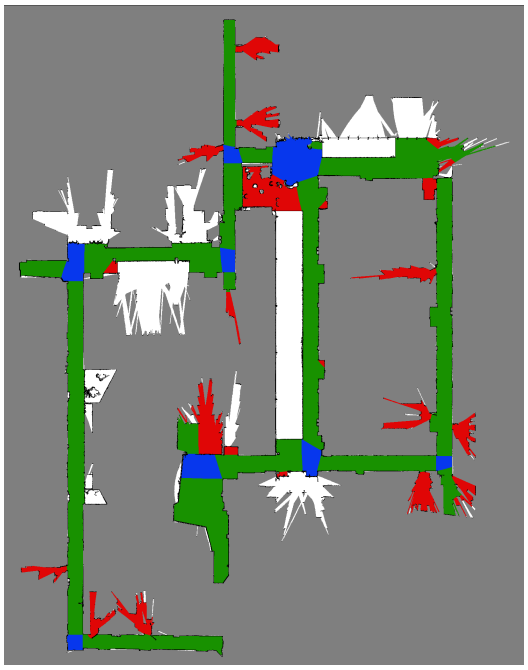


(c) MCMC labels

Figure A.13: Maps used for evaluation of ggb1.

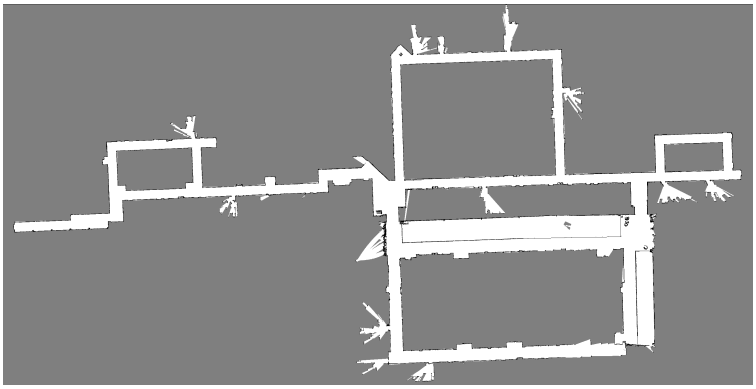


(a) Metric map (b) Ground-truth labels

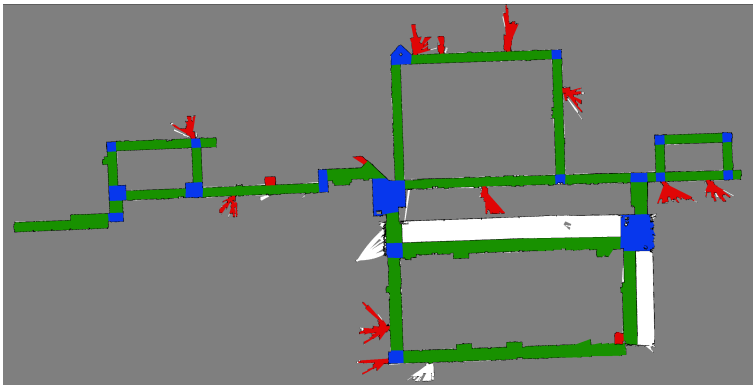


(c) MCMC labels

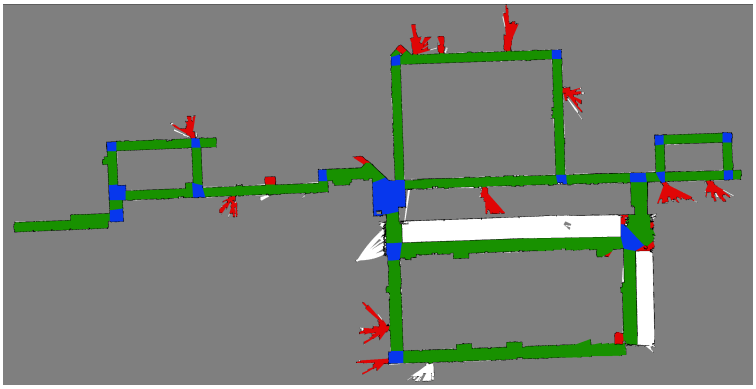
Figure A.14: Maps used for evaluation of ggb2.



(a) Metric map

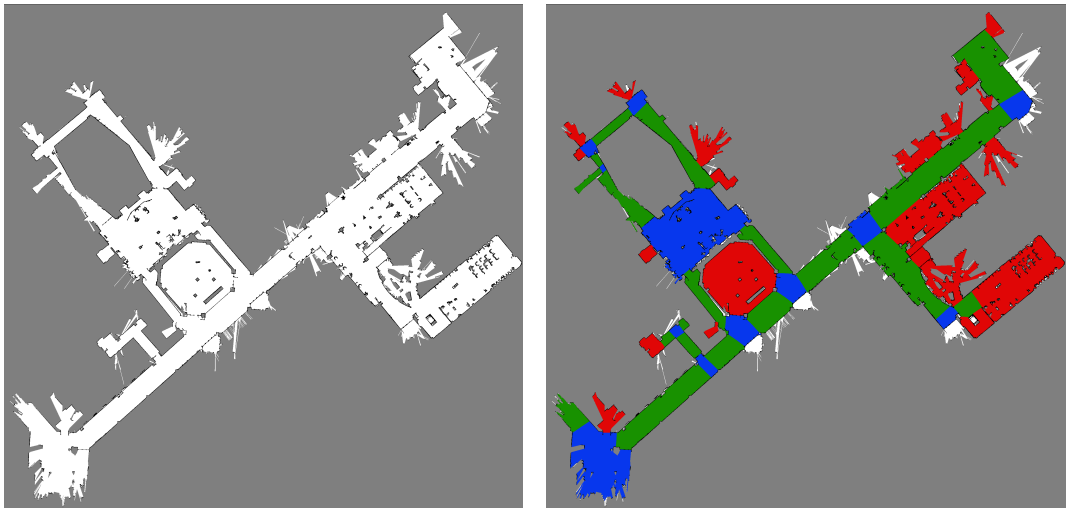


(b) Ground-truth labels



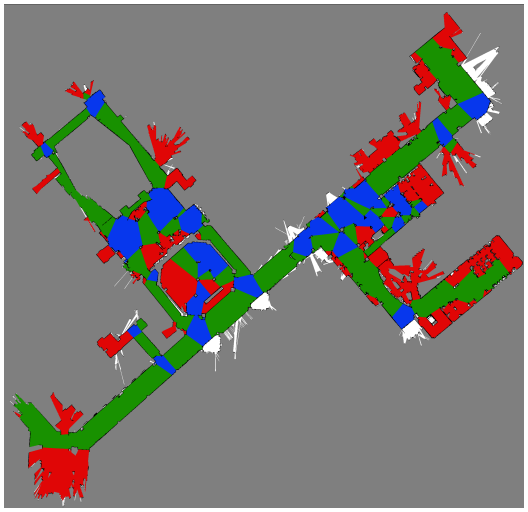
(c) MCMC labels

Figure A.15: Maps used for evaluation of ggb3.



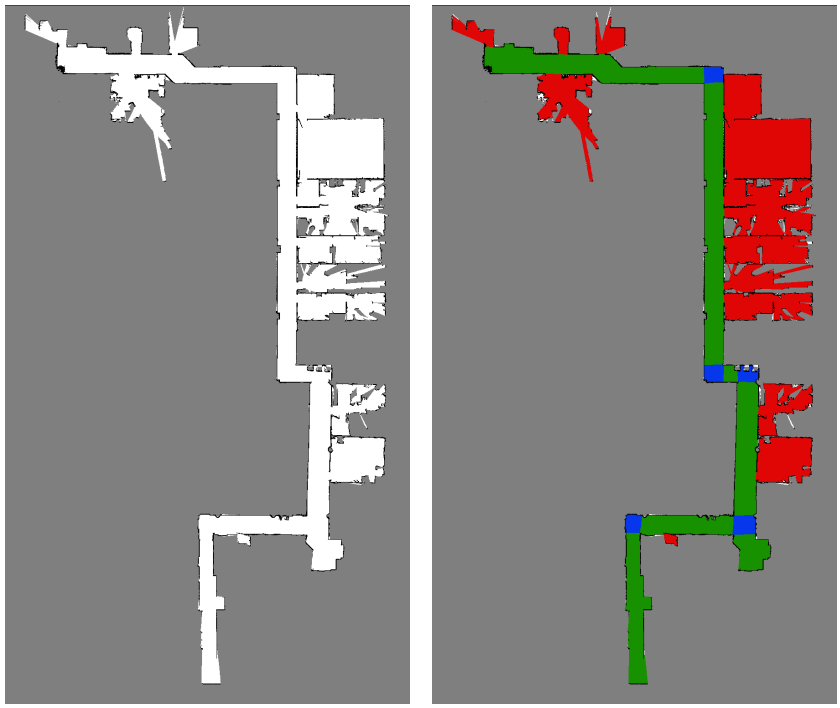
(a) Metric map

(b) Ground-truth labels

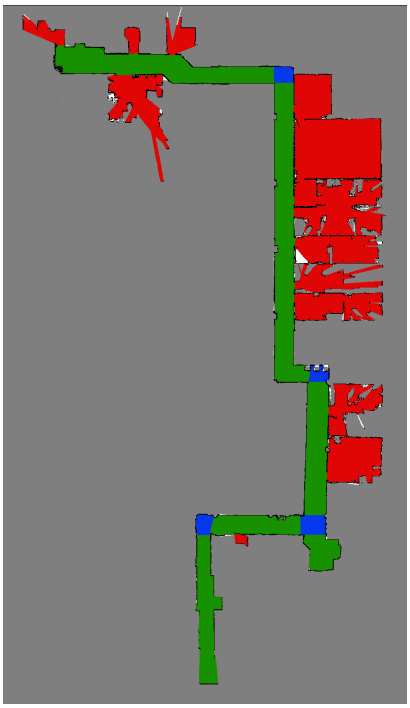


(c) MCMC labels

Figure A.16: Maps used for evaluation of pierpont1.



(a) Metric map (b) Ground-truth labels

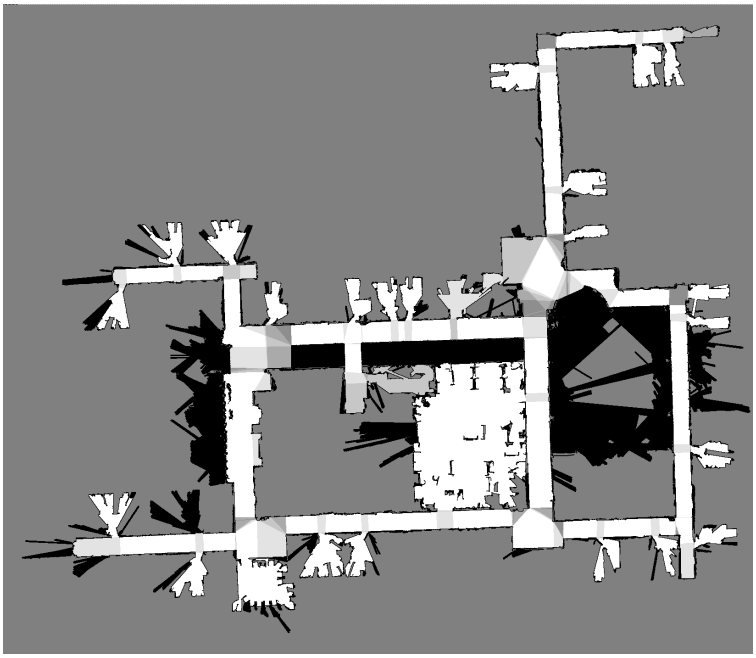


(c) MCMC labels

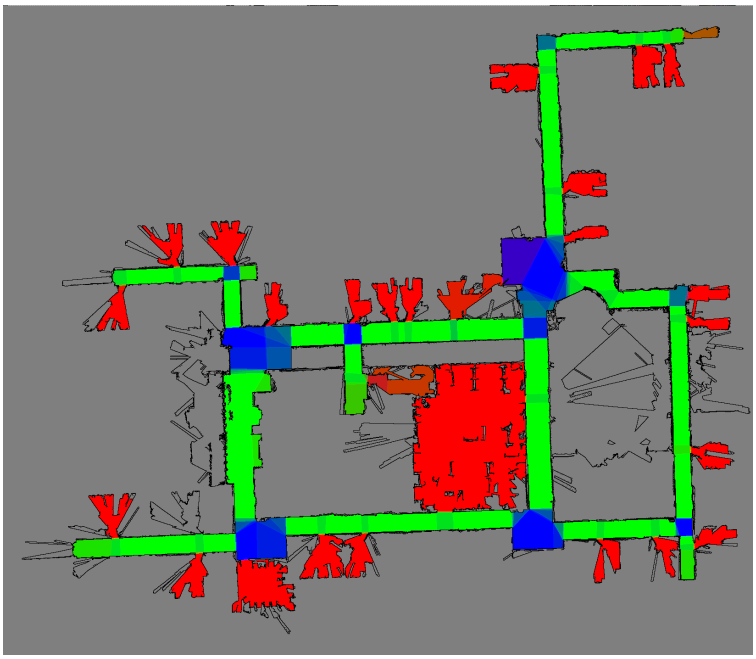
Figure A.17: Maps used for evaluation of tufts3.

A.2 Human Variability Results

The results for the human variability experiment are shown using two figures. The first figure (a) shows the consensus amongst the different labelings. White indicates complete agreement between all maps, while light gray is maximum disagreement (all classes with the same number of votes), the grayscale value is set using $L_{max} - L_{min}$. Black areas in the map are free space cells without a semantic label because they correspond to areas on the other side of glass walls or errant beams during mapping. (b) shows what labels were assigned by the human labelers. Each labeled map votes green (path segment), red (destination), or blue (decision point) for each cell in the combined map. Thus, path segment/decision point ambiguity is teal, decision point/destination ambiguity is purple, and path segment/destination ambiguity is brownish.

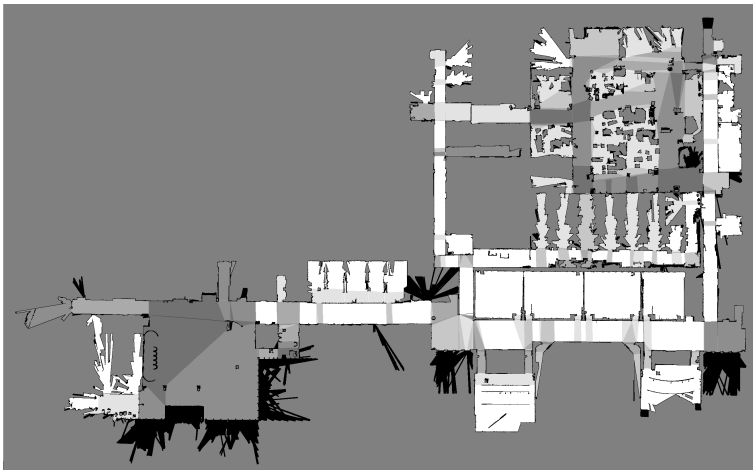


(a)

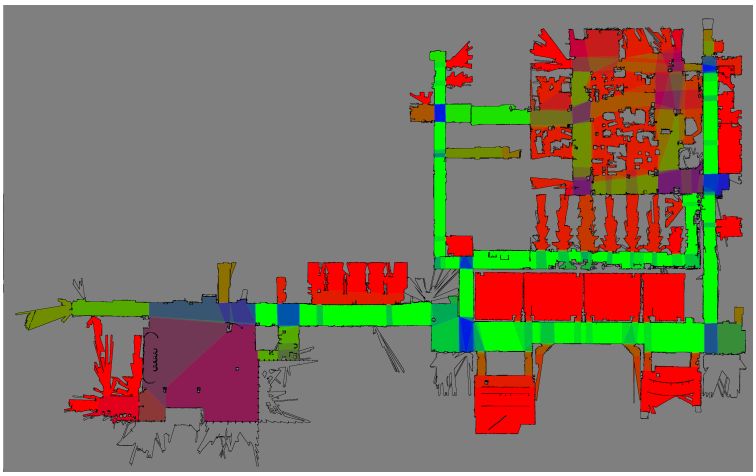


(b)

Figure A.18: Human variability results for bbb3.

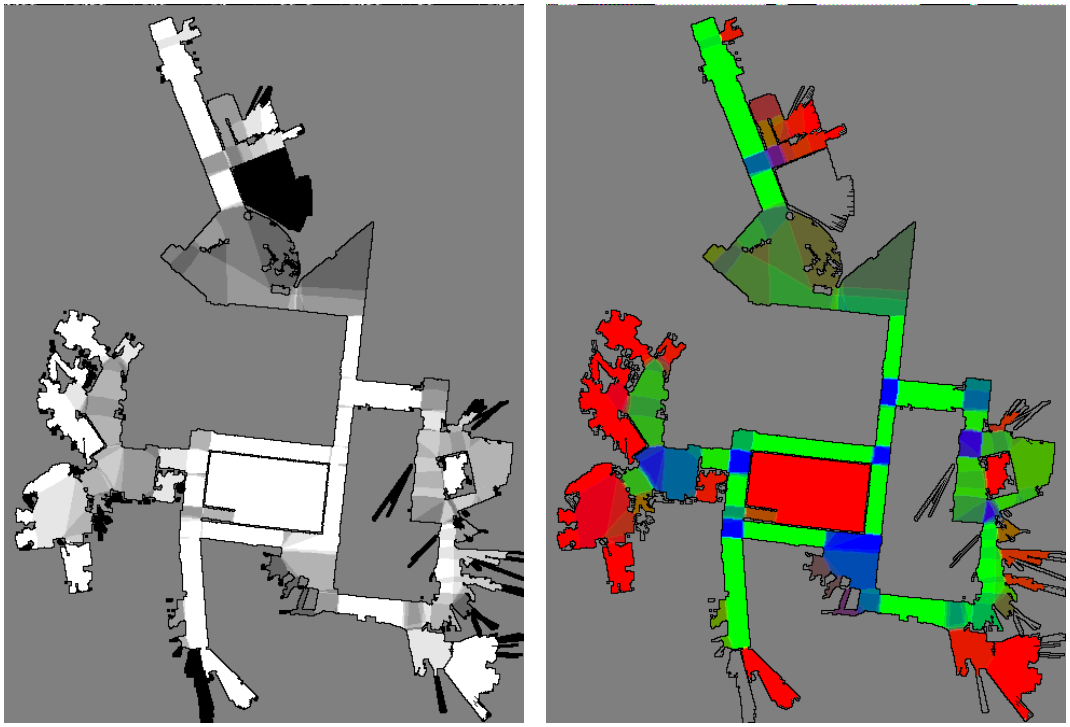


(a)



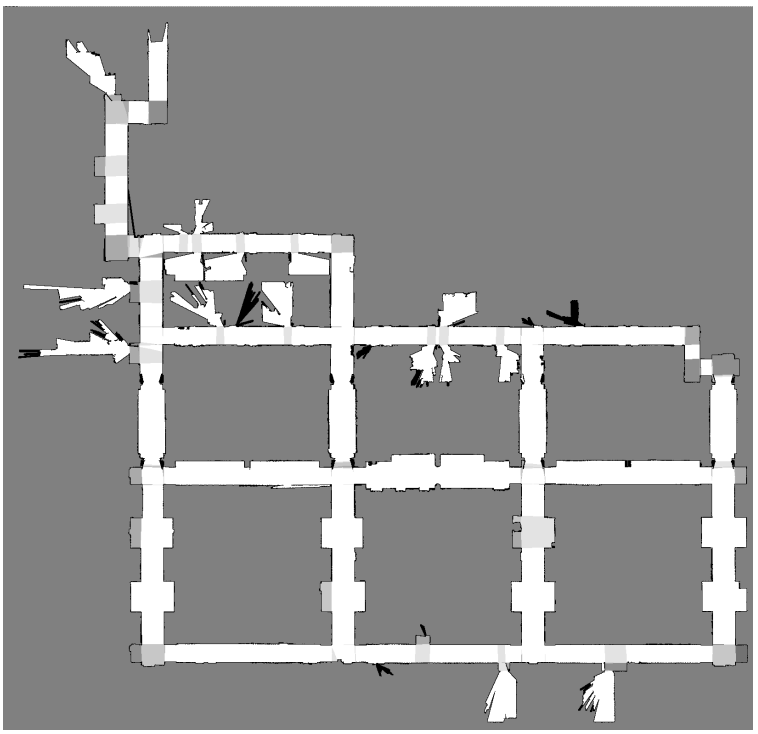
(b)

Figure A.19: Human variability results for bbbdow1.

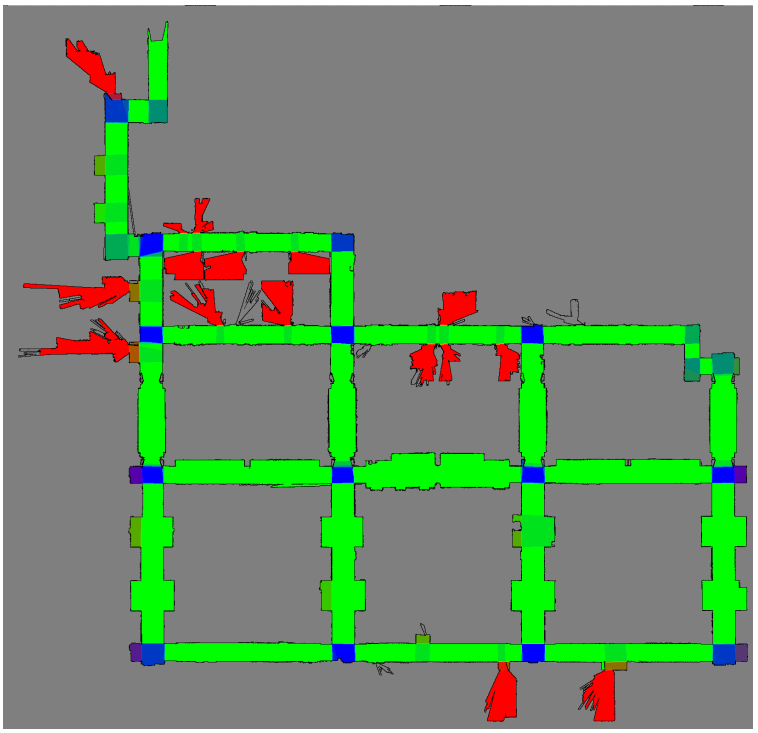


(a) (b)

Figure A.20: Human variability results for csail3.



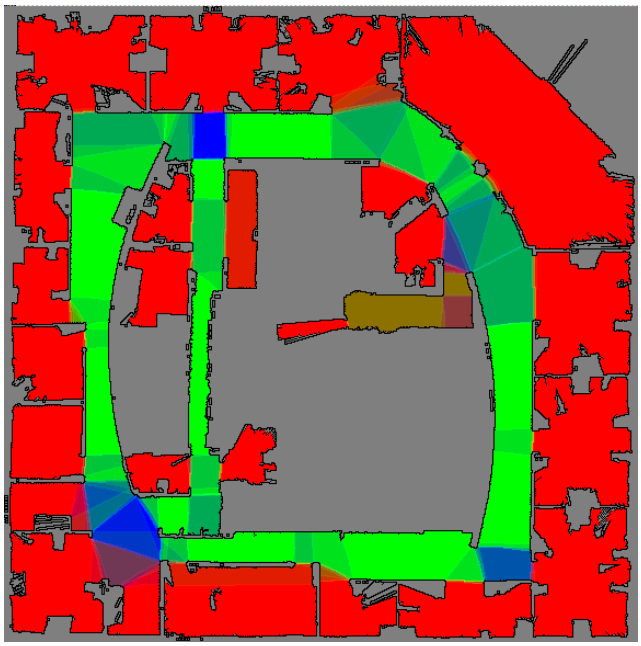
(a)



(b)

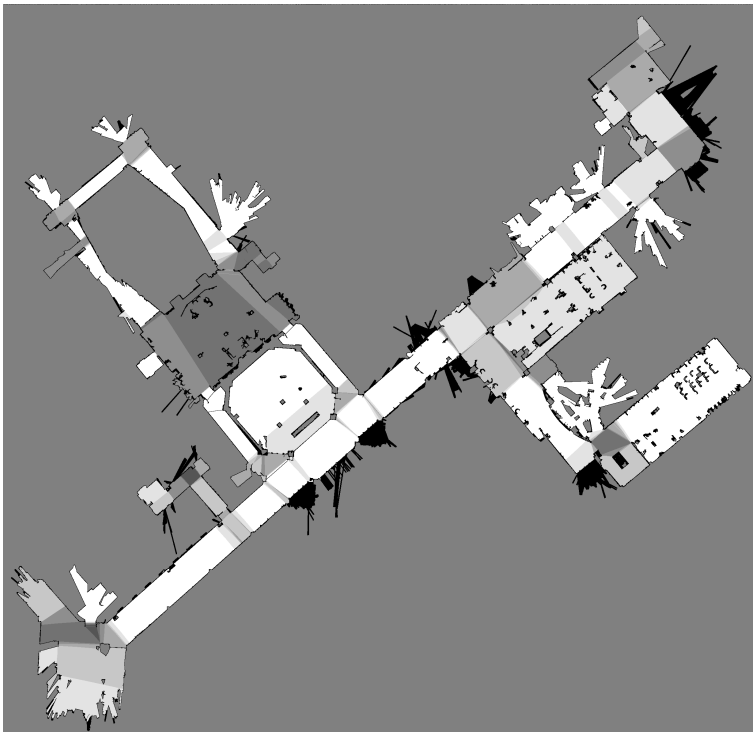
Figure A.21: Human variability results for eecs3.

(a)

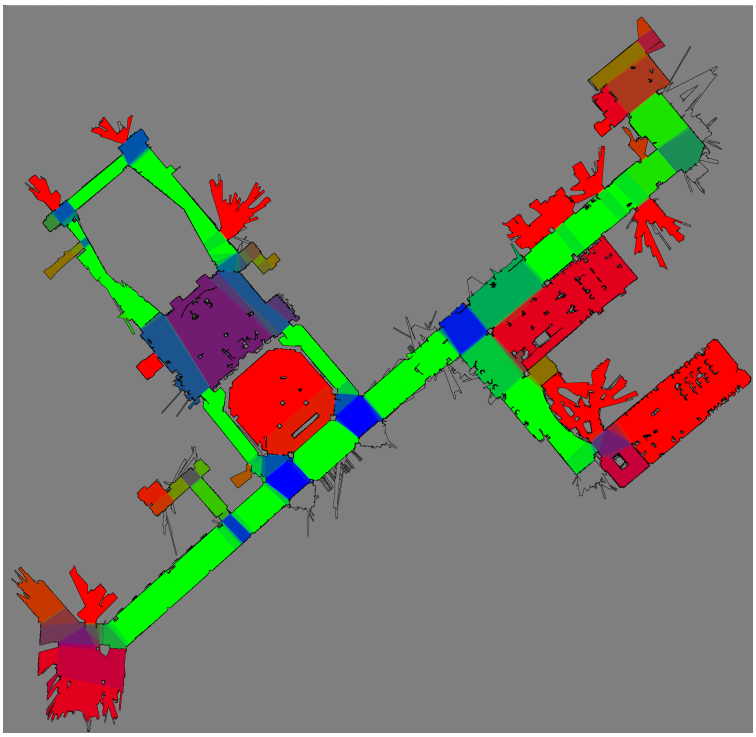


(b)

Figure A.22: Human variability results for intel.



(a)



(b)

Figure A.23: Human variability results for pierpont1.

APPENDIX B

Multi-laser Pedestrian Tracking

We model the robot's environment as a tuple $E_t = \langle M_t, \mathbf{O}_t \rangle$. M_t is a metric map of the static environment, either constructed *a priori* or being built concurrently with the object tracking using an online SLAM algorithm. $\mathbf{O}_t = \{\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_K\}$ are the dynamic objects being tracked by the robot. Each object, $\mathbf{o}_k = \langle x, y, v_x, v_y, a_x, a_y, S \rangle$, is represented by its position (x, y), velocity (v_x, v_y), and acceleration (a_x, a_y) in the x-y plane, and a shape model, S.

At each timestep t, the input to the tracking algorithm is the robot's pose $\mathbf{x}_t$, a collection of objects, $\mathbf{Z}_t^N$, segmented from a laser scan, the environment map, M_t, and the previously estimated objects $\mathbf{O}_{t-1}$. The goal of the tracker is to estimate $\mathbf{O}_t$.

We treat each moving object as independent and estimate each object's state using a linear Kalman filter. The following sections detail the methods used for detecting objects in a laser scan, building models of the detected objects, associating newly detected objects with existing tracked objects, and the process and measurement models used for the Kalman filter.

B.1 Object Detection and Modeling

For each update, we segment a laser scan, L_t, received at time t into a set of measured objects $\mathbf{Z}_t^N = \{z_t^0, z_t^1, ..., z_t^N\}$, where $z_t^n = \langle x, y, \Sigma_{xy}, P, S, \Sigma_S \rangle$, representing the estimated position plus uncertainty (x, y, Σ_{xy}), the measured laser endpoints P, and a shape fit to the laser endpoints plus uncertainty (S, Σ_S).

Segmentation of the laser scan begins by splitting the laser measurements into a set of measurements that can be explained by obstacles in the static environment map, M_t, and those that fall into space estimated to be free in M_t. For example, in an occupancy grid representation, those measurements whose endpoint lands in a cell with $p(occ > 0.5)$ are classified as static measurements, and measurements whose endpoint lands in a cell with

$p(occ <= 0.5)$ are classified as moving.

Given the measurement classification, individual objects are identified by clustering adjacent moving laser points whose difference in measured radial distance is less than some threshold ΔR_{max}. The output of the clustering process is N clusters of points $P^1, P^2, ..., P^N$. We use these point clusters to estimate the position and shape of the objects measured by the laser scan.

The most straightforward way to calculate the position of an object is to use the mean of the endpoints in the cluster. However, this approach estimate the true position only in rare due to the functionality of the laser scan. As shown in Figure B.1, the laser scan can never observe the full boundary of an object. Consequently, the mean endpoint point will almost always underestimate the true distance to the object.

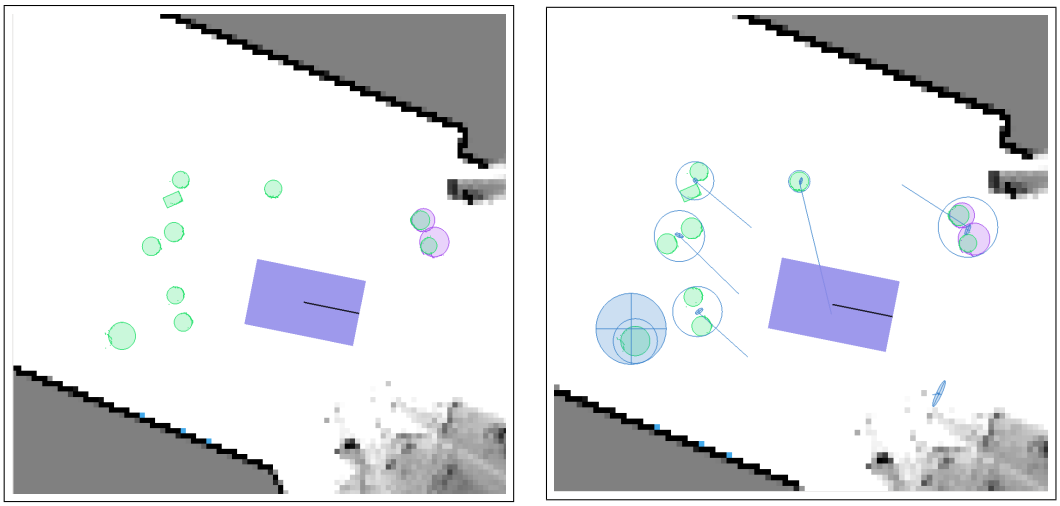
A more principled approach is to estimate the underlying shape of the object being measured by the laser points, that is to find some shape S that minimizes the measurement error between the shape boundary and the measured laser points. The shape S can take many forms. We choose models that reasonably describe the most common moving objects in the environment, like legs and carts. We calculate three models for each point cluster and select the model with the lowest RMS error:

- A circle with radius in the range $[R_{min}, R_{max}]$.
- Two circles with two radii (R_1, R_2), each in the range $[R_{min}, R_{max}]$.
- A bounding rectangle.

For each model, we calculate the shape that minimizes the RMS error between the shape boundary and the measured points. We use the circle-fitting algorithm created by Chernov and Lesort [97]. The two-circle model splits the points into two disjoint sets that minimize the total RMS error. The two-circle model accounts for cases where two legs exist in the same point cluster.

For the bounding rectangle, we use a variation on the rotating calipers-based minimum area bounding rectangle algorithm [98]. Instead of selecting the bounding rectangle with the minimum area, we select the rectangle that minimizes the RMS error between the rectangle boundary and the measured points.

Once the shape of the object is calculated, the position of the object is taken to be the center of the shape. Al-Sharadqah and Chernov [99] describe how uncertainty propagates in the circle fitting algorithm we use. The uncertainty of the position and radius of the fitted circle is a function of the angular spread of the measured points, The greater the maximum angular difference between the points, the lower the uncertainty.



(a) (b)

Figure B.1: (a) shows the dynamic objects detected around the robot. The color indicates the laser that detected the object on the robot. Green is the back laser and magenta is the front laser. Individual legs can clearly be seen. A shape-fitting error results in one leg being estimated to be a rectangle. (b) shows the estimated position of each object, along with the estimated velocity vector and position covariance. For those objects with two detected legs, the position and velocity are accurately measured to be the center of the line connecting the legs. For single leg cases, the estimate is more uncertain (left side) or results in an incorrect velocity based on the center switching from the center of two legs to the center of a single leg (top). Merging of the estimated from both lasers can be seen in the upper right corner.

The uncertainty calculation in [99] uses the angle from the circle center to the measured points for determining the amount of the circle boundary that was measured. We adapt this approach for finding the uncertainty of the fitted rectangle by projected the measured points onto the rectangle boundary. We use the angle from the center to the projected point for the uncertainty calculation.

B.2 Data Association

We match each detected object $z_t^n \in Z_t^N$ with a tracked object o_k by considering the distance between $P_{z_t^n}$ and the boundary of each tracked object. We associate a detected object with the closest tracked object if the distance is less than a certain threshold.

$$\arg\min_k \sum_P dist(p, S_{o_k}) \tag{B.1}$$

B.3 Constant-Acceleration State Estimation

We use a linear Kalman filter for estimating the state of each object. Below we describe each component of the tracking filter. Because Kalman filter notation varies, we list the Kalman filter equations in (B.2)-(B.6) for clarity. u_t, which normally appears in (B.2), is left out because the underlying action being taken by the object is unknown.

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} \tag{B.2}$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + V_k \tag{B.3}$$

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + W_k)^{-1} \tag{B.4}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - H_k \hat{x}_{k|k-1}) \tag{B.5}$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \tag{B.6}$$

Process Model Our process model, (B.7), for predicting the next object state uses the classical equations of motion for an object moving with constant acceleration.

$$F_k = \begin{pmatrix} 1 & 0 & \delta_t & 0 & \frac{1}{2}\delta_t^2 & 0 \\ 0 & 1 & 0 & \delta_t & 0 & \frac{1}{2}\delta_t^2 \\ 0 & 0 & 1 & 0 & \delta_t & 0 \\ 0 & 0 & 0 & 1 & 0 & \delta_t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.7}$$

where $\delta_t = t_k - t_{k-1}$.

We use a constant zero-mean Gaussian process noise:

$$V_k = diag(\sigma_{pos}, \sigma_{pos}, \sigma_{vel}, \sigma_{vel}, \sigma_{accel}, \sigma_{accel}) \tag{B.8}$$

Measurement Model In addition to the measured position in z_t^n, we take the first and second time derivatives of the recent history of position measurements to provide a measurement of the object's velocity and acceleration.

$$y_k = \begin{pmatrix} x & y & v_x & v_y & a_x & a_y \end{pmatrix}^T \tag{B.9}$$

Because the measurement state is the same as the object state, $H_k = I$.

BIBLIOGRAPHY

[1] J. S. Gutmann, E. Eade, P. Fong, and M. E. Munich, "Vector Field SLAM : Localization by learning the spatial variation of continuous signals," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 650–667, 2012.

[2] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, July 2013.

[3] M. Pfingsthorn and A. Birk, "Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints," *The International Journal of Robotics Research*, 2015.

[4] N. Sünderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 2012, pp. 1879–1884.

[5] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 62–69.

[6] M. Cummins and P. Newman, "Appearance-only SLAM at large scale with FAB-MAP 2.0," *The International Journal of Robotics Research*, vol. 30, no. 9, pp. 1100–1123, 2011.

[7] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1643–1649.

[8] A. Ranganathan and F. Dellaert, "Online probabilistic topological mapping," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 755–771, 2011.

[9] S. Tully, G. Kantor, and H. Choset, "A unified bayesian framework for global localization and SLAM in hybrid metric/topological maps," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 271–288, 2012.

[10] C. Johnson and B. Kuipers, "Efficient search for correct and useful topological maps," in *IROS*. IEEE, 2012, pp. 5277–5282.

[11] P. Beeson, J. Modayil, and B. Kuipers, "Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy," *The International Journal of Robotics Research*, vol. 29, no. 4, pp. 428–459, 2010.

[12] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using adaboost," in *ICRA*. IEEE, 2005, pp. 1730–1735.

[13] S. Friedman, H. Pasula, and D. Fox, "Voronoi random fields: Extracting topological structure of indoor environments via place labeling." in *IJCAI*, vol. 7, 2007, pp. 2109–2114.

[14] A. Ranganathan, E. Menegatti, and F. Dellaert, "Bayesian inference in the space of topological maps," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 92–107, Feb 2006.

[15] D. Marinakis and G. Dudek, "Pure topological mapping in mobile robotics," *IEEE Transactions on Robotics*, vol. 26, no. 6, pp. 1051–1064, 2010.

[16] B. Kuipers and Y.-T. Byun, "A robust, qualitative method for robot spatial learning," in *AAAI*, 1988, pp. 774–779.

[17] S. Tully, G. Kantor, H. Choset, and F. Werner, "A multi-hypothesis topological SLAM approach for loop closing on edge-ordered graphs," in *IROS*, oct. 2009, pp. 4943 – 4948.

[18] K. Lynch, *The Image of the City*, ser. Harvard-MIT Joint Center for Urban Studies Series. MIT Press, 1960.

[19] B. Kuipers, "Modeling spatial knowledge," *Cognitive Science*, vol. 2, no. 2, pp. 129 – 153, 1978.

[20] J. J. Park, C. Johnson, and B. Kuipers, "Robot navigation with model predictive equilibrium point control." in *IROS*, 2012, pp. 4945–4952.

[21] B. Kuipers, "The spatial semantic hierarchy," *Artificial intelligence*, vol. 119, no. 1, pp. 191–233, 2000.

[22] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping SLAM: toward exact localization without explicit localization," *IEEE Transactions on Robotics*, vol. 17, no. 2, pp. 125–137, 2001.

[23] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "Robotic exploration as graph construction," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 6, pp. 859 –865, dec 1991.

[24] Ö. Erkent and H. I. Bozma, "Bubble space and place representation in topological maps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 672–689, 2013.

[25] A. Ranganathan and F. Dellaert, "Bayesian surprise and landmark detection," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2017–2023.

[26] J. L. Blanco, J. A. Fernndez-Madrigal, and J. Gonzlez, "Toward a unified bayesian approach to hybrid metric–topological slam," *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 259–270, April 2008.

[27] H. Karaoguz and H. I. Bozma, "Reliable topological place detection in bubble space," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2015, pp. 5462–5467.

[28] R. Goeddel and E. Olson, "Learning semantic place labels from occupancy grids using CNNs," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[29] L. Shi and S. Kodagoda, "Towards generalization of semi-supervised place classification over generalized voronoi graph," *Robotics and Autonomous Systems*, vol. 61, no. 8, pp. 785 – 796, 2013.

[30] Y. Liao, S. Kodagoda, Y. Wang, L. Shi, and Y. Liu, "Place classification with a graph regularized deep neural network," *IEEE Transactions on Cognitive and Developmental Systems*, vol. PP, no. 99, 2016.

[31] A. Pronobis, O. M. Mozos, B. Caputo, and P. Jensfelt, "Multi-modal semantic place classification," *IJRR*, vol. 29, no. 2-3, pp. 298–320, Feb 2010.

[32] E. Brunskill, T. Kollar, and N. Roy, "Topological mapping using spectral clustering and classification," in *IROS*. IEEE, 2007, pp. 3491–3496.

[33] M. Liu, F. Colas, and R. Siegwart, "Regional topological segmentation based on mutual information graphs," in *ICRA*. IEEE, 2011, pp. 3269–3274.

[34] Z. Liu and G. von Wichert, "Extracting semantic indoor maps from occupancy grids," *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 663–674, 2014.

[35] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA'12)*, Saint Paul, MN, USA, may 2012.

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

[37] A. Rituerto, A. Murillo, and J. Guerrero, "Semantic labeling for indoor topological mapping using a wearable catadioptric system," *Robotics and Autonomous Systems*, vol. 62, no. 5, pp. 685–695, 2014.

[38] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *Proceedings of the Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*. New York, NY: Elsevier Science, 1986, pp. 267–288.

[39] G. Dudek, P. Freedman, and S. Hadjres, "Using local information in a non-local way for mapping graph-like worlds," in *IJCAI*, 1993, pp. 1639–1647.

[40] D. Hahnel, S. Thrun, B. Wegbreit, and W. Burgard, "Towards lazy data association in SLAM." in *ISRR'03*, 2003, pp. 421–431.

[41] J. Biswas and M. M. Veloso, "Localization and navigation of the cobots over long-term deployments," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1679–1694, 2013.

[42] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *Robotics, IEEE Transactions on*, vol. 21, no. 3, pp. 354–363, 2005.

[43] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[44] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[45] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach," in *Intelligent Robots and Systems (IROS), 2011 IEEE International Conference on*, September 2011, pp. 2014–2019.

[46] N. E. D. Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, Feb 2012.

[47] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human-robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[48] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[49] E. T. Hall, "The hidden dimension," 1966.

[50] M. K. Lapinski and R. N. Rimal, "An explication of social norms," *Communication theory*, vol. 15, no. 2, pp. 127–147, 2005.

[51] F. Zanlungo, T. Ikeda, and T. Kanda, "A microscopic "social norm" model to obtain realistic macroscopic velocity and density pedestrian distributions," *PLOS ONE*, vol. 7, pp. 1–10, 12 2012.

[52] M. Shiomi, F. Zanlungo, K. Hayashi, and T. Kanda, "Towards a socially acceptable collision avoidance for a mobile robot navigating among pedestrians using a pedestrian model," *International Journal of Social Robotics*, vol. 6, no. 3, pp. 443–455, 2014.

[53] P. A. Lasota and J. A. Shah, "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration," *Human factors*, vol. 57, no. 1, pp. 21–33, 2015.

[54] M. Joosse, A. Sardar, M. Lohse, and V. Evers, "BEHAVE-II: The revised set of measures to assess users attitudinal and behavioral responses to a social robot," *International Journal of Social Robotics*, vol. 5, no. 3, pp. 379–388, 2013.

[55] E. Pacchierotti, H. I. Christensen, and P. Jensfelt, "Evaluation of passing distance for social robots," in *ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication*, Sept 2006, pp. 315–320.

[56] A. Sardar, M. Joosse, A. Weiss, and V. Evers, "Don't stand so close to me: Users' attitudinal and behavioral responses to personal space invasion by robots," in *2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, March 2012, pp. 229–230.

[57] T. Kruse, A. Kirsch, H. Khambhaita, and R. Alami, "Evaluating directional cost models in navigation," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. ACM, 2014, pp. 350–357.

[58] S. Y. Chung and H. P. Huang, "Incremental learning of human social behaviors with feature-based spatial effects," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 2417–2422.

[59] B. Okal and K. O. Arras, "Formalizing normative robot behavior," in *International Conference on Social Robotics*. Springer, 2016, pp. 62–71.

[60] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon, "A human aware mobile robot motion planner," *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 874–883, 2007.

[61] T. Kruse, P. Basili, S. Glasauer, and A. Kirsch, "Legible robot navigation in the proximity of moving humans," in *2012 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, May 2012, pp. 83–88.

[62] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[63] F. Zanlungo, T. Ikeda, and T. Kanda, "Social force model with explicit collision prediction," *EPL (Europhysics Letters)*, vol. 93, no. 6, p. 68005, 2011.

[64] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1688–1694.

[65] S.-Y. Chung and H.-P. Huang, "A mobile robot that understands pedestrian spatial behaviors," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5861–5866.

[66] B. Kim and J. Pineau, "Socially adaptive path planning in human environments using inverse reinforcement learning," *International Journal of Social Robotics*, vol. 8, no. 1, pp. 51–66, 2016.

[67] C. Dondrup, N. Bellotto, M. Hanheide, K. Eder, and U. Leonards, "A computational model of human-robot spatial interactions based on a qualitative trajectory calculus," *Robotics*, no. 4, pp. 63–102, 2015.

[68] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *arXiv preprint arXiv:1703.08862*, 2017.

[69] C. Park, J. Ondřej, M. Gilbert, K. Freeman, and C. O'Sullivan, "Hi robot: Human intention-aware robot planning for safe and efficient navigation in crowds," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 3320–3326.

[70] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2016.

[71] ——, "Fast discovery of influential outcomes for risk-aware mpdm," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6210–6216.

[72] J. J. Gibson, *The Ecological Approach to Visual Perception.* Houghton, Mifflin and Company, 1979.

[73] E. Uğur and E. Şahin, "Traversability: A case study for learning and perceiving affordances in robots," *Adaptive Behavior*, vol. 18, no. 3-4, pp. 258–284, 2010.

[74] G. Tsai, C. Johnson, and B. Kuipers, "Semantic visual understanding of indoor environments: from structures to opportunities for action," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference on*. IEEE, 2014, pp. 373–380.

[75] M. L. Benedikt, "To take hold of space: isovists and isovist fields," *Environment and Planning B*, vol. 6, no. 1, pp. 47–65, 1979.

[76] B. Lau, C. Sprunk, and W. Burgard, "Improved updating of euclidean distance maps and Voronoi diagrams," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/lau10iros.pdf

[77] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[78] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 04 2000.

[79] A. Turner, M. Doxa, D. O'sullivan, and A. Penn, "From isovists to visibility graphs: a methodology for the analysis of architectural space," *Environment and Planning B: Planning and design*, vol. 28, no. 1, pp. 103–121, 2001.

[80] M. Batty, "Exploring isovist fields: space and shape in architectural and urban morphology," *Environment and planning B: Planning and Design*, vol. 28, no. 1, pp. 123–150, 2001.

[81] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[82] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[83] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[84] Ö. Erkent and H. I. Bozma, "Long-term topological place learning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5462–5467.

[85] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003. [Online]. Available: http://radish.sourceforge.net/

[86] L. Shi, S. Kodagoda, and G. Dissanayake, "Laser range data based semantic labeling of places," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010, pp. 5941–5946.

[87] P. Foster, Z. Sun, J. J. Park, and B. Kuipers, "Visagge: Visible angle grid for glass environments," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2213–2220.

[88] Q. Zhang, D. Whitney, F. Shkurti, and I. Rekleitis, "Ear-based exploration on hybrid metric/topological maps," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 3081–3088.

[89] F. Savelli and B. Kuipers, "Loop-closing and planarity in topological map-building," in *IROS*, vol. 2, sept.-2 oct. 2004, pp. 1511 – 1517 vol.2.

[90] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[91] E. Remolina and B. Kuipers, "Towards a general theory of topological maps," *Artificial Intelligence*, vol. 152, no. 1, pp. 47–104, 2004.

[92] T. Williams, C. Johnson, M. Scheutz, and B. Kuipers, "A tale of two architectures: A dual-citizenship integration of natural language and the cognitive map," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '17. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 1360–1368.

[93] J. J. Park, "Graceful navigation for mobile robots in dynamic and uncertain environments," Ph.D. dissertation, University of Michigan – Ann Arbor, 2016.

[94] G. Ferrer and A. Sanfeliu, "Bayesian human motion intentionality prediction in urban environments," *Pattern Recognition Letters*, vol. 44, pp. 134–140, 2014.

[95] K. Konolige, "A gradient method for realtime robot control," in *IROS '00*, vol. 1, 2000, pp. 639–646 vol.1.

[96] A. Aydemir, P. Jensfelt, and J. Folkesson, "What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 4675–4682.

[97] N. Chernov and C. Lesort, "Least squares fitting of circles," *Journal of Mathematical Imaging and Vision*, vol. 23, no. 3, pp. 239–252, 2005.

[98] G. T. Toussaint, "Solving geometric problems with the rotating calipers," in *Proc. IEEE Melecon*, vol. 83, 1983, p. A10.

[99] A. Al-Sharadqah, N. Chernov *et al.*, "Error analysis for circle fitting algorithms," *Electronic Journal of Statistics*, vol. 3, pp. 886–911, 2009.