Learning Large-Scale Topological Maps Using Sum-Product Networks

Kaiyu Zheng

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science with Departmental Honor

University of Washington
2017

Advisors
Dr. Andrzej Pronobis
Prof. Rajesh Rao

Program Authorized to Offer Degree:

Paul G. Allen School of Computer Science & Engineering

Abstract

Contents

1 Introduction

The fundamental motivation of robotics is two-fold. First, robots can take risks in the place of humans, do what humans cannot or struggle to do. Second, robots can assist humans to achieve their goals more efficiently and effectively. The continuing favor of machines in the last several decades saw an explosion of interest and investment in robotics both in commercial applications and academic research [?]. With little doubt, one of the directions of continuously heavy research is autonomous robots. In this paper, we consider a subfield of autonomous robots, which is that of mobile robots in indoor environments. Mobile robots are desirable to fulfill both aspects of the motivation, especially the second; They have the potential to provide various kinds of services, and recently attempts have been made to apply mobile robots to real-world problems such as helping older people [?], guiding passengers in airports [?], and telepresence [?]. However, these robots are still far from fully autonomous, and only perform limited actions with specially designed tasks.

To enable autonomous mobile robots to exhibit complex behaviors in indoor environments, it is crucial for them to form an understanding of the environment, that is, to gather and maintain spatial knowledge. The framework that organizes this understanding is called the spatial knowledge representation. Additionally, it is also crucial for the robots to be able to learn about the spatial knowledge representation in a probabilistic manner. In this paper, we use the Deep Affordance Spatial Hierarchy (DASH) [?], a recently proposed hierarchical spatial representation that spans from low-level sensory input layer to high-level human semantics layer. The ultimate goal of our research is to use a unified deep generative model to capture all layers of DASH, an approach fundamentally different from the traditional where an assembly of independent spatial models exchange information in a limited way [?]. Specifically, in this paper, we devote our time on learning the layer of topological map, a mainstream graphical representation of the information of places and their connectivity for a full floor map [?]. We have chosen to use sum-product networks (SPN) [?], a new class of deep probabilistic models. It is favorable for our purpose primarily because of three reasons. First, it is a deep generative model, therefore probabilistic by nature. Second, it allows tractable training and inference time with respect to the size of the network. Third, it is simple to combine multiple SPNs into one, unified SPN with interpretable layers, which matches our purpose nicely.

Learning a topological map means learning the structure of the large-scale space and dependency between places, for example, how the evidence of a group of places influence the attributes of other places. This is an important step towards planning complex actions in the environment. By itself, it is also useful for tasks such as inferring missing information of a place based on the surrounding places. In the context of DASH, it is can be used for correcting the classification results of the local environment model[1]. Although there has been considerable effort in the extraction of topological maps [?][?][?], only a few work is done to actually learn it with semantic information in the past two decades [?][?], and we have not found any work on learning topological maps probabilisticly to allow inference.

There are three main challenges in this work. First, the topological maps have varying sizes (number of places and connections), and they may be rotated arbitrarily, because the underlying environment may have different dimensions, and there is essentially no loss of dependency information if a topological map is rotated as a whole. Second, there is no existing method that uses SPN to model[2] the topological map, essentially a sparse, undirected graph. Third, there is not much available data with annotated floor plans and raw robot sensory readings with which we can create a dataset of topological maps.

We address the above challenges through our contributions. We present two methods to use SPN to model a topological map: the place grid method, and the template-based method. Both methods

[1]In DASH, above the layer of low-level sensory input, there is a layer called "peripersonal layer" which converts sensory input into robot-centric representation of the immediate surroundings, i.e. local environment. More details about DASH is given in section 3.2.

[2]In this paper, "modeling" a topological map means the same as "learning" it.

aim to deal with the variability of topological maps but with different emphasis on the dependency between places. In the place grid method, we project the topological map onto a grid of fixed size, and generate a SPN structure using random decomposition of the grid. In the template-based method, we train an SPN on a template, or a hierarchy of templates. Each template is a basic graph structure that is assumed to prevail in the topological map. For each topological map instance, we construct an instance-specific full SPN using the template SPNs as children, and do inference on the full SPN. To our knowledge, these are the first methods to use SPN to model a sparse graph. We evaluate the two methods by experiments and our results show their capability of enabling robots to do inference of environment attributes with promising output, showing the understanding of the spatial relations between places. In addition, we created a large dataset with fully annotated localization, sensory information and virtual scans, a low-level geometry representation of robot-centric environment[3]. This dataset is used to evaluate our methods.

1.1 Thesis outline

Section 2 - Related Works We review related works in the use of topological maps as a part of spatial knowledge, learning of topological maps, and recent machine learning techniques used to model graphs.

Section 3 - Background We discuss details about the theoretical background related to our work. First, we describe details of sum-product networks. Then, we present an overview of the Deep Affordance Spatial Hierarchy. Finally, we formalize the notion of topological map and describe the method we use to extract topological maps.

Section 4 - Problem Statement & Proposed Solutions We formalize the problem that we try to solve in this thesis. Then, we describe the details of the two presented methods that model topological maps using SPNs: the place grid method and the template-based method.

Section 5 - Experiments We describe the dataset, methodology, and results of our experiments.

Section 6 - Conclusion & Future Work We conclude our thesis by providing a summary of the presented methods and experiment results. Finally, we discuss improvements to be made in the future.

[3]We plan to publish this dataset to IJRR soon.

2 Related Works

The use of topological map in navigation and mapping dates back to the work by Kulpers and Byun [?] in 1991, where they first considered the use of qualitative maps to avoid error-prone geometrically precise maps. Topological maps then began to be considered as a form of spatial knowledge [?]. More recently, Beeson et. al. [?] proposed a hybrid approach of spatial knowledge representation that uses metrical representation for local environment and topological representation for large environment. Pronobis et. al. [?] outlined criteria for good spatial knowledge representation where the use of topological map to represent global environment is also promoted.

Aydemir et. al. [?] investigated the statistical properties of indoor environments by analyzing two large floor plan datasets consisting of over 197 buildings and 38,000 rooms in total. They pointed out that indoor environments are not yet well understood. They discovered that local complexity remains unchanged for growing global complexity in real-world indoor environments. This finding provides important statistical support for our template-based method discussed later. They also proposed two methods for predicting room categories for a given topological map, but their methods rely on the maintenance of a large dataset of possible graphs, and do not involve learning of the actual topological map. In contrast, our methods can learn the dependency of place categories from a limited amount of data.

Sum-product networks (SPNs), proposed by Poon and Domingos [?] are a new class of probabilistic model that guarantee tractable inference. Despite their competitive or state-of-the art results on tasks such as image completion [?], image classification [?], and language modeling [?], the advantages of SPNs have not been exploited much in robotics, besides [?]. Moreover, for neural networks, the problem of learning from data naturally structured as graphs has been on the radar for around a decade. Scarselli et. al. [?] proposed graphical neural networks, and recently there is an increase of intererst for generalizing convolutional neural networks (CNNs) beyond grid data such as images to model data in graph domains [?][?], showing promising progress. In contrast, there has been no research on using sum-product networks to learn similar data. This work offers a straightforward first step in broadening the scope of applications for SPNs.

3 Background

3.1 Sum-Product Networks

3.1.1 Definitions and Properties

SPN, proposed by Poon and Domingos [?], is a new class of probabilistic graphical models with built-in properties that allow *tractable* inference, a major advantage over traditional graphical models such as Bayesian networks. The idea is built upon Darwiche's work on network polynomial and differentials in arithmetic circuit representation of the polynomial [?]. Here, I provide the definition of SPN and several of its key properties.

Definition 3.1 (SPN)**.** [?] Let $\mathcal{X} = \{X_1, \cdots X_n\}$ be a set of variables. A sum-product network (SPN) defined over $\mathcal{X}$ is a rooted directed acyclic graph. The leaves are indicators $[X_p = \cdot]$. The internal nodes are sum nodes and product nodes. Each edge (i, j) from sum node i has a non-negative weight w_{ij}. The value of a sum node is $\sum_{j \in Ch(i)} w_{ij} v_j$, where $Ch(i)$ is the children of i. The value of a product node is the product of the values of its children. The value of an SPN is the value of its root.

We use S to denote an SPN as a function of the indicator variables (i.e. the leaves). Let $\mathbf{x}$ be an instantiation of the indicator variables, a full state. Let $\mathbf{e}$ be an evidence (partial instantiation). For a given node i, we use S_i to denote the sub-SPN rooted at i. Also, we use x_p^a to mean $[X_p = a]$ is true, and use $\bar{x}_p^a$ to mean the negation, for simplicity. We define the following properties of SPN.

Definition 3.2 (Validity)**.** An SPN is *valid* if and only if it always correctly computes the probability of evidence: $S(\mathbf{e}) = \Phi_S(\mathbf{e})$, where $\Phi_S(\mathbf{e})$ is the unnormalized probability of $\mathbf{e}$.

Definition 3.3 (Consistency)**.** An SPN is *consistent* if and only if for every product node i, there is no variable X_p that has indicator x_p^a as one leaf of the sub-SPN S_i and indicator x_p^b with $b \neq a$ as another leaf.

Definition 3.4 (Completeness)**.** An SPN is *complete* if and only if all children of the same sum node have the same scope. (The scope of an SPN is the set of variables in $\mathcal{X}$ that the indicators of an SPN are defined on.)

Definition 3.5 (Decomposability)**.** An SPN is *decomposable* if and only if the children of every product node have disjoint scopes.

The above definitions can constrain an SPN so that it is no longer an arbitrary multi-linear map from the indicators to a real number. Poon and Domingos proved that if an SPN is complete and consistent, then it is valid [?]. A more restricted theorem for validity is that if an SPN is complete and decomposible, then it is valid [?]. We will apply the latter theorem in to solve our problem, since it is easier to guarantee that the children of a product node have disjoint scopes.

Hidden Variables Given a complete and consistent SPN S and an arbitrary sum node i, we know:

$$S_i(\mathbf{x}) = \sum_{j \in Ch(i)} w_{ij} S_j(\mathbf{x}) \tag{1}$$

If $\sum_{j \in Ch(i)} w_{ij} = 1$, we can view the value of the sum node i as summing out a hidden variable Y_i, where $P(Y_i = j) = w_{ij}$. Also, in this case, the partition function $Z_S = S(1 \cdots 1) = \sum_{\mathbf{x}} S(\mathbf{x})$=1, because all of the indicators have value 1, and the product nodes and sum nodes all output 1. It follows that the value of every sub-SPN is a normalized probability. Therefore, the SPN rooted at sum node i can be viewed as a mixture model, where each child j is a mixture component with distribution $S_j(\mathbf{x})$.

3.1.2 Inference

We can perform marginal inference and MPE inference with a valid SPN in time linear to its size. This uses the same idea as Darwiche's derivation of partial differentiations in arithmetic circuits [?].

Given an SPN S and an arbitrary intermediate node i, let $S_i(\mathbf{x})$ be the value of the node on state $\mathbf{x}$. Let $Pa(i)$ be the parent nodes of i. Then,

$$\frac{\partial S(\mathbf{x})}{\partial S_i(\mathbf{x})} = \sum_{k \in Pa(i)} \frac{\partial S(\mathbf{x})}{\partial S_k(\mathbf{x})} \frac{\partial S_k(\mathbf{x})}{\partial S_i(\mathbf{x})} \tag{2}$$

If node i is a product node, $\frac{\partial S_k(\mathbf{x})}{\partial S_i(\mathbf{x})} = w_{ki}$. If node i is a sum node, $\frac{\partial S_k(\mathbf{x})}{\partial S_i(\mathbf{x})} = \prod_{l \in Ch_{-i}(k)} S_l(\mathbf{x})$. We can compute $\frac{\partial S(\mathbf{x})}{\partial S_k(\mathbf{x})}$ by first going upwards from leaves to root then going downwards from root to node i.

Marginal inference Suppose node i is an indicator x_p^a. Then, as derived in [?],

$$P(X_p = a|\mathbf{e}) = \frac{1}{S(\mathbf{e})} \frac{\partial S(\mathbf{e})}{\partial S_i(\mathbf{e})} \propto \frac{\partial S(\mathbf{e})}{\partial S_i(\mathbf{e})} \tag{3}$$

We can also infer the marginal of hidden variables. Suppose node i is a sum node which marginalizes out a hidden variable Y_i. For child j branching out from i, we can consider the value of indicator $[Y_i = j] = S_j(\mathbf{e})$. This holds because the indicators are actually real-valued [?][?], and this is important when we take the derivative with respect to an indicator [?]. Thus, we use the partial differentials derived by Darwiche [?] to obtain the following. Note that by definition, $\mathbf{E}$ and $\mathbf{Y}$ must be disjoint.

$$P(Y_i = j, \mathbf{e}) = \frac{\partial S(\mathbf{e})}{\partial S_j(\mathbf{e})} \tag{4}$$

$$\begin{aligned} P(Y_i = j|\mathbf{e}) = \frac{P(Y_i = j, \mathbf{e})}{P(\mathbf{e})} &= \frac{1}{S(\mathbf{e})} \frac{\partial S(\mathbf{e})}{\partial S_j(\mathbf{e})} \\ &= \frac{1}{S(\mathbf{e})} \frac{\partial S(\mathbf{e})}{\partial S_i(\mathbf{e})} \frac{\partial S_i(\mathbf{e})}{\partial S_j(\mathbf{e})} \\ &= \frac{w_{ij}}{S(\mathbf{e})} \frac{\partial S(\mathbf{e})}{\partial S_i(\mathbf{e})} \\ &\propto w_{ij} \frac{\partial S(\mathbf{e})}{\partial S_i(\mathbf{e})} \end{aligned} \tag{5}$$

MPE inference The task of MPE inference is to find the most likely assignment of unobserved variables given the evidence. Suppose we have evidence $\mathbf{e}$ for a set of observed variables $\mathbf{E}$, we have hidden variables $\mathbf{Y}$, and we have the unobserved input variables $\mathbf{X} = \mathcal{X} - \mathbf{E}$. We are computing $\text{argmax}_{\mathbf{x},\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\mathbf{e})$.

This can be done by replacing each sum operation with a maximization operation, such that $S_i(\mathbf{e}) = \max_j w_{ij} S_j(\mathbf{e})$. First, we compute $S_i(\mathbf{e})$ for every node from bottom to top. Next, we traverse recursively from the root to the leaves: At a sum node, we pick its child that led to the value of that sum node; At a product node, we select all of its children for traversal.

If S is consistent, it is guaranteed that there is no conflicting values among the children of a product node, which means that the MPE result is a valid instantiation of the variables. In this case, the obtained instantiation $\hat{\mathbf{x}}$ has the maximum value of $S(\hat{\mathbf{x}}|\mathbf{e})$ [?].

3.1.3 Learning Parameters

Gradient descent and backpropagation Suppose S is a valid SPN over the set of variables $\mathcal{X}$, and suppose w_{ij} represents the weight of the edge from sum node i to its j-th child. Suppose we have observations of states $\mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$. We aim to estimate the weights $\mathbf{w}$ by maximizing the log-likelihood function:

$$l(\mathbf{w}|\mathbf{x}) = \sum_{p=1}^{n} logP_{\mathbf{w}}(\mathbf{x}_p) = \sum_{p=1}^{n} log\Big(\frac{S(\mathbf{x}_p)}{Z}\Big) \tag{6}$$

where Z is the partition function that equals to $S(1 \cdots 1)$. The second equal sign holds because S is valid. Now we take the gradient with respect to a single weight w_{ij}:

$$\frac{\partial l(\mathbf{w}|\mathbf{x})}{\partial w_{ij}} = \sum_{p=1}^{n} \frac{\partial}{\partial w_{ij}} log\Big(\frac{S(\mathbf{x}_p)}{Z}\Big) = \sum_{p=1}^{n} \frac{1}{S(\mathbf{x}_p)} \frac{\partial S(\mathbf{x}_p)}{\partial w_{ij}} - \sum_{p=1}^{n} \frac{1}{Z} \frac{\partial Z}{\partial w_{ij}} \tag{7}$$

The value of the sum node i is $S_i(\mathbf{x}_p) = \sum_{j \in Ch(i)} w_{ij} S_j(\mathbf{x}_p)$. We can thus use the chain rule to obtain:

$$\frac{\partial S(\mathbf{x}_p)}{\partial w_{ij}} = \frac{\partial S(\mathbf{x}_p)}{\partial S_i(\mathbf{x}_p)} \frac{\partial S_i(\mathbf{x}_p)}{\partial w_{ij}} = \frac{\partial S(\mathbf{x}_p)}{\partial S_i(\mathbf{x}_p)} S_j(\mathbf{x}_p) \tag{8}$$

Using (2), we can compute $\partial S(\mathbf{x}_p)/\partial S_i(\mathbf{x}_p)$ by first computing the partial derivative with respect to the parents of i. Computing $\partial Z/\partial w_{ij}$ follows similar derivation. This is naturally a backpropagation process. Alternatively, we can ensure that $S(\mathbf{x}_p)$ to be a normalized probability by renormalizing the weights after each full update, so that we can discard the Z in the derivation.

Hard EM As discussed previously, a sub-SPN rooted at sum node i in an SPN S can be viewed as a mixture model. Poon and Domingos found that their EM method does not work well in practice, so they opt to use MPE inference instead of the marginal inference in the E step, and changes the M step accordingly. The algorithm can be described as follows.

(1) In E step, compute MPE inference:

$$j^* = \operatorname*{argmax}_j P(\mathbf{x}_p, Y_i = j) = \operatorname*{argmax}_j \frac{\partial S(\mathbf{x}_p)}{\partial S_j(\mathbf{x}_p)} = \operatorname*{argmax}_j w_{ij} \frac{\partial S(\mathbf{x}_p)}{\partial S_i(\mathbf{x}_p)} \tag{9}$$

A count is kept for every child of the node i to record the number of times the index of that child equals to j^*.

(2) In M step, the count of child j^* increments by one.

3.1.4 Learning Structure

LearnSPN The most commonly used structure learning algorithm is LearnSPN [**?**]. This algorithm recursively divides up the training data either by rows (instances) or columns (features). A product node is added when the variables can be partitioned into approximately independent subsets. If this is not the case, the instances are partitioned into similar subsets, and sum node is added on top of the subsets. This algorithm requires a large amount of training data to yield good partition results. This is not very feasible in our problem because there is very little data for the topological maps and it is difficult to acquire a massive amount of such data.

Random-Decomposition We use a different approach in learning the SPN structure, similar to the one used in [?]. This algorithm is visualized in figure 3. Given a set of variables $\mathcal{X}$, we decompose it into random subsets multiple times. A sum node is used to model a mixture of these decompositions, summing up the output of SPNs built upon each of these decompositions. This process is done recursively for each subset until singleton. Product nodes combine the output of the SPNs on each subset to union their scopes. Weights are shared for each sum node at the same level, and edges with zero weight are pruned once the parameter learning is finished.

3.2 Deep Affordance Spatial Hierarchy

Pronobis et. al. [?] proposed a hierarchical spatial representation named Deep Affordance Spatial Hierarchy (DASH). In their work, they also discussed the properties of a desired representationI summarize these properties as follows:

(1) Minimal;

(2) Hierarchical (It allows long-term, high-level, global planning to break down to short-term, lower-level local planning);

(3) Correlating abstraction with information persistence (more dynamic means less persistent, which requires higher level of abstraction);

(4) Probabilistic;

(5) Representing the unknowns;

Guided by the above principles, the authors showed DASH, which aims to be used by deep learning frameworks. DASH consists of four layers.

(1) Perceptual layer represents the robot's raw sensor readings. In the paper, the authors used occupancy grid to represent the laser-range observations, centered at the robot's position.

(2) Peripersonal layer represents objects, obstacles, and landmarks in the space immediately surrounding the robot. The authors used a polar occupancy grid representation which is a simplification of this layer's information.

(3) Topological layer represents the graphical structure of the map that connects places together. It maintains information of the places, including the eight views of a place, and the connectivity between places. It also contains placeholders. In the paper, a topological map is constructed from first sampling places in a window centered at the robot, according to the probability distribution $P(E|G)$, where E is the variable for existance of places, and G is the perceptual[4] occupancy grid. Then, according to navigation affordance, places reachable from the robot are added. The affordance is determined by A* search over the potential field $P(E|G)$.

(4) Semantic Layer represents human concepts about the places, such as place categories. The knowledge represented in this layer can be used for complex human-robot interaction tasks. In the paper, the authors implemented a probabilistic relational data structure but the details are not shown.

When the robot is handed with a DASH, ideally it is supposed to be able to navigate around and complete tasks that involve interaction with humans. And the robot should be able to explore the environment and build up the topological layer on its own. However, it does not have the ability to model uncertainty in the environment, and infer semantics of placeholders or other latent variables.

[4]I think this should be peripersonal, from the polar occupancy grid. But there is little difference between the first two layers due to the simplification in the peripersonal layer.

To enable this, the authors used DGSM (Deep Generative Spatial Model) [?] to represent the default knowledge about the environment. Default knowledge can be understood as the knowledge that the system assumes before it is provided explicitly. In the context of the paper, default knowledge means place semantics that the model infers for all places (including placeholders) based on its training data. DGSM is an architecture that essentially connects the four layers together, with sensory readings as input, and semantics as output.

3.3 Topological Maps

A topological map is a graph that captures the overall geometry of the map as well as important places in the map. In the context of robotics, the nodes in this graph are the places accessible by the robot, and edges indicate connectivity between places, i.e. the feasibility of navigating from one place to another. Figure 11 shows examples of simplified topological maps.

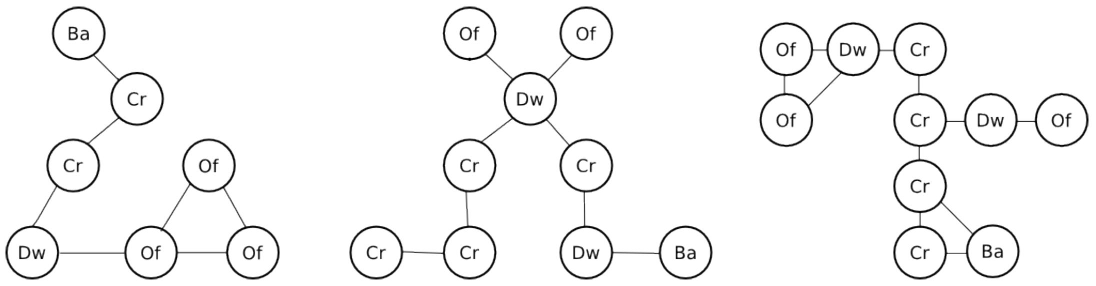

Figure 1: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).

Pronobis et.al. [?] propose a framework of spatial knowledge representation, called Deep Affordance Spatial Hierarchy (DASH), in which topological map is used as a layer to capture the geometry of the map and connectivity between places. They describe a mapping algorithm that builds a topological map. As the robot explores an environment, the mapping algorithm expands the topological map by adding *placeholders* [?] as nodes, according to a probability formulation as follows

$$P(E|G) = \frac{1}{Z}\prod_i \phi_I(E_i)\phi_N(E_i,\mathcal{E}) \tag{10}$$

where $P(E|G)$ is the probability of placeholder locations E given robot-centered occupancy grid G which represents laser-range observations. The potential function $\phi_I(E_i)$ models the existence of placeholder at location i, denoted as $E_i \in \{0,1\}$[5]. The potential function $\phi_N(E_i,\mathcal{E})$ models the probability of placeholder E_i in the neighborhood of the set of existing places $\mathcal{E}$. It is defined as

$$\phi_N(E_i,\mathcal{E}) = \sum_{p\in\mathcal{E}} \exp\Big(-\frac{(d(i,p)-d_n)^2}{2\sigma^2}\Big) \tag{11}$$

where $d(i,p)$ is the distance between location i and place p. The key point to note here is that $\phi_N(E_i,\mathcal{E})$ promotes places that are of certain distance d_n apart from existing places. This fact is important when we describe the place grid method in section 4.2.

3.3.1 Challenges

There are several challenges in modeling topological maps due to their scale and variability. A full map of a workspace may contain dozens of nodes and hundreds of edges. The number of places in

[5]For details about how $\phi_I(E_i)$ is defined, refer to the original paper [?].

one map likely differs from that in another. Different topological maps may represent environments of different dimensions. The structure of places in one map may appear very differently compared to another. Even for similar environments, the topological maps may be rotated differently, because rotation does not lose information in a topological map.

4 Problem Statement & Proposed Solutions

4.1 Problem Statement

4.2 Place Grid Method

To address the challenge that topological maps are of different sizes and shapes, we present a simple idea which is to map them to fixed-size grids. A grid can be considered as a generic representation of topological maps. It approximates the topological map, but preserves most of the adjacency relations between places. It is straightforward to use SPN to model this grid, using the algorithm in 3.1.4, and thus indirectly model topological maps as a whole.

4.2.1 From Topological Map to Place Grid

In this section, I first define a *place* in a topological map, then define a *place grid*, and then describe in detail how can a topological map be mapped to a place grid. My definition of a place follows from the polar occupancy grid representation for local environment escribed by Pronobis and Rao in [**?**].

Definition 4.1 (Place)**.** A *place* p in a topological map is a tuple (L, V) where L is the location of p with respect to the topological map's coordinate system, and V is the robot-centric polar occupancy grid that is divided into 8 views.

An example polar occupancy grid of a place is shown in figure 2.

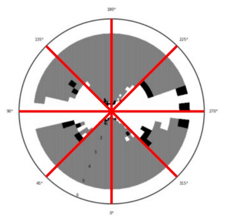

Figure 2: Example place, a polar occupancy grid divided into 8 views. (Image adapted from [**?**])

Definition 4.2 (Place Grid)**.** A place grid G with resolution r (meter per cell) is an $m \times n$ matrix where each cell contains a place.

Given a 2D place grid G with resolution r, we can map a place p in topological map M with $L = (p_x^M, p_y^M)$ to a cell in G at (p_x^G, p_y^G) simply by

$$(p_x^G, p_y^G) = \Big(\lfloor\frac{p_x^M - p_{x_{min}}^M}{r}\rfloor, \lfloor\frac{p_y^M - p_{y_{min}}^M}{r}\rfloor\Big) \tag{12}$$

where $p_{x_{min}}^M$ is the x coordinate for the place with minimum x coordinate (similar goes for $p_{x_{min}}^M$).

To avoid gaps between places mapped to the grid which disturb adjacency relations between places, we can map the topological map to a place grid with resolution at most d_n. (Recall from equation (11) that places in a topological map are separated by a predefined distance d_n.)

To deal with the situation where multiple places map to the same cell, we can do either of the following:

- Pick the place with the highest value of $\phi_I(E_p)$, as shown in equation (10).
- Create a hybrid place by sampling views from these places with probability according to $\phi_I(E_p)$.

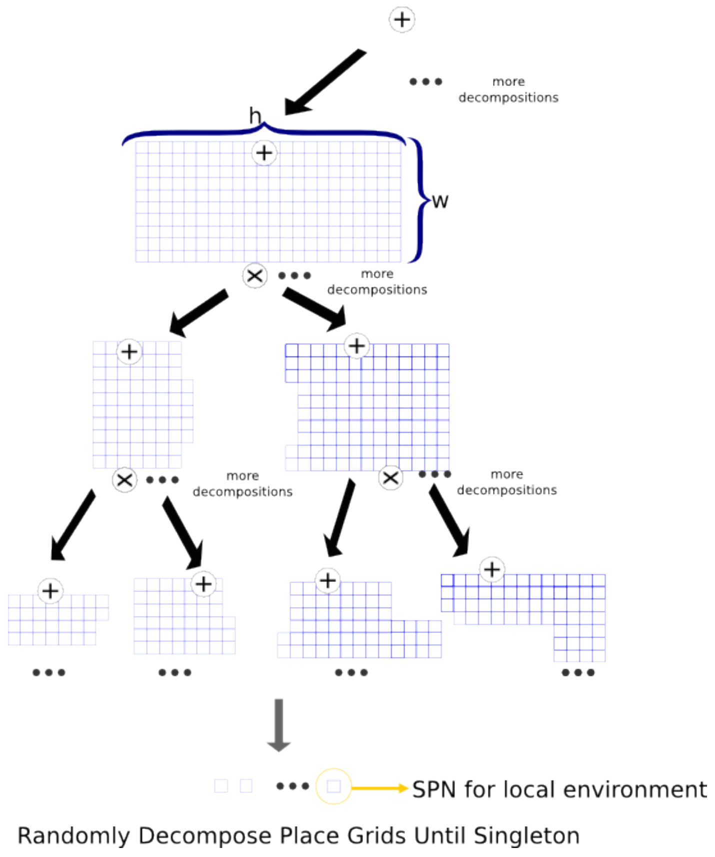

Figure 3: Process of constructing a valid SPN using the random-decomposition algorithm. Each sum node models a mixture of different decompositions of the variables in its scope. Each product node unions the scopes its children by multiplying them together. The leaf (singleton) variable, in our context, is a place which can be modeled by an SPN in DGSM [?].

Example

4.2.2 Modeling a Place Grid Using Rotation-Invariant SPN

We can use the random-decomposition algorithm to construct an SPN that models a place grid, as shown in figure 3. One issue with using the place grid is that two topological maps may represent similar spatial information but are rotated with respect to each other, and this leads them to map to drastically different place grids. Therefore, we need some way of dealing with the rotation of topological maps.

There are multiple ways to can resolve this. We could rotate all the topological maps before hand, based on the prior knowledge that human environments typically have right angle corners and straight walls. Another way is to incorporate rotation-invariance into the SPN that models the place grid. Here, we present the construction of a rotation-invariant SPN based on a place grid.

Rotation-invariant SPN We consider k number of major directions (e.g. 8). At the top level of the SPN, we add a max node with k children of sum nodes where each sum node models the place grid for a certain orientation. Suppose one child A is a grid resulted from rotating another child B by angle θ. Then, a cell (x_B, y_B) in B corresponds with (has the same value as) a cell (x_A, y_A) in A by the following transform:

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} \lfloor cos(\theta)x_A + sin(\theta)y_A \rfloor \\ \lfloor -sin(\theta)x_A + cos(\theta)y_A \rfloor \end{bmatrix} \tag{13}$$

With the above relation, we are not adding more inputs to the SPN. We simply connect a cell in the grid with the corresponding input for the place. There is a case where this relation causes (x_B, y_B) to be out of bound, which means that there are some cells in the "rotated" grid B that have no corresponding cells in A. We simply ignore the out of bound cells, and feed 1 as input for those cells that have no correspondence. One way to reduce the effect of this on the trained SPN is by using a place grid with large dimension where there are loops of cells that are supposed to be empty. The point is that these cells do not really affect the SPN's representation of the useful parts in the place grid where mapping happens.

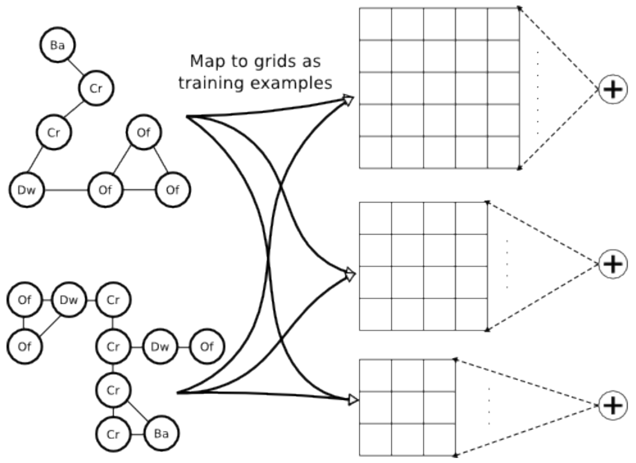

Figure 4: High-level process of the Place Grid Method. We have several place grids with different resolutions, each is modeled by an SPN using the random-decomposition algorithm. The topological maps are mapped to the grids to provide training examples for the SPNs.

At this point, we can certainly map any topological map to grids of different resolutions, and use an SPN to model each. Lower resolution means the grid represents relations of places at a more general level. This effectively results in an ensemble of SPNs (see figure 11)). If a query concerns, for example, what category should a place at a location in the topological map be given its surroundings, we can have the ensemble of SPNs to each come up with an answer and combine them in a certain way (e.g. weighted voting) to produce the final answer.

4.3 Template-Based Method

One shortcoming of the place grid method is its rely on rotation-invariant SPNs. Besides, the grid itself is in the middleground between being accurate and approximate: Althougth the grid is metric, the cells do not accurately represent the location of places. This may affect the quality of inference.

To some extent, it is unnecessary to have an accurate absolute representation of the topological map, because (1) it is hard to maintain this accuracy given the variability of topological maps, and (2) topological maps are usually used in conjunction with metric grid maps for robot planning and navigation, serving as a reference for high-level planning. Therefore, we propose a template-based relative representation of the topological map, which models some certain basic structure of a graph, and expands as needed.

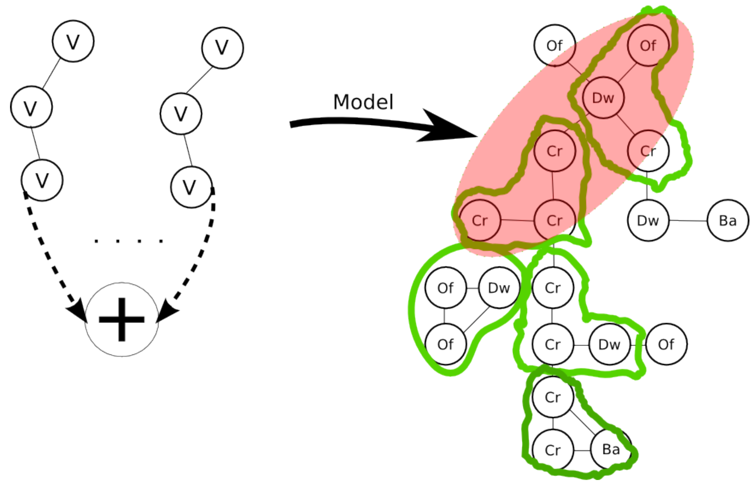

Figure 5: Illustration of the Template-Based Method. On the left side, the three node structure is the basic structure B, and the pair of these basic structures is the super structure S, on which the SPN is constructed. On the right side, a graph instance can be partitioned into a supergraph using B, and the SPN essentially models the red region on the graph.

Consider a *basic structure* $\mathcal{B}$, such as the three-node structure in figure 5. The only constraint that this structure has is the connectivity of the nodes; How the views are connected is not constrained. Therefore, this basic structure is rotation-insensitive. We can partition a given graph instance using $\mathcal{B}$. Note that $\mathcal{B}$ should be a very simple structure. One reason is that a simple structure likely occurs very often in the topological map. Another reason is that finding if a graph is a subgraph of another (subgraph isomorphism problem) is NP-complete. Linear time solution exists but only works for planar graphs [**?**], and a topological map is not necessarily planar. Examples of such basic structure are shown in figure 6. Algorithm 1 is a greedy algorithm to partition the graph when B is very simple. In this algorithm, **match_graph** is assumed to be a function that is specific to $\mathcal{B}$; it could be hard-coded. The output of this algorithm is a graph H where nodes are the basic structures and edges indicate their connectivity (see the green groups in figure 6).

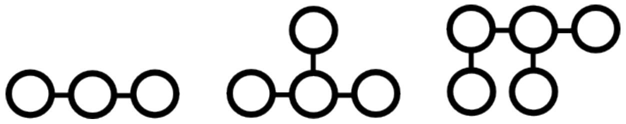

Figure 6: Some simple basic structures.

Now, consider another simple graph structure $\mathcal{S}$ that consists of $\mathcal{B}$. For example, $\mathcal{S}$ is a pair of $\mathcal{B}$ connected as shown in the red region of the topological map in figure 6. We can again partition the graph H by $\mathcal{S}$ using the same algorithm. Therefore, the given graph instance can be partitioned into a graph with nodes that represent $\mathcal{S}$'s.

We can think of $\mathcal{B}$ as the template, and $P(\mathcal{S})$ is the joint distribution of the templates defined in $\mathcal{S}$. The probability $P(\mathcal{S})$ is the *expansion model* of the template that dictates how a template expands to form a graph. We don't have to use a transition model between the templates, because there is no notion of "order" between the places in a topological map. The independence assumption is that any $\mathcal{B}$ in $\mathcal{S}$ is independent from other $\mathcal{B}$'s in the partitioned graph given the other $\mathcal{B}$'s in $\mathcal{S}$.

We can use the random-decomposition method to construct an SPN for $\mathcal{S}$, treating each template

Algorithm 1: Graph Partition by a Basic Structure

```
1  function Graph-Partition ($G, \mathcal{B}$);
   Input  : A graph $G = (V_G, E_G)$ and a graph $\mathcal{B}$ representing a basic structure.
   Output: A graph $H = (V_H, E_H)$ resulting from the partition of $G$ using $\mathcal{B}$.
2  $V_{available} \leftarrow V_G$;
3  while $V_{available} \neq \varnothing$ do
4      $v \leftarrow$ draw from $V_{available}$ randomly;
5      $V_{used} \leftarrow \mathcal{B}.\mathbf{match\_graph}(G, v)$;
6      if $V_{used} \neq \varnothing$ then
7          $V_{available} \leftarrow V_{available} - V_{used}$;
8          $u \leftarrow$ a node that represents $V_{used}$;
9          $V_H \leftarrow V_H \cup \{u\}$;
10         foreach $v' \in V_{used} \cup \{v\}$ do
11             $E_H \leftarrow E_H \cup (u, M.\mathbf{get}(v'))$;
12             $M.\mathbf{put}(v', u)$;
13         end
14     end
15     $V_{available} = V_{available} - \{v\}$;
16 end
17 return $(V_H, E_H)$
```

as a variable. The inputs of the SPN for $\mathcal{S}$ are sub-SPNs that model template $\mathcal{B}$. This will give us a valid SPN. With this method, the SPN is quite small, and we can do inference at most at the scale of S. It is definitely ok to construct an SPN for a full global map by reusing weights – We recursively partition the full global map by B and S in alternating fashion, and the weights for the SPN representing B and S are all shared.

5 Experiments

5.1 Data Collection

In support of the larger goal of our research to learn the Deep Affordance Spatial Hierarchy (DASH), we collected a large dataset of fully-annotated sensory (RGB, laser rangefinder readings, odometry), localization, and virtual scans, a low-level representation of the environment immediate to the robot. For this thesis, we utilized this dataset to generate topological maps with nodes labeled by place categories.

The dataset consists of 100 sequences and total of 11 different floor plans in three buildings in three different locations: Stockholm, Sweden, Freiburg, Germany, and Saarbrucken, Germany. Summary of the sequences is shown in table 1. For each sequence, among the types of data contained, there is a corresponding canonical map and labeled floor plan, on which the labeled localized poses with frequency of 30Hz are generated using adaptive Monte-Carlo Localization (AMCL). We use the canonical map and the canonical map poses to generate topological maps, one for each sequence. We used 1.0m as the desired separation between place nodes when generating topological maps. We trained and tested our methods mostly on the Stockholm sequences as the building in Stockholm has much larger number of rooms and bigger in scale.

	Stockholm	Freiburg	Saarbrucken
# sequences	42	26	32
# floor plans	4	3	4
Avg. # topological map nodes per sequence	103.62	66.12	-[6]
Avg. # topological map edges per sequence	159.90	104.86	-

Table 1: Details of the dataset

5.2 Methodology

5.2.1 Software

We used LibSPN [?], a Python library that implements learning and inference with SPNs on GPUs. This library provides the functionality to generate a dense SPN structure using the random-decomposition method as described in section 3.1.4. We use this functinoality to generate SPN on top of a place grid as well as on templates. We implemented ourselves weight-sharing and instance-specific full SPN construction as needed for the template-based method.

5.2.2 Experiments for Place Grid Method

Setup We converted topological maps from Stockholm sequences into place grids. We used place grids of size of 15 rows by 10 columns with resolution of 1.5m per cell. The resolution is lower than the 0.8m of desired place separation in topological map to ensure the connectivity of places when mapped to the grid, as suggested in section 4.2.1. Since the size of our dataset is not very large, we lowered the number of place categories to four: doorway, corridor, small office, and large office. so that we can better assess the learning of structure and dependency.

To verify the consistency of the place grid method, we used cross-validation by training the network on three floors and test it on the other one, for each combination of the four floors in the Stockholm dataset.

We did not implement the rotation-invariant SPN when testing the place grid method, as the canonical maps in the Stockholm dataset have the same orientation, and the topological map data

[6]We did not generate topological maps for the Saarbrucken sequences.

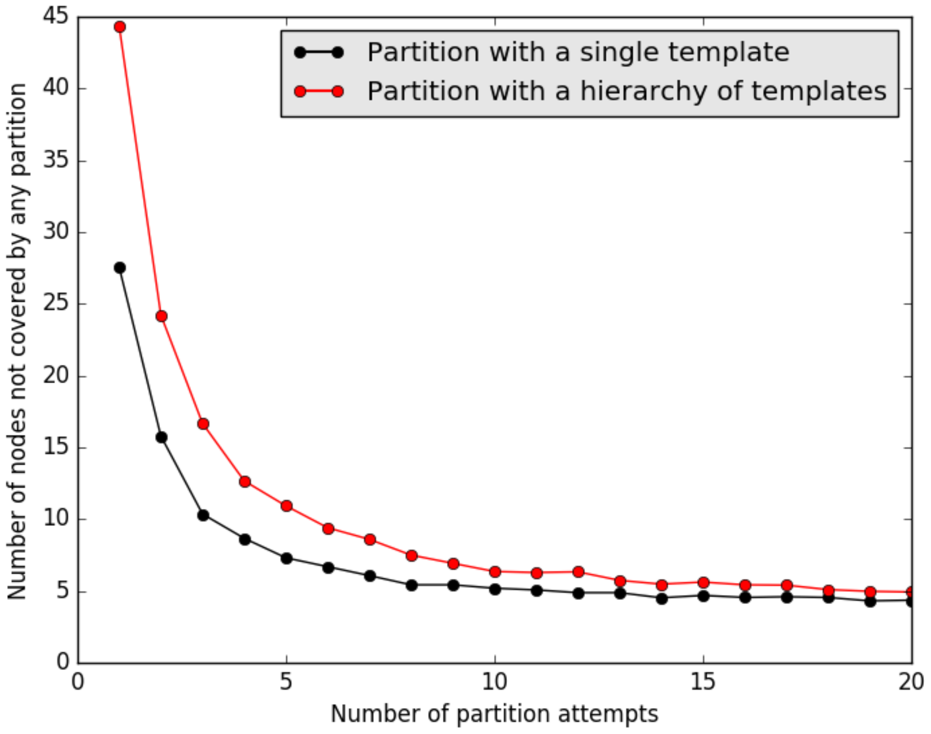

Figure 7: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).

examples are aligned well with each other. In the future, we plan to let the SPN learn the topological maps from two datasets (e.g. Stockholm and Freiburg) and test it on another (e.g. Saarbrucken), and implement the rotation-invariant SPN.

Tasks We test the method by using it for two main types of tasks, map completion and novel map structure detection. For map completion, we use the SPN to do MPE inference for the full map and the occluded part of the map. Also, we randomly occlude cells in the place grid that may or may not have been mapped directly from the topological maps, and test if the network can infer their values correctly. For novel structure detection, we construct test examples of topological maps by swapping the labels of pairs of classes, such as doorway and corridor, large office and small office, and small office and corridor.

5.2.3 Experiments for Template-Based Method

Setup As described in section 4.3, during training, we train an SPN for a single template, or a hierarchy of templates. Therefore, we created a dataset where each example is an assignment of the template variables. We do this by partitioning the available topological maps using the templates. The partition operation is done 10 times for each topological map, to capture different combinations of nodes.

During testing, we partition the given topological map test instance by our templates, and construct a full SPN for this instance. We let the weights of the sub-SPNs that model templates shared. To ensure that different combinations of nodes are captured, the full SPN is constructed over two different partitions of the test graph. The full SPN is created by using a product node over all the templates, as doing so unions the scope of the variables (i.e. nodes). To ensure completeness of the full SPN, the sub-SPN for each partition needs to cover all the nodes in the topological map. Since there are nodes that are covered by during each partition operation, we initialize another SPN with dense structure which simply takes all unused nodes as inputs with zero weights. As shown in figure **??**, the number of uncovered nodes drops quickly as the number of partitions is attempted for a graph instance.

Besides, since we are ignoring several place categories such as kitchen and meeting room, there is a significant amount of nodes that would be all labeled unknown (the default label). Therefore, we test both cases of ignoring and not ignoring the unknown class during construction of topological maps examples.

5.3 Results

5.3.1 Place Grid Method

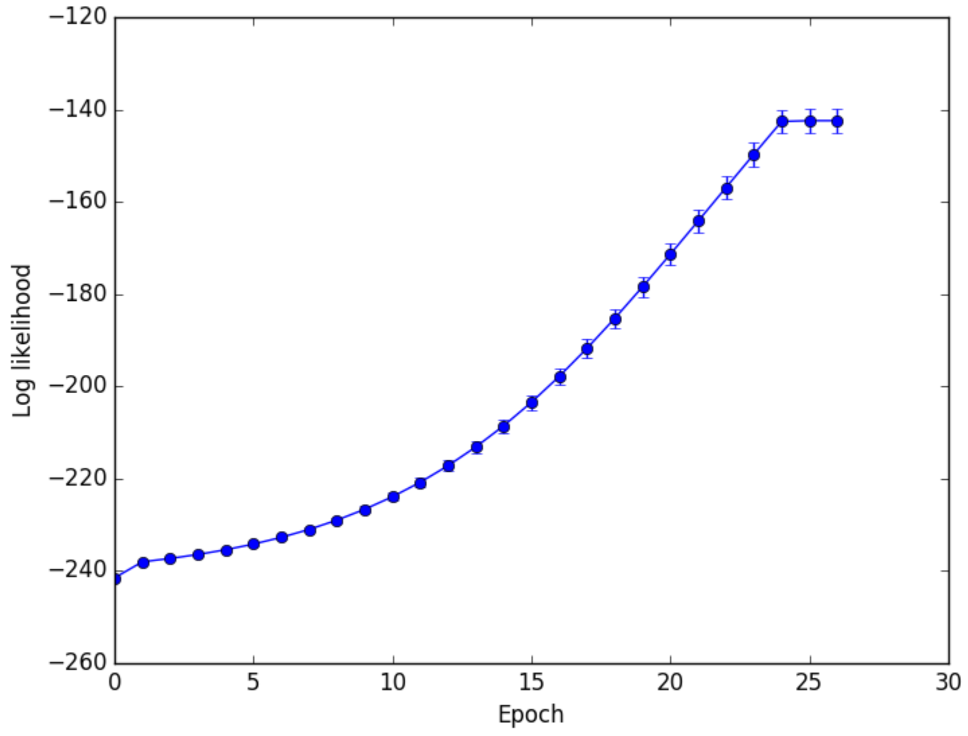

Figure 8: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).

Map completion

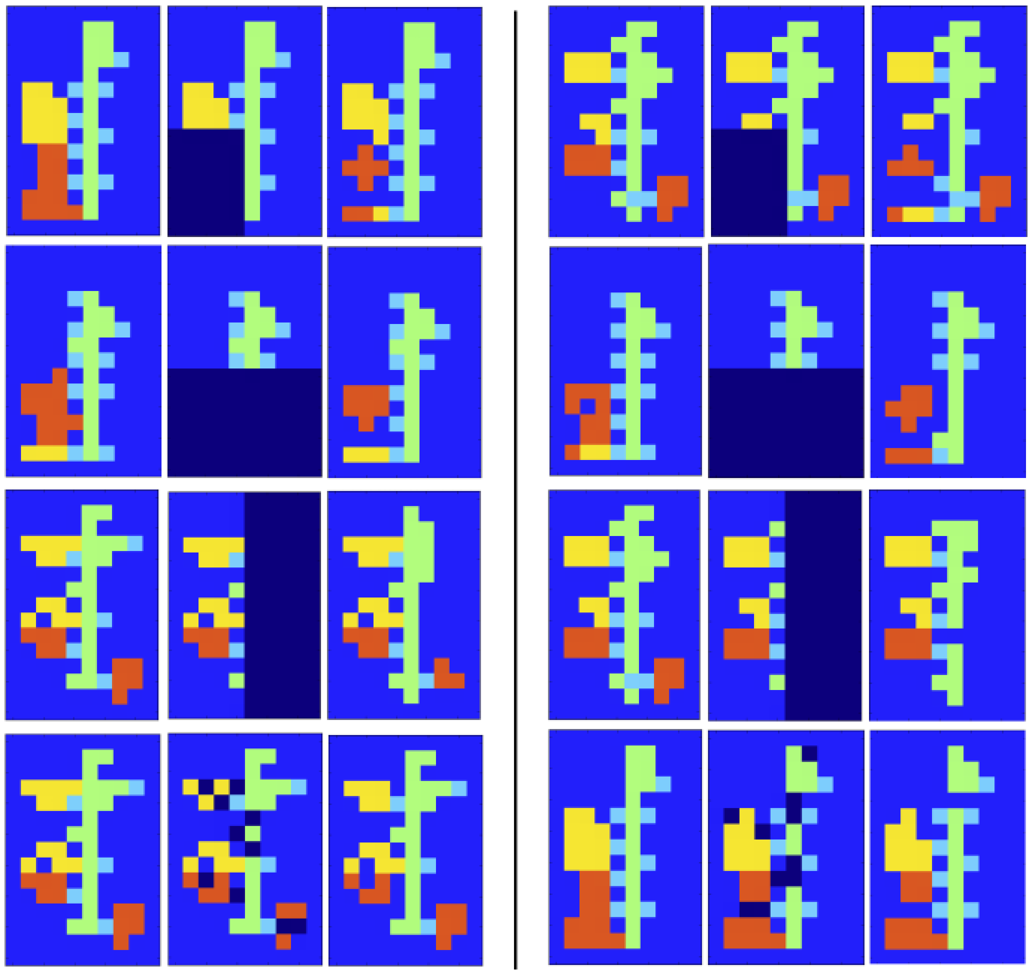

Figure 9: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).

6 Conclusion & Future Work

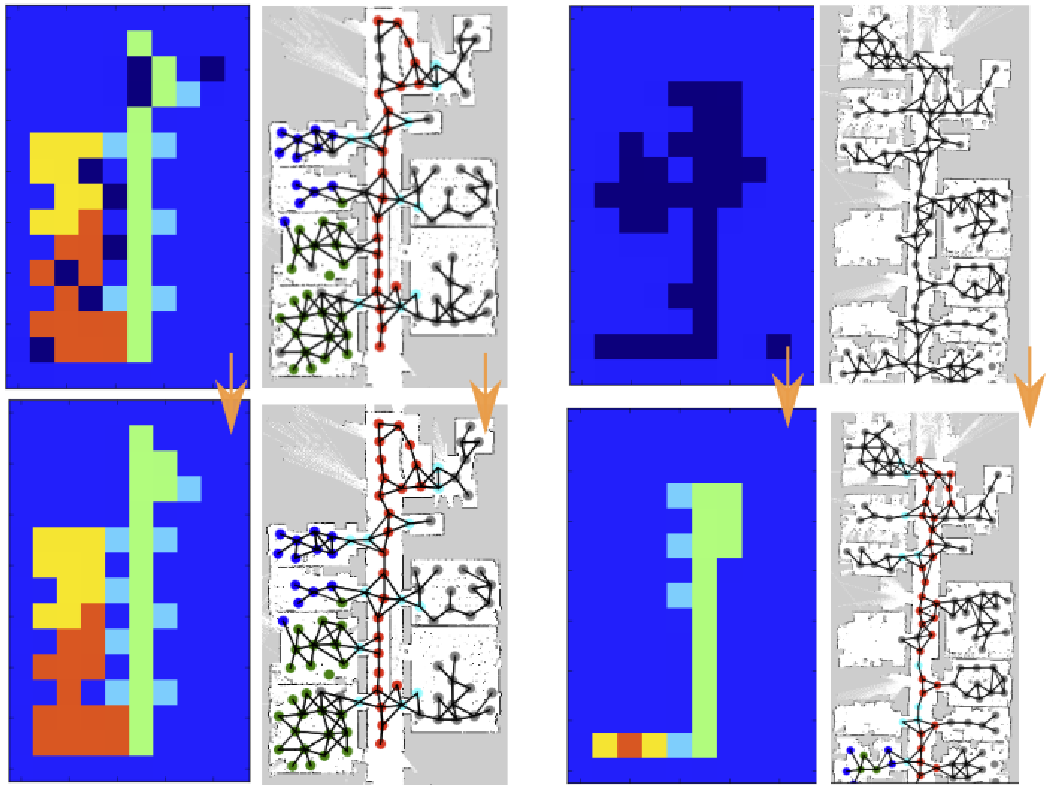

Figure 10: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).

7 Acknowledgements

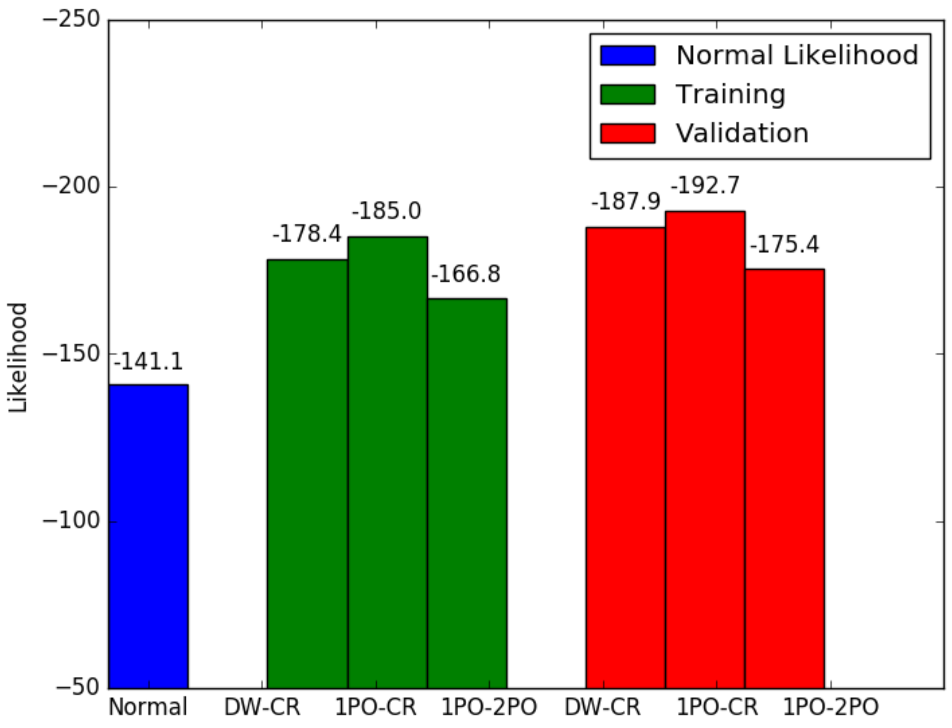

Figure 11: Simplified example topological maps. The text on a node indicates the category of the place (Ba = bathroom, Cr = corridor, Dw = doorway, Of = office).