Semantic Mapping in ROS

XAVIER GALLART DEL BURGO

Master’s Thesis at CVAP/CAS,
The Royal Institute of Technology, Stockholm, Sweden
Supervisor: Dr. Andrzej Pronobis
Examiner: Prof. Danica Kragic

August, 2013
Abstract

In the last few years robots are becoming more popular in our daily lives. We can see them guiding people in museums, helping surgeons in hospitals and autonomously cleaning houses. With the aim of enabling robots to cooperate with humans and to perform human-like tasks we need to provide them with the capability of understanding human environments and representing the extracted knowledge in such a way that humans can interpret. *Semantic mapping* can be defined as the process of building a representation of the environment, incorporating semantic knowledge obtained from sensory information. Semantic properties can be extracted from various sources such as objects, topology of the environment, size and shape of rooms and room appearance.

This thesis proposes an implementation of semantic mapping for mobile robots which is integrated in a framework called Robot Operating System (ROS). The system extracts spatial properties like rooms, objects and topological information and combines them with common-sense knowledge into a probabilistic framework which is capable of inferring room categories. The system is tested in simulations and in real-world scenarios and the results show how the system explores an unknown environment, creates an accurate map, detects objects, infers room categories and represents the results in a map where each room is labelled according to its functionality.
Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Andrzej Pronobis, for giving me the opportunity to develop my thesis in one of my passions, the robots. Thank you for your guidance, your advices, your supervision, your patience and for introducing me to robotics research.

There are many people who have made my time in Sweden rich of enjoyable moments. I would like to thank all the people of the Computer Vision and Active Perception Lab for their kindness and hospitality that made my time in KTH as if I were at home. Especially to the “innebandy crew” for showing me this sport, teaching me and for all the great moments playing every Wednesday. To my Swedish family, Nina, Jessica and Nichlas for helping me from the first day in Stockholm. Thank you for all the amazing things you have showed me and for being so kind and generous. It was a pleasure to share my time in Stockholm with my neighbours, the “Kista family”. I will not forget every moment we spent together doing shared dinners, watching movies, skating, going out, travelling, etc.

I would also like to thank all the people who have helped me during my studies in Barcelona, specially to Gerard, Henar and Javi for all the moments we have spent in class, in the laboratory, in the library and at the bar. All these years would not have been the same without you.

Last but not least, I want to thank my family; to my parents and brothers for understanding all my decisions and giving me the chance to study abroad. Thank you for all your love and support.

Finally, I want to thank Júlia, for always being by my side. For your constant love in good and bad moments, for your patience during these months and for always helping me when I needed it most.

Xavier Gallart del Burgo
Stockholm, August 2013
List of Acronyms

**ROS**  Robot Operating System  
**SLAM**  Simultaneous Localization and Mapping  
**HOG**  Histogram of Oriented Gradients  
**SURF**  Speeded Up Robust Features  
**SVM**  Support Vector Machine  
**DGM**  Directed Graphical Model  
**UGM**  Undirected Graphical Model  
**PDGM**  Partially Directed Graphical Model  
**DAG**  Directed Acyclic Graph  
**CVAP**  Computer Vision and Active Perception Laboratory
Chapter 1

Introduction

Each day, robots are increasingly more present in our daily lives. In the near future we expect them to perform more advanced actions and to exhibit human-like behaviours. To achieve that goal, we need them to reason and understand the world in a human-like fashion. Moreover, if robots understand human concepts, human-robot communication becomes possible and we can transfer them knowledge about the world. An important concept for robots working in our environments is to understand them, which is called spatial understanding. Spatial understanding is an important feature for robots to perform more complex tasks such as exploration, navigation, object search and human-robot interaction.

There are different ways in which spatial knowledge can be represented. A metric map is a well-known strategy but it is limited to geometrical information about the environment. Another spatial representation is a topological map, which segments space into discrete places that are connected to one another by paths. Our objective is to extend those representations by providing human-like conceptualization of space. In indoor environments, humans tend to perceive space in terms of discrete areas such as rooms. Rooms play an important role in spatial understanding because we label them with specific names such as “corridor”, “bedroom”, “kitchen”, “office” or “living-room”, typically according to their specific functionality: for example, kitchen is to cook, bedroom is to sleep and office is to work. The presence of certain objects in rooms conveys a lot of information about their functionality (e.g. a room where we find a microwave oven, a fridge and a cooker is a kitchen). Consequently, it is of vital importance for a robot to be able to extract this information from the environment.

Semantic knowledge can be extracted, not only from objects, but also from a wide variety of sources such as metric and topological properties, size, shape and geometry of rooms and human knowledge. Many recent advances permit robots to obtain all this information: mapping processes build maps using laser scanners, place classification systems perceive size, shape, geometry and appearance of places.
and computer vision algorithms detect objects in images. Human knowledge can be provided as an asserted information from the user and also from search engines (e.g. searching on Google a description for “kitchen”). The process of extracting semantic knowledge from an environment, combining it with other sources of knowledge and representing all in an intuitive map, is what we call semantic mapping.

In the last few years, an open source robot operating system called ROS
\(^1\) (Robot Operating System) has become more popular among robot researchers. ROS is a framework to develop robotic software that is becoming more popular among robot researchers. It is based on a peer-to-peer network of processes and it provides various tools to manage the complexity of developing a robotic system. Management tools perform tasks such as navigate the file system, create source code and define dependencies. Visualization tools monitor the communications between ROS components and show the data flow. Moreover, ROS permits programming in three programming languages: C++, Python or Lisp. Using all these features, robot researchers have created a wide number of robot applications which are integrated in ROS and are available online, some of which are: mapping, exploration, navigation, grasping, image processing and object manipulations. Another advantage of using ROS is that the same system can be easily run on numerous robot platforms.

The goal of this thesis is to create a ROS framework capable of performing semantic mapping using mobile robots. We aim to create a system that uses information from diverse sensory sources to extract spatial properties from the environment, combines the perceptions using semantic relationships and reasons about the explored areas. Our system extracts information from sensors such as laser and cameras to detect objects, extract topology features and detect doors to segment space in different rooms (e.g. a room which is connected to the corridor and contains microwave and a fridge is more likely to be a kitchen than a meeting room). These properties are combined into a conceptual map which maintains relationships between sensed information and human concepts. A probabilistic chain graph models the conceptual map with common-sense knowledge to perform spatial reasoning and to infer room categories. This allows us to jointly reason about the presence of objects, room formation and functional categories of rooms.

The system has been tested offline and online. The experimental results show how the semantic mapping algorithm extracts semantic information, creates a metric map, segments space into rooms, detects objects and combines all in a probabilistic fashion. The probabilistic framework infers spatial reasoning and creates a map where rooms are labelled with their functionality (kitchen, corridor, office and meeting room).

\(^1\)http://www.ros.org/
1.1 Related Work

The semantic mapping problem covers different fields such as mapping, localization, exploration, object recognition, geometric features extraction, place segmentation and spatial reasoning, among others. Regarding the mapping problem, there is a wide literature on mapping methods for mobile robots. To build a map of an unknown environment, a technique known as simultaneous localisation and mapping (SLAM) can be used. In [1], the mathematical problem of the SLAM is explained in detail and it presented a solution based on the Kalman-filter approach which is tested in a realistic scenario.

Topological maps are also commonly employed to represent environments. In literature we can find methods that extract the topological map in real time [2] and other approaches that generate topological maps from a previous metric map [3]. In order to exploit the advantages of both metric and topological maps, hybrid maps are proposed. Hybrid maps combine different types of maps and in [4], we can find a theoretical definition and a classification of hybrid maps. In [5], a specific implementation for hybrid maps was described. The system used a hybrid map based on two levels (global and local). The global level contained a topological map of the explored space and the local level was composed by images stored at each node of the topological map.

Semantic mapping has been widely researched in the last few years. Frameworks differ in terms of which semantic properties are extracted from the workspace. An early approach was described by Torralba et al. in [6]. It proposed a vision system that recognizes objects and places to categorize space and labels them with a name such as office, corridor or street. The space categorization proposed could be used for object search algorithms; their search could be optimized searching objects in rooms where they are typically found (e.g. starting searching books in offices).

In [7], Galindo et al. presented an approach for building a semantic map using mobile robots. The system extracts information from the environment and creates a metric and a topological map. It also uses a camera to obtain object information. It is composed by two hierarchies (spatial and semantic) where the anchoring concept is used to relate them. Figure 1.1 illustrates the two hierarchies and their connections. This structure is used to decide room categories depending on the detected objects and it also infers which objects are typically found in a specific room.

Another framework, by Zender et al. [8], included some of the concepts presented in [7] but it also added place geometry properties and human input as a source of semantic information. It created a conceptual representation of indoor environments based on multiple layers: metric map, navigation, topological map and conceptual map. A speech recognition system was used to add information in the conceptual map. In Figure 1.2 we can see the structure of that system. In [9] a
CHAPTER 1. INTRODUCTION

Figure 1.1: Spatial and conceptual hierarchies. Anchoring creates a link between sensory information and semantic models. Reproduced from [7].

A fully probabilistic representation of the space was presented. A Bayesian approach was used for space conceptualization and classification. It created a hierarchical concept-oriented representation based on detected objects. Objects were used to capture spatial semantics and places were formed as a collection of objects. Using this information, the system performed spatial abstractions to create a conceptual representation of the environment.

Figure 1.2: Structure of the system proposed in [8]. The perception components send information to three layers (metric, navigation and topological map). The reasoning part (conceptual map) takes information from the other layers as well as from human inputs. Reproduced from [8].
1.2. CONTRIBUTION OF THE THESIS

In [10], Meger et al. described a spatial-semantic map created from collected images by a mobile robot. The mapping and localization functions were performed with a FastSLAM algorithm [11]. The main improvement of this system, compared to the ones explained above, was the innovative vision system. However, it did not contain any information about topology or conceptual information. Nüchter & Hertzberg [12] proposed a system for semantic mapping where the sensor was a 3D laser scanner. A 6D SLAM system was used to represent a 3D map of the environment. Additionally, features such as walls and objects were extracted from that 3D map.

Nieto-Granda et al. [13] presented a method to create a semantic map using information provided by a human guide. A topological map was built on top of a metric map and regions were labelled according to the user information. Civera et al. [14] focused on extracting semantic information using a monocular camera. It used a camera to perform a monocular SLAM system which extracted geometric information from the environment and detected objects.

A more recent project by Pronobis and Jensfelt [15] presented a semantic mapping framework which includes more sources of information than any method described above. The system is able to extract semantic knowledge such as objects, place appearance, room geometry (size and shape) and topology of the environment. It combines all this concepts into a probabilistic chain graph model. This graph model not only permits reasoning about explored space, but also inferring concepts about unexplored areas. It uses a layered structure starting from a sensory layer that contains information such as laser scans, odometry data and images; above, a place layer contains topological information of the space and a categorical layer includes shape, object and appearance models. On top of all these layers, there is the conceptual layer that relates all the properties from lower levels. The proposed system reasons about all the extracted concepts and maintains a semantic map where rooms are categorized into eleven room categories: anteroom, bathroom, computer lab, robot lab, conference hall, hallway, kitchen, meeting room, double office, single office and professor’s office.

1.2 Contribution of the Thesis

The contribution of the thesis is a semantic mapping framework for mobile robots that is integrated in ROS. It consists of different parts (mapping, object detection and spatial reasoning) that combining them, they can perform semantic mapping. The framework is available online at [https://github.com/pronobis/rocs-ros](https://github.com/pronobis/rocs-ros). Our system is capable of segmenting an environment in different rooms and labelling them as kitchen, office, living room and corridor, for example. All our system is integrated in ROS, taking advantage of existing packages and using the same system in different robot platforms. The mapping part is based on a metric map generated by a ROS component that perform simultaneous localization and
mapping (SLAM) [16]. A topological mapping engine is created on top of the metric map and it performs room segmentation. Room segmentation is done thanks to a door detection method [17], which permits the algorithm to create rooms when a new door is detected.

In order to extract more information from the environment, two object detection methods are evaluated. The first one is called RoboEarth [18] that uses a SURF-based detectors to compare object models with incoming images. The second method is called Latent SVM [19] and the original models have been modified to make them simpler and consequently, reduce the detection time. To train object models, two image databases have been used. The two methods were successfully implemented and evaluated in an indoor environment.

A probabilistic chain graph has been implemented to combine the extracted information (topology and objects) with common-sense knowledge. This common-sense knowledge includes information such as that a monitor is more likely to be in an office than in a bathroom or that corridors connect more than one room, for example. Thus, the chain graph infers spatial reasoning and obtains whether a room is a kitchen, a corridor or an office, among others. All these results are represented in a final map that gives a perspective of the explored environment. The system is evaluated in simulation and in real-world scenarios in the Computer Vision and Active Perception Laboratory (CVAP) at the Royal Institute of Technology (KTH) in Stockholm, Sweden.

1.3 Outline

The thesis begins in Chapter 2 explaining basic information as a background for other chapters. It describes different techniques to represent spatial knowledge; it also explains the object detection problem and shows various techniques. To complete the background chapter, probabilistic graphical models are introduced. Chapter 3 presents the reasons for choosing ROS, detailing its components, tools and basic packages. Chapter 4 deals with the implementation of a semantic mapping system integrated in ROS. The structure of our framework is detailed and explained component by component. Chapter 5 explains the experimental set-up and shows the results of the experiments. As a conclusion, Chapter 6 summarizes the thesis and presents future work to improve our system.
Chapter 2

Background

This chapter explains basic knowledge about involved fields in the semantic mapping implementation. It starts describing how to represent spatial knowledge using different types of maps; it also explains the problem of detecting objects and some techniques to solve it. Then, probabilistic graphical models are introduced, defining the existing types and focusing on the relevant ones for our project.

2.1 Mapping

Mapping is a technique to create a representation of an environment. In the literature, several methods have been presented with their advantages and disadvantages. In this section three types of maps are briefly explained: metric, topological and hybrid maps. Regarding metric maps, we can find different representations and implementation such as line maps, occupancy grid maps or 3D metric maps. In this thesis we have focused on 2D maps and more specifically in occupancy grids, which are detailed in the next section.

2D Metric Map

A 2D metric map creates a two-dimensional representation of the world. A specifically type of 2D metric map is an occupancy grid map which can be defined as a pair \( M = (P, O) \) where \( P = \{p_1, \ldots, p_n\} \) is a list of cell positions \( p_i = (x, y) \) and \( O = \{o_1, \ldots, o_n\} \) is a list of values \( o_i \) that represent if a cell is occupied, free or unknown. Figure 2.1 shows an example of an occupancy grid map.

The mapping process is not a trivial problem because for building a map, a robot needs a good estimation of its location and for localization, it needs a consistent map. To know how the robot is moving during the mapping process, trajectory information of the robot can be obtained from odometry data. Odometry data estimates the change in the robot position over time. However, odometry data is sensitive to errors that are accumulated over the time. The accumulated errors can
be solved by mapping algorithms using observations from the environments (e.g. laser scans). Another problem to solve while building a map is the loop closing, which is illustrated in Figure 2.2. Subfigure A represents the real space and the trajectory of the robot during the mapping process. Before closing the loop, the map contains errors because the robot has not been localised in the map (subfigure B). In C the loop has been closed and in D the algorithm corrects the accumulated errors to create an accurate map.

The SLAM technique tries to solve the issues of creating a map. It builds a map while localizing the robot in that map. A ROS component called gmapping solves the SLAM problem using Rao-Blackwellized particle filters [21]. It acquires laser scans and odometry data from the robot and is able to create an accurate map of the explored space.

Metric maps are easy to build for local environments, but they are hard to maintain in dynamic and large-scale environments. The reason is that to maintain the level of accuracy, any single change in the world should be updated and for large spaces we need to store a large volumes of data. Moreover, a 2D metric map is a compact representation of the world and it is easy to visualize. Metric maps are also appropriate for local tasks such as navigation, local path planning and obstacle avoidance.
2.1. MAPPING

**Topological Map**

A topological map $T$ is typically defined by a pair $T = (N, E)$ where $N = \{n_1, \ldots, n_n\}$ is a set of nodes which describes discrete places and $E = \{e_1, \ldots, e_m\} \subseteq N \times N$ is a set of edges, where each edge connects two nodes. Thus, topological maps represent an environment using discrete units.

There are different ways to create a topological map such as extracting geometry properties from a metric map (e.g. using voronoi diagrams), grouping places based on their visual appearance and assigning points on top of a metric map. Figure 2.3 shows an example of how to extract geometry properties from a metric map and build a topological map using a voronoi diagram. An example of a visual based topological map is described in [22], where an outdoor topological map based on appearance properties is built for a trajectory of 1,000 km. The topological map used in this thesis is built on top of a 2D metric map. Space is segmented into nodes, defined as $n_i = (x, y)$, representing their position into the map from a defined coordinate center.

![Image of topological mapping process using a voronoi diagram](image.png)

**Figure 2.3:** Topological mapping process using a voronoi diagram. Reproduced from [23].

If topological maps are not based on metric maps, they perform better scalability in large environments than metric maps because they do not keep a high level of details. They are also easy to maintain because the stored data is only a set of nodes and edges. Moreover, global path planning tasks are much easier than in metric maps. In semantic mapping, we can extract semantic information from the topology of environments (e.g. a room that connects various rooms might be a corridor). However, metric maps are more convenient for local planning because localization and obstacle avoidance is easier. To fully exploit the advantages and minimize the disadvantages of metric and topological maps, hybrid maps are created. A hybrid map integrates different types of maps and permits to take advantage of their benefits.
2.2 Object Detection

In the process of extracting knowledge from an environment, one of the relevant information for our system is the presence of objects. Objects carry a lot of semantic information (e.g. microwave ovens are more likely to be in a kitchen than in an office and monitors are not usually in bathrooms). For that reason, it is important for us to create a robust object detection system.

A first step to create an object detector is to create a model of an object. A model can be described as a set of features that are common within a specific class of objects and different to other objects. Features define properties of a specific point in an image. To define a model we have to find those features from images and use them to create the final model. There are different types of features that we can extract from an image: local or global features. Global features are extracted from a whole image and local features are derived from a locally part of the image, based on remarkable parts.

The process of extracting local features starts extracting interest points from an image. Interest points define characteristic aspects that should be re-detected even if the object has been rotated or scaled. Once the interest points have been extracted, a descriptor defines robust features of each interest point. To detect an object, we extract the descriptor of an image, which is matched and compared with the object descriptor. On the other hand, global features build a representation of the whole patch corresponding to the object into a single vector. This vector is then compared with an image to find the modelled object. In our system we have used two methods to detect object, one is a local features-based and the other uses global features.

2.2.1 Speeded Up Robust Features (SURF)

Speeded Up Robust Features (SURF) \[24\] is an interest point detector and descriptor that is invariant to image scaling and rotation. The SURF detector relies on the Hessian matrix determinant for its good performance in computation time and accuracy. In an image \(I\), at the point \(x = (x, y)\) at scale \(\sigma\), the Hessian matrix \(\mathcal{H}(x, \sigma)\) is defined as follows:

\[
\mathcal{H}(x, \sigma) = \begin{bmatrix}
L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\
L_{xy}(x, \sigma) & L_{yy}(x, \sigma)
\end{bmatrix}
\]

(2.1)

where \(L_{xx}(x, \sigma)\) is the convolution of the image \(I\) in the point \(x\) with the Gaussian second order derivative \(\frac{\partial^2}{\partial x^2} g(\sigma)\). Interest points are localized in the image using a non-maximum suppression in a \(3 \times 3\) neighbourhood. Once the interest points are found, the next step is to initialize the SURF descriptor for every interest point. The descriptor constructs an oriented square region centred on the interest point and is divided in \(4 \times 4\) sub-regions. Then the Haar wavelet response is calculated
2.2. OBJECT DETECTION

for each square. The descriptor corresponds to the $2 \times 2$ sub-divisions, which are the sums of $dx$, $|dx|$, $dy$ and $|dy|$ (see Figure 2.4).

\[
\sum dx \\
\sum |dx| \\
\sum dy \\
\sum |dy|
\]

Figure 2.4: SURF descriptor. Reproduced from [24].

After that, the matching process compares the descriptor of the input image with a previously processed image to detect matching points. The matching process is based on comparing the distance between vectors. An example of a detected object using SURF features is shown in Figure 2.5. The SURF detector is one of the two tested methods in this thesis; it is implemented by a framework called RoboEarth, which will be detailed in Chapter 3.

Figure 2.5: Detection of an energy drink can using SURF.
2.2.2 Histogram of Oriented Gradients (HOG)

Histogram of Oriented Gradients (HOG) are feature descriptors based on edge orientation and local intensity gradients which are invariant to translations and rotations. To obtain these descriptors, an image is divided in small regions (cells) and the edge orientation or the histogram of gradient directions is accumulated. The HOG descriptor is composed of all the cell histograms after a normalization process. [25]

An object detector method using this descriptor was proposed in [25], where a HOG descriptor and a linear SVM (Support Vector Machine) classifier were used for detecting humans. In Figure 2.6 we can see an example of this method.

![Figure 2.6: Example of HOG. At the left the original image; in the middle the HOG descriptor and at the right the HOG descriptor after using a SVM classifier. Reproduced from [25].](image)

In 2010, Felzenszwalb et al. [19] presented an improved method based on HOG detectors, called Discriminatively Trained Deformable Part Models. It proposed to enrich the models created in [25] with part filters at higher resolution to detect smaller details of an object. This method uses a sliding window approach which consist in comparing the filter model at all the positions and scales of an image. Figure 2.7 represents how the sliding window approach works. The method described in [19] is the second object detector used in this thesis and it is explained in detail in the following section.

![Figure 2.7: Sliding window approach.](image)
2.2. OBJECT DETECTION

2.2.3 Discriminatively Trained Deformable Part Models

In [19], star-structured part-based models are described. They are composed of a root filter and part filters at higher resolution to detect details. All these filters specify weights for HOG features and are trained using a variation of Multiple-Instances Support Vector Machines (MI-SVM) [26] that they call Latent SVM. This system was awarded on the PASCAL object detection challenge\(^1\) in 2010.

The Latent SVM detector is based on a more basic detector called Dalal-Triggs detector [25]. Latent SVM has been extended to include part filters and to create mixture models which make this detector more accurate. An example of a two-component bicycle model is illustrated in Figure 2.8 where each row represents one component. Each component refers to a different view of a bicycle: the first component captures sideways views and the second frontal views.

![Figure 2.8: A two-component bicycle model. (a) and (d) represent root filters, (b) and (d) are part filters at higher resolution and (c) and (f) are spatial models that define the location of the different parts. Reproduced from [19].](image)

Matlab code for detection and model creation is available in [27]. There is also an adapted version (in C++) inside the OpenCV library\(^2\) which has been used during this project. Although this method provides high accuracy detecting objects, the detection time using OpenCV is too long (around 15 seconds per model) to perform real time applications. In order to reduce the detection time, modifications to the original models have been made during this project. These modifications consist in creating simpler models, that can reduce the computation time. The following sections describes the models and its training process.

\(^1\) [http://pascallin.ecs.soton.ac.uk/challenges/VOC/](http://pascallin.ecs.soton.ac.uk/challenges/VOC/)
\(^2\) [http://docs.opencv.org/modules/objdetect/doclatent_svm.html](http://docs.opencv.org/modules/objdetect/doclatent_svm.html)
Models

After modifying the original models, each object category is defined by a single filter on HOG. Thus, the complexity of our model has been reduced considerably. The model filter $F$ is an array of weight vectors and to evaluate its response at different scales and positions, a feature pyramid $H$ is used. The response of that process is a list of scores (also called confidence levels), which indicates how probable is that the object model has been detected.

“Let $F$ be a filter $w \times h$, $H$ be a feature pyramid, $p = (x, y, l)$ specify a position $(x, y)$ in the $l$th level of the pyramid. Let $\phi(H, p)$ denote the vector obtained by concatenating the feature vectors in the $w \times h$ subwindow of $H$ with top-left corner at $p$ in row-major order.” [19] The score of $F$ at any position $p$ is:

\[
\text{score}(p_0, \ldots, p_n) = \sum_{i=0}^{n} F'_i \cdot \phi(H, p_i),
\]

(2.2)

where $F'$ is the vector obtained by concatenating the weight vectors in $F$ in row-major order.

Given the scores for all the positions, the detection of an object is done by thresholding these scores. If we obtain a score above the threshold, we have a possible detection and we can know where the object located is in the image. A visual description of the detection process is presented in Figure 2.9: an input image is processed into a feature map; the feature map is compared with the model filter computing the dot product. The result of this operation is the response, where higher scores are represented by brighter pixels.

Figure 2.9: Detection process using a single filter. Reproduced from [19].
2.3 Probabilistic Graphical Models

Training Process

The aim of the training process is to obtain a vector of model parameters $\beta$ that define our model. Consider a classifier which scores an example $x$ with:

$$f_{\beta}(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z),$$

(2.3)

where $z$ are latent values and $Z(x)$ defines all the possible latent values for $x$. The model parameters are trained analogy to classical SVM, minimizing the latent SVM function:

$$\frac{1}{2} \| \beta \|^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i f_{\beta}(x_i)),$$

(2.4)

where $\max(0, 1 - y_i f_{\beta}(x_i))$ represents the standard hinge loss, $C$ is a control constant and $y_i \in \{-1, 1\}$.

The training process requires a large number of images to obtain good models. It needs a set of positive images with bounding boxes; bounding boxes describe where the object is placed in an image and permits the training process to select this part of the image. Negative images are also needed during the process; a negative image means that the object is not placed in that image. To obtain all these images, two databases have been used: PASCAL database and LabelMe. In Section 5.3.2 the created models are explained in detail.

2.3 Probabilistic Graphical Models

In order to create a probabilistic semantic mapping system, graphical models are going to be used to integrate discrete variables corresponding to spatial concepts to reason about them and infer probabilities about room categories. During this thesis we have used a probabilistic chain graph which has been later converted into a factor graph with the purpose of using inference methods.

A graphical model is a data structure that is able to represent complex probability distributions into a set of nodes and edges. The nodes $\mathcal{X} = \{X_1, \ldots, X_N\}$ represent the variables and the edges $\mathcal{E}$ connect pair of nodes $X_i, X_j$ and define the probabilistic interaction between variables. The edges can be either directly $X_i \rightarrow X_j$ or indirectly $X_i - X_j$. Another important characteristic of the graphical models is whether they are cyclic or not. A cyclic graph is when there is a path $X_1, \ldots, X_K$ where $X_K = X_1$, that means that we can make a loop that ends in the same node as the origin of the loop. If there is not a cyclic path, the graph is called acyclic. All these features define different types of graphical models:

- **Directed graphical models** (DGMs) are graphs which only contain directed edges (see Figure 2.10a) which represent condition independence properties.
In that kind of graph, given an edge \((X_i \rightarrow X_j)\), \(X_j\) is the child of \(X_i\) and \(X_i\) the parent of \(X_j\). When these graphs have no cyclic path, they are called Directed Acyclic graph (DAG), also known as Bayesian Networks.

- **Undirected graphical models** (UGMs), also called Markov random fields, have only undirected edges as it is shown in Figure 2.10b. Given an edge connecting \((X_i, X_j)\) both \(X_i\) and \(X_j\) are neighbours. Edges represent associative relations between variables without indicating causality. They are better for inference tasks and for representing independence relations between variables. A graph called tree is an UGM that is also acyclic.

- **Partially directed graphs** (PDGs) : as illustrated in Figure 2.10c, these graphs admit both directed and undirected edges. If a PDG is also acyclic, it is called partially directed acyclic graph (PDAG) or Chain graph \([28]\). The advantage of these graphs is that combining directed and undirected edges, they can represent causal hypotheses, associative relations and conditional independence. The joint density \(f\) in a factor graph that satisfies the Markov property can be written as \([28]\):

\[
 f(x) = \prod_{\tau \in T} f(x_\tau | x_{pa}(\tau)), \tag{2.5}
\]

where \(\tau\) represents a node whose parents are \(pa(\tau)\). That comes from an outer factorization of the chain graph converted into a directed acyclic graph where \(X_\tau\) represents random variables for all the chain components \((T)\). Each factor \(f(x_\tau | x_{pa}(\tau))\) can be written as:

\[
 f(x_\tau | x_{pa}(\tau)) = \frac{1}{Z(x_{pa}(\tau))} \prod_{\alpha \in A(\tau)} \phi_\alpha (x_\alpha), \tag{2.6}
\]

where \(A(\tau)\) is a set of vertices in the undirected graph \(G_{\tau \cup pa(\tau)}\), normalized by the factor \(Z\), that every pair of vertices are connected by an edge.

Figure 2.10: Types of graphical models.
2.3. PROBABILISTIC GRAPHICAL MODELS

2.3.1 Factor Graph

A factor graph (see Figure 2.11) is an undirected graphical model that has two types of nodes: one to represent variables and another to represent factors (functions) which influence the variable nodes that are neighbours. Variables and factor nodes are joined by edges. These edges can only connect a variable node with a factor node and never two nodes of the same type.

Figure 2.11: Example of a factor graph.

Factor graphs are useful to calculate marginal probabilities efficiently and exactly. To calculate them we can use the Sum-Product algorithm, also known as Belief Propagation [29]. The algorithm is based on passing messages between nodes and computing the desired marginal probabilities. To deal with factor graphs a free and open source library is proposed by Mooij [30]. It permits the user to describe a graphical model and use a large number of inference methods such as brute force enumeration, junction-tree methods, Fractional Belief Propagation [31], Loopy Belief Propagation [29], Conditioned Belief Propagation [32], Mean Field Approximation and MAP inference.

Factors graph can be obtained from the different types of graphs described above: directed, undirected and chain graph. The process is to assign the distributions that represent the edges to the factor nodes. In Figure 2.12 we can see the process of converting a directed and an undirected graph to a chain graph. In Figure 2.13 it is shown the conversion used during this thesis that converts a chain graph to a factor graph. This conversion only works for a certain type of chain graphs as the one represented here.

In our system, a factor graph has been used to model the spatial properties detected by the robot. The graphical model enables the system to reason about the environment in a probabilistic fashion. Using the Loopy Belief Propagation algorithm we can infer marginal probabilities for our variables. Section 4.3 explains in detail the structure of the graphical model used in this thesis.
Figure 2.12: Conversion of a directed graph to a factor graph (a) and an undirected graph to a factor graph (b). Reproduced from [33].

Figure 2.13: Conversion of a chain graph to a factor graph.
Chapter 3

Robot Operating System (ROS)

The Robot Operating System (ROS)\(^1\) is an open source robot operating system that allows to create software for robots.\(^2\) It is developed by Willow Garage\(^3\) and was based on Switchyard, which was a part of a project called STanford Artificial Intelligence Robot (STAIR)\(^2\) and was written by Morgan Quigley at Stanford. The semantic mapping system, which is the aim of this thesis, has been designed to fully run on ROS. This robot operating system provides a standardized API and communication interface for robotic components, and its main features are:\(^4\)

- Hardware abstraction for typical robotic hardware
- Low level device control
- Implementation of commonly-used functionalities
- Message passing between processes
- Package management
- Libraries and tools for writing, building and running code

There are various advantages that make ROS an appropriate framework for robots. The first one is that it is easy to integrate in different robots due to its hardware abstraction. Another advantage is that “Willow Garage is strongly committed to developing open source and reusable software”\(^5\) and code from universities, companies or users is available. Currently, ROS has software for mapping, navigation, exploration, object recognition, motion planning for arms, simulation and machine learning, for example. ROS is flexible since it offers the possibility to program in three programming languages: C++, Python or Lisp.

\(^1\)http://www.ros.org/
\(^2\)http://stair.stanford.edu/
\(^3\)http://www.ros.org/
\(^4\)http://stair.stanford.edu/
3.1 ROS Concepts

There are three levels of concepts in ROS. The first one is how the different files are organized on the computer (Filesystem Level), the second is the components of the peer-to-peer network (Computational Graph Level) and the last is how users share software to allow everyone to use it (Community Level).

Filesystem Level

The files organization is mainly divided in two important units called packages and stacks:

- Packages: the lowest and the main units of the ROS file system. Packages contain source code, libraries, executables (nodes), configuration files and launch files.
- Stacks: a stack is a group of packages that are combined to perform high-level functionalities.

Another important file in ROS is the manifest. They can belong to a package or a stack and they describe general information about a specific package or stack, such as a brief description of what it does, its type of license and the dependencies with other packages. These dependencies are important because usually packages depend on other packages and they must be installed to work perfectly.

Computational Graph Level

The Computational Graph level describes the infrastructure that ROS components use to communicate. It is based on a peer-to-peer network of processes (nodes) and it is composed of various units:

- Nodes: they are executables in the ROS system and carry out the computational part. They can be connected to other nodes using topics or services.
- Messages: they are composed of primitive types of data such as integers, booleans or arrays and are the data transmitted between nodes.
- Topics: nodes share information passing and receiving messages. These messages are published with a special name called topic. Thus, nodes can publish or subscribe to a specific topic to get the desired messages. Many nodes can be subscribed to the same topic to receive the messages that they need.
- Services: using services, two nodes can communicate using a request-response method. When a node wants to call a service, it has to call that service and wait for the reply. Once the service node gets the call, it answers to the calling node with the requested data. Unlike the communication via topic, only two nodes are involved in this communication.
3.1. ROS CONCEPTS

- Master: it enables the ROS network to work and is the first process that should be run when we are using ROS. Its function is to register all the nodes that are executed and the existing topics and services. This is essential to interconnect nodes and to monitor all the units in the system.

In Figure 3.1 a basic communication scenario in the ROS Graph level is represented. There are two nodes running, node 1 is publishing messages on a topic and the second node is subscribed to that topic to get the messages. The other way of communication is calling the service provided by the node 2. Node 1 calls the service and node 2 replies with the requested data.

![Figure 3.1: ROS communication infrastructure. Reproduced from [36].](image)

Community Level

An important feature of ROS is the community that shares software and code, making ROS one of the biggest robot communities. There are different ways to obtain ROS resources. ROS distributions (see table 3.1) are a compilation of stacks that one can easily install by using `apt-get install`. There are hundreds of repositories (usually using svn\(^3\) or github\(^4\)) where one can find software for robot utilities. The main source of information is the ROS Wiki \([36]\) where one can find information and tutorials of the available packages. There is also a ros-users mailing list and a web page \([\text{answers.ros.org}]\) where users share questions, answers and comments.

<table>
<thead>
<tr>
<th>Name of distribution</th>
<th>Release date</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROS Box Turtle</td>
<td>March 2, 2010</td>
</tr>
<tr>
<td>ROS C Turtle</td>
<td>August 2, 2010</td>
</tr>
<tr>
<td>ROS Diamondback</td>
<td>March 2, 2011</td>
</tr>
<tr>
<td>ROS Electric Emys</td>
<td>August 30, 2011</td>
</tr>
<tr>
<td>ROS Fuerte Turtle</td>
<td>April 23, 2012</td>
</tr>
<tr>
<td>ROS Groovy Galapagos</td>
<td>December 31, 2012</td>
</tr>
</tbody>
</table>

Table 3.1: ROS Distributions.

\(^3\)http://subversion.apache.org/
\(^4\)https://github.com/
3.2 ROS Tools

ROS provides tools to facilitate the implementation of our robot system. Its tools permit the users to simulate, launch several ROS components at the same time and visualize the data flow between components. Some of these tools are explained below.

Launch Files

In a ROS project, one may want to run various ROS nodes at the same time to perform more complex systems. ROS has a tool called roslaunch that allow users to run numerous nodes, set configuration parameters of each node, rename the default topic names and even change the name of the node. The purpose of this is to easily set up the global system.

In Listing 3.1 we can see a basic example of a launch file. To start a node we define its package “pkg=”, its name into that package “type=” and we can change its default name “name=”. Moreover, we can also configure different node parameters.

```
<launch>
  <!-- ROSARIA: start the robot Pioneer 3-DX -->
  <node pkg="ROSARIA" type="RosAria" name="RosAria" output="screen"/>
  <!-- Start Gmapping -->
  <node pkg="gmapping" type="slam_gmapping" name="slam_gmapping"/>
  <!-- Run Rviz tool -->
  <node pkg="rviz" type="rviz" name="rviz" output="screen"/>
</launch>
```

Listing 3.1: Example of Launch File.

Rosbag

Rosbag\(^5\) is a set of tools for recording and playing data from ROS topics. The data is stored into bag files and command-line tools for working with bags are available. Rosbag avoids deserialization and reserialization of messages and its main tools are:

- **rosbag info**: it displays the contents of bag files such as recorded topics, start and end time, number of messages, frequency and statistics of compression.
- **rosbag record**: it writes the contents of all the messages published on those topics that we want to record and the information is stored in a .bag file.
- **rosbag play**: it reads a bag file and publishes the information on ROS topics in a time-synchronized fashion. The ROS system is able to use this information as if it was in real-time.

\(^5\)http://www.ros.org/wiki/rosbag
3.2. ROS TOOLS

During this thesis, rosbag files have been created to adapt information from a database called COLD-Stockholm and to make it compatible with ROS (for more information see Section 5.1.1).

**RVIZ**

RVIZ is a 3D visualization tool for ROS that allows to visualize maps, robots, objects, laser data, images from cameras, point clouds and markers. RVIZ provides a simple interface to choose the information that we want to be displayed. Figure 3.2 shows an example of the RVIZ interface. On the left we can see all the elements that RVIZ is displaying. Each element has to be configured with the name of a specific topic. Then RVIZ subscribes to that topic and visualizes its information. On the middle there is the main window where all the information is displayed and on the left we can choose tool properties and change between different views. RVIZ also permits the user to create plugins to add new display capabilities.

![Figure 3.2: RVIZ simulator.](image)

**Stage**

Stage is a robot simulator that provides sensory information from a virtual world. It permits the users to simulate an environment defined in a .world files. A world file describes the geometry properties of the environment, robots and sensors. Stage provides various models of sensors, including laser range finder, bumpers, grippers and cameras, among others. More information about the .world file syntax and the available models can be found in [http://playerstage.sourceforge.net/doc/Stage-3.2.1/](http://playerstage.sourceforge.net/doc/Stage-3.2.1/).

The ROS node for Stage simulates the defined environment and publishes the virtual information from sensors into ROS topics. Odometry information of the
CHAPTER 3. ROBOT OPERATING SYSTEM (ROS)

robot is published on the topic “odom”, laser scans on the topic “base_scan” and
the real position of the robot in the map is published on “base_pose_ground_truth”. This information can be used for example to simulate the process of building a map, as it is explained in Section 5.2. In Figure 3.3 an example of how Stage simulates a world is shown.

![Stage simulator](image)

**Figure 3.3:** Stage simulator.

3.3 ROS Packages

As mentioned before, ROS has a great number of packages/stacks that are available to use. This section aims to explain in detail the most relevant packages that we have used in the semantic mapping implementation.

3.3.1 Coordinate Frames

A coordinate frame is an important concept in ROS. Any robot may have various components such as a laser, camera, sonar or arms and they may have a coordinate frame attached. Many ROS algorithms require to keep track of all these coordinate frames. Let us explain this with an example. Consider a simple mobile robot with a laser on top of it (see Figure 3.4a). We may define two coordinate frames, one for the robot and another for the laser. Let “base_link” be the coordinate frame attached to the center of the mobile base and “base_laser” the one attached to the laser, which is 10 cm forward and 20 cm above the center of the robot. Imagine that the laser has data about how far is a wall and we want to know how far is the robot from that wall. We must apply to the incoming data an inverse translation of (x: -10 cm, y: 0 cm, z: -20 cm).

The tf package is able to track and maintain the relationship between multiple coordinate frames. Its role is to provide tools and functions to define all the coordinate frames of our robot and to transform data from one frame to another. This information may be needed for ROS packages. A tree structure (see Figure 3.4b) is created
3.3. ROS PACKAGES

with all the coordinate frames of our system and all the transforms are published via the topic “tf”. Thus, any package which need to know the relationship between coordinate frames can listen the topic “tf” and obtain the needed information.

(a) Simple robot with “base_link” and “base_laser” frames.  
(b) Tree structure.  

Figure 3.4: Basic coordinate frames in a robot. Reproduced from [36].

3.3.2 ROSARIA

ROSARIA\textsuperscript{7} is a ROS wrapper package that provides communication with MobileRobots platforms\textsuperscript{8}, in our case, a Pioneer 3-DX has been used. It permits the system to send velocity commands, get odometry information, read bumper states and read data from a sonar. It uses a C++ library for mobile robots called ARIA\textsuperscript{9} (Advanced Robot Interface for Applications). Figure 3.5 illustrates the topics (in black) and the messages (in blue) that ROSARIA uses to communicate with ROS.

In order to move the platform, messages of the type “geometry_msgs/Twist” have to be sent to the topic “cmd_vel”. To move the platform with a joystick, the Joy\textsuperscript{10} node can be used. This node reads the state of a joystick and publishes its information to “cmd_vel”.

\textsuperscript{7}http://www.ros.org/wiki/ROSARIA  
http://www.mobilerobots.com/  
\textsuperscript{8}http://robots.mobilerobots.com/wiki/ARIA  
\textsuperscript{9}http://www.ros.org/wiki/joy
3.3.3 Gmapping

This package provides SLAM (Simultaneous Localization and Mapping) based on laser and odometry data. A node `slam_gmapping` creates a 2D occupancy grid map (see Section 2.1) and estimates its position in the map [21]. The map can be obtained listening the topic “map” or calling the service “dynamic_map”. To run the node we need to publish odometry data, laser information and the relationship between coordinate frames.

Odometry data is not published directly in a topic, but it is provided by publishing the relationship between the odometry frame and the base of the robot. Then `gmapping` will publish the map on the topic “map” and it can also be asked via service. Figure 3.6 gives us an overview of this node.

![Figure 3.6: Overview of the slam_gmapping node.](image)

The gmapping node requires the relation between robot coordinate frames. It needs two input transformations that the user have to configure and it also provides the estimated pose of the robot using a transformation from the origin of the map:

- **Required transforms:**
  - Transform between the frame of the incoming laser scans (`base_laser`) and the base of the robot (`base_link`).
  - Relation between the `base_link` and the odometry frame (`odom`).

- **Provided transforms:**
  - Gmapping provides the current pose of the robot, publishing the relation between the map frame (`map`) and the odom frame (`odom`).

In order to complete the action of creating a map, a node called `map_saver` can be used to save the map into a file. It creates an occupancy map file (into a PNG image) where white pixels mean free space, grey pixels are unknown space and black ones are occupied. An example of a 2D map obtained using `gmapping` and `map_saver` is shown in Figure 3.7.
3.3. ROS PACKAGES

Figure 3.7: Example of an occupancy map using gmapping and map_saver.

3.3.4 RoboEarth

RoboEarth [38] is a ROS stack that provides algorithms for object detection and object modelling. In Figure 3.8 we can see the structure of RoboEarth, which is divided in three main parts: object scanning, object detection and the RoboEarth database.

Figure 3.8: RoboEarth structure. Reproduced from [18].
CHAPTER 3. ROBOT OPERATING SYSTEM (ROS)

Object Scanning

The object scanning part is able to create models of objects. It uses a Microsoft Kinect camera to record an object, which is located in a marker template. The process of creating a model consists in placing the object in the marker template and rotating it 360°. The node `re_object_recorder` uses the point clouds obtained from Kinect and the markers in the template to create a 3D model. In Figure 3.9 we can see the process of modelling a box.

![Box in the marker template.](image1)
![3D model.](image2)

Figure 3.9: Modelling process in RoboEarth.

Object Detection

After creating models of objects, the object detection part can be used. It is divided in three packages:

- **re_vision**: it contains a node called `ObjectDetector`, which is capable of detecting objects using SURF. This node reads images from a monocular camera and returns the pose of the object and its location in the image.

- **re_kinect_object_detector**: this method uses point clouds from Kinect and returns the location of the object detected.

- **re_object_detector_gui**: it is an user interface to interact with all the available options in RoboEarth such as detecting objects using one of the two methods explained above, downloading models from the RoboEarth database and loading them into the system.

Communication

The `re_comm` package is an interface to communicate with the RoboEarth's database. In that database we can find hundreds of models, which have been created by other users and we can download and test them in our system.
3.3. ROS PACKAGES

Camera Drivers

In ROS, there are packages that contain driver nodes for cameras. These nodes configure a camera and publish the images into a ROS topic. Our robot has three cameras: a Grey Chameleon CMLN-13S2C, a Prosilica GC1380c and a Microsoft Kinect. For each camera we have to choose the appropriate ROS package:

- Kinect: a node called `openni_node` launches the Kinect camera and publishes its RGB images on the topic `rgb/image_raw`, its depth image in `depth/image_raw` and the infra-red in `ir/image_raw`.

- Chameleon CMLN-13S2C: the package `camera1394` has a node that publishes obtained images on the topic `camera/image_raw`.

- Prosilica GC1380c: similar to the previous camera, a node called `prosilica_node` publishes images on the topic `camera/image_raw`.

3.3.5 OpenCV

OpenCV\textsuperscript{11} is a popular computer vision library. It is released under a BSD license and includes code for a great number of methods for image and video processing. According to the OpenCV web page, it has more than 47 thousand active users. Its capabilities for image processing make OpenCV an important library to include in our system. For example, in this thesis it has been used to detect objects.

ROS contains a specific stack (`vision_opencv`) that provides bridge functions between ROS messages and OpenCV data. To transform ROS messages to OpenCV images, we can create a node that listen to messages which contain images; then we can call a function such as `cv_data = cv_bridge::toCvCopy(image_msg, encoding)` which transforms a ROS image (`image_msg`) to OpenCV data (`cv_data`) according to the defined encoding. On the other hand, to convert an OpenCV image to ROS message we can use the function `toImageMsg (ros_image)`. Figure 3.10 illustrates the bridge process between ROS and OpenCV.

\textbf{Figure 3.10:} Bridge between ROS and OpenCV. Reproduced from [36].

\textsuperscript{11}http://opencv.org/
Chapter 4

Semantic Mapping Package for ROS

This chapter deals with the implementation of a semantic mapping system for mobile robots in ROS. We can define semantic mapping as the process of extracting semantic information from the environment, reasoning about it and representing it. The implementation is available at [https://github.com/pronobis/rocs-ros](https://github.com/pronobis/rocs-ros). The semantic mapping algorithm used in this thesis is based on the framework presented in [15]. Our system focuses on two spatial properties: objects and the topology of the environment. Objects are meaningful to understand space (e.g. a fridge is more likely to be in a kitchen than in an office or a stapler is unlikely to be in a bathroom) and the topology gives us an idea about how space is structured. All this information is integrated with common-sense knowledge in a probabilistic graphical model which reasons about all the obtained concepts and infers room categories such as kitchen, office, corridor and meeting room.

There are different ways in which we can represent spatial knowledge. As proposed in [15], spatial knowledge can be represented in four layers, which is represented in Figure 4.1. The lower level (sensory level) comprises the sensor information such as images and laser scans. Above this layer, the topological layer segments the space in discrete places (nodes). The categorical layer contains models for various objects that we may find in indoor environments, which are used for an object detection algorithm. On top, the conceptual layer relates sensed instances (objects, nodes and rooms) to human concepts.

The conceptual layer maintains relations between concepts (e.g. office has-object monitor), relates observed instances with concepts (e.g. object 1 is-a monitor), it groups nodes in rooms (node1 and node2 is-in room1) and links room categories using undirected links (e.g. office has-conn. corridor). The relations in that layer can be either acquired or inferred. An acquired relation means that the property has been extracted from a perceptual process and inferred relations are generated as a result of inference process. Another distinction between relations is whether they are probabilistic or non-probabilistic. Probabilistic information is extracted from
common-sense knowledge such as that monitors are more likely to be in offices, bathrooms commonly contain toilet paper and corridors connect various rooms, among others.

![Figure 4.1: Structure of the spatial representation. Adapted from 15.](image)

The structure of the system and the data flow between its components are represented in Figure 4.1. A metric SLAM system, performed by gmapping, gathers laser scans and odometry information to create and maintain a metric map. On top of that metric map, a topological map is built. The topological map segments space in different rooms thanks to a door detector. Cameras collect images that are sent to the object recognition process to detect objects in the environment. The conceptual mapping and reasoning component is the core of the system; it collects information from all the other parts and combines them with predefined common-sense knowledge to obtain the final representation. In the following sections, all these parts are explained in detail.
4.1 Metric and Topological Mapping

The maintenance of an accurate representation of our environment is an important feature to perform any spatial reasoning in mobile robots. Therefore, we create two maps to fully understand the environment: a metric and a topological map. The metric map permits to perform localization of the robot in the map and avoid obstacles. However, the metric map has not any semantic meaning itself and that is the reason for creating a topological map.

As explained in Chapter 3, one of the advantages of using ROS is that we can use a wide variety of open source code. For creating a metric map, we have used a ROS package called gmapping to create a 2-D occupancy grid map. As explained in section 3.3.3, gmapping uses laser scans and odometry information to build a map and to estimate the robot’s pose in that map.

From the topology of an environment, we can extract semantic information (e.g. a corridor is a room that connects other rooms). In order to extract this knowledge, a topological map is built on top of the metric map. The topological mapping process uses a door detection algorithm to segment the explored space in rooms. This door detector (proposed by Jensfelt [17]) uses laser scans to detect when the robot crosses a door. If the robot have crossed a door means that it has changed from one room to another.

Our topological map is a pair $T = (N, E)$ where $N = \{n_1, \ldots, n_n\}$ is a set of nodes and $E = \{e_1, \ldots, e_m\} \subseteq N \times N$ is a set of edges. A node is defined as $n_i = (pos, node_id, room_id, is_door)$ where $pos = (x, y)$ defines the position in
the map, node_id is the identifier of each node, room_id defines in which room is placed that node and the value of is_door is “0” for a normal node and “1” if a node represents a door. An edge $e_i = (src, dst)$ describes the path between two nodes where src is the source node and dst the destination.

The topological map algorithm works as follows: when the robot is exploring an environment, we create a new node every one meter of distance. If a door is found, we create a door node which also represents that the robot may enter to another room. If the robot crosses a door, the algorithm updates the room_id value considering if the current area was explored previously or not. Thus, we solve a problem that can occur if the robot explores the same room twice. The algorithm defines edges connecting the two closest nodes. In Figure 4.3 we can see the process of building both metric and topological map. Nodes are painted in different colors to distinguish different rooms and doors are represented by red dots.

![Figure 4.3: Topological map with built on top of a metric map.](image)

**Implementation in ROS**

The mapping process is composed of two main components, the metric and the topological mapping process. A metric map is obtained using gmapping. Gmapping uses information from the robot such as laser scans and odometry to provide an accurate metric map and to estimate the robot’s pose in that map. The position of the robot is used by the topological mapping component to divide the continuous space in different nodes. Each node is created when the robot has moved one meter from the last node and nodes are grouped in rooms. Rooms are segmented thanks to a door detector that uses information from the laser to detect if the robot is crossing a door. The topological mapping process creates a list of nodes and edges; each node contains its identifier, the position of the map where it is created and
4.2. OBJECT DETECTION

the room where it is placed. Edges describe paths from one node to another. Both
nodes and edges are published on ROS topics and saved in a file. In Figure 4.4 we
can see the structure of the mapping component.

![Figure 4.4: ROS implementation of the mapping component.](image)

4.2 Object Detection

In order to perform the object detection system, two methods have been tested
during this thesis. One of the widely examined frameworks during this project but
not used in the final experiments is a ROS package called RoboEarth (explained in
Section 3.3.4). The second method is called Latent SVM [19] and was used during
the final experiments. The two methods, their advantages and disadvantages are
explained below.

4.2.1 RoboEarth

RoboEarth provides object detection methods and model creation tools. It is spe-
cially recommended because all the system is integrated in ROS and it is easy to
combine with other components. To evaluate the performance of RoboEarth, we
have modelled objects such as a rice box, a wooden box and a painting pot. More-
over, a model of a game cover is downloaded from the online database¹.

The detection process works as follows. The first step is to run a camera driver
node that publishes camera images into a defined ROS topic. Then the RoboEarth
detector analyses the incoming images and compares them with the loaded models.
When an object is detected it is highlighted with a red area. In Figure 4.5, object
detection results are shown for four models: rice box, game cover, wood box and
painting pot. As we can see the detector finds successfully the presence of a model
and indicates its location.

¹[http://api.roboearth.org/](http://api.roboearth.org/)
Despite the fact that this framework works successfully, it is not possible to create categorical models. What we would like is to have one model for each category of objects, that means one model for cups, books, chairs or monitors, for example. To illustrate this disadvantage of RoboEarth, imagine that we want to detect different kinds of cups in our kitchen; to do so we have to create every single model for every cup. Moreover, if one day we buy another cup but with little differences to all the created models, the detector may not detect it. Another disadvantage is that increasing the number of models increases the detection time. To solve these drawbacks, the Latent SVM method has been tested and it is detailed in the following section.

### 4.2.2 Latent SVM

The second method for performing object detection is Latent SVM. In Section 2.2.3 we have described in detail how this method works. In this section we describe how this method has been integrated in the ROS system. In contrast to RoboEarth, the Latent SVM detector allows to generate models for object categories; it enables to create models that can detect different objects from the same class. To train the object models, Matlab code has been used and for detection, the OpenCV library provides functions to use the Latent SVM models\(^2\). In the following section, the implementation in ROS is described.

---

\(^2\) [http://docs.opencv.org/modules/objdetect/doc/latent_svm.html](http://docs.opencv.org/modules/objdetect/doc/latent_svm.html)
4.2. OBJECT DETECTION

Implementation in ROS

The detection process works as follows. The object recognition system uses images from a camera and processes them with all the object models. The detection process returns a confidence level or score that is thresholded to obtain object detections. When the object detection algorithm finds a possible detection, it attaches to this detection the robot’s pose and uses this information to find which is the node in the topological map that is closer. Then, the object detector component publishes on a ROS topic the detections. Each message contains the name of the detected object, the map position where the object has been found, the closest node to the detection position and the room identifier where the object has been detected. The results are saved into a file that it is used to perform the reasoning part. Figure 4.6 illustrates the structure of the implementation in ROS.

![Figure 4.6: ROS implementation of the object detector component.](image)

The detection method is capable of evaluating a model in an image in 3-4 seconds. The detection time may vary due to the image size and the model complexity. During real-world experiments (see Section 5.3), the detection process is evaluated offline. Figure 4.7 shows three examples of detection performed with the Latent SVM detector.

![Figure 4.7: Detections using latent SVM.](image)

(a) Bottle detection. (b) Plate detection. (c) Person detection.
4.3 Conceptual Mapping and Reasoning

The conceptual mapping and reasoning part combines extracted information from the environment with common-sense knowledge into a probabilistic chain graph (see Figure 4.8). Its aim is to integrate the spatial knowledge representation (presented in Figure 4.1) in a probabilistic framework. Each room found by the topological mapping algorithm is represented by a random variable for room categories. They are connected to one another following the topological structure of the environment. The knowledge about the topology of the environment and the detected objects is extracted from the saved files mentioned above.

The potential functions $\phi_{rc}(\cdot, \cdot)$ represent how certain categories of rooms are typically connected; for example that a bathroom is more likely to be connected to a corridor than to another bathroom or that offices are usually connected to corridors. The presence of objects is also represented by random variables directly connected to the room where they have been detected. We assign to these object variables a probability of 0.8 that the object has been detected correctly and 0.2 that it is a false positive detection. $p_{oi}(\cdot | \cdot)$ describes knowledge about the probability of finding certain objects in a certain room (e.g. is more likely to find a microwave in a kitchen than in a corridor or that monitors are usually in offices).

Once the chain graph is built, it is converted into a factor graph to use an open source library for graphical models, described in [30]. This library permits to use inference techniques such as Loopy Belief Propagation that infers probabilities about random variables. The chain graph infers which is the probability that each room pertains to a certain category. Imagine that we define three possible categories (corridor, kitchen and office); after running the inference method we will obtain three values representing the probability that the room is a corridor, a kitchen or an office. Using these results we can perform a spatial representation of our environment, labelling each room with its category.

Figure 4.8: Structure of the chain graph used in the conceptual mapping and reasoning part.
Chapter 5

Experiments

This chapter describes the experimental part of our semantic mapping system. The system has been tested both in simulations and in real-world experiments. Simulations have been used for testing the metric and topological mapping and real-world experiments evaluated the whole system in the CVAP department at the Royal Institute of Technology (KTH) in Stockholm, Sweden.

5.1 Experimental Setup

The robot used in this thesis (called “Dora”) is based on a Pioneer 3-DX mobile base. The original platform is extended with an extra structure to support cameras and a laptop where the system is running. Moreover, on the Pioneer platform there is a Hokuyo laser scanner. In Figure 5.1 we can see Dora and all the components, which are detailed in the following sections.

![Diagram of Robot Dora and its components](image)

**Figure 5.1:** Robot Dora and its components.
The semantic mapping systems runs on a laptop with the following features: it has a processor Intel Core i7-2640M, 2.80GHz and the Operative System is Ubuntu 12.04 LTS. Over Ubuntu we run ROS in the Fuerte version. To control the robot during the experiments a Logitech cordless rumblepad joystick is used.

Pioneer 3-DX

The Pioneer 3-DX is a popular mobile robot used for research. It is a two-wheel two-motor differential drive robot and it has a front sonar, wheel encoders and bumpers to avoid collisions. The software to control the robot in ROS is the package called ROSARIA which was detailed in Section 3.3.2 and it allows us to drive the robot and get odometry data.

Cameras

The visual system is based on three cameras:

- Point Grey Chameleon CMLN-13S2C: it is a complete and cost effective camera with a USB 2.0 interface. It is a colour camera with a CCD sensor of 1/3” working at 18 frames per second and resolution of 1296 x 964. In our system, it is used to perform object detection tasks.

- Prosilica GC1380c: it is a compact and high resolution camera (1.4 Megapixel and 1360 x 1024 resolution). The interface is Gigabit Ethernet and it runs at 20 frames per second. The sensor is the Sony ICX285 CCD and the size 2/3”. This camera will be used for future place recognition functionalities.

- Kinect: the popular camera made by Microsoft. It provides RGB images and point clouds thanks to its infra-red structured light sensor. It is used for detecting objects and also to make models with RoboEarth.

Laser Range Finder

The laser used is the Hokuyo URG-04LX; it provides distance readings via RS-232 or USB interfaces. The specifications of this laser scanner are in Table 5.1. The laser scans are used for mapping and door detection purposes.

<table>
<thead>
<tr>
<th>Table 5.1: Hokuyo URG-04LX Specifications.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection Range</td>
</tr>
<tr>
<td>Scan area</td>
</tr>
<tr>
<td>Scan time</td>
</tr>
<tr>
<td>Resolution</td>
</tr>
<tr>
<td>Accuracy</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Angular Resolution</td>
</tr>
</tbody>
</table>
5.2. EVALUATING TOPOLOGICAL MAPPING IN SIMULATION

5.1.1 COLD Database

To evaluate the semantic mapping system, information from the COLD-Stockholm Database\(^1\) has been used. This database consists of sequences of image, laser range and odometry data. The sequences were acquired in the Computer Vision and Active Perception Laboratory (CVAP) at the Royal Institute of Technology (KTH) in Stockholm, Sweden. The robot used was a MobileRobots Powerbot robot platform which was equipped with a multi-camera system and a laser range scanner \(^2\). The database contains information about four different floors of an office environment under different illumination conditions (night, cloudy and sunny weather). In this thesis, real-world experiments are based on the sequence of the 6th floor of the CVAP department under cloudy conditions.

The sequences from the COLD-Stockholm database are synchronized, which means that every sample has a sequence number identifier, and that permits us to adapt adapting them to a ROS format. A bag file (see Section 3.2) has been created to enable the system to read the information. Then, the ROS system uses the database information in a real-time fashion. This database has been used because it contains all the needed information to run the semantic mapping system.

Additionally, two different data based have been used to train object models for the object detection system. These datasets are unrelated to the testing environment, and they are called PASCAL and LabelMe database:

- PASCAL database \(^3\): it is composed of more than 9,000 images where twenty object classes are found. Since our thesis is focused in indoor environments we have selected the following categories from this database: person, bottle, chair and monitor.

- LabelMe database \(^4\): it is an online database where anyone can contribute and upload images with annotated objects. There are more than 1,000 images with hundreds of different objects. From these database we have trained models for bin, bookshelf, microwave and whiteboard.

5.2 Evaluating Topological Mapping in Simulation

Simulations have been used to provide an initial evaluation of the metric and topological mapping. To simulate the mapping process, various ROS components have to be used. Figure 5.2 shows the structure used to perform the simulations. As explained in Section 3.2, stage provides sensory information from a virtual world. To move the robot in the environment, a node called teleop reads the keyboard and publishes the information in the topic “cmd_vel”. Stage reads that information and moves the robot around the world. The virtual information created by stage is used

\(^1\)http://www.pronobis.pro/data/cold-stockholm

41
CHAPTER 5. EXPERIMENTS

Figure 5.2: ROS configuration to evaluate the mapping component.

by the mapping components. A metric map is created using gmapping and the map saver node stores the map into an image file. The topological mapping node creates the topological map using a door detector.

As an example of the simulations, in Figure 5.3 we can see a metric and a topological map of the 7th floor of the CVAP. In this topological map, the room segmentation algorithm is not used. During this thesis various metric maps were created using this process and are shown in Figure 2.1, Figure 3.7 and in Figure 4.3.

Figure 5.3: Metric and topological map of the 7th floor of the CVAP department.
5.3 Real-world Experiments

Our system has been evaluated in a real-world environment on the 6th floor of the CVAP department at the Royal Institute of Technology (KTH) in Stockholm. To perform the experiments, the COLD-Stockholm database explained above has been used. We have used a dataset where the robot went through nine different rooms under cloudy conditions collecting images, odometry and laser data. The following sections show how our system is capable of performing semantic mapping in this database.

This section is divided in three parts: the first one describes the metric and topological mapping results, the second part explains the object detections and the last one shows how all this information is integrated into a probabilistic chain graph and how the final semantic map is obtained.

5.3.1 Metric and Topological Map

The COLD-Stockholm database has been evaluated using a single sequence. The information stored in a bag file has been used by the mapping system to create a metric and a topological map. The topological mapping algorithm is able to detect eight doors and to segment the explored environment in the nine existing rooms. Figure 5.4 shows both metric and topological maps where doors are represented by red dots and each room is painted in different colors.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{metric_topological_map}
\caption{Metric and topological map with room segmentation.}
\end{figure}
5.3.2 Object Detections

The object detection system implemented in the final version is based on the Latent SVM method. As described in Section 4.2.2, models of objects have to be created to use the Latent SVM algorithm. Aiming the semantic mapping algorithm proposed in this thesis to work in indoor environments, various objects have been modelled.

The training process requires a large number of images to obtain good models. It needs a set of positive images with bounding boxes; bounding boxes describe where the object is placed in an image and permits the training process to select this part of the image. Negative images are also needed during the process; a negative image means that the object is not placed in that image. To obtain all these images, two databases have been used, which are unrelated to the testing environment:

The training process has been made in Matlab using the available tools in the third version of latent SVM \(^2\). Some of these models are represented in Figure 5.5.

The resulting models have been converted into a xml format to be used within the OpenCV code. Using OpenCV we have integrated the object detector system in ROS and hence we can integrate it with the other parts of our framework.

![Trained models using latent SVM.](http://cs.brown.edu/~pff/latent-release3/)

Figure 5.5: Trained models using latent SVM.

The list of all the trained models is:

- Bin
- Bookshelf
- Bottle
- Chair
- Microwave oven
- Monitor
- Person
- Plate
- Whiteboard

After creating the models, the object detection system is evaluated using images from the COLD-Stockholm database. The images are processed for each model in

\(^2\)http://cs.brown.edu/~pff/latent-release3/
5.3. REAL-WORLD EXPERIMENTS

non real-time mode. The result is a list of scores or confidence levels that represents possible detections. These results are checked in order to select the appropriate threshold for each object and consequently, obtain low false positive detections. As explained in Section 4.2.2, the object detection system attaches to each detection its localization in the map and the room where it has been detected. Figure 5.6 illustrates how the system detects objects and stores information about its location; we can see examples of detected objects and the place where they have taken place. A list of all this information is stored to be combined into the probabilistic graphical model.

![Topological map with various detected objects.](image)

Table 5.2 shows that there are four detectors (bin, bookshelf, microwave oven and monitor) that have detected more than the 59% of the existing objects. However, in this group, the only which has no false positive detections is the microwave oven de-

In Table 5.2 we can see the results of the object detection process for the sequence extracted from the COLD-Stockholm described in Section 5.1.1. It is illustrated the total number of each object that we can find in the database, as well as the true and false positives detections obtained with our method.

Table 5.2 shows that there are four detectors (bin, bookshelf, microwave oven and monitor) that have detected more than the 59% of the existing objects. However, in this group, the only which has no false positive detections is the microwave oven de-
Table 5.2: Detections results.

<table>
<thead>
<tr>
<th>Model</th>
<th>Objects in database</th>
<th>True positives</th>
<th>False positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bin</td>
<td>8</td>
<td>6 (75%)</td>
<td>2</td>
</tr>
<tr>
<td>Bookshelf A</td>
<td>12</td>
<td>5 (42%)</td>
<td>0</td>
</tr>
<tr>
<td>Bookshelf B</td>
<td>12</td>
<td>12 (100%)</td>
<td>5</td>
</tr>
<tr>
<td>Chair</td>
<td>40</td>
<td>12 (30%)</td>
<td>2</td>
</tr>
<tr>
<td>Microwave oven</td>
<td>2</td>
<td>2 (100%)</td>
<td>0</td>
</tr>
<tr>
<td>Monitor</td>
<td>17</td>
<td>10 (59%)</td>
<td>2</td>
</tr>
<tr>
<td>Whiteboard</td>
<td>6</td>
<td>1 (17%)</td>
<td>3</td>
</tr>
</tbody>
</table>

tector. The detectors such as chair and whiteboard have a low rate of true positive detections (less than 31%). The low true positive detections of the chair detector is due to the large within-class variability. Another reason that could explain the low number of true detections in these models is that various of the objects are occluded in the dataset and as a consequence the threshold is too high to accept them as a true detection.

Regarding the bookshelf detector, we have created two different models (A and B). The difference between these models is the dataset used for the training process. The model A has detected five out of twelve bookshelves and the B has detected all of them. However, the second model has five false positives and the first has none. For the final evaluation we have selected the model A, since we aim to have the lowest rate of false positive detections. A fact that explains false positive detections is when a similar object to our model is scored as a positive detection. As an illustration of this, in Figure 5.7 we can see how a plant pot is detected as a bin, a painting as a monitor and a bulletin board as a whiteboard.

![False positives examples](image)

**Figure 5.7:** False positives examples. In a) a plant pot is detected as a bin, in b) a painting is detected as a monitor and in c) a bulletin board is confused with a whiteboard.
5.3. REAL-WORLD EXPERIMENTS

5.3.3 Conceptual Mapping and Reasoning

As explained in Section 4.3, the main function of this component is to combine the extracted information (objects, topology and rooms) with common-sense knowledge and represent sensed instances as human semantic relationships. The extracted knowledge is modelled into a probabilistic chain graph that infers room categories.

The information used to create the chain graph comes from saved files about the topology of the environment and the detected objects. The tested sequence is composed of nine rooms, where all the rooms are connected to the first one (the corridor). Different objects are detected in all the rooms except the room number four where the detector did not find any object. The system constructs the chain graph following the topological structure and attaches detected objects to the appropriate room. In Figure 5.8 we can see the structure of the chain graph corresponding to our experiment.

![Figure 5.8: Structure of the chain graph for the experimental results.](image)

The extracted information is combined in the graph with common-sense knowledge which defines probabilities about how rooms are typically connected or which objects are more likely to be found in certain rooms. The common-sense knowledge that defines rooms connectivities ($\phi_{rc}(\cdot, \cdot)$) is set according to values proposed in [15]. This probabilities are extracted evaluating the COLD-Stockholm database, so they describe an indoor office environment. The values of $p_o(\cdot | \cdot)$, which de-
scribe the probability that a certain object is found in a certain room category, are configured after analysing a small dataset of images from the 6th floor of the CVAP.

In our experiments, the room category variables are configured to have four possible values: corridor, office, meeting room and kitchen. The chain graph is transformed into a factor graph (see Section 2.3.1) and the, the Loopy Belief Propagation method is used to infer the room category probabilities. The obtained results are:

Table 5.3: Inferred probabilities by the probabilistic chain graph.

<table>
<thead>
<tr>
<th>Room</th>
<th>Corridor</th>
<th>Office</th>
<th>Meeting room</th>
<th>Kitchen</th>
<th>Inferred category</th>
<th>Real category</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>Corridor</td>
<td>Corridor</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.99</td>
<td>Kitchen</td>
<td>Kitchen</td>
</tr>
<tr>
<td>3</td>
<td>0.00</td>
<td>0.41</td>
<td>0.58</td>
<td>0.00</td>
<td>Meeting room</td>
<td>Meeting room</td>
</tr>
<tr>
<td>4</td>
<td>0.07</td>
<td>0.37</td>
<td>0.33</td>
<td>0.23</td>
<td>Office</td>
<td>Office</td>
</tr>
<tr>
<td>5</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>Office</td>
<td>Office</td>
</tr>
<tr>
<td>6</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>Office</td>
<td>Office</td>
</tr>
<tr>
<td>7</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>Office</td>
<td>Office</td>
</tr>
<tr>
<td>8</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>Office</td>
<td>Office</td>
</tr>
<tr>
<td>9</td>
<td>0.00</td>
<td>0.99</td>
<td>0.00</td>
<td>0.00</td>
<td>Office</td>
<td>Office</td>
</tr>
</tbody>
</table>

The results in Table 5.3 show how the semantic mapping system is capable of performing an accurate reasoning of the explored environment. All the rooms have been successfully categorized. The first room is categorized as a corridor with a high probability (99%) due to the fact that it is connected to all the other rooms. Regarding the second room, the algorithm defined this room as a kitchen thanks to the two microwave ovens found in it. The third room has a probability of 0.41 of being an office and 0.58 a meeting room. The cause is that these two rooms usually have the same objects (chairs, bin, monitors), but in this case a whiteboard was detected in that room and the configured common-sense knowledge estimates more probable to find a whiteboard in a meeting room than in an office.

In room number four, the object detector system could not detect any object. That is the reason why there is not a high probability for any of the room categories, but the room is still an office because the room connectivity probabilities were defined in an office environments, which makes more probable to find an office than any other room. The rest of the rooms have been correctly categorized as offices. The detected objects in these rooms were monitors, bookshelves, bins and chairs.

Once the reasoning process has been made and the system knows all the room categories, only one part is missing that is to represent the results. To do so, we represent the topological map and each room is labelled according to the inferred results. The final map, which is represented in Figure 5.9, shows an accurate representation of
5.3. REAL-WORLD EXPERIMENTS

the environment and permits anyone to understand the explored space.

Figure 5.9: Topological map with room categories.
Chapter 6

Conclusions and Future Work

This thesis described the process of creating a semantic mapping framework integrated in ROS. Semantic mapping methods allow robots to comprehend human environments using information extracted from sensors. To create a representation of an environment, robots have to be capable of modelling spatial concepts such as objects, rooms, geometry of the space, appearance of rooms and topology of the environment. All this information combined with common-sense knowledge should permit robots reason about the environments.

As proposed in [15], the presented framework is divided in four layers: sensory, topological, categorical and conceptual. The sensory layer contains sensory information such as laser scans and images, the topological level maintains a topological map, the categorical level has object models and on top of all of them the conceptual layer relates sensed instances with humans concepts. Our framework focuses on extracting the following spatial concepts: topology of the space and objects. These concepts are integrated in a probabilistic fashion to perform reasoning functions.

As mentioned above, the semantic mapping system is integrated in ROS. ROS is an operating system for robots that provides numerous tools to develop any robot application. It contains hundreds of robot algorithms that can be easily used and integrated in the system such as mapping, exploration, navigation, object detection or machine learning, among others. It also has simulation and monitor tools which permit debugging functions in all the system. ROS is compatible with a great number of robot platforms thanks to its hardware abstraction.

Gmapping is an example of a ROS component used during this thesis. This ROS package uses SLAM techniques to create a 2-D occupancy grid map of the explored space. It permits us to have an accurate metric map while localizing the robot in that map. On top of that metric map, a topological map has been created. The topological map discretizes continuous space into nodes and edges. A door detector is used to segment the topological map in different rooms and hence, obtain seman-
tic information about how rooms are connected.

Objects are an important concept for semantic mapping. During this project, two object detection methods have been evaluated: RoboEarth and Latent SVM. The first one is a ROS package which uses a Kinect camera to create object models and uses SURF detectors to find objects in incoming images. During this thesis, different models have been created and successfully detected, but the RoboEarth platform did not allow to create categorical models. To solve that, the Latent SVM method has been used. This method is integrated in the OpenCV library and allows to easily integrate it in ROS. However, the detection time of the original code was too long (around 15 seconds). Thus, we have modified the original models in order to create simpler ones and to reduce the detection time. After training models using two image datasets (PASCAL and LabelME), we have obtained and evaluated nine models: bin, bookshelf, bottle, chair, person, plate, microwave oven, monitor and whiteboard.

The extracted spatial knowledge is combined with common-sense knowledge in a probabilistic framework. It represents the extracted properties into a probabilistic graphical model; in our case a chain graph and a factor graph have been used. Graphical models permit the system to reason about the explored environment and infer probabilities about room categories.

The presented framework have been experimentally evaluated in simulations and in real-world scenarios. The results show how the framework maintains an accurate representation of the explored environment, segments it in rooms, detects objects, combines all these spatial concepts into a graphical model and infers room categories. At the end, it is able to build a semantic map where rooms are successfully labelled as kitchen, office, corridor or meeting room.

6.1 Future Work

There are several directions in which the developed semantic mapping system can be improved. In our thesis we have focused on extracting spatial knowledge from objects and the topology of the environment. Despite the fact that these properties carry a lot of semantic mapping, we can extract information from other properties like the size and shape of a room, as in [15]. These properties will enable the system to decide whether rooms are small, medium, large, rectangular, square or round and consequently, distinguish between single office, double office and big or small meeting room, for example. Another improvement for our system could be to implement a visual appearance-based place categorization algorithm. It is based on comparing incoming images with room models to discriminate if the appearance of the image is corridor-like, kitchen-like, office-like or meeting room-like.
6.1. FUTURE WORK

Human-robot interaction is another direction for future research in the semantic mapping process. We could use information from a human as an input in our system. If we perform a robust speech recognition system, while we are driving the robot around the environment, we could tell the robot “this is the kitchen” or “in this room there are four chairs”. Then the robot could incorporate this information as another source of knowledge.

Regarding the object detection, in this project we have trained nine object models. A direction to extend the capabilities of our system could be to create more models and reduce the detection time. Cloud computing techniques could be used to detect objects: the robot system could send images to the cloud where all the computation part would be done and then the cloud shall send to the robot information about the detected objects.

To improve the sensory layer, 3D techniques can be used. Using these methods we could create a 3D map or extract geometrical information from the environment. Recent advances allow us to use a Kinect camera to extract depth information, process its information and extract geometrical features.

The probabilistic graphical model presented in this project infers probabilities of the explored rooms; an important improvement would be to predict unexplored space ([13]). It would predict which objects are more likely to be found in the next room or if the next room will be a kitchen or an office, for example.

The foremost improvement for the semantic mapping systems is to be able to develop and integrate the future work described above and perform systems capable of understanding complex space, cooperating and helping humans: guiding people, helping humans shopping in supermarkets or developing assistant functions for disabled people.
Bibliography


BIBLIOGRAPHY


## List of Figures

1.1 Spatial and conceptual hierarchies. Anchoring creates a link between sensory information and semantic models. Reproduced from [7].

1.2 Structure of the system proposed in [8]. The perception components send information to three layers (metric, navigation and topological map). The reasoning part (conceptual map) takes information from the other layers as well as from human inputs. Reproduced from [8].

2.1 Occupancy grid map; white pixels represent free space, black ones indicate an occupied area and grey pixels are unknown spaces.

2.2 Process of building a metric map. Reproduced from [20].

2.3 Topological mapping process using a voronoi diagram. Reproduced from [23].

2.4 SURF descriptor. Reproduced from [24].

2.5 Detection of energy drink can using SURF.

2.6 Example of HOG. At the left the original image; in the middle the HOG descriptor and at the right the HOG descriptor after using a SVM classifier. Reproduced from [24].

2.7 Sliding window approach.

2.8 A two-component bicycle model. (a) and (d) represent root filters, (b) and (d) are part filters at higher resolution and (c) and (f) are spatial models that define the location of the different parts. Reproduced from [19].

2.9 Detection process using a single filter. Reproduced from [19].

2.10 Types of graphical models.

2.11 Example of a factor graph.

2.12 Conversion of a directed graph to a factor graph (a) and an undirected graph to a factor graph (b). Reproduced from [33].

2.13 Conversion of a chain graph to a factor graph.

3.1 ROS communication infrastructure. Reproduced from [36].

3.2 RVIZ simulator.

3.3 Stage simulator.

3.4 Basic coordinate frames in a robot. Reproduced from [36].
### List of Figures

3.5  Subscribed and published topics by ROSARIA ........................................ 25
3.6  Overview of the *slam_gmapping* node. ........................................ 26
3.7  Example of an occupancy map using *gmapping* and *map_saver*. .......... 27
3.8  RoboEarth structure. Reproduced from [18]. ....................................... 27
3.9  Modelling process in RoboEarth .......................................................... 28
3.10  Bridge between ROS and OpenCV. Reproduced from [36]. .................... 29

4.1  Structure of the spatial representation. Adapted from [15]. ....................... 32
4.2  Structure and data flow of the semantic mapping system .......................... 33
4.3  Topological map with built on top of a metric map .................................. 34
4.4  ROS implementation of the mapping component ................................. 35
4.5  Object detections using RoboEarth. The first row shows original images and the second row the results of the object detection ........................................ 36
4.6  ROS implementation of the object detector component .......................... 37
4.7  Detections using latent SVM ................................................................. 37
4.8  Structure of the chain graph used in the conceptual mapping and reasoning part .......................................................... 38

5.1  Robot Dora and its components .............................................................. 39
5.2  ROS configuration to evaluate the mapping component ........................... 42
5.3  Metric and topological map of the 7th floor of the CVAP department .......... 42
5.4  Metric and topological map with room segmentation .............................. 43
5.5  Trained models using latent SVM .......................................................... 44
5.6  Topological map with various detected objects ....................................... 45
5.7  False positives examples. In a) a plant pot is detected as a bin, in b) a painting is detected as a monitor and in c) a bulletin board is confused with a whiteboard .......................................................... 46
5.8  Structure of the chain graph for the experimental results ........................ 47
5.9  Topological map with room categories ................................................ 49
List of Tables

3.1 ROS Distributions ............................................ 21
5.1 Hokuyo URG-04LX Specifications .......................... 40
5.2 Detections results ............................................. 46
5.3 Inferred probabilities by the probabilistic chain graph 48