Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

João Carlos da Cunha Sanches Fernandes

Semantic Mapping with a Mobile Robot Using a RGB-D Camera

Dissertation submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra in partial fulfillment of the requirements for the Degree of Master of Science

Coimbra

February 2019

Universidade de Coimbra

Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

Mapeamento semântico com um robô móvel utilizando um sensor RGB-D

por

João Carlos da Cunha Sanches Fernandes

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação.

Supervisor: Prof. Doutor Rui Paulo Pinto da Rocha

Jury

President: Prof. Doutor Urbano José Carreira Nunes

Vogals: Prof. Doutor Rui Alexandre de Matos Araújo

Prof. Doutor Rui Paulo Pinto da Rocha

Fevereiro 2019

Acknowledgements

This work would not have been possible without the inspiration and support of a number of wonderful individuals — my thanks and appreciation go to all of them for being part of this journey and making this dissertation possible.

In first place, I would like to thank all my family. I am forever indebted to my mother, grandmother and sister for giving me the opportunities and experiences that have made me who I am. They selflessly encouraged me to explore new directions in life and seek my own destiny. This journey would not have been possible if not for them, and I dedicate this milestone to them.

I owe my deepest gratitude to my supervisor Prof. Rui Rocha. Without his encouragement, support and continuous optimism, this thesis would hardly have been completed.

I am forever thankful to all my colleagues at the Mobile Robitics Lab., for their friendship and support, and for setting up a creative working environment, where new ideas and knowledge were constantly shared.

Finally, I express my deep and sincere gratitude to all my closer friends. I am very thankful for their unconditional friendship, support and patience through all these years, thanks for the wonderful times we shared, you made my studying journey unforgettable.

Resumo

Atualmente o principal foco da Robótica é o de integrar robôs de serviço na sociedade e nas nossas vidas quotidianas, de forma a que estes nos possam ajudar nas mais diversas actividades. Com este propósito, um agente robótico deveria, idealmente, entender e lidar com o seu ambiente da mesma forma que um ser humano. Para tal é necessário que os robôs não só sejam capazes de categorizar divisões do espaço de trabalho e entendam em que tipo de contextos estas são utilizadas, mas que também prevejam ainda que tipo de objectos tendem a ser encontrados nessas divisões. Ao detectar e registar este tipo de conhecimentos em mapas robóticos mais informativos – mapas semânticos –, os robôs de serviço podem tornar-se mais eficazes e úteis num grande leque tarefas. Os mapas semânticos podem vir a ser utilizados, por exemplo, com o intuito de modelar o comportamento de um robô quando este é chamado a executar a tarefa de procurar um determinado tipo de objecto. Dado o "conhecimento" do robô, através do mapa semântico, de que pratos tendem a existir em cozinhas, o robô deverá procurar por estes em divisões pertencentes a esta categoria.

A recente disponibilidade de datasets de grande dimensão, em cojunto com a existência de processadores gráficos (GPUs) cada vez mais poderosos, tem influenciado de forma positiva técnicas de aprendizagem máquina baseadas em redes neuronais profundas e estas tem alcançado níveis de desempenho sem precedentes na resolução de diversos problemas.

O foco principal desta dissertação de mestrado foi estudar e desenvolver um sistema robótico móvel capaz de construir mapas semânticos, neste caso particular mapas com informações acerca de ocorrências de objectos e de categorias de divisões anotadas, combinando métodos clássicos utilizados na robótica móvel com métodos de aprendizagem máquina baseados em redes neuronais profundas. Foi ainda desenvolvido um algoritmo baseado em ontologias para tornar a procura de objectos por parte do robô mais rápida e eficaz.

O sistema de mapeamento semântico desenvolvido foi validado e, nos cenários de teste realizados, demonstrou uma boa capacidade no registo de ocorrências de objectos e categorias de divisões nos mapas semânticos. O algoritmo de procura de objectos foi avaliado num conjunto de simulações a partir das quais foi possível demonstrar que os relacionamentos entre objectos e divisões podem melhorar significativamente o desempenho de um robô de serviço na execução da tarefa de procurar por um objecto específico.

Palavras-chave: *Mapeamento Semântico*, *Detecção de Objectos*, *Reconhecimento de cenas*, *ROS*, *Redes neuronais profundas.*

Abstract

Nowadays, the major focus in robotics is to make service robots ubiquitous in our daily lives to assist us in different activities, in public and domestic spaces. With this purpose, robots should understand their surroundings in a human-like manner. This requires that robots are able to categorize rooms, understand in which kind of context humans use those, and predict which objects can be found in each area category. By detecting and registering such spatial properties in more informative robotic maps – semantic maps –, service robots become more capable to perform useful tasks and assist humans. For instance, the generated semantic maps can be used to modulate the robot's behavior when seeking a given type of object, e.g. given the common-sense knowledge that plates are usually found in a kitchen, the robot will search them firstly in places belonging to that category.

Deep learning, i.e. deep neural networks, leveraged by the availability of large datasets and increasingly powerful GPUs, have attained unprecedented performance in object detection and scene recognition in images.

The major focus of this M.Sc. dissertation was to study and develop a mobile robotic system endowed with the capability of building semantic maps, in this particular case, maps with object occurrences and place categories annotated, by combining classical methodologies used in the mobile robotics field with state-of-the-art deep learning techniques. Furthermore, object-places relationships were also studied and an ontology-based robotic object searching algorithm was developed.

The developed semantic mapping framework was validated through experiments conducted over real-world data and, in those test scenarios, it showed its ability to successfully attain a fairly good representation of the object occurrences and place categories present in the robot's workspace. The ontology-based robotic object searching algorithm was evaluated in a set of simulated scenarios where it has proven that object-places relationships can enhance the performance of a service robot when asked for seeking a specific object.

Keywords: *Semantic Mapping*, *Object Detection*, *Scene Recognition*, *ROS*, *Deep Learning*.

List of Acronyms

ISR	Institute of Systems and Robotics
MRL	Mobile Robotics Laboratory
ROS	Robot Operating System
LRF	Laser Rangefinder
SVM	Support Vector Machine
DNN	Deep Neural Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
CNN	Convolutional Neural Network
ReLu	Rectified Linear Unit
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
OD	Objectness Detection
SOD	Salient Object Detection
COD	Category-Specific Object Detection
R-CNN	Region Convolutional Neural Network
SPPnet	Spatial Pyramid Pooling Network
RoI	Region of Interest
RPN	Region Proposal Network
YOLO	You Only Look Once
SSD	Single-Shot Multibox Detector
PASCAL VOC	Pattern Analysis, Statistical Learning and Computational Learning Visual Object Classes
MS-COCO	Microsoft Common Objects in Context
IoU	Intersection over Union
CRF	Conditional Random Fields
RDF	Randomised Decision Forest
SLAM	Simultaneous Localization and Mapping
IR	Infrared

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Until now, most robots have been limited to working on factories, on industrial bounded areas segregated from humans, where they repeatedly do the same activities. The current new focus in robotics is to free these robots of their "cages", and start to use them for public and domestic services, often in interaction with humans and in uncertain, dynamic environments. Advances in these areas should automate non-differentiating mundane tasks, so that humans can be more focused on higher-level activities, not spending precious life time on dull activities.

Gray Scott, a futurist and techno-philosopher of the 21th century, announced progress in robotics in the following way [1]:

> "Robots will harvest, cook, and serve our food. They will work in our factories, drive our cars, and walk our dogs. Like it or not, the age of work is coming to an end."

These ideas clearly demonstrate the human dream that, in a near future, robots will work side by side and interact with humans, and be capable of automating and replacing humans in tedious tasks, or assist humans and collaborate in other tasks. To perform complex tasks, mobile service robots should develop an understanding of their surroundings that goes beyond the ability to avoid obstacles, autonomously navigate, or build maps [2]. They need to understand and reason about the environment where they coexist with humans, and be able to model environments through human-like representations that can be used to interact with humans. Therefore, mobile service robots will not only keep track of their world position in a metric way, but also perform spatial understanding, which requires modeling and representing semantic information [3] about the spatial properties of the environment.

Humans tend to discretise and label indoor areas in rooms according to their specific function. In order to improve the human-robot interaction, robots should also categorize rooms, understand in which kind of contexts humans use those, and predict what can be found in those bounded areas. Consequently, for the next level of robot intelligence and intuitive user interaction, robots would recognize bedrooms, have the common-sense knowledge that the areas where they are found are meant for humans to rest, and will predict that in those places there will be most certainly a wardrobe near a bed.

In order to attain this level of intelligence, robots could use various sources of spatial knowledge, such as occurrence of objects, size, shape, appearance and topology of rooms. By representing these spatial properties in semantic maps and reasoning about them, it is possible to enhance significantly the robot's behavior during navigation tasks, e.g. given the common-sense knowledge that plates are usually found in a kitchen, the robot will search them firstly in a kitchen.

In consideration of the foregoing, this work intends to program a mobile robot to be capable of building semantic maps, i.e. maps with semantic information annotated about the spatial properties mentioned, combining classical methodologies used in the mobile robotics field with state-of-the-art deep learning techniques.

1.1 Objectives

As previously referred, the main focus of this thesis is to program a mobile robot to build semantic maps. In order to do so, the robot should not only be capable of localizing itself in the workspace; it should also:

- Recognise and locate certain types of objects used by humans in their daily life.
- Categorize room divisions, and in order to do so, it must perform scene recognition;
- Represent and handle the different acquired semantic properties, i.e. object occurrences and place categories, in a temporal coherent semantic map;
- Take advantage of the semantic map in order to find objects in a faster and more competent manner.

The system is intended to be developed in Robot Operating System (ROS), a collection of software packages and tools for robot software development [4], and should run in a Pioneer P3-DX[1] mobile robot endowed with a Hokuyo URG-04LX Laser Rangefinder (LRF) and a Microsoft Kinect[2] RGB-D camera.

1.2 Document Overview

This manuscript is organized in 5 chapters.

After this introductory chapter, chapter 2 starts by introducing machine learning and deep learning in particular. Afterwards, state-of-the-art object detection and scene recognition methods are briefly covered. A survey with the most relevant semantic mapping

[1]More information about the Pioneer P3-DX can be found at `https://www.generationrobots.com/en/402395-robot-mobile-pioneer-3-dx.html`

[2]More information about the Microsoft Kinect RGB-D sensor can be found at `https://en.wikipedia.org/wiki/Kinect`

frameworks that can be found in literature is then presented. In the final chapter's sections ROS framework and the Kinect sensor are briefly introduced.

Chapter 3 starts by describing the developed system in a broad manner. Each system's comprising module considered is presented afterwards in greater detail.

Chapter 4 presents the experiments in which the system developed was tested and evaluated as well as the results and conclusions obtained.

Finally, Chapter 5 presents a reflection about the work developed in this M.Sc. thesis and points out and discusses future research directions.

Chapter 2

Background and Related Work

2.1 Machine Learning

For decades, humanity has dreamed of building intelligent machines that mimic human brain cognition [5]. However, to build these artificially intelligent machines we have to solve some of the most complex computational problems we have ever dealt with, and new approaches to program them have to be adopted. We do not know how to write a program to categorize an object as an umbrella because we do not know exactly how our brain is doing that. We know that an umbrella is an umbrella because, through our entire life, we have seen and used lots of umbrellas – we have learned the concept of an umbrella.

Machine learning, a subarea of the artificial intelligence field, tackles problems in a similar manner: learning from example. The goal of a machine learning algorithm is to predict the mapping function $\boldsymbol{y} = f(\boldsymbol{x})$, where:

- $\boldsymbol{x}$ is the input data, it could be sensor readings, images, etc.
- $\boldsymbol{y}$ is a discrete value, i.e. class label, and we are solving a classification problem, or it is a continuous value and we are dealing with a regression problem.

The mapping function $\boldsymbol{f}$ is predicted using a set of training samples, also known as the training dataset, and can be characterized with a set of parameters $\boldsymbol{\gamma}$. In supervised learning, from the set of training samples $\{\boldsymbol{x}_i\}$ with their annotated target outputs $\{\boldsymbol{y}_i\}$, $\boldsymbol{\gamma}$ is estimated at the training stage.

The target function $\boldsymbol{f}$ learned from the training samples sometimes does not generalize well to new data. Generalization refers to how well the concepts learned apply to examples not seen by the model in the training stage. The learned model should generalize well from the training data because, for most of the problems, the number of possible input configurations is immense, and "new examples are almost surely going to be different from any of the training examples" [6]. This would allow us to make predictions on new data never seen by the model.

The learned model can fail to solve our problem in two different ways:

- The training error, i.e. the error obtained when validating the learned model on the training subset, remains big, which is a symptom that our model was not able to pick up the problem's proper characteristics and cannot solve it in an efficient manner – *underfitting*.

- The training error is small, but there is a relevant discrepancy between the training and test error. The test error validates our model on data not used in the learning phase; it is a metric that gives an idea of how well the learned model is generalizing. As expected, most of the times the learned model achieves lower training error than test error, but if this discrepancy is significant, that is a clear symptom that our model does not generalize well – *overfitting*.

It is possible to control if a model is more likely to overfit or underfit by altering its capacity. A model's capacity can be seen as its ability to fit a wide variety of functions [6]. Overffiting can be explained by the mismatch between the learning capacity and the dimension of the training dataset [7]. A machine learning algorithm will perform better when its capacity is appropriate for the real complexity of the problem and for the amount of training data it is provided with.

As the dimensionality of the training samples increases, the number of parameters as well as the learning capacity of $\boldsymbol{f}$ increases as well. This phenomenon makes the overfitting problem even worse and is known as "*the curse of dimensionality*" [7].

Nowadays, large datasets are available for training machine learning algorithms, and people observed that the performance of $\boldsymbol{f}$ on the training subset got improved when the dimensionality of the input data increased. This is known as *"the blessing of dimensionality"* [8] and it can be explained due to the fact that larger training data, due to its disparity, also requires larger learning capacity. However, some of the machine learning models with shallow structures (e.g. Support Vector Machines (SVMs) and Boosting) get their performance saturated when training data becomes very large because of their limited learning capacities [7]. In this case, underfitting happens, the learned model will not represent a good solution to the problem it is trying to solve. On the other side, Deep Neural Networks (DNNs), with their deep architectures, can have a huge number of parameters, resulting in a larger learning capacity. Therefore, when large scale training data is available, DNNs could perform better compared with other machine learning methods. On the other hand, when the training data is small they face the overfitting problem and could perform even worse than other machine learning methods.

In the last decade, with the emergence of large scale training data and increasingly more powerful Central Processing Units (CPUs) and Graphics Processing Units (GPUs), DNNs have started to be more exploited and have won several contests in different domains, such as speech recognition, handwritten classification, pattern recognition, object detection and scene recognition [9, 10]. For this reason, the work developed in this thesis on object detection and scene recognition is based on those state-of-the-art deep learning methods. In the next

section, we briefly introduce a deep learning architecture that has been widely used for these two problems: the Convolutional Neural Network (CNN).

2.2 Convolutional Neural Network

The CNN is one of the most studied deep learning architectures. The first CNN, the NeoCognitron, was introduced by Fukushima [11] in 1982 . Sixteen years after, LeCun et al. created a CNN for character recognition known as LeNet-5 [12], which became one of the most famous examples of this technique. The LeNet-5's basic structure of altering convolutional layers, also known as conv layers, and pooling layers, also known as subsampling layers, followed by fully connected layers for the classification of the input image into characters, is still used in most of the today's CNNs, although with some modifications. The LeNet-5's CNN structure can be visualized in Fig. 2.1.

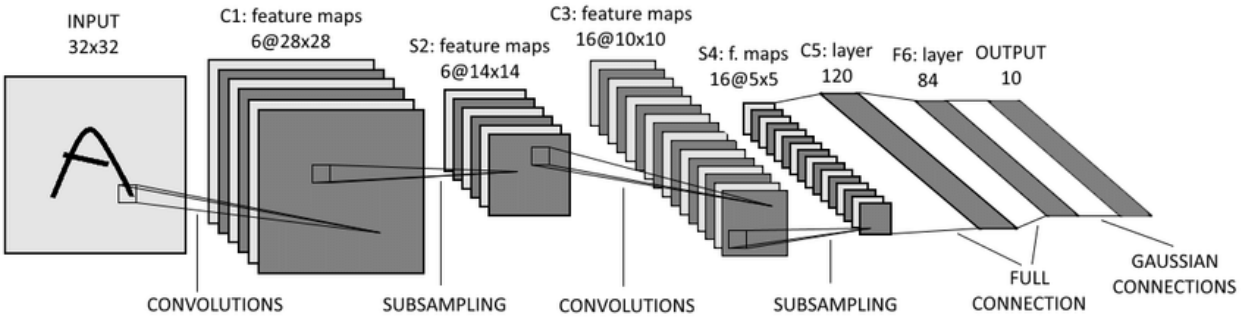


Figure 2.1: A Typical CNN structure, the LeNet-5's structure. Image reproduced from [12].

With some exceptions, CNNs are typically structured in two parts: the feature extraction comprising convolutional and pooling layers; and the last part, classification, comprising fully connected layers. Each different type of layer has different rules for forward and error backward signal propagation.

2.2.1 Convolutional Layers

Convolutional layers, as the name suggests, employ convolution operations on input volumes. Convolutions are performed on the input volumes using filters, also known as kernels, which work as feature extractors. The result from a convolution is obtained by summing the dot products within a local patch from the input volume with the desired kernel. The patch selection is then slided (towards the right, or downwards when the boundary of the matrix is reached) by a certain amount called the stride value. The process is repeated until the entire input volume has been processed [13]. The output of a convolution operation is typically called a feature map.

Even when using a stride of 1, the dimension of the output is reduced when performing a convolution. To preserve the dimensions of the input volume to the output volume, the

most widely used technique is the zero padding, which adds zeros around the borders of the input volume.

A Convolutional layer produces k feature maps, also denoted as activation maps or channels, where k is the number of kernels in that layer's filter bank. The filter weights and bias are learned during the training phase of the CNN. The number of filters, k, as well as the filter's squared dimension e, the stride, s, and padding, p, used are hyperparemeters of each convolutional layer.

After extracting the features with the learned kernels, the bias is added to the resulting feature map. A non-linear activation function is used on that feature map and the result is the final feature map extracted. Nowadays, the most used non-linear activitation function in convolutional layers, is the Rectified Linear Unit (ReLu). This function replaces all negative values with 0s, and outputs the input if the input has positive values.

Although the filter depth, n_c, is usually omitted in the notation, its value is k^{l-1}, where k^{l-1} is the number of the feature maps produced in layer $l-1$ (same as the depth of the input volume). For instance, when feeding a RGB image to a convolutional layer, the first convolutional layer's kernels have always $n_c = 3$ due to the number of feature maps of its input being $k^0 = 3$, where each color channel is considered a different feature map. Considering the output volume of layer l of dimensions $d^l = w^l \times h^l \times k^l$, the dimensions of output volume of the forwarding convolutional layer are $d^{l+1} = w^{l+1} \times h^{l+1} \times k^{l+1}$ [5], where $w^{l+1} = \lceil (\frac{w^l + 2p^{l+1} - e^{l+1}}{s^{l+1}} + 1) \rfloor$, $h^{l+1} = \lceil (\frac{h^l + 2p^{l+1} - e^{l+1}}{s^{l+1}} + 1) \rceil$, and k^{l+1} is the number of kernels used in layer's $l+1$ filter bank. To demonstrate this in action an example of a convolutional layer is provided in Fig. 2.2.

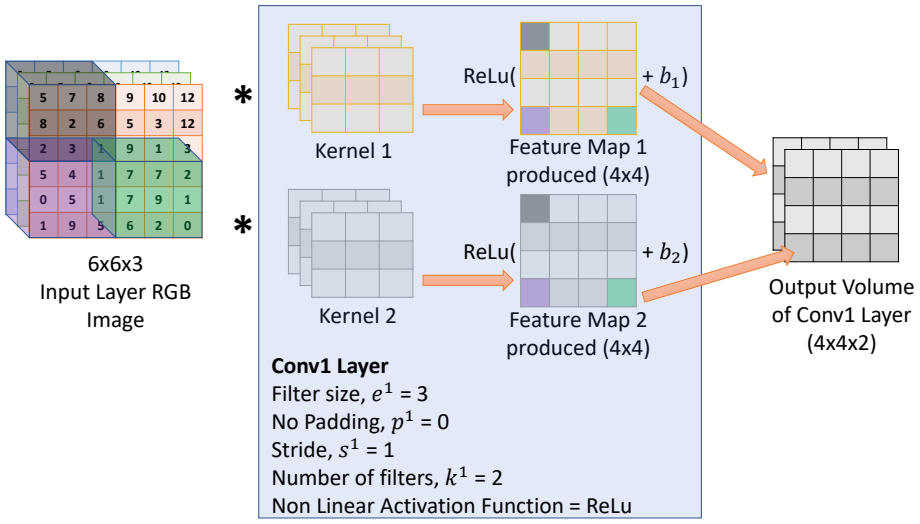


Figure 2.2: Convolutional layer example.

In Fig. 2.2, *Conv1* layer convolves an input volume, in this case an RGB image, using two different kernels. Each kernel has a squared dimension of 3×3 and 3 channels due to the input's volume depth. In this convolutional layer, convolutions are performed with a stride of 1 and no padding. Therefore, this convolutional layer produced two feature maps

with dimensions of 4×4, by another others it produced a volume of dimension $4 \times 4 \times 2$. As the output volume has a depth of 2, if there was another convolution layer connected to this output, that forwarding convolution layer would need to have kernels with 2 channels.

While the first convolutional layers tend to capture low-level features such as edges, lines and corners, the deeper layers learn high-level features by combining the low-level ones produced in the previous layers [10].

2.2.2 Pooling Layers

Most of pooling layers do not involve any learning process; they are used to reduce the size of feature maps. The input of these layers is divided into multiple elements, and units within each element are used to create a single unit of the output.

The type of operation that is performed on each element determines the type of pooling layer. This operation can be averaging, selecting maximal value, performing an Euclidean norm (also known as L2 norm) within the selected elements, etc. Selecting the maximal value is the most common type of pooling operation and, in that case, the layer is denoted as a max-pooling layer accordingly. The principle of a max-pooling operation is illustrated in Fig. 2.3.

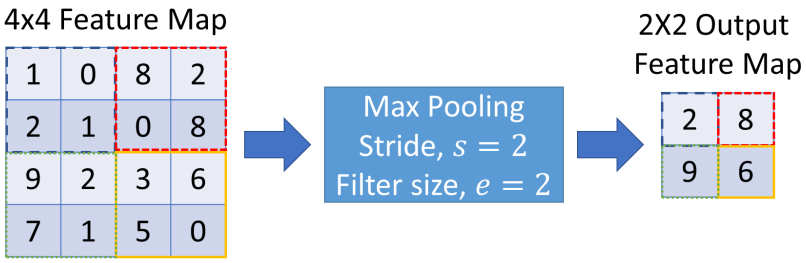


Figure 2.3: The max-pooling operation described uses a filter size, e, and a stride, s, equal to 2. If the input feature maps dimensions are even, with these parameters, pooling layers will reduce the input feature map dimensions to half.

In all cases, a pooling layer, replacing an output of the net at a certain location with a summary statistic of the nearby outputs, helps to make the feature representation invariant to small translations of the input and helps to control overfitting [6].

2.2.3 Fully Connected Layers

The output from the last convolutional and pooling layers represent high-level features of the input image. Most of these features might be good to perform the final classifications, but combinations of those might be even better. This is achieved by adding the final fully connected layers in the network. Therefore, the purpose of the final fully connected layers is to use the computed features either to acquire the final classification (in the last layer with softmax) or just to learn non-linear combinations of those.

As fully connected layers have lots of parameters (each neuron in a layer is connected to all the neurons of the previous layer), overfitting can easily happen. An efficient technique to prevent overfitting in fully connected layers is the dropout method, which can be used during the training phase [14].

2.2.4 Popular CNNs

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [15] is one of the most important challenges in computer vision and has drawn lots of attention in recent years, especially after the breakthrough success of AlexNet in 2012 [14]. Since then, several CNN architectures have been benchmarked on ImageNet challenges and the following works deserve to be highlighted by its achievements:

- AlexNet [14]: AlexNet won the 2012 ILSVRC obtaining a top-5 classification error rate[1] of 15.3%, outperforming the second best with more than 10%. AlexNet architecture has 5 convolutional layers, max-pooling layers following the first, second, and fifth convolutional layers. Two fully connected layers follow those and the network ends with a softmax layer with 1000 neurons (2012 ILSVRC image classification task had 1000 categories). This work was the first that used a GPU implementation of the convolution operation and dropout to prevent overfitting. Moreover, it was proved that ReLu is more effective than hyperbolic tangent as the nonlinear activation function in convolutional layers and that it can make training faster.

- VGGnets [16]: It has two famous architectures, VGGNet-16 and VGGNet-19. The former has been widely used due to its simpler structure, which has 13 convolutional layers, 5 pooling layers, and 3 fully connected layers. The authors of VGGNets [16] won the localization and classified second in the classification ILSVRC 2014 tracks.

- GoogLeNet [17]: GoogLeNet introduced a novel level of organization in CNNs, known as "Inception modules". In each "Inception module", different filter sizes can be used, which allows to preserve more spatial information from the input features. Despite of the deep GoogLeNet's architecture (50 convolutional layers inside 22 "Inception Modules"), this network has 12 times fewer parameters than Alexnet [14].

- ResNet [18]: The deeper a neural network is, the more difficult is to train it. This problem was tackled in [18]. The main idea behind ResNet is that each layer should not learn the whole feature space transformation but only a residual correction to the previous layer, which allows training much deeper networks efficiently and faster. ResNet extremely deep representations (up to 152 layers) obtained excellent generalization performance and led it to win several contests in 2015.

[1] In image classification tasks top-5 error rate is usually used for evaluation, classification is counted as correct if the ground truth label is among the top five classes predicted by the algorithm

2.3 Object Detection

Despite of the object's appearance variance due to change in pose, texture, illumination, deformation, and occlusion, humans can identify objects easily with low effort. Conversely, machines and robots cannot do that by themselves, and object detection remains one of the fundamentals challenges in the computer vision community.

The research work developed in object detection in the past few decades can be approximately divided into three different directions represented in Fig 2.4 [10]:

- Objectness detection (OD) desires to detect all possible objects in a given image, without classifying those in a specific category;
- Salient object detection (SOD) aims to detect objects that draw our attention in a given image. Like OD, SOD does not classify the objects in a specific category;
- Category-specific object detection (COD) has the purpose of, not only identifying the image regions that may contain the objects of interest, but also categorizing the detected object at each image region.

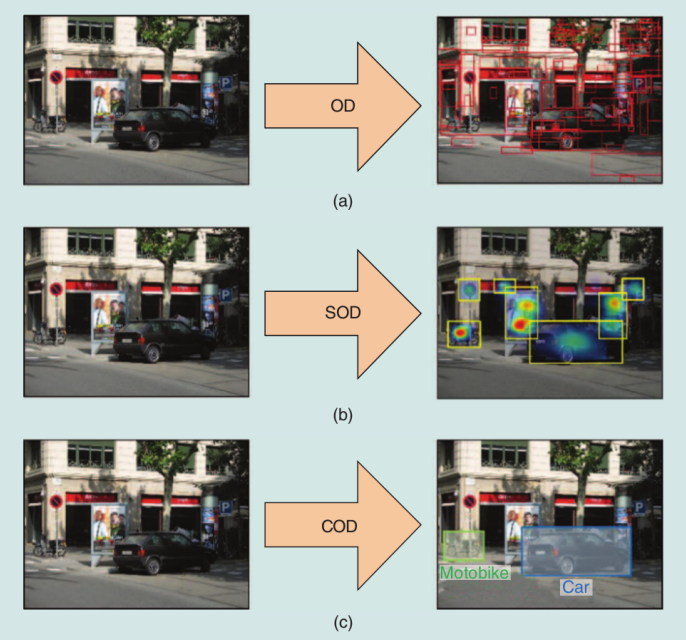


Figure 2.4: The developed semantic mapping framework should recognise different categories of objects and locate them in the semantic map. In order to map those entities, there is a need of knowing the image regions where they are contained, thus we are interested in COD methods. Image reproduced from [10].

COD frameworks can be generally divided in two main categories:

- Proposal-based COD methods: Firstly, a set of image regions that could contain objects is generated – OD. Afterwards, each one of those image regions is used by a fine-

tuned classifier to determine whether they represent backgrounds or a category-specific object.

- Regression-based COD methods: This approach predicts bounding-boxes and the object category scores in each one of those image regions in a unified classifier.

2.3.1 Proposal-based COD methods

Regarding COD's proposal-based methods, we can start by highlighting the region-CNN (R-CNN) proposed by Girshick et al. [19]. When proposed in 2014, this framework achieved significant performance improvements over all the previously published works on several benchmark data due to its rich feature extractor, a CNN. The R-CNN framework, represented in Fig. 2.5, can be described with a chain of conceptually simple steps. Firstly, 2000 region proposals (boundig boxes) that probably contain objects are generated by the selective search method proposed in [20]. Once the proposals are generated, R-CNN resizes those regions to a standard square size (i.e. 227x227). A fine-tuned CNN model, based on AlexNet [14], is used to extract features for each resized region. Next, a category-specific linear SVM classifies whether the region proposal contains an object or not, and if so what object. Finally, in order to improve object localization accuracy, the region proposal is refitted using a bounding-box regressor.

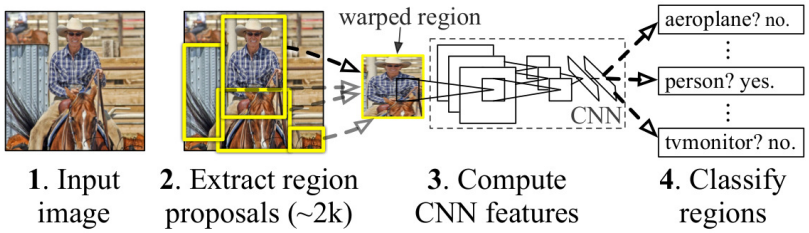


Figure 2.5: R-CNN Framework. Image reproduced from [19].

Leaving aside its achievements, R-CNN still had several drawbacks:

- A forward pass to the used CNN was required for each of the 2000 region proposals. This approach made the object detection quite slow.
- Its pipeline is extremely hard to train. Three different models are trained separately: the CNN that generates features, the SVM classifier that predicts the objects categories, and, finally, the regression model to improve precision of the bounding boxes.

In order to decrease the training and execution time of R-CNN, several methods were proposed afterwards [21–23].

The Spatial Pyramid Pooling Network (SPPnet) introduced the spatial pyramid pooling layer in order to eliminate the requirement that the input must have a fixed-size [21]. Just like that, SPPnet can generate fixed-length convolutional feature representations regardless

of the image size and scale. Furthermore, SPPnet computes the feature maps from the entire image once, and then pools the features in arbitrary-regions, such as the region proposals, to generate a fixed-length feature representation for each region proposal, thus avoiding the several CNN's forward passes to classify each of the region proposals. The major drawback of SPPnet was that the fine-tunning algorithm of SPPnet could only update the fully connected layers, which made impossible to train the CNN feature extractor and the SVM classifier together to further improve performance [10]. Towards overcoming this problem, the Fast R-CNN [22], represented in Fig. 2.6, was proposed, which is an end-to-end trainable version of SPPnet.

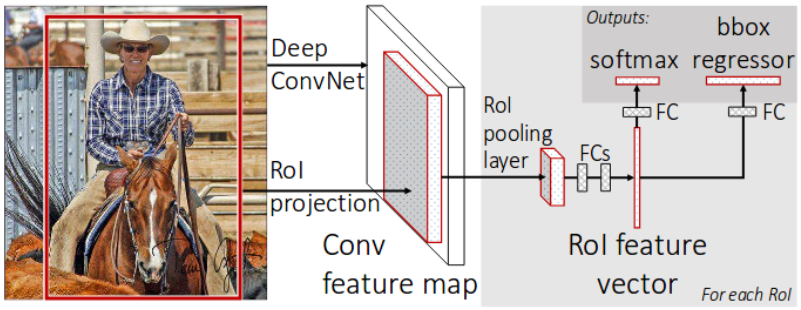


Figure 2.6: Fast R-CNN framework. Image reproduced from [22].

The Fast R-CNN replaced the SVM classifier with a softmax layer, and added a linear regression layer to obtain the refitted bounding boxes. This pipeline still had the drawback of requiring the input of regions that may contain objects - regions of interest (RoIs). Like in SPPnet, each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers. The softmax layer uses that feature vector in order to obtain the probability of its corresponding RoI containing category-specific objects. While R-CNN [19] has different models to extract the image features (CNN), classify (SVM), and refit bounding-boxes, Fast R-CNN instead uses a single network to compute all the three. This approach improved the learning process and the detection accuracy compared to the R-CNN framework.

Both R-CNN [19] and Fast R-CNN [22] pipelines need the inputs of region proposals and that turned out to be the bottleneck of the entire pipelines [23]. Faster R-CNN [23] tackled that problem by introducing the Region Proposal Network (RPN) that shares convolutional features with the detection network, thus enabling cheaper region proposals. RPN is a fully convolutional network that had the purpose of predicting RoI and its objectness scores, that are then used by Fast R-CNN for detection. RPN and Fast R-CNN are merged in a single network, sharing their convolutional features. This pipeline not only achieved better accuracy on several benchmark test data than the other frameworks previously proposed, it is also much faster due to the sharing of the same computed feature maps by the region proposal component and the final classification modules. While R-CNN runs a detection in 47 seconds, Faster R-CNN can perform it at about 5Hz [23].

2.3.2 Regression-based COD methods

Regression-based COD methods are formulated as a single regression problem with spatially separated bounding boxes and associated class probabilities [24, 25]. The multiple bounding boxes scores and class probabilities are predicted simultaneously in a single network.

These frameworks, due to the fact that they do not need the RoI's subsequent feature resampling stages, tend to be faster and have much simpler structures [25]. However, the loss functions used, when training these models, tend to be more complex as they have to address simultaneously the proposal localization and object classification problems, thus multitask loss functions are much more commonly used in regression-based COD methods [24].

You Only Look Once (YOLO) [24] and Single-Shot MultiBox Detector (SSD) [25] are the two most emblematic COD regression-based methods that can be found in the literature.

YOLO divides an input image into an $S \times S$ grid. For each object that is present on the image, one grid cell is "responsible" for predicting it, which is the cell where the center of the object falls into. Furthermore, each grid cell predicts B bounding boxes as well as C class probabilities. Each bounding box prediction has 5 components - $b_x, b_y, b_w, b_h, prob$. While b_x and b_y represent the distance between the center of the box relative to its grid cell, b_w and b_h represent the bounding box dimensions (normalized relative to the image size). Finally, *prob* reflects the confidence about the presence or absence of an object of any class in that bounding box. In that way, the network predictions are encoded as an $S \times S \times (B \times 5 + C)$ tensor. Notice that each grid cell is only responsible for detecting one object, this does not depend on the number of bounding boxes predictions. This resulted in a strong spatial constraint that limits the number of nearby objects that YOLO could predict [24]. YOLO's CNN architecture, represented in Fig. 2.7, was inspired by GoogLeNet [17] architecture used for image classification. It has 24 convolutional layers responsible by extracting features from the image, 4 pooling layers, and 2 fully connected layers that predict the object detections and its bounding boxes coordinates. Notice that the CNN architecture, represented in Fig. 2.7, was crafted for use in the Pattern Analysis, Statistical Modelling and Computanional Learning Visual Object Classes (PASCAL VOC) dataset which contains 20 object categories, therefore $C = 20$. The authors decided to use $S = 7$, $B = 2$ and that explains the size of the output tensor being $7 \times 7 \times (2 \times 5 + 20)$.

Later, SSD [25] was proposed and was able to improve detections on small-sized objects and localization accuracy, when compared with YOLO. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales at each feature map location. SSD predicts the scores for the presence of each object category in each default box and refits those to better match the object appearances. Furthermore, the network combines predictions from multiple feature maps with different resolutions to be capable of handling objects with various sizes.

More recently by performing several changes in the YOLO framework, such as modifying the network architecture and changing the methodology to estimate the bounding boxes, the

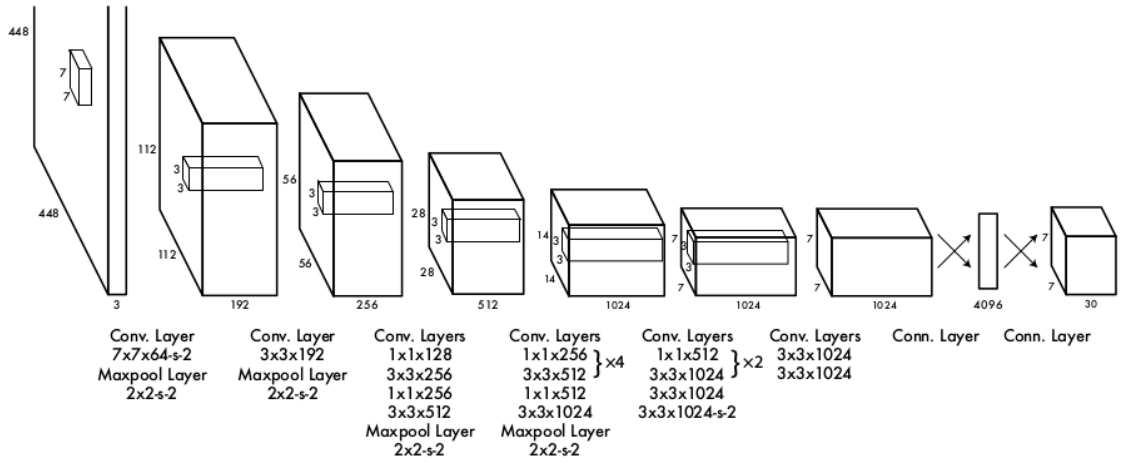


Figure 2.7: YOLO CNN Architecture. Image reproduced from [24].

authors of YOLO were able to improve its object detector not only in execution time, but also in terms of accuracy [26, 27].

2.3.3 Benchmarks and evaluation metrics

The impressive results obtained by deep learning based methods could only be achieved due to the recent availability of large datasets. The three most commonly used datasets for deploying and benchmarking COD methods are the following ones:

- PASCAL VOC 2007 [28] - This dataset contains a total of 9,963 images from 20 object categories. Those images are divided in the training and validation set (5,011) and the testing set (4,952). All of these images contain object's ground-truth bounding boxes, which were manually labeled.

- PASCAL VOC 2012 [29] - It is an extension of the previous PASCAL VOC 2007, which contains a total of 22,531 images. 11,540 images belong to the training and validation set and the remaining 10,991 made the testing set. However, the testing set does not contain ground-truth labels, therefore to evaluate a model the results obtained in the test set should be submitted to the PASCAL VOC evaluation server.

- Microsoft Common Objects in Context (MS-COCO) [30] - A newer object detection benchmark proposed in 2014 that contains more than 200,000 images and 80 object categories. The training set comprises nearby 80,000 images, the validation set 40,000 and the test set the remaining 80,000. Like in PASCAL VOC 2012, the evaluation of models in this benchmark needs to be done on MS-COCO's evaluation server.

The two most widely used metrics to evaluate object detection methods are average precision (AP) and mean average precision over all object classes (mAP). They are properly

designed to take into account false negatives, duplicated detections of one object instance and false positives.

Precision measures the fraction of detections that are correct. It takes into account the number of true positives detections (TP) and the number of false positive detections (FP):

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall takes into account the number of correct detections achieved by the object detector (TP) and the total number of instances, in which some were not detected thus corresponding to false negatives (FN):

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

Intersection over Union (IoU) measures how good a prediction is in terms of its bounding box location:

$$IoU = \frac{area(B \cap G)}{area(B \cup G)}, \tag{2.3}$$

where B is the predicted bounding-box and G is the ground truth bounding box. A detection is only considered to be a true positive if its IoU is higher than a predefined threshold.

The final model performance is represented by a precision-recall (PR) curve, and summarized by the area under the PR curve – AP or mAP if it is over all the object classes.

In PASCAL benchmarks, IoU threshold is set to 0.5. MS-COCO's challenges, when launched, used the same IoU threshold, but nowadays they average mAP over ten different IoU thresholds, from 0.5 to 0.95. This new metric emphasizes more the object's localization. When compared to the previous metric it becomes more difficult to obtain high AP/mAP values for a given model.

Tables 2.1 to 2.3 show the detection results obtained on PASCAL VOC 2012, PASCAL VOC 2007, and MS-COCO, respectively, using several state-of-the-art COD methods. Table 2.2 also compares computational costs of these methods.

Table 2.1: PASCAL VOC 2012 object detection results. VGGNet-16 is the default backbone CNN model for all methods (except for YOLO, YOLOv2 and Faster R-CNN (Resnet 101)). Results taken from [10].

Method	aero	bike	bird	boat	bott	bus	car	cat	chair	cow	table	dog	horse	mbik	pson	plant	sheep	sofa	train	tv	mAP
Fast R-CNN	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2	68.4
Faster R-CNN	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5	70.4
Faster R-CNN (Resnet 101)	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6	73.8
SSD300	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5	72.4
SSD512	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0	74.9
YOLO	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8	57.9
YOLOv2	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7	73.4

It is observed in these tables that regression-based COD methods, such as SSD [25], and YOLOv2 [26], could obtain better performances compared with proposal-based methods, such as Fast R-CNN [22] and Faster R-CNN [23]. SSD512 achieved the best accuracy on two of the three benchmarks. Table 2.2 suggests that regression-based COD methods have

Method	mAP	FPS
Fast R-CNN	70.0	0.5
Faster R-CNN	73.2	7
Faster R-CNN (Resnet 101)	76.4	5
SSD300	74.3	46
SSD512	76.8	19
YOLO	63.4	45
YOLOv2 288x288	69.0	91
YOLOv2 544x544	78.6	40

Table 2.2: Accuracy vs speed comparison on PASCAL VOC 2007 test set. Results taken from [26].

Method	mAP @0.5:0.95	mAP @0.5	mAP @0.75
Fast R-CNN	20.5	39.9	19.4
Faster R-CNN	21.9	42.7	-
ION	23.6	43.2	23.6
SSD300	23.2	41.2	23.4
SSD512	26.8	46.5	27.8
YOLOv2	21.6	44.0	19.2

Table 2.3: Object detection results on the MS-COCO test-dev2015. The VGGNet-16 is the default CNN model for all methods (except for YOLOv2). Results taken from [10].

much lower computational cost than proposal-based methods with a comparable, or even better accuracy. YOLOv2 is the most competitive real-time detector, and the fact that it can be fed with images of several resolutions allows for an easy tradeoff between speed and accuracy.

2.4 Scene Recognition

Scene recognition tries to distinguish different scene types, i.e. the goal of scene recognition is to categorize a place in which we can move and perform certain tasks, assigning semantic meaningful values such as bedroom, working space, outdoors, beach, etc., to different spaces.

Likewise the object detection problem, scene recognition is one of the hallmark tasks in the computer vision field [31] and has been extensively studied from different angles.

By looking at the pictures in Fig. 2.8, Torralba et al. found out a very interesting fact: the visual context of an image can help in performing a reliable object recognition [32].

(a) Isolated object (b) Object In context (c) Low-res Object

Figure 2.8: In (c) it is harder to recognise the coffee machine present in all the pictures. The visual context of the scene can help in categorizing the object correctly. Image reproduced from [32].

Furthermore, they proposed a vision system capable of identifying familiar locations (e.g., office 108, bathroom 32), categorize new environments (e.g., office, bath) and use that information to provide contextual priors for a reliable object detection system.

In [33], Lazebnik et al. proposed spatial pyramid matching (SPM) to include a spatial layout into bag-of-word (BoW) representations for scene recognition. Six years after, Partizi et al. [34] designed a reconfigurable version of SPM, wich associated different BoW representations to different image regions.

Pedro Sousa et. al [35] built a system capable of categorizing environments as rooms or corridors. A set of features are extracted from LRF scans, and a SVM trained in a supervised regime uses those to categorize places as rooms or corridors.

Quattoni et al. [36] studied the problem of indoor scene recognition by modeling the spatial layout of scene components. A novel CNN architecture with a spatially unstructured layer was introduced to deal with variations caused by spatial layout changes. Furthermore, they produced a new benchmark for indoor scene recognition, the MIT Indoor-67 which contains 67 categories of indoor images, with 80 images per category available for training.

Singh et. al [37] exposed a set of discriminative patches which can work as mid-level visual representations of a scene. The patches could correspond to parts of images, objects, "visual phrases", etc., but were not restricted to any of them. The learning of these visual representations was treated as an unsupervised discriminative clustering problem on a dataset of image patches.

Recently, motivated by the success of CNNs in the image classification and object detection tasks, CNNs started to be more exploited for the scene recognition task by Zhou et al. [31]. A large-scale scene recognition dataset, the Places, was produced. In this work, they also evaluated the deep feature's performance of a CNN trained on ImageNet, compared to a CNN with the exact same structure but trained on Places for the scene recognition task, and came to the conclusion that object-centric and scene-centric neural networks differ in their deep representations. Furthermore, they deduced that using a CNN trained on a scene-centric datasets allows to achieve much better results in scene classification problems. Three years later, they introduced another more challenging dataset [38] with more categories and images, called as Places2. Nowadays, Places database contains 4 subsets where each one can work as a different benchmark for scene classification:

- Places205: It was introduced in [31] and contains 2.5 million images from 205 scene categories. Each class contains from 5,000 to 15,000 images. The training set has 2,448,873 images, the validation set 100 images per category and, finally, the test set contains 200 images per category.

- Places88: This benchmark contains 88 common scene categories between ImageNet [39], SUN [40] and Places205 datasets. It is mostly used to perform comparisons across different scene-centric databases [38].

- Places365-Standard: Proposed in the second version of the Places database [38]. It contains 1,803,460 training images divided between 365 scene categories, where each category has 3,068 to 5,000 images. While the validation set contains 50 images per

class, the test set contains 900 images per class.

- Places365-Challenge: Contains the same categories as the Places365-Standard. The validation and test sets are exactly the same as Places365-Standard, however the training set is significantly larger with a total of 8 million training images.

Wang et al. [41] achieved better performances than [31] training VGGNet CNN models [16] on the Places205.

Gangopadhyay et al. [42] proposed a dynamic scene classification algorithm, CNN feature maps were extracted for each frame of the video. An aggregation scheme was used in order to obtain a robust feature descriptor for the video and classify the video-scenes.

Table 2.4 presents the classification results obtained by several CNN architectures, trained on Places205, on several benchmarks.

Table 2.4: CNN architectures trained on Places205. The model's performance is described by the top-5 accuracy rate, wherein one test sample is counted as correctly classified if the ground-truth label is among the top 5 predicted labels of the model, or by the top-1 accuracy where the test sample is counted as correctly classified only if the top predicted label is the ground-truth label. When validating the models on different benchmarks than the Places205, the final classification layer of the CNNs (softmax layer) was replaced by a linear SVM which uses the CNN computed features to perform the final classifications [38]. SUN205 represents a subset of the SUN397 dataset which contains the similar 205 categories of Places205. Values taken from [38, 41, 43].

CNN architecture	Places205		SUN205		SUN397	MIT67
	top-1	top-5	top-1	top-5	top-1	top-1
AlexNet	50.04	81.10	67.52	92.61	54.32	68.24
GoogLeNet	55.50	85.66	71.6	95.01	57.00	75.14
VGGNet-11	59.0	87.6	–	–	65.3	**82.0**
VGGNet-13	60.1	88.5	–	–	66.7	81.9
VGGNet-16	60.3	88.8	**74.6**	**95.92**	**66.9**	81.2
VGGNet-19	**61.2**	**89.3**	–	–	–	–

2.5 Semantic Mapping

In order to improve robot's intelligence and human robot-interaction, semantic mapping has been widely researched in the last few years.

Galindo et al. [3], divided the environment representation in two perspectives: a spatial perspective used for path planning and navigation, and the semantic perspective endowed with human-like knowledge and inference capabilities on symbolic data (e.g., a bathroom

is a room that contains a toilet). In Fig. 2.9, those two perspectives are represented and explained. Using this framework, Galindo et al. [3], were capable of, not only performing place categorization based on the occurrence of certain types of objects in the room, but also ring which objects are usually found in a specific room.

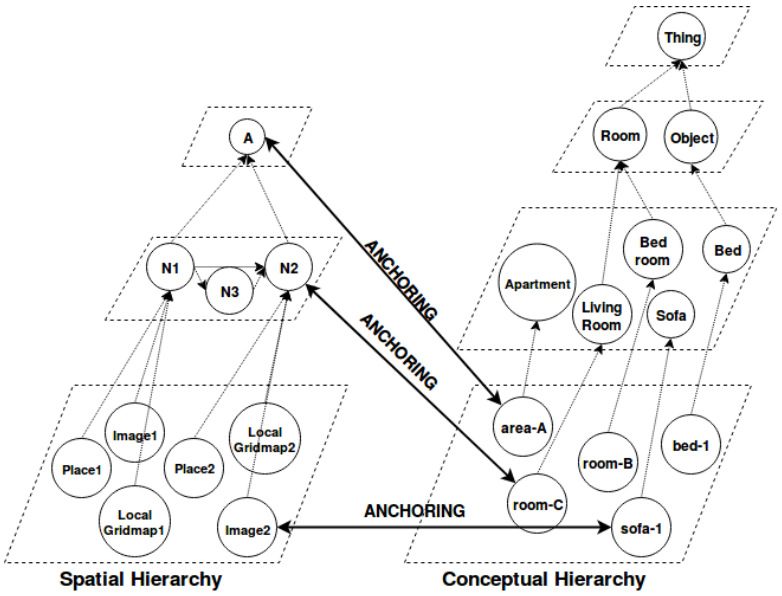


Figure 2.9: On the left, its represented the Spatial Hierarchy, that contains sensor information in different levels of detail – from simple sensory information like grid maps and camera images, to the topology of the environment, to the whole environment represented on an abstract node. On the right, the Conceptual Hierarchy contains semantic information, this hierarchy represents concepts (categories and instances) and their relationships, modeling the knowledge about the robot's environment. The anchoring concept was used to relate the two perspectives of the environment. Image reproduced from [3].

Zender et al. [44], proposed a novel spatial representation for indoor mobile robots, divided into three different layers, representing different levels of abstraction from sensory input to human-like concepts. It was the first framework that used geometric cues and human assertions as sources of semantic information. The spatial representation proposed is illustrated in Fig. 2.10.

In [45], a fully probabilistic representation of space was used. Objects were used to capture the spatial semantics. Clustering and Bayesian network classifiers were utilized to learn the object-places relationships, enabling the robot to conceptualize and classify its environment based on the occurrence of objects.

In [46], the first semantic mapping framework that used a 3D laser scanner was proposed. A 6D SLAM technique was used to build a 3D map of the environment. Some coarse scene features, such as walls and floors, were extracted from the map and registered, and more delicate spatial entities such as objects were detected by a trained classifier, localized and registered in the final 3D semantic map. Similar 3D semantic mapping techniques were

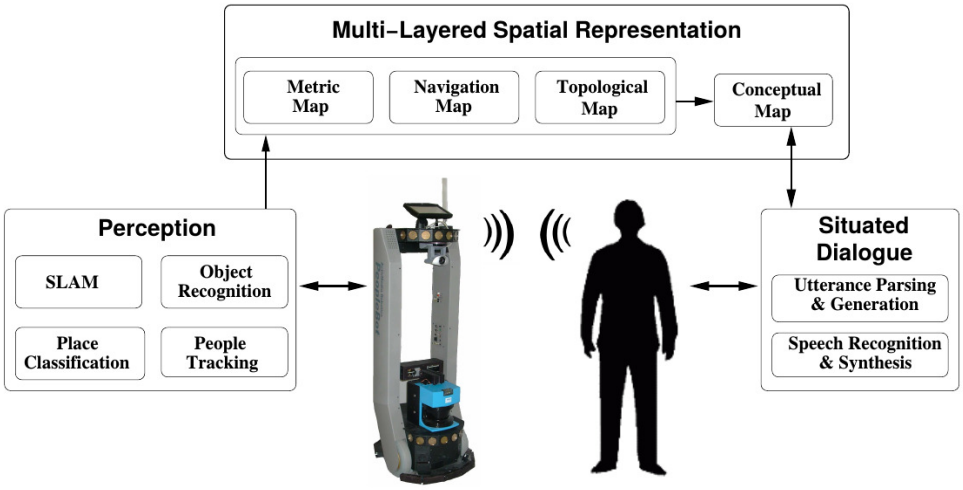


Figure 2.10: The perception subsystem is meant to sensory input evaluation, the communication system is used for situated spoken dialog, and the multy-layered conceptual mapping subsystem made the connections from sensor-based maps and human-like spatial representations. The metric, navigation and topological maps are built based on the information from the perception layer. The conceptual map is built reasoning about those maps and about the human inputs. Image reproduced from [44].

afterwards proposed and can be found in [47, 48]. In [47], a CNN was used to predict the semantic classes of each cell of the 3D map using the RGB-D images, while in [48] a 2-D semantic segmentation of the RGB-D images is done using Random Decision Forests (RDFs) and 2D dense Conditional Random Fields (CRFs), finally a 2D-3D label transfer is executed in order to obtain the final semantic segmentation of the 3D map.

Nieto-Granda et al. [49] built a 2D semantic mapping framework where each grid cell had a probability distribution, modeled as a Gaussian model, of a set of semantic labels. Each of the Gaussians in their model was based on robot's sensor data when it was provided a label by a human guide. In their work, humans were responsible to provide the correct labels for the robot positions (e.g., office, corridor). If the robot had not enough confidence that he was on a region, with a known label, it would sign that and request a human operator to input the label of the current region. Furthermore, with the map properly built, a human had the possibility to request the robot to navigate to a known position like "office".

Niko Sünderhauf et al. [2] contributed with a ROS package for the semantic mapping and place categorization problem. A CNN, trained in the Places205 dataset [31], classified the images individually into their respective classes. A Bayesian filter was responsible for maintaining temporal coherence and removing spurious false classifications. Their mapping subsystem gradually builds the semantic map with the resulting place labels assigned for each grid cell.

Andrezj Pronobis and Patric Jensfelt [50] proposed a fully probabilistic semantic mapping framework that used more sources of information than all the ones referred before. Their

system used the occurrence of objects, the size, shape, appearance, and topology of rooms, and also human assertions as input for the semantic mapping system. Moreover, a chain-graph is used to represent all of these semantic properties on a conceptual map and perform inference over it. With this representation, their system is able to infer room categories, predict existence of objects and values of other spatial properties, as well as reason about unexplored space.

Another 3D semantic mapping framework was recently proposed in [51]. The solution to this problem incorporated a CNN and a state-of-the-art SLAM system, ElasticFusion [52]. ElasticFusion provided correspondences from 2D frames into a globally consistent 3D map, which allowed the CNN's semantic predictions from different viewpoints to be probabilistically fused into a dense semantically annotated map. An example of the final results obtained by this framework is presented in Fig. 2.11.

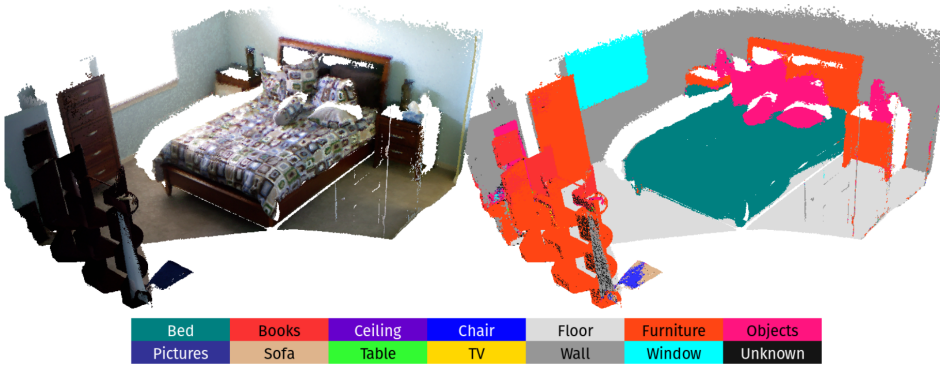


Figure 2.11: Dense 3D Semantic Mapping with Convolutional Neural Networks, SemanticFusion - Results example. Image reproduced from [51].

The previously presented works differ in terms of which semantic properties are extracted from the environment, in the way the reasoning is done about them, and finally, in the way they are annotated in the final semantic map. From that point of view, it is extremely hard to define metrics and benchmark all of these pipelines to conclude which is better than the others. However, it is a fact that the higher number of information sources that a semantic mapping framework takes advantage of, the more effective it can be:

- Object Occurrences: The number of useful actions that a robot can perform highly increases if a robot is capable of recognizing objects, and even more if the robot has in memory the locations where certain objects can be found. Object occurrences can also be useful to categorize the different scenes in the robot's environment.

- Place categories: Acknowledging the different places categories contained in the robot's environment can also be very helpful in human-robot interactions (e.g. even if a robot does not know where a toothpaste can be found, it should look for it in a bathroom firstly).

- Room's topology: To infer place categories without visiting those, a robot can also use room's topology information (e.g., the robot knows that a certain room is connected to a large number of other rooms, by that order it is very likely that that room is a corridor).
- Place geometry and shape: In order to improve robot's navigation capabilities it can be useful to know the shape and geometry of places (e.g. a robot should not navigate in the same manner in elongated and large environments as it navigates in narrow places). Moreover, if a robot knows that there is a narrow room, likely to be full of people, between its location and its navigation goal, it should avoid passing in that room, otherwise it can be chaotic to reach its destination.
- Human assertions: The different spatial properties acquired might be wrong or incomplete. Human assertions can be used to increase the robots knowledge about its environment, or to correct some misclassifications that might lead to dangerous situations.

Taking into account the number of information sources used for the creation of the semantic map, Pronobis et. al work [50] is the one that stands out the most because their framework uses all of the knowledge's sources listed above. In terms of the reasoning performed about the spatial properties acquired, their work stands out again: it is the only one capable of reasoning about unexplored spaces and fuses all the knowledge sources in order to verify if a certain property acquired makes sense or not. Nevertheless, 3D semantic map representations such as [46–48, 51] can be very interesting also, as they can contain more detailed information about the objects present in the robot's workspace such as its shape and 3D dimensions.

2.6 ROS

The production of software for robots is becoming increasingly difficult as the scale and scope of robotic applications are increasing in a continuous manner. There is a need for structures that can ease the development and reuse of software for robotics purposes. This is the main goal of the Robot Operating System (ROS) [4], which was used to implement the semantic mapping system described in chapter 3. ROS is an open-source robotics middleware (i.e. a collection of software frameworks) that offer many tools for communication between robots and other devices or software through a TCP/IP network. One of the major advantages when using ROS is its TCP/IP based communication network. There are a few key concepts that one should master in order to understand the ROS' communication protocol:

- Nodes: They work as processes that can perform certain computations, execute some tasks and communicate to other nodes thanks to the ROS network. They are registered to the network with a unique id and a list of topics and/or services that they want to use to send/receive messages to/from.

- Master: The master is a special node that is launched when the ROS network is started. It has the goal of handling registration, subscription and disconnection of every other nodes in the network. Links for each topic or service are also created by the master, so that messages can arrive to its targets successfully.

- Messages: Represent packets sent over the network. Each message is defined by a data structure containing the typed fields in that message. Furthermore, a message can also have in their fields other message types.

- Topics: Topics are a means of asynchronous communication. A node can receive messages subscribing to the topic where those messages are being published. There is no limit on the number of topics where a node is publishing or subscribing. Messages, sent by a publisher, will be received by all the subscribers of that topic. Each published message on a topic has a template, and it is not possible to send other kind of messages throughout that topic.

- Services: Unlike topics, services provide synchronous communication between two nodes. A node that uses a service will only receive data in response to the requests it sends. The response may very depending on the request sent, thus service messages describe both the request and response message types.

ROS organizes software in packages. Each package might contain ROS nodes, configuration files such as message types, independent libraries, or anything else that logically might constitute a useful module. Each package should provide useful functionalities in an "easy-to-consume" manner so that the software contained in that package could be easily reused. The introduced concepts and ROS architecture are illustrated in Fig. 2.12.

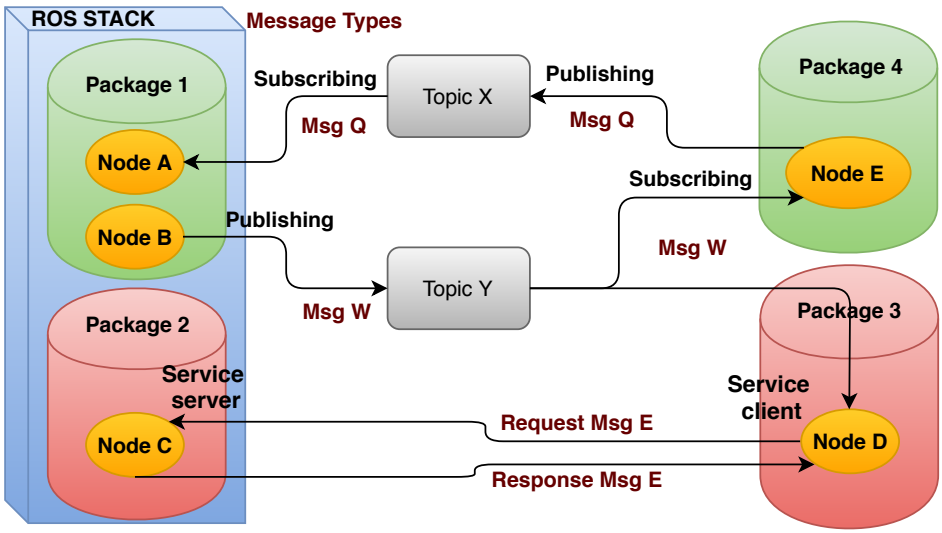


Figure 2.12: ROS architecture and key concepts

2.7 Kinect sensor

In order to compute an estimate of the objects' localization, the system presented in chapter 3 uses a RGB-D sensor: the Microsoft Xbox 360 Kinect. Kinect was originally designed as a video game peripheral, and it is composed of an RGB camera and a depth sensor.

The depth sensor is composed of an infrared (IR) emitter and a IR camera [53] to detect the depth of the corresponding observed surfaces. The emitted IR pattern is captured by the IR camera and is correlated against a previously defined reference pattern [54]. The reference pattern is obtained when calibrating the camera by capturing a plane at a known distance from the sensor, and is stored in the sensor's memory. When a speckle is projected into an obstacle, whose distance to the sensor is higher or smaller than that of the reference plane, its position in the IR image will be shifted accordingly to that difference. A disparity image can be obtained by calculating all the speckle's shifts. The depth of each image's pixel is computed, using the disparity image and the reference's plane depth, with a triangulation model described in [54]. Finally, a 3D Pointcloud representing the observed scenario, can be obtained using the depth image.

The depth resolution is not constant. It is inversely proportional to the distance from the sensor to the obstacle, thus the depth information for objects farther away from the camera is less precise than those which are closer [54]. Table 2.5 presents some Kinect specifications.

Table 2.5: Kinect Specifications. Values retired from [55].

Specification	**Value**
RGB and Depth Image Resolution	640×480
Frame rate	30Hz
Ideal Depth Operating Range	0.8m - 4.6m
RGB Horizontal and Vertical Angles of View (AoV)	62°/ 48°
Depth Horizontal and Vertical AoV	58°/ 44°

2.8 Summary

Throughout this chapter, an overview of the theoretical foundations and relevant methods for this dissertation was presented. The object detection and scene recognition problems were introduced and described. The semantic mapping problem and which related approaches can be found in the literature were also presented. Finally, two key tools in the course of this work – ROS and the Kinect sensor – were also introduced. In the next chapter, the solution developed towards the semantic mapping problem is presented in detail, which makes use and combines methods, techniques and tools presented throughout this chapter.

Chapter 3

Proposed Semantic Mapping System

The semantic mapping framework proposed in this chapter acquires and represents in a semantic map two different spatial properties: object occurrences and place categories. These two semantic properties were selected as they can greatly increase the number of human-robot interactions where a robot may come to intervene. Furthermore, these properties correspond to human known concepts and can provide a layer of spatial semantics shared between the user and the robot.

Fig. 3.1 presents a general overview of the system, clarifying system components and their dataflow.

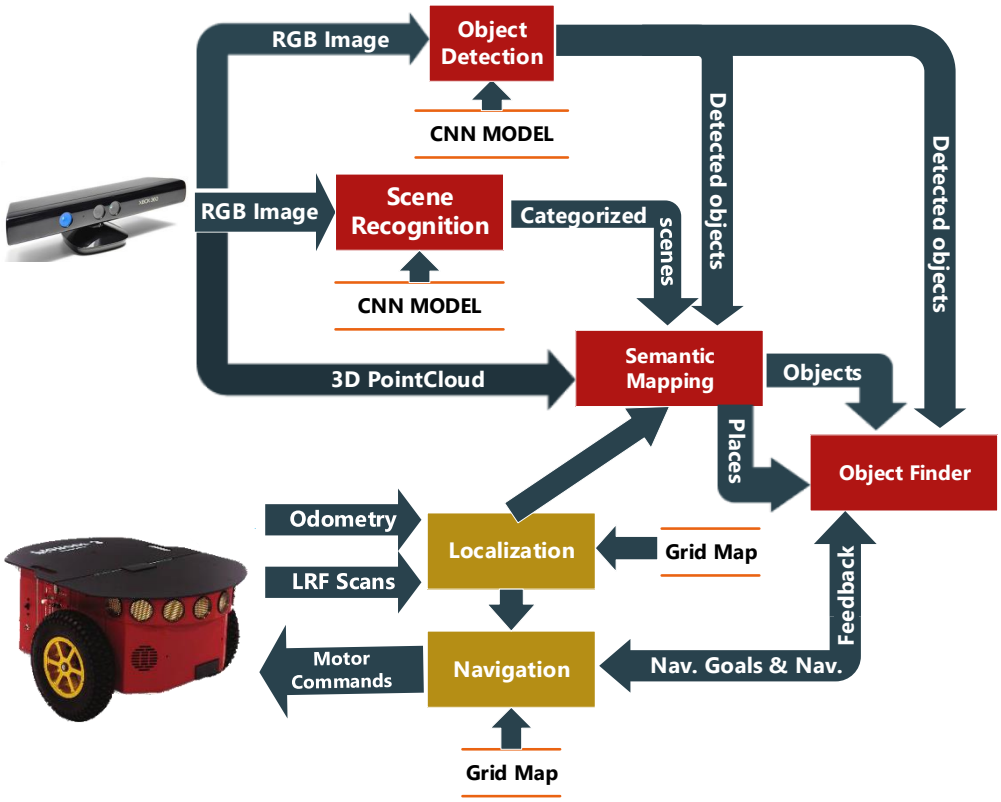


Figure 3.1: System components and dataflow.

The developed semantic mapping system can be divided into six major modules: *Lo-*

calization and *Navigation*, which are out of the scope of this work, *Object Detection*, *Scene Recognition*, *Semantic Mapping* and finally the *Object Finder*.

While exploring a known environment, the system constantly gathers RGB images and 3D Pointclouds provided by the RGB-D sensor. The RGB images are used to detect objects and to classify the different scenes observed. Those classifications flow to the *Semantic Mapping* component which combines that with the localization of the robot, place segmentation information, several frame transformations and Pointclouds in order to register object occurrences and place categories in a temporal coherent semantic map. Finally, the mobile robot can also search for objects. In order to find the desired objects the robot takes advantage of the semantic map and of common-sense knowledge rules given in the form of an ontology (e.g. even if the robot does not have information relative to the occurrence of a banana in the semantic map, it will search for bananas firstly in kitchens).

The developed *Object Detection*, *Scene Recognition*, *Semantic Mapping* and *Object Finder* components, represented in Fig. 3.2, are distributed into four ROS packages, forming a ROS stack called *Semantic Mapper MRL-ISR*, which corresponds to the main contribution of this dissertation work.

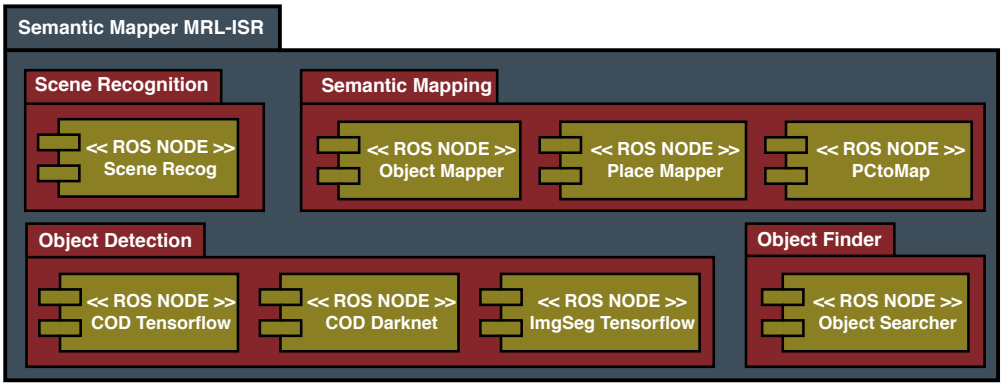


Figure 3.2: *Semantic Mapper MRL-ISR* ROS' Stack components. The developed ROS nodes comprised in each package are written in Python. *PCtoMap* node is an exception and is written in C++.

Each developed ROS package will be explained in greater detail throughout the following sections.

3.1 *Object Detection* Package

Deep learning is rapidly evolving and new CNN based COD methods that outrun COD's current state-of-the-art are constantly being produced and made publicly available for deep learning frameworks such as *TensorFlow*[1], *Darknet*[2] and *Caffe*[3]. This was a major concern

[1]Additional information about *TensorFlow* is available at `https://www.tensorflow.org/`
[2]Additional information about *Darknet* is available at `https://pjreddie.com/darknet/`
[3]Additional information about *Caffe* is available at `http://caffe.berkeleyvision.org/`

when building the *Object Detection* package: the developed software should easily follow these changes, facilitating the use of future models proposed. For that purpose, *TensorFlow Object Detection API* is used in part of the *Object Detection* package.

At *TensorFlow Object Detection*[1] *API* github's repository, several models trained on MS-COCO, such as SSD and Faster R-CNN models, are constantly made available. However, as the system might also be rewarded by using YOLO and as YOLO is only available for *Darknet API*, two different ROS nodes, represented in Fig. 3.3, were built. While *COD Tensorflow* node can work with models, trained on MS-COCO, prepared to be loaded using Tensorflow's libraries such as the ones made available at the *Tensorflow Object Detection API*, *COD Darknet* node can load and use YOLO models, also trained on MS-COCO.

Despite of Image Segmentation not being deeply studied in this dissertation, a ROS node capable of using Image segmentation models, also available in the *Tensorflow Object Detection API* and trained on MS-COCO, was developed. Those models also generate segmentation masks for each instance of a detected object in the image, which could then be used, in future studies, to incorporate objects' 6D poses and objects' 3D reconstructions in the generated semantic map.

All this package components and a state machine describing their behavior is represented at Fig. 3.3.

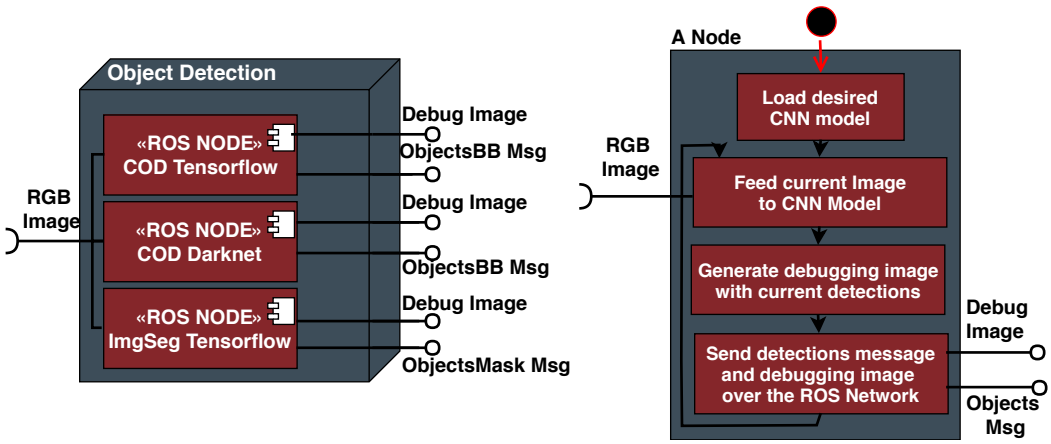


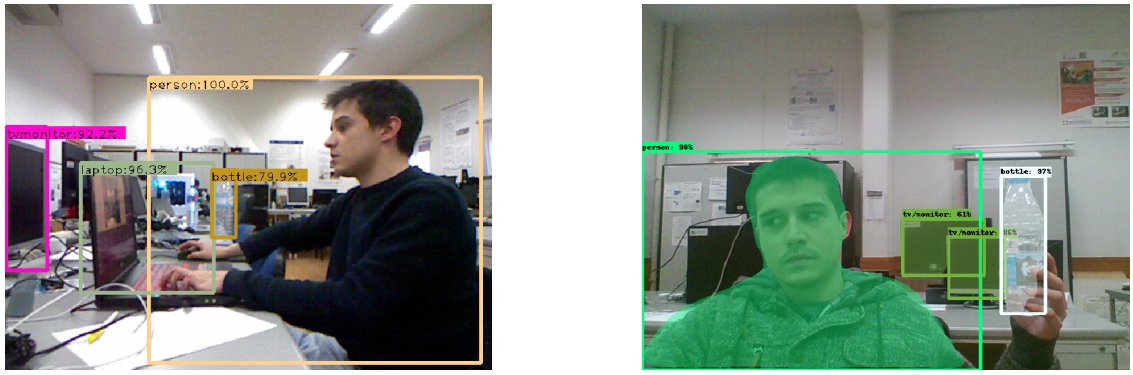
Figure 3.3: *Object Detection* package's components and functionalities.

COD Tensorflow, *COD Darknet* and *ImgSeg Tensorflow* nodes work in a similar manner: they load the desired object detection models and use them to detect objects in the images gathered by the robot's camera.

After each detection, an image, such as the ones represented in Fig. 3.4, with the detected objects and its regions highlighted (i.e. masks and/or bounding-boxes) is produced and published in a ROS topic, so that one could see how the object detector is performing.

The detected objects confidences and their image regions, i.e. bounding-boxes in case

[1]Additional information about *TensorFlow Object Detection* API is available at `https://github.com/tensorflow/models/tree/master/research/object_detection`



(a) YOLOv3 loaded by *COD Darknet* node

(b) Mask R-CNN ResNet50 loaded by *ImgSeg Tensorflow* node

Figure 3.4: Generated debugging images with the object detections.

of *COD Darknet* node and *COD Tensorflow* node or bounding-boxes and masks in case of *ImgSeg Tensorflow* node, are also sent over the ROS network, via topic publishing, so that other nodes can use those informations for their own purposes.

By using CNN models trained on MS-COCO with the developed Object Detection ROS package, the system is capable of detecting 80 different types of objects such as remotes, laptops, monitors, chairs, persons, bottles, backpacks, bicycles, umbrellas, dogs, apples, etc.

3.2 *Scene Recognition* Package

As discussed in section 2.4, Deep Learning has also achieved exceptional results in scene recognition problems. For this reason, *Scene Recog* node was developed with the purpose of working with several CNNs trained on Places205, namely GoogleNet, AlexNet and VGGNets which are available for *Caffe* framework.

Scene Recog node works in a similar manner to the nodes comprised in the *Object Detection* package. It loads the desired CNN model (GoogleNet, AlexNet, VGGNet11, VGGNet13, VGGNet16 or VGGNet19)[1], and classify the images, gathered by the robot's camera, amongst the 205 scene categories contained in Places205 dataset.

Moreover, the obtained classifications and a generated debugging image, such as the one represented in Fig. 3.5, are transmitted to the ROS network.

3.3 *Semantic Mapping* Package

The ROS nodes included in the *Semantic Mapping* package use the spatial properties acquired by the two previously described packages, i.e. object detections and scene categoriza-

[1]These models are available at `https://github.com/BVLC/caffe/wiki/Model-Zoo` and `https://github.com/wanglimin/Places205-VGGNet`

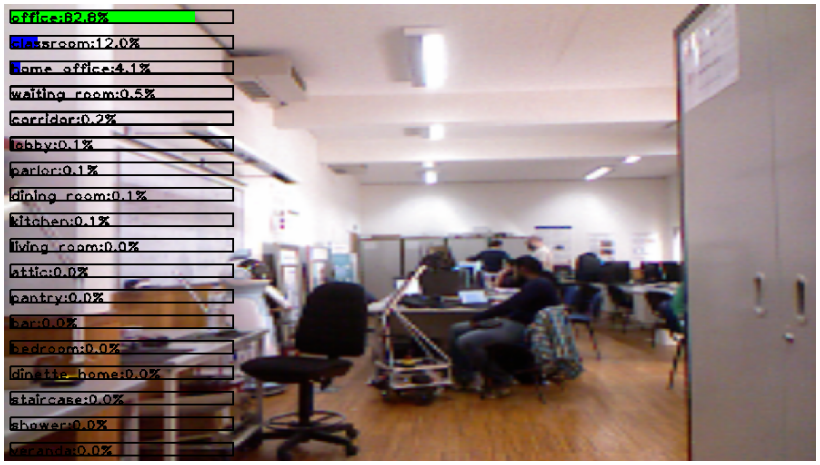


Figure 3.5: In this example, VGGNet19 is classifying the Mobile Robotics Lab (MRL) of ISR - University of Coimbra. It had 82.8% confidence that MRL was an office, 12.0% that it was a classroom, etc.

tions, and integrates them in order to properly register and keep track of object occurrences and place categories in the semantic map.

The object occurrences and place categories differ in the way they are integrated, used, and visualized in the semantic map, thus the *Semantic Mapping* module has two major components (two different ROS nodes), the *Objects Mapper* node and the *Place Mapper* node. While the former is designed to deal with object occurrences in the robot's workspace, the latter has the goal of taking care of place categories in the semantic map.

The required and provided interfaces of the nodes comprised in the *Semantic Mapping* module are represented in the components diagram in Fig. 3.6.

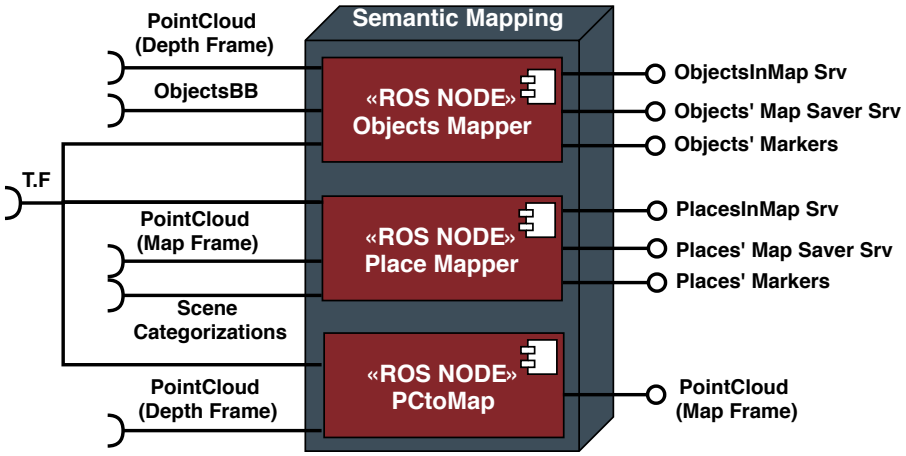


Figure 3.6: *Semantic Mapping* package components diagram

3.3.1 *Objects Mapper* Node

As can be observed in Fig. 3.6, the *Objects Mapper* node uses the object detections published by the used object detector node. Furthermore, it combines the object detections with Pointclouds and frame transformations in order to locate, register, and keep track of object occurrences in the semantic map. The system does not perform 3D reconstructions of the detected objects and the generated semantic map does not have information relative to objects' 3D dimensions, orientation and shape. It is not also a 3D semantic map; nevertheless the objects' localizations, i.e the objects' 3D geometrical centers, are predicted.

For visualization and debugging purposes, as illustrated in Fig. 3.7, object occurrences are annotated in the semantic map adding markers on top of the metric map. Each marker observed in the semantic map might represent a different object occurrence. Markers characteristics differ for each object class (e.g. persons are represented with a green cylinder with 1.80m of height and a radius equal to 0.50m, apples with a green sphere with radius of 0.15m, etc.). The marker dimension and orientation used to represent the detected object is only for visualization purposes and do not depend on the real object dimension and orientation, because these properties are not acquired by the system. Moreover, the system adds text markers describing the object categories to ease the reading and understanding of the built semantic map by human operators.

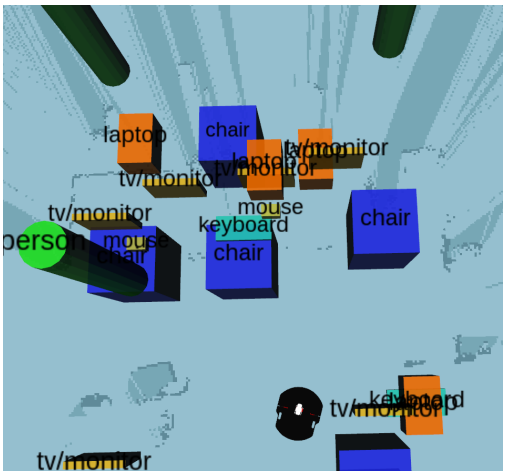


Figure 3.7: Semantic Map Visualization - At this point the system acknowledge the presence of 3 persons, 5 chairs, mouses, keyboards, monitors, etc.

The objects' markers are sent to the ROS network and can be visualized using *RVIZ*[1], this node functionality is represented in Fig. 3.6 with the ***Objects' Markers*** provided interface.

This node is also the server of two ROS services, already represented in Fig. 3.6:

- ***ObjectsInMap Srv***: Clients of this service, depending on their request message, can

[1]RVIZ is a 3D visualizer for displaying sensor values, camera's data, state informations, etc, from ROS Topics. More information about RVIZ is available on the ROS wiki at `http://wiki.ros.org/rviz`

obtain several information about the object occurrences registered in the semantic map. A client of this service can query:

- The full list of object occurrences known in the semantic map;
- Information about object occurrences of a certain category, e.g. information about laptops in the semantic map;
- Information about object occurrences located within a desired distance of the robot, e.g objects within 5m of the robot;
- Information about object occurrences in a specified bounded area of the map, e.g. objects located in $(x > -3) \wedge (x < 12) \wedge (y > 5) \wedge (y < 17)$.

- ***Objects' Map Saver Srv***: This service can be called to save the object occurences registered in the semantic map in a file. The generated file can then be loaded when launching this node. With this, when the robot resumes the exploration of its workspace, it can already know the presence of objects in certain previously explored places, not having to explore its workspace from scratch.

Objects' Localization

The object's 3D geometrical center is estimated using the Pointcloud's patch that contains the 3D points layed on the bounding box of the detected object. The full procedure used to estimate the localization of an object is illustrated in Fig. 3.8.

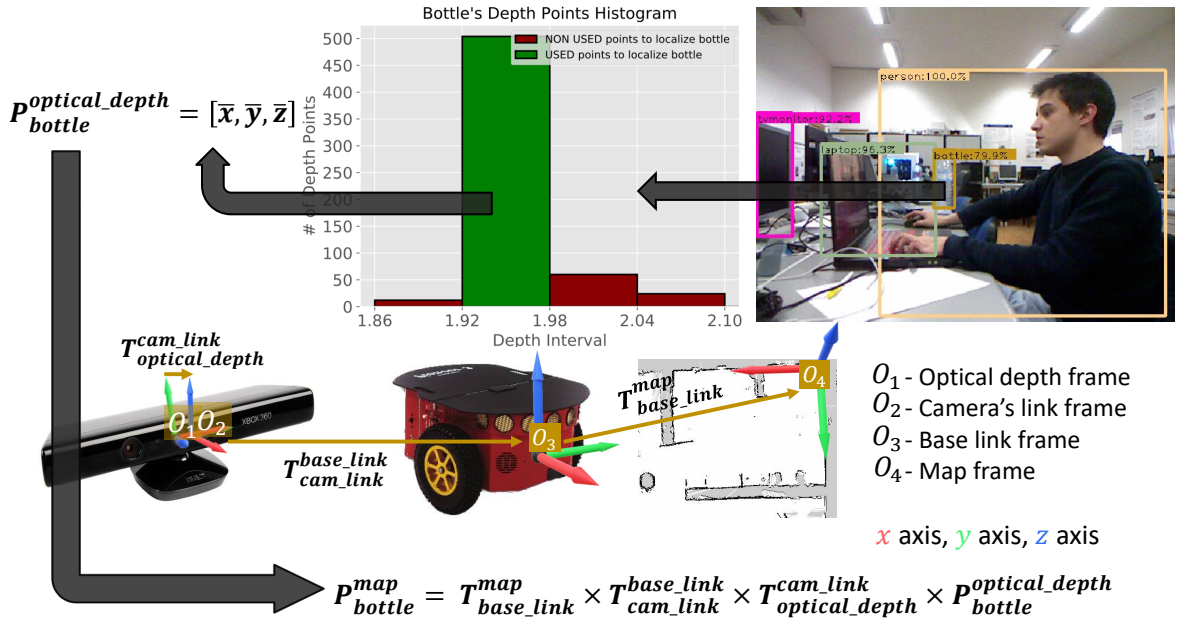


Figure 3.8: Objects localization method.

An histogram of the Z (depth) coordinates, in the camera's optical depth frame, is done on the object's corresponding Pointcloud's patch. Afterwards, it is obtained the depth's interval where most of the patch's points belong. The points that lay on that interval are used to estimate, performing an average mean of them, the 3D localization of the object in the camera's optical depth frame. Finally, the object localization, in the map frame, is obtained using a chain of frame transformations:

$$P_{object}^{map} = T_{base\_link}^{map} \times T_{cam\_link}^{base\_link} \times T_{opttical\_depth}^{cam\_link} \times P_{object}^{optical\_depth}$$

$T_{cam\_link}^{base\_link}$ and $T_{optical\_depth}^{cam\_link}$ are static transformations where the former corresponds to the transformation between the camera's link and robot's base link (it depends on the camera's position and orientation relative to the robot), and the latter is the transformation between the camera's link and the camera's depth frame (depends on the RGB-D sensor used). $T_{base\_link}^{map}$ transformation tell us where the robot is located in the map (it is provided by the localization module used).

A crucial step in the developed objects' localization method is the histogram's bin size selection. The bin's size should not depend only on the object's image region size, but also on the distribution of the depth points in that image region. For instance, in Fig. 3.8, the bootle's depth points do not have a big variance, however the depth points contained in the person's bounding box have a bigger variance, which difficult the person's localization estimation using this approach. Choosing a proper function to estimate the histogram bin size could allow us to obtain better results using this method. The developed objects' localization method was evaluated in section 4.1 using three histogram bin estimators that take into account the data's size and interquartile range (*Freedman-Diaconis* [56]), or data's size and standard deviation (*Scott* [57]), or data's 3rd-moment skewness (*Doane* [58]). The test scenarios were inconclusive and the developed method achieved similiar performances and predictions with the different histogram bin estimators. By default, the system uses the *Freedman-Diaconis* function to estimate the number of histogram bins, however there is a node parameter which one could later change to test and use the system with one of the other two functions.

Treating object occurrences in the semantic map

The developed system treats objects as dynamic properties of the environment. At a certain time instant objects can be somewhere located but they can move to other places, or just be consumed and disappear. Consequently, after registering an object in the semantic map, the system should constantly verify if the known objects are still in their locations.

From different views, i.e. different robot poses, the localization estimated of the same object can be different. The developed object localization method has a certain error associated, which difficults the task of finding out if the incoming object detections correspond to already known and mapped objects, or objects that the system did not knew

they exist. To tackle this problem, it is assumed that an incoming object detection is referring to an object already registered in the semantic map, if these two are of the same object class and have an estimated localization difference smaller than a certain threshold (e.g. a monitor is detected in the current image and the system predicts its localization $P_{d\_monitor} = [x_d, y_d, z_d]$, there was already a monitor registered at $P_{o\_monitor} = [x_o, y_o, z_o]$, if $dist = \sqrt{(x_d - x_o)^2 + (y_d - y_o)^2 + (z_d - z_o)^2} < \kappa$ the system considers that the monitor detection is actually referring to the already known monitor, thereafter it will just going to update that known object occurrence's location and confidence, and will not add a new monitor to the map). After several experimental tests, κ was defined to be equal to 75cm.

If an incoming object detection corresponds to an object already registered in the semantic map, the object occurrence localization is updated with an iterative arithmetic mean over all the localizations estimated in the difference detections, for the known object. The object occurrence confidence, C_k, is updated taken into account its previous confidence, C_{k-1}, and the confidence in the current detection (softmax probability), D_k, using the following heuristic:

$$C_k = \max(D_k \cdot \phi + C_{k-1}, \nu) \tag{3.1}$$

C_k does not represent a probability and the maximum value it can take is equal to ν. ν and ϕ are constants that should be tuned given the used object detector's model accuracy and speed. For instance, when using YOLOv3, good values for ν and ϕ turned out to be $\nu = 3$ and $\phi = 0.8$.

If a detected object was never recognized by the system, the system registers that object occurrence on the semantic map. The confidence in that object occurrence is also set using eq. 3.1 where C_{k-1} is assumed to be zero.

While evaluating the system, it was noticeable that the object detectors can sometimes generate duplicate detections of the same object, e.g. the same chair is detected as two or more chairs that are nearby one another. When this happens, the detections of the same chair lead to nearby locations. The system tries to prevent the semantic map from having duplicated object occurrences registered on it by making some assumptions:

- When two or more object detections are referring to the same object occurrence, i.e they are nearby an object of the same class already registered on the semantic map, the system will only use the one that is nearest to the object already registered to update its confidence and position. The other detections will not be used for any purpose as the system will "think" it is dealing with a duplicate detection and it is not beneficial of registering that on the semantic map.

- When two or more object detections are not referring to any object occurrence already registered, but they are of the same class and are nearby one another, the system will only register one object on the semantic map which the corresponding detection is the one that have higher confidence. Once again, the other detections are discarded in order to prevent duplicated occurrences on the semantic map.

The objects' dynamic feature leads the system to reflect on a question not as trivial and easy to answer as it seems at a first glance. After registering an object in the semantic map, if the system is no longer detecting such object and such object is within the camera's field of view (FoV), is the object now occluded or has it disappeared? Only with a 3D dense reconstruction of the registered objects it would be possible to infer the portion of objects' surface points that are represented in robot's view, and try to infer mathematically if an object already registered in the semantic map is not being detected due to total occlusion, partial occlusion or due to the fact that it is no longer more in its predicted location.

To deliberate if an object, already registered on the semantic map, should be being detected, the system firstly verifies if its estimated 3D geometrical center, in the camera's optical depth frame, $P_{obj}^{opt\_depth} = [P_{obj_x}^{opt\_depth}, P_{obj_y}^{opt\_depth}, P_{obj_z}^{opt\_depth}]$, is within the camera's FoV:

$$P_{obj_z}^{opt\_depth} > 0, \theta_x = \arctan\left(\frac{P_{obj_x}^{opt\_depth}}{P_{obj_z}^{opt\_depth}}\right) < \left|\frac{FoV_x}{2}\right|, \theta_y = \arctan\left(\frac{P_{obj_y}^{opt\_depth}}{P_{obj_z}^{opt\_depth}}\right) < \left|\frac{FoV_y}{2}\right| \tag{3.2}$$

If these conditions are all verified it is assumed that the object is totally in the camera's FoV. To deliberate if an object in the camera's FoV is not being detected due to occlusion, the system uses its localization in the camera's optical depth frame, $P_{obj}^{opt\_depth}$. Considering the camera's pinhole model, the system then uses the camera's intrinsics matrix $\mathbf{K}$ to project the object's localization into the image plane to a pixel p. A window of pixels, $\mathbf{W}$, is defined surrounding the estimated projected pixel, p, and it is verified, using the Pointcloud, if any 3D point from $k \subset \mathbf{W}$ has lower depth than $P_{obj_z}^{opt\_depth}$:

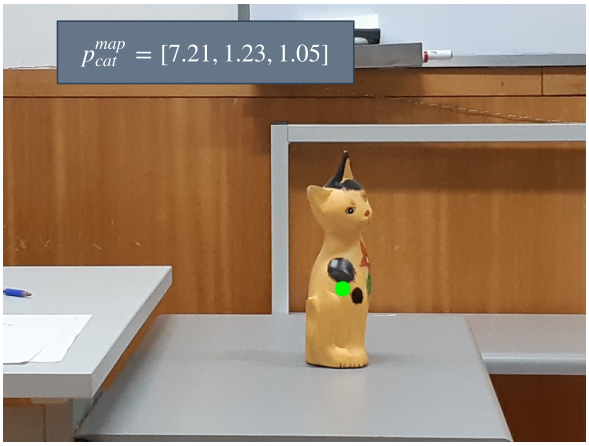
$$k_z < P_{obj_z}^{opt\_depth}, \tag{3.3}$$

If any $k \subset \mathbf{W}$ satisfies the above condition, the known object is treated as occluded, thus its confidence will not be decreased. The window used to compare the depth values, $\mathbf{W}$, is a 10×10 window. The intuition behind this approximation is represented in Fig. 3.9.

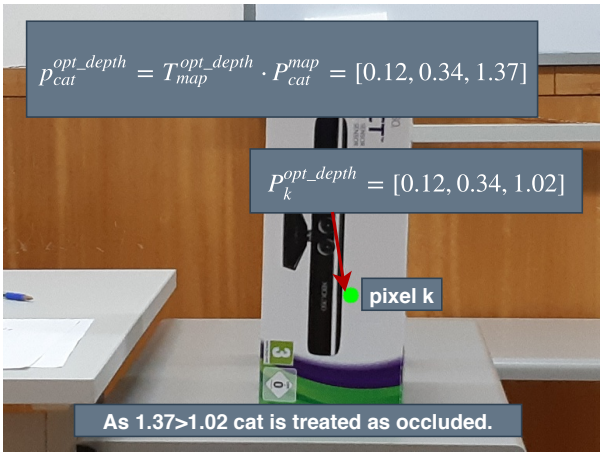
If an object already mapped is not being detected, is assumed to be in the camera's FoV, and using the occlusion method above explained, is said to be not occluded, such object might no longer be in its previous location. Thus, the system decreases its occurrence's confidence to C_k using its last confidence C_{k-1}, with the following equation:

$$C_k = C_{k-1} - \xi \tag{3.4}$$

Such as ν and ϕ on eq. 3.1, ξ is a positive constant that should be tuned given the used object detector's accuracy and speed. Using YOLOv3, a good value to this constant turned out to be 0.45. When $C_k \leq 0$ the system removes the object from the semantic map.



(a) The cat present in the image was detected and registered on the semantic map on location P_{cat}^{map} at time $t = t_k$.



(b) At time $t = t_{k+1}$, the cat is no longer being detected. From this view, P_{cat}^{map} can be transformed to $P_{cat}^{opt\_depth}$ and be projected in the image plane to pixel k. As $P_{cat_z}^{opt\_depth} > k_z$, we could think that some other object is in between the camera and the cat, therefore the system will treat this object as occluded and will not decrease the confidence in that object occurence.

Figure 3.9: Intuition behind the occlusion verification method.

Additional notes about camera's and object localization uncertainties

As described in section 2.7, the depth resolution of the RGB-D sensor used in this work, the Microsoft Kinect, is inversely proportional to the distance from the sensor to the obstacle. The ideal depth of this RGB-D sensor operating range is between 0.8m and 4.6m. Given this, as predicted object localizations tend to be more unreliable outside of this depth range, the system only registers objects that are within this depth range from the camera. Nevertheless, if an object already registered outside of this range is confronted with a currently detected object, the system updates that object occurrence confidence. Already registered objects that are far away, despite of being correctly detected in the images, its predicted localization could be erroneous and the system might not have the capability of recognizing that the far away detected object is an object that is already registered on the map. For that reason, the system also only decrease confidence of objects that are within the camera's ideal depth range.

When verifying if an object already registered and not being detected is being occluded, sometimes the 3D points layed in $\mathbf{W}$ can not be properly obtained, this could happen on metal/ glass surfaces, dark spots, if they are outside of the camera's ideal depth operating range, etc. When none of these points is obtainable, the system does not decrease the object's occurrence confidence.

A summarized flowchart that does not describe the way the system prevents duplicated occurrences and objects poorly localized, but instead it tries to describe the explained procedure and its reasoning in global manner is represented in Fig. 3.10.

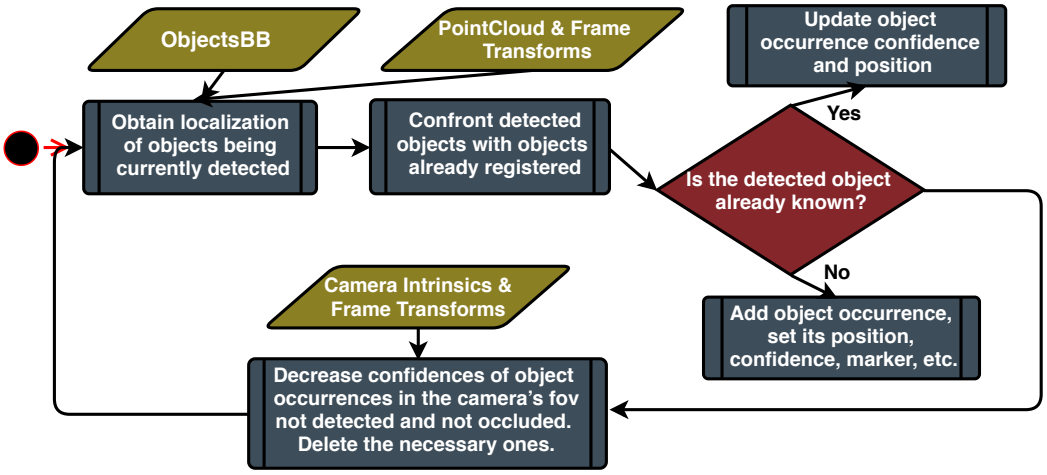


Figure 3.10: Object's registration global flowchart.

3.3.2 *Place Mapper* Node

Place Mapper node is responsible for integrating the scene categorizations performed by the scene recognition component, and categorizing the different meaningful places present in the robot's environment. For that purpose, a place segmentation built on top of the occupancy grid map, describing each place's boundaries, such as the one represented in 3.11, is required. Each place, contained in the robot's environment, must be described with 4 coordinates (that form a rectangle) in a file that is loaded when launching this node.

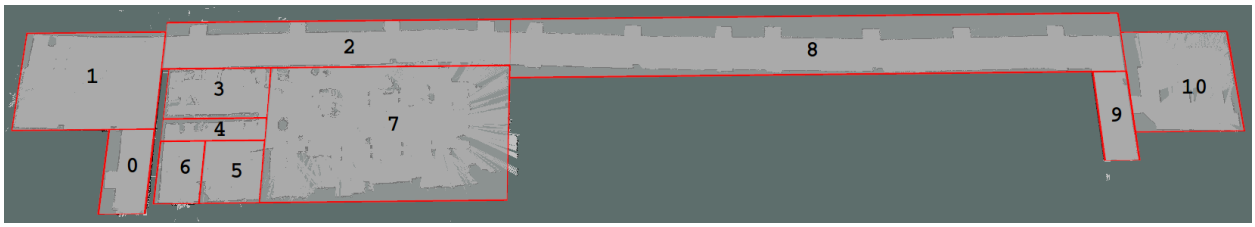


Figure 3.11: Occupancy grid map and place segmentation example.

Fig. 3.11 presents the occupancy grid map which the localization module uses to localize the robot while performing the final system evaluations. In order to categorize the different places present in the robot's environment, the occupancy grid map needs to be previously discretized into the different places one wants to classify, either if they are a closed-room or just a specific meaningful part of it.

The system then uses the place's boundaries to reason about image's quality when it comes to its capacity to describe a place. If a certain robot's view, i.e. an image, represents

mainly one place, that view will be used to classify that place. On the other coin, if the robot's view represents several places, that image fails when describing a singular place and its classification will not be used. To infer about this image's quality, the system uses the Pointclouds provided by the RGB-D sensor. *PctoMap* node transforms the acquired Pointclouds from the camera's depth optical frame to the map frame (Fig. 3.8) and publishes, in a specific topic, the transformed Pointclouds. Furthermore, *Place Mapper* node uses the transformed Pointclouds in order to verify the place where each of its containing points belongs to. This procedure is illustrated in Fig. 3.12.

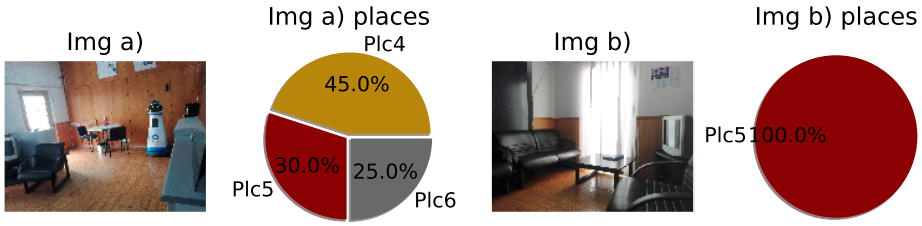


Figure 3.12: Image's capacity to describe a place

Image a), represented in Fig. 3.12, contains parts of several places. In that order, it should not be used to classify any of those places. Image b) only represents one place, it is a perfect image to be used in that place categorization. The 3 places represented on Image a) describe a demanding case scenario, where is difficult to obtain images that classify each place individually. Performing several evaluations on these 3 places, we found out that if an image contains about 75% of its points in a singular place, such image can be safely used to categorize that place.

In a first attempt to integrate the scene classifications conveyed by the scene recognition module, the system used the Bayesian Filter proposed in [2] as this belief updating strategie was developed to a similar problem, the problem of categorizing places for each grid map cell. Although, when using this strategie, it was noticeable that the place categories beliefs could change dramatically after updating the place beliefs with a low number of scene classifications and this can be explained due to the assumption of a first order Markov property in the Bayesian filter. For this reason, in its final version, the *Place Mapper* node uses another place categories belief updating method, a mean average over all the scene categorizations performed in a place. Assuming that most of the scene categorizations performed by the used CNN are correct, this belief updating method might allow the system to obtain better performances when categorizing the different places. The mean average filter belief updating equation can be described as follows:

$$p_j(\hat{x}_i|I_{1:k}) = \frac{p_j(\hat{x}_i|I_{1:k-1}) \cdot (k-1) + p(\hat{x}_i|I_k)}{k}, \text{ where:} \tag{3.5}$$

$p_j(\hat{x}_i|I_{1:k})$ is the discrete probability distribution over all possible place labels $\hat{x}_i$ given all the observed images that have quality to classify the place j from the past until now $I_{1:k}$, and $p(\hat{x}_i|I_k)$ is the network's prediction for the current image.

The update of $p_j(\hat{x}_i|I_{1:k})$ only takes place when the robot's view has quality to describe place j and when the robot reaches a localization (i.e. position and orientation) that was not used recently to categorize that place. Therefore, an history containing the last 50 localizations where $p_j(\hat{x}_i|I_{1:k})$ was updated is kept. With this verification, the system reduces the implications that might occur when the robot has a misrepresentation of a place (e.g., looking only at a wall) and the images are being misclassified during long periods of time.

The explained methodology to update the places' category beliefs, $p_j(\hat{x}_i|I_{1:t})$, is moreover represented in a flowchart diagram in Fig. 3.13.

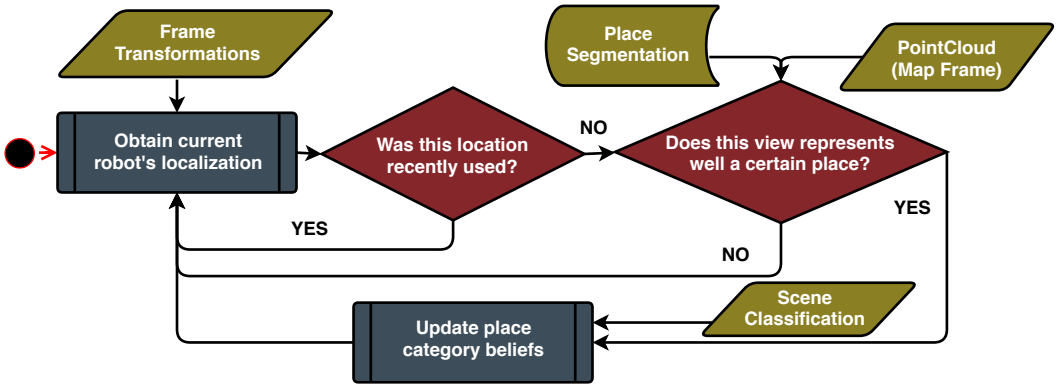


Figure 3.13: *Scene Mapper* node flowchart

Additionaly, as previously represented in Fig. 3.6, *Place Mapper* node provides 3 interfaces. It is the server of two ROS services and has the goal of publishing markers describing the place categories within the robot's environment:

- ***Places' Markers:*** For visualization purposes, place categories are annotated also using markers on top of the metric map. Each of the segmented places contained in robot's environment is "painted" with a color relative to its Top-1 predicted category. For instance, offices are represented in a blueish color, this is the reason why in Fig. 3.7 the floor is painted in that color. While the robot was detecting and registering objects on the semantic map, it was also categorizing the place where it was located as an office. Moreover, the system adds text markers describing the Top-1 predicted place category and its belief to ease the reading and understanding of the built semantic map by human operators. This markers are published in a specified topic and, as well as the objects' markers, the places' markers can be visualized using *RVIZ*;

- ***PlacesInMap Srv:*** Clients of this service can obtain information about each segmented place known in the semantic map. The response message will contain information referring to each place's boundaries and its categories beliefs;

- ***Places' Map Saver Srv:*** This service can be called to save the places' categories, already known in the semantic map, in a file. The generated file can then be loaded when launching this node. With this, when the robot resumes its exploration, it can already know the categories of places previously explored.

3.4 *Object Finder* Package

Throughout this dissertation, it was mentioned several times that semantic maps can boost the performance of object searching systems. For that reason, *Object Finder* package was developed and it has the major goal of providing the robot with the ability of finding objects in a more competent manner.

In the pursuance of obtaining $p(\hat{o}_i|\hat{x}_k)$, i.e. the probability of existing one or more objects of class i in a place of category k, two different approaches were firstly taken out. Obtaining $p(\hat{o}_i|\hat{x}_k)$ would highly boost the performance of the object searching system as even without having prior knowledge about locations of the objects to be found, the system could then infer the places where objects of that class are more likely to be present. However, none of the strategies to predict $p(\hat{o}_i|\hat{x}_k)$ achieved satisfactory results and, for that reason, they are only explained in the Appendices, sections 6.1.1 and 6.1.2.

A knowledge graph, also referred as an ontology, represents semantic relations between concepts (i.e. words or phrases of natural language) in a network. The graph can be described by a set of vertices, also known as nodes, which represent concepts, and with a set of edges, which represent semantic relations between different concepts, thus connecting the concepts (vertices) via different mapping or connecting semantic fields (edges).

To find the desired objects, with the developed object searching algorithm, the robot not only uses its workspace's semantic map but also a set of common sense rules, that are given in the form of an ontology. The developed ontology is explained in the next section.

3.4.1 Developed Ontology

The developed ontology comprises 289 concepts, of which 205 are places (Places205 categories), 80 are objects (MS-COCO categories) and 4 that divide objects into 4 major object categories (food, vehicles, electronics and animals), therefore providing another semantic layer that can be shared by the user and the system. The ontology have 3 different types of edges, where each one adds a different type of common sense rule that could relate the different concepts:

- **IsA**: These ontology edges can relate concepts to its categories, e.g. Office IsA Place, Apple IsA Object, Apple IsA Food. There are 324 edges of this type;

- **LocatedAt**: These relations are established between object concepts and place concepts. When there is a high probability of finding a giving object in a giving place, that information is added to the ontology with a connection between those two concepts with an edge of this type, e.g. Banana LocatedAt Kitchen, Dining Room, Supermarket, etc., Laptop LocatedAt Office, Home Office, Living room, etc. There are 201 edges of this type.

- **NearBy**: These ontology edges relate objects that can be found nearby other objects, e.g. Mouse and Keyboard NearBy Laptop, Apple NearBy Banana, etc. The developed knowledge graph comprises 12 relations of this type.

The 3 different types of edges are all asymmetric, i.e. the directionality of the edge is relevant. Furthermore, the edges are all deterministic as the approaches to estimate $p(\hat{o}_i|\hat{x}_k)$ failed and, for that reason, there was no possibility to incorporate $p(\hat{o}_i|\hat{x}_k)$ in the *LocatedAt* edges. Fig. 3.14 illustrates the previously explained ontology.

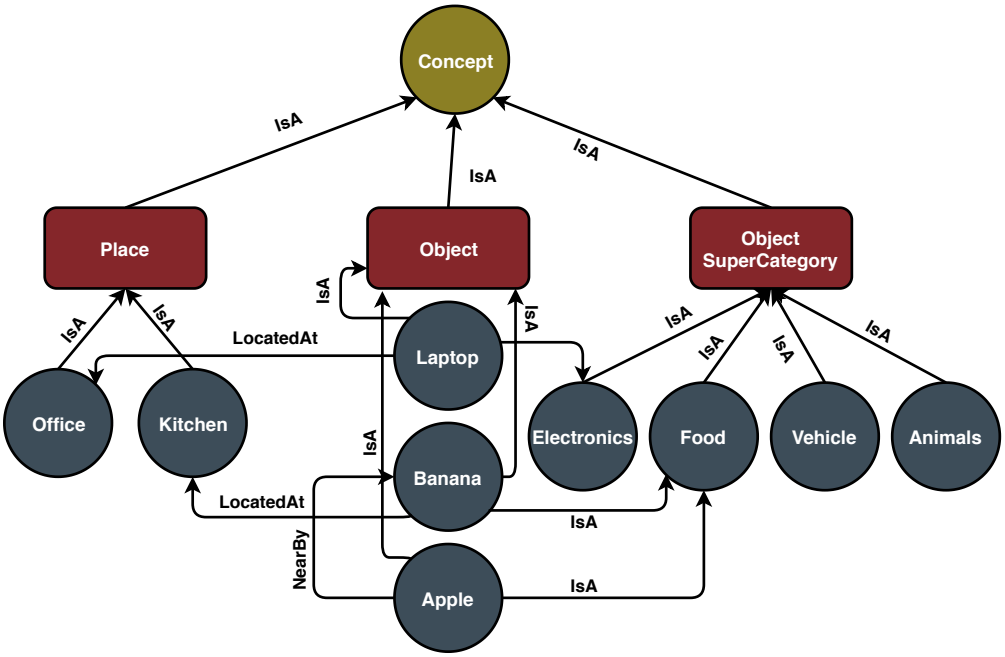


Figure 3.14: Developed ontology illustration

3.4.2 *Object Finder* node

The developed *Object Finder* package comprises only one node, which the required and provided interfaces are represented in the components diagram presented in Fig. 3.15.

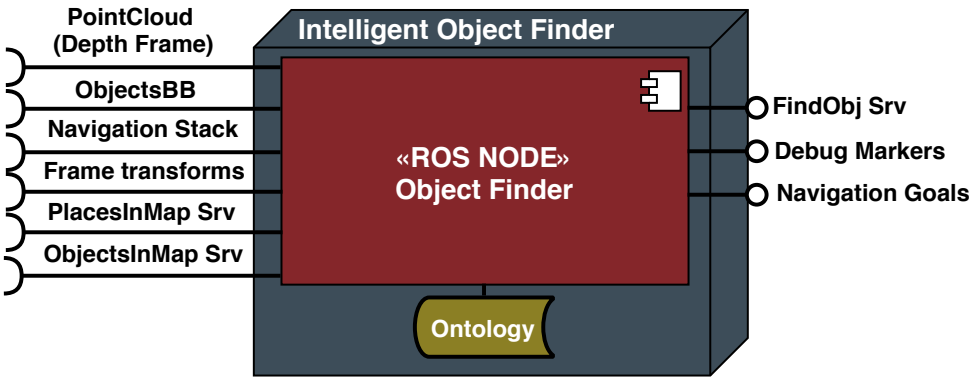


Figure 3.15: *Object Finder* package components diagram.

The *Object Finder* node is the server of a ROS service which may be called when a human operator desires the robot to start searching for a specific object. When the service is called, the user is prompted to introduce the concept which is trying to find. The introduced concept could be a MS-COCO object category or an object supercategory also available in the developed ontology. When the user advances with a supercategory, the system will prompt the user to introduce the object concept within its supercategory (e.g. user searched for food, system prompts to choose within banana, apple, sandwich, etc...).

At this point, *Object Finder* node starts to listen to the object detections published by the developed *Object Detection* component. If an object of the desired category is detected, the major goal of finding the object succeeded.

Using the semantic map and the develop ontology, the system then infers the robot's workspace locations where there is a high probability of finding the desired object. For that purpose, the system starts by performing a querie to ***ObjectsInMap srv*** requesting for locations of known object occurrences of the category to be searched and of categories that are the linked by a NearBy edge to that category (e.g., when searching for an apple, the system sends a request message to ***ObjectsInMap srv*** in order to find, not only the locations where apples are present, but also the locations where bananas are known to be). Afterwards, the system performs a request to ***PlacesInMap srv*** and obtains the list of known places and their categories. The explored places which have a top-1 predicted category which is linked via a *LocatedAt* edge to the object's category to be found, will also be used to search the object (e.g., when there is a known office in the robot's workspace and the robot is searching for a keyboard, as "keyboard" is a concept which is linked to "office" concept by a *LocatedAt* edge, that office's center will also be a navigation goal to search the keyboard). The system then considers that all the locations, obtained with this procedure, are locations where there is an equally high probability of finding the desired object and, for that reason, it can use all of them to find the desired object. The list of these locations is ordered given the robot's distance to those and, finally, the system sets the robot's navigation goal to the nearest location obtained with this procedure.

When the robot reaches a position nearby its current navigation goal, it starts rotating over it (360^{o}) in order to obtain a more complete view of that location. If, after the rotation, the robot did not found yet the desired object, it will start moving to the next nearest location where the object might be found. If the robot searched all the desired locations and the object was still not found, the major goal of finding the object failed.

For debugging and visualization purposes, *Object Finder* node publishes markers which can be visualized using *RVIZ*. The systems adds markers on the locations which will be explored. After exploring a location where the object was not found, that location's marker is removed. If during this procedure the desired object was found, the system estimates its location and signals that information with a marker.

The explained object searching algorithm is illustrated in a flowchart diagram in 3.16.

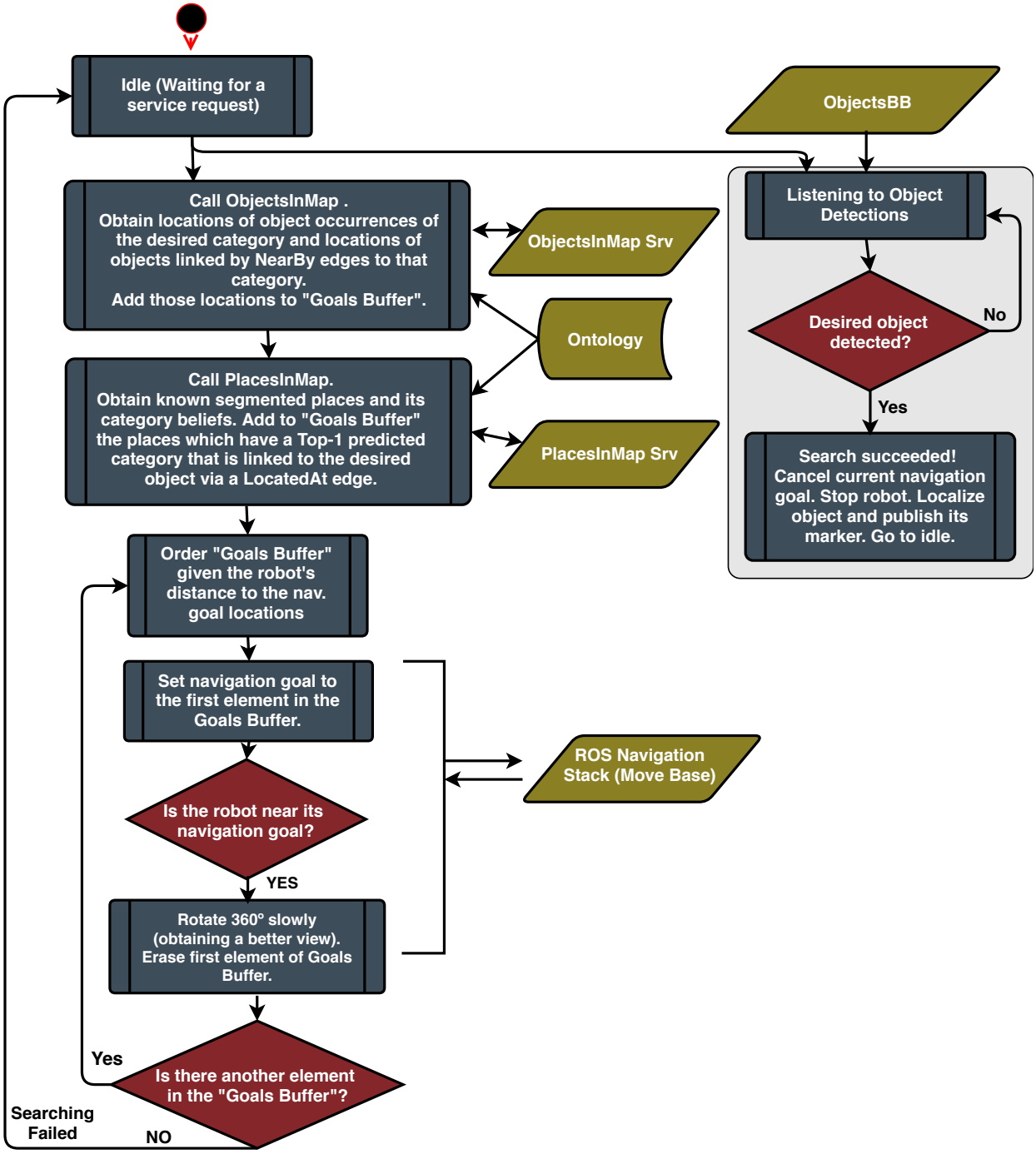


Figure 3.16: Developed object searching algorithm

3.5 Summary

Throughout this chapter the developed *Semantic Mapping* ROS stack was presented. In the first section, the proposed system was explained in a broad manner. The following sections presented all the system components in a major detail and explained all the rational that supported several design decisions. In the following chapter, the system will be validated experimentally.

Chapter 4

Experimental Evaluation and Result Analysis

The developed system was evaluated using a Pioneer P3-DX mobile platform endowed with a Hokuyo URG-04LX-UG01 LRF. As previously mentioned, the RGB-D sensor used was the Microsoft's XBOX 360 Kinect and the laptop mounted on top of the robot was a MSI GP72MVR-7RFX, gifted with a NVIDIA GeForce GTX 1060, an i7-7700HQ Quad-Core, and 16GB of RAM.

In order to evaluate the performance of the system, four different sets of experimental tests, explained in the following sections, were carried out. For the tests explained in section 4.2 and section 4.3, i.e the objects' registration tests and the place categorization tests, *rosbag*[1] tool was used in order to record real data while the robot explored its environment via teleoperation. The RGB and depth images, scans, map data, and frame transformations were recorded, ensuring that this data would later be played back to test the developed system. This allowed to, while using real-world data, repeatedly test, tune parameters, and improve the developed system's performance.

4.1 Objects' Localization Tests

The accuracy and robustness of the developed objects' localization method was evaluated in two different scenarios where the localization of a person, a monitor, and a chair are known. In each test scenario, measurements of the points which represent the objects' 3D geometrical centers, relative to the camera's link frame, were taken. *RVIZ* was then used to verify if those points were a good representation of the objects' centers. The objects and its image regions were detected using YOLOv3, and the objects' localization method was evaluated using three histogram bin estimators.

In Figs. 4.1 and 4.2 the real objects' 3D geometrical centers and its predictions are

[1]Adittional information about *rosbag* tool can be found on the ROS Wiki at `http://wiki.ros.org/rosbag`

represented above the Pointcloud. While the real objects' localizations are represented by green spheres, their different predictions using the Freedman-Diaconis, Scott, and Doane histogram's bin estimators are represented respectively, with blue, orange and red spheres. Tabs. 4.1 and 4.2 show the real objects' localizations and their different predictions. Furthermore, they present the error, in this case an Euclidean distance in meters, between the objects real localizations and its predictions.

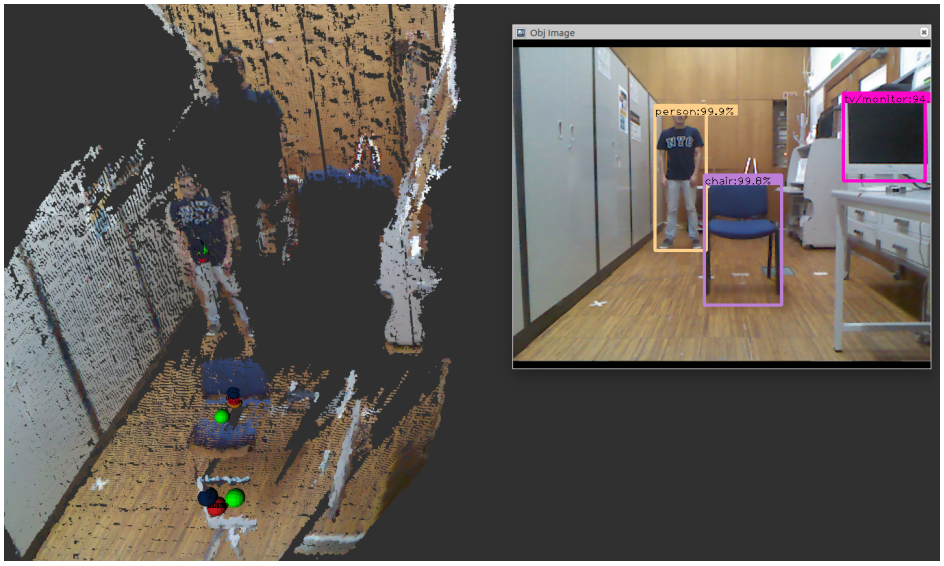


Figure 4.1: Objects' Localization Test Scenario 1

Table 4.1: Objects' Localization Test Scenario 1 Results

Test Scenario 1	**Histogram Bin's estimator**											
	Freedman-Diaconis				**Scott**				**Doane**			
	x	**y**	**z**	**error**	**x**	**y**	**z**	**error**	**x**	**y**	**z**	**error**
Person $[x, y, z] = [3.93, 0.48, 0.38]$	3.918	0.511	0.432	**0.062**	3.971	0.513	0.308	**0.090**	3.950	0.503	0.297	**0.088**
Chair $[x, y, z] = [2.48, -0.15, -0.10]$	2.709	-0.149	-0.085	**0.229**	2.698	-0.147	-0.157	**0.225**	2.705	-0.142	-0.135	**0.228**
Monitor $[x, y, z] = [1.45, -0.70, 0.27]$	1.419	-0.576	0.308	**0.133**	1.416	-0.611	0.242	**0.098**	1.416	-0.611	0.242	**0.098**

Table 4.2: Objects' Localization Test Scenario 2 Results

Test Scenario 2	**Histogram Bin's estimator**											
	Freedman-Diaconis				**Scott**				**Doane**			
	x	**y**	**z**	**error**	**x**	**y**	**z**	**error**	**x**	**y**	**z**	**error**
Person $[x, y, z] = [4.15, -0.10, 0.30]$	5.626	-0.010	0.788	**1.557**	5.626	-0.010	0.788	**1.557**	5.626	-0.010	0.788	**1.557**
Chair $[x, y, z] = [2.48, -0.15, -0.10]$	5.653	-0.596	0.110	**3.211**	5.594	-0.655	0.066	**3.159**	5.653	-0.596	0.110	**3.211**
Monitor $[x, y, z] = [1.45, -0.70, 0.27]$	1.606	-0.592	0.277	**0.190**	1.579	-0.608	0.272	**0.159**	1.577	-0.609	0.271	**0.157**

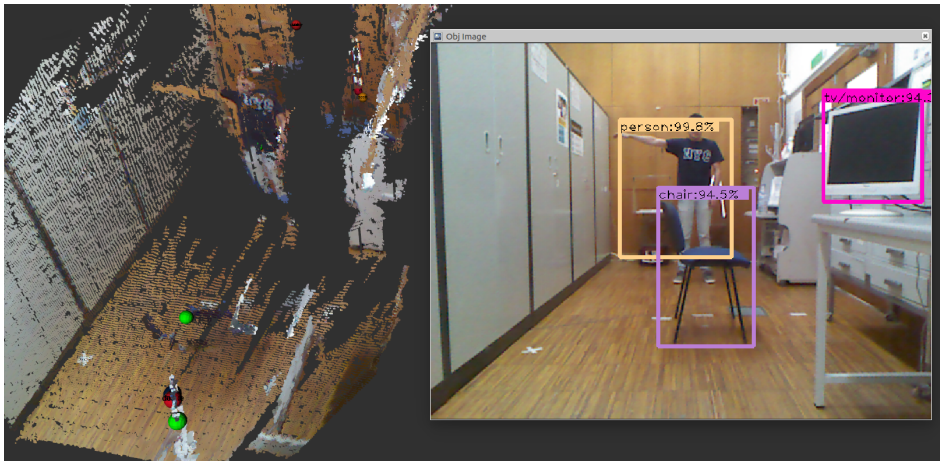


Figure 4.2: Objects' Localization Test Scenario 2

4.1.1 Objects' Localization Discussion

By visually comparing the object's ground-truth localization and its predictions, represented on Fig. 4.1, it becomes evident that, in the first test scenario, the developed method was able to estimate properly, i.e with small error values, the localization of the different objects independently of the histogram's bin estimator used. With a further analysis of Table 4.1, it is verified that using Scott's histogram bin estimator the developed method predicted the localization of the three objects with an average error equal to $0.1377m$. Doane's and Freedman-Diaconis bin estimators also allowed the method to obtain satisfactory low distance errors, respectively, $0.138m$ and $0.1433m$.

Looking into Fig. 4.2 and Tab. 4.2, its noticeable that the localization of the chair and the person were never well predicted in the second test scenario. The chair and the person have been projected onto the wall in front of those, and this can be explained due to the fact that their predicted image regions contained a large amount of points on the wall. The depth interval, chosen by the method, to predict the localizations of these two objects was the interval that contained the wall points.

These two figurative test scenarios are enough to conclude that the developed objects' localization method can sometimes have good performances. Although, when there are overlapped bounding-boxes of different objects or when an object's image region contains a significant number of points that do not belong to the object, this method is not that robust.

Every methodology chosen to deal with the occurrence of objects, and the manner in which the object occurrences confidences are updated, were thought considering this poor object localization performances. With the developed system, it is possible that the robot, when obtaining different views for the already registered objects, can correct/remove from the map objects poorly localized.

4.2 Objects' Registration Tests

In order to evaluate the generated semantic map regarding the object occurrences registered on it, the system was tested in part of ISR-MRL (places 3 and 7 in Fig. 3.11). The system was tested using two object detectors, i.e two different CNN models loaded by the object detection module. YOLOv3 and SDD (Resnet 50) were chosen due to the fact that they are known for having a good tradeoff between accuracy and computational cost.

The objects which can be detected and so registered on the semantic map during the robot's navigation were manually annotated on a 2D raw plan, where the exact localization of each object is not accurately known. The average precision and recall regarding the objects occurrences registered on the semantic map were, finally, obtained by visually comparing and analyzing the built semantic map with the previously built 2D raw plan.

The following table presents the precision and recall metrics obtained after confronting the registered object occurrences with the objects that the robot could observe during its navigation, i.e. the objects present in the 2D raw plan. Furthermore, Figs. 4.3 and 4.4 show the objects registered on the semantic map after this evaluation using YOLOv3 and SSD (Resnet 50) respectively.

Table 4.3: Precision and recall of the semantic map regarding its object registrations

	YOLOv3					**SSD (Resnet 50)**				
	TP	**FP**	**FN**	**Precision**	**Recall**	**TP**	**FP**	**FN**	**Precision**	**Recall**
chairs	16	2	1	0.889	0.941	16	2	1	0.889	0.941
laptops	2	4	1	0.333	0.667	2	3	1	0.400	0.667
tv/monitors	10	0	5	1.000	0.667	7	0	8	1.000	0.467
bottles	3	1	0	0.750	1.000	1	0	2	1.000	0.333
mouses	2	0	2	1.000	0.500	1	0	3	1.000	0.250
keyboards	7	0	3	1.000	0.700	4	0	6	1.000	0.400
dining tables	0	3	0	–	–	0	6	0	–	–
persons	6	1	0	0.857	1.000	4	2	2	0.667	0.667
backpacks	1	0	1	1.000	0.500	0	0	2	–	–
remotes	0	1	0	–	–	0	1	0	–	–
suitcases	0	0	0	–	–	0	1	0	–	–
refrigerators	0	0	0	–	–	0	1	0	–	–
cats	0	0	0	–	–	0	2	0	–	–
couches	0	1	0	–	–	0	0	0	–	–
Semantic Map	47	13	13	**0.783**	**0.783**	35	18	25	**0.660**	**0.583**

4.2.1 Objects' Registration Discussion

The results presented in Tab. 4.3 show the developed system's ability to successfully attain a global representation of the objects present in robot's workspace. Using the developed

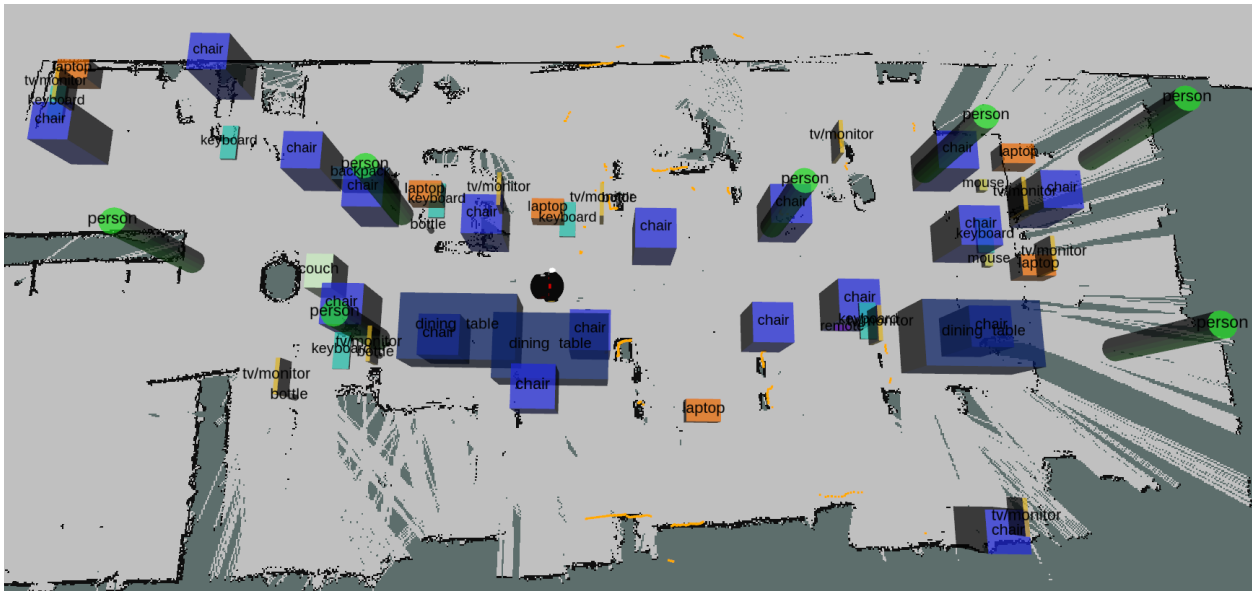


Figure 4.3: Objects registered on the semantic map using YOLOv3 as the CNN model

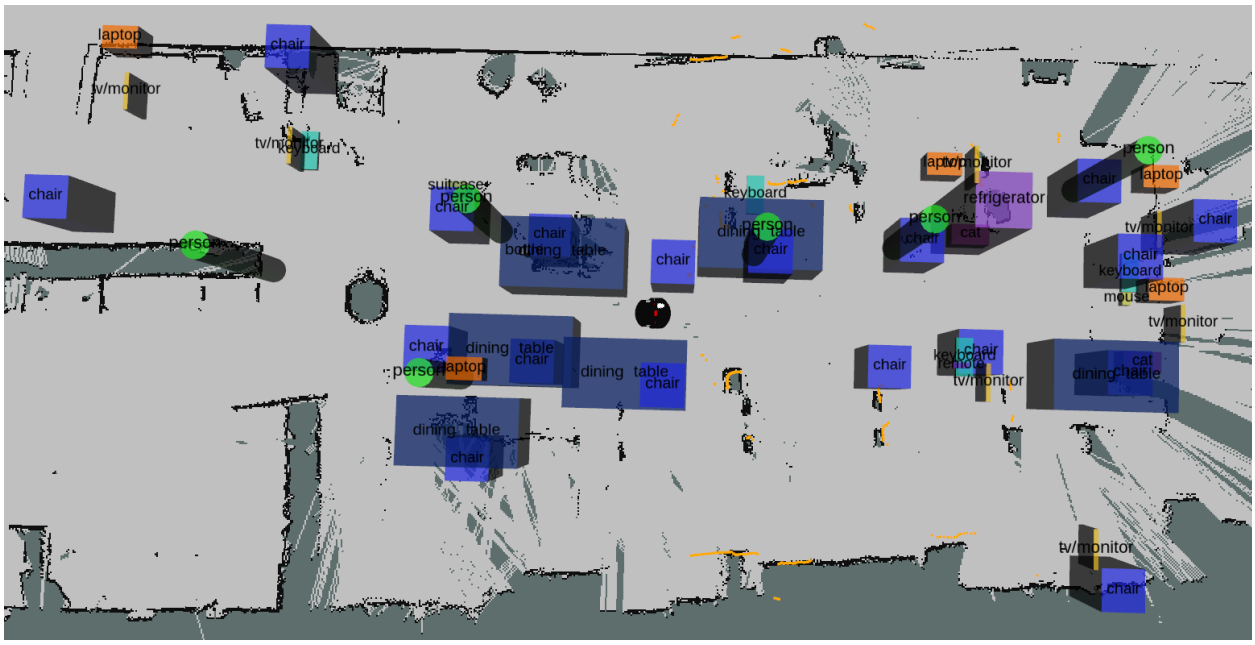


Figure 4.4: Objects registered on the semantic map using SSD (Resnet 50) as the CNN model

techniques in *Objects' Mapper* node and YOLOv3 as the object detector, the semantic map achieved fairly good average precision and recall values, both equal to 78.3%. When using SSD (Resnet 50) the semantic map attained not as good metrics, nevertheless the values of precision and recall, 66.0% and 58.3%, are also quite acceptable.

While performing these evaluations, several screenshots of the semantic map and of the object detections were taken. Several of those screenshots, of the semantic map being built using YOLOv3 as the CNN model are presented in the Appendices, section 6.2. Through a deep analysis of those, several conclusions about the developed system can be taken and

will now be mentioned.

The developed system is capable of recovering from false positive object occurrences and erase such object occurrences from the map or correct its localizations. However, there are still several system weaknesses. The major one, as expected, is when verifying if an object is occluded or not. The occlusion verification can fail, when this happens, objects incorrectly registered can be said as occluded, and the system will not decrease its confidences, which do not allow the system to erase this false positives and furthermore improve the semantic map precision. Another weakness is when there are several objects of the same class close to each other. When this happens, the system will perceive it is dealing with duplicated occurrences of the same object and will only register one of them. Nevertheless, this verifications are necessary in the system, or otherwise the semantic map will have a larger number of duplicated object occurrences registered on it due to object localization uncertainties or due to duplicated object detections.

4.3 Place Categorization Tests

The different places contained in the robot's workspace were categorized with VGGNet-19 as the scene recognition CNN model and using two different place's belief updating strategies, the Bayesian filter described in [2] and the mean average filter. Fig. 4.5 shows the obtained results when the system is categorizing two different places, a corridor and an office.

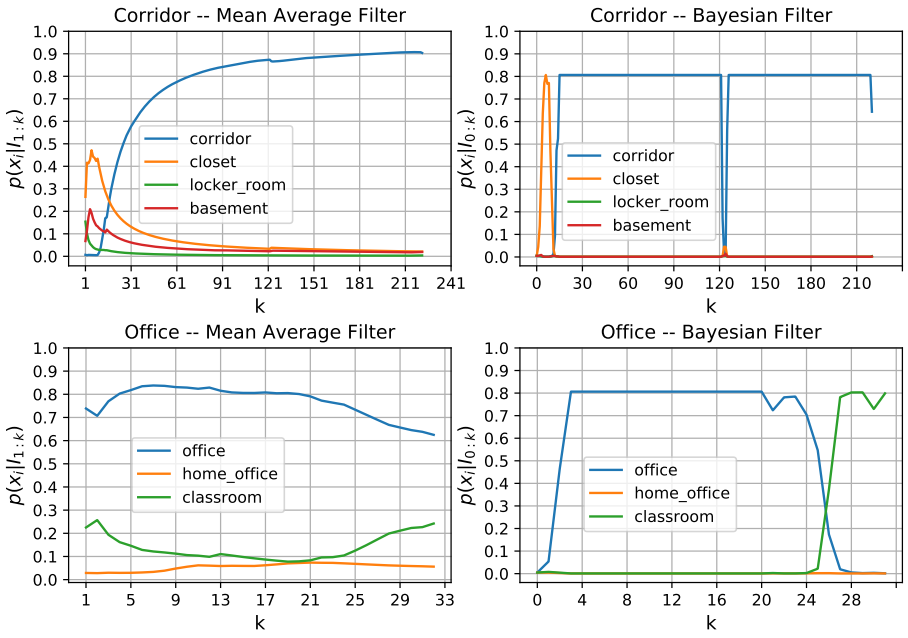


Figure 4.5: Mean average filter (right) vs Bayesian filter (left).

The results in Fig. 4.5 are a good example of what can happen when using the Bayesian Filter approach. Place 2, a corridor, was being correctly categorized until $k \approx 120$, a few scene misclassifications were then responsible by a fast belief decrease of place 2 being a

corridor. However, the system was also able to recover from that quickly as the next scene classifications were classifying that place correctly. When categorizing the MRL office, the last images were categorizing that place as a classroom, the Bayesian filter quickly changed the top predicted category from office to classroom and did not recovered from that. These problems did not happened when using the mean average filter, as expected, its response is smoother. However, for the same reason, the mean average filter will have more difficulties in recovering from situations where the place is misclassified after using a large number of scene categorizations.

These two belief updating methods were evaluated with and without endowing the system with prior knowledge. When endowing the system with prior knowledge, the scene categorizations performed by the CNN are normalized within a subset of indoor scene categories comprised in Places205 dataset. This prior might be helpful as a large number of Places205 are outdoor categories or irrelevant indoor categories such as "dock", "basilica", "bus interior", "butchers shop", etc., which are place categories where the system developed is not intended to run on. In this case, the system will categorize the different places between the following categories: attic, auditorium, bar, bedroom, classroom, corridor, dining room, dinette, home office, kitchen, living room, lobby, office, pantry, parking lot, parlor, patio, pavilion, staircase, veranda and waiting room. The place categories beliefs of classes that are not in the priors of the system are all set to 0.

The obtained results, after categorizing 8 different places contained in our test scenario, are represented by a modified top-1 and top-5 predicted categories accuracy metric, where for a category being considered as a top-1 or top-5 category it has also to have a category belief higher than 15%, in Tab. 4.4. Fig. 4.6 illustrates the obtained results when giving prior knowledge to the system and using the mean average filter. A table comprising the top-5 category predictions for each of the places categorized is presented in the Appendices, section 6.3.

Table 4.4: Place Categorization Results

	Belief Updating Strategy			
	Baysian Filter	**Bayesian Filter with prior knowledge**	**Mean Average Filter**	**Mean Average Filter with prior knowledge**
Top-1 accuracy (%)	44.44	66.67	77.78	77.78
Top-5 accuracy (%)	44.44	66.67	88.89	88.89

4.3.1 Place Categorization Discussion

The results presented in Tab. 4.4 confirm the system's ability to properly label the explored places by the robot when using the mean average filter belief updating strategy. When using the Bayesian filter, the desired belief's temporal coherence cannot be obtained and it is a random factor if the places are correctly categorized or not in the end of the system evaluation

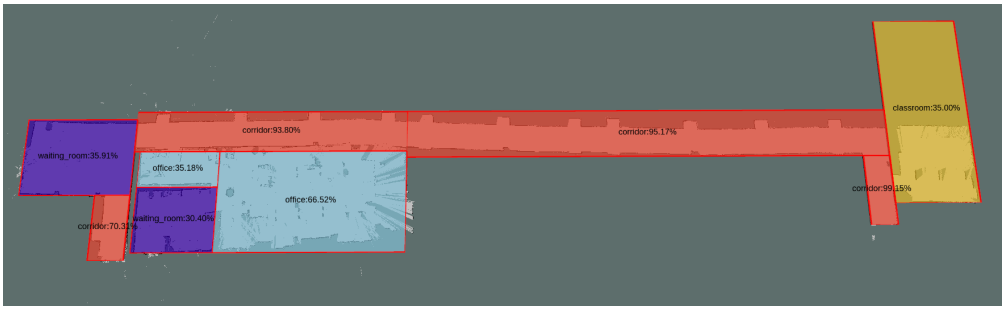


Figure 4.6: Top-1 predicted place categories registered in the Semantic Map. The red, blue, purple and yellow colors represent respectively, corridors, offices, waiting rooms and classrooms as top-1 predicted categories.

due to the fact that they will mostly depend only on the last used scene classifications.

4.4 Object Searching Algorithm Tests

Several tests were performed to evaluate the developed object searching algorithm in ISR-MRL with real data. While conducting those evaluations, the system revealed several weaknesses regarding the robot's localization and its navigation capabilities. As these problems are not in the main core of this work and there was no simple solution to them, the developed object searching algorithm was tested in several simulated scenarios. The major goal when validating the developed algorithm by this via was to find out how fast can it be in comparison to a similar one that does not take advantage of object-places relationships.

The simulated test scenarios are based on the following assumptions:

- Each scenario contains only one object instance of the object class to be searched.
- The object is localized on a place where it is likely to be found. For instance, a laptop can be in auditoriums, bedrooms, classrooms, home offices, living rooms and offices. If the simulated environment contains more than one of those places, the object will be localized in one of them which is chosen randomly.
- When reaching the place's center where the object is located, it is assumed that the robot then takes 41.79[1] seconds to fully explore that place and detect the desired object. After those 41.79 seconds the object is considered as detected.
- The only difference between the ontology-based and the brute-force object searching algorithms is that, while the ontology-based algorithm will only fully explore the places where the ontology "says" the object might be located, the brute-force algorithm will search, i.e. fully explore by 41.79 seconds, every single place where it passes into.

[1]This value was chosen due to the fact that, taking into account all the simulated scenarios, the robot took in average 41.79 seconds to travel between different place's centers.

The occupancy grid map, place segmentation and place categorization, used in all the eighteen different test scenarios performed are illustrated in Fig. 4.7. Moreover, the results obtained in each test scenario are presented in Tab. 4.5.

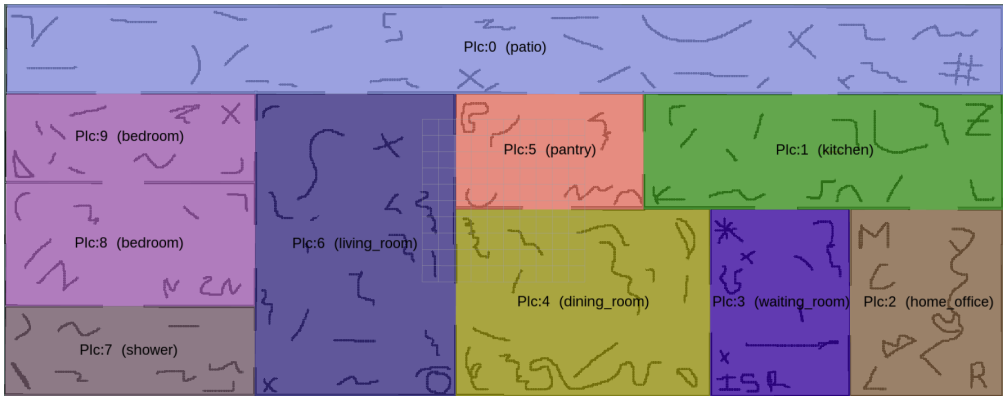


Figure 4.7: Simulated Environment - A 61×24m occupancy grid map segmented in 10 places of 9 different categories.

4.4.1 Object Searching Algorithm Discussion

In seventeen of the eighteen different test scenarios, the ontology-based object searching algorithm was faster in finding the desired object. Averaging the results obtained in all the test scenarios, we see that developed algorithm was 2.644 times faster than the algorithm which does not take advantage of object-places relationships. This is explained by the fact that, with the ontology-based object searching algorithm, the robot does not fully explore all the places where it passes into, but only the places where the desired object is likely to be found. While the ontology-based algorithm before finding the desired object fully explored 1.722 rooms in average, the other (brute-force) algorithm fully explored 6.444 rooms.

Notice that this values can be relative as they depend on several environment properties such as its places' categorization, its topology, its dimensions and furthermore on the average time a robot takes until finding an object in a given place. Nevertheless, the obtained results have the ability to give experimental evidence that, it is expected that the developed object searching algorithm is faster in finding a desired object compared to an algorithm which search for an object in all the locations possible.

Table 4.5: Ontology-based object searching algorithm vs Brute-force object searching algorithm benchmark. n_r and n_b reflect the number of places fully explored before finding the desired object when using the developed ontology-based and the brute-force object searching algorithms respectively. t_r and t_b present the time spent, measured in seconds, that the robot took until the object was found.

Object searched, object pose [x,y], and object place	Object can be located at:	Initial robot pose	n_r, n_b	t_r, t_b	t_b/t_r
sink [23.28, 4.33] plc:1 (kitchen)	kitchen and kitchenette	[−13.78, −8.90] plc:6 (living room)	$n_r = 1$ $n_b = 7$	$t_r = 222.16s$ $t_b = 771.26s$	3.47
toaster [13.52, 5.46] plc:1 (kitchen)	kitchen and kitchenette	[−26.31, −9.50] plc:7 (shower)	$n_r = 1$ $n_b = 10$	$t_r = 243.27s$ $t_b = 865.05s$	3.56
bed [−17.45, −5.46] plc:8 (bedroom)	bedroom and hotel room	[−2.96, 9.32] plc:0 (patio)	$n_r = 2$ $n_b = 8$	$t_r = 218.65s$ $t_b = 657.36s$	3.01
fork [24.18, 4.39] plc:1 (kitchen)	dining room, kitchen, kitchenette and home dinette	[13.71, −8.11] plc:3 (waiting room)	$n_r = 2$ $n_b = 3$	$t_r = 232.04s$ $t_b = 236.67s$	1.02
keyboard [25.85, −5.1315] plc:2 (home office)	auditorium, bedroom, classroom, home office, living room and office	[15.39, 2.06] plc:1 (kitchen)	$n_r = 1$ $n_b = 3$	$t_r = 155.056s$ $t_b = 212.87s$	1.37
wine glass [4.42, −4.69] plc:4 (dining room)	bar, coffee shop, dining room, home dinette, kitchen, kitchenette, and living room	[−25.78, 3.52] plc:9 (bed room)	$n_r = 2$ $n_b = 8$	$t_r = 183.44s$ $t_b = 561.348s$	3.06
tie [−28.53, 3.01] plc:8 (bedroom)	bedroom and closet	[25.88,-3.21] plc:2 (home office)	$n_r = 1$ $n_b = 7$	$t_r = 224.57s$ $t_b = 642.23s$	2.86
toilet [−23.20, −10.89] plc:7 (shower)	shower	[1.58, 1.60] plc:5 (pantry)	$n_r = 1$ $n_b = 6$	$t_r = 186.45s$ $t_b = 505.93s$	2.71
cake [6.82, 2.47] plc:5 pantry	dining room, cafeteria, kitchenette, supermarket, coffee shop, bar, dinette home, kitchen, pantry and kitchen restaurant	[11.20, −8.24] plc:4 (dining room)	$n_r = 2$ $n_b = 5$	$t_r = 124.67s$ $t_b = 379.47s$	3.04
sofa [−19.29, −1.66] plc:6 (living room)	waiting room, coffee shop, lobby, living room and bar	[−23.09, 4.83] plc:9 (bedroom)	$n_r = 1$ $n_b = 4$	$t_r = 89.35s$ $t_b = 261.97s$	2.93
apple [26.60, 0.80] plc:1 (kitchen)	bar, cafeteria, coffee shop, dining room, home dinette, kitchen, kitchenette, restaurant kitchen, and supermarket	[−23.09, 4.83] plc:7 (shower)	$n_r = 2$ $n_b = 10$	$t_r = 266.78s$ $t_b = 839.85s$	3.15
hair drier [−23.09, 4.83] plc:9 (bedroom)	bedroom and shower	[−8.47, −1.40] plc:6 (living room)	$n_r = 3$ $n_b = 10$	$t_r = 251.98s$ $t_b = 861.96s$	3.42
laptop [25.37, −6.94] plc:2 (home office)	auditorium, bedroom, classroom, home office, living room and office	[4.77, 8.33] plc:0 (patio)	$n_r = 4$ $n_b = 5$	$t_r = 552.54s$ $t_b = 346.88s$	0.63
oven, [25.37, −6.94], plc:1 (kitchen)	kitchen and kitchenette	[−10.35, −5.44] plc:6 (living room)	$n_r = 1$ $n_b = 7$	$t_r = 143.28s$ $t_b = 636.06s$	4.44
pizza [14.10, 4.97] plc:1 (kitchen)	dining room, cafeteria, dinette home, kitchen, kitchenette, supermarket, living room, coffee shop, bar, and kitchen restaurant	[22.97, 4.97] plc:3 (waiting room)	$n_r = 1$ $n_b = 3$	$t_r = 114.29s$ $t_b = 210.28s$	1.84
scissors [28.04, −10.06] plc:2 (home office)	classroom, auditorium, home office and office	[13.22, −7.17] plc:3 (waiting room)	$n_r = 1$ $n_b = 2$	$t_r = 89.64s$ $t_b = 134.57s$	1.50
mouse [26.10, −3.97] plc:2 (home office)	auditorium, bedroom, classroom, home office, living room and office	[−21.16, 5.02] plc:9 (bedroom)	$n_r = 4$ $n_b = 8$	$t_r = 367.34s$ $t_b = 628.22s$	1.71
toothbrush [−29.39, −9.24] plc:7 (shower)	shower	[17.53, 1.84] plc:1 (kitchen)	$n_r = 1$ $n_b = 10$	$t_r = 244.66s$ $t_b = 944.72s$	3.86

Chapter 5

Conclusion and Future Work

In section 1.1, the main objectives of this work were defined. Throughout this chapter a reflection upon them and on the developed system's performance is carried out.

Firstly, the developed system had to be capable of detecting objects used by humans in their daily lives. By building the *Object Detection* package, the system was endowed with this capability. Furthermore, as the nodes comprised in that package use state-of-the-art deep learning object detection methods, the system is able to recognize objects with high levels of precision and accuracy.

The second requirement stated that the system had to be able of performing scene recognition. This was achieved by designing the *Scene Recog* node. Once again, using state-of-the-art deep learning methods, the system was gifted with the ability of performing scene recognition with high levels of performance.

The developed semantic mapping framework had to be capable of representing the different acquired semantic properties, i.e. object occurrences and place categories, in a temporal coherent semantic map. These requirements were accomplished by designing the *Semantic Mapping* package. The system's temporal capability of keeping track of object occurrences in the semantic map was tested and the system demonstrated its ability to successfully obtain and maintain a fairly good representation of the object occurrences present in the robot's workspace. The system's ability of categorizing the different places explored by the robot was also evaluated and, using the developed methodologies and the mean average belief updating strategy, the system was capable of classifying correctly most of the places present in the robot's workspace.

Finally, the last requirement stated that the robot should take advantage of the semantic map in order to find objects in a faster and more efficient manner. For that purpose, an ontology-based object searching algorithm was developed. The developed object searching algorithm was benchmarked in a set of simulated scenarios where it proved that object-places relationships can enhance an object searching robotic system.

5.1 Future Work

Despite of the overall satisfactory obtained results, there is a large room for future improvements in the developed system.

First and foremost, it would be interesting to study and test different methods to predict the localization of the detected objects. The system could also extract more valuable information about the detected objects. For instance, it would be really interesting acquiring objects' 3D dimensions and their 6D poses.

Deep learning methods, such as the ones used in this work, return their scores from softmax layers that are proportional to the models confidence in a given detection. However, softmax values are not calibrated probabilities and there is no possibility to estimate its uncertainties unlike is usual to do with most other robot's sensor measures. Bayesian deep learning field tries to tackle that problem and properly estimate uncertainties from deep learning architectures endowing a robotic system the capability to properly fuse neural network predictions, with its uncertainties, with other sensor's data. With this, it would be of major interest a study of Bayesian deep learning architectures. Furthermore, the system would be rewarded by changing the developed object occurrence and place categories confidences updating strategies to incorporate deep neural network uncertainties.

There is a lot of room for future improvements on the *Object Finder* package. Currently, the robot navigates to locations where there is a high probability of finding the desired object and then it performs a rotation in order to obtain a more complete view of that location. The system would be compensated by using better exploration techniques as the current one might not be sufficient, in a large number of cases, to find the desired objects. Obtaining the object-places relationships in a probabilistic manner, changing the ontology to incorporate those, and updating the developed object searching algorithm to reason about those and about navigation costs would be remarkable. When searching for a laptop, should a robot move to an office 50m far away where there is a probability of existing a laptop of 95%, or should the robot firstly move to a nearest living room at 15m where that probability is of only 50%. This optimization decision problem was not addressed in this work, but it certainly is a stimulating one, and future work might be done on this area.

Chapter 6

Appendix

6.1 What is the probability of existing a given object in a given place?

6.1.1 Deep Analysis of Places205 Dataset

In a first attempt to estimate $p(\hat{o}_i|\hat{x}_k)$, i.e. the probability of existing one or more objects of class i in a place of category k, an object detector was run in the Places205 testset images and the amount of objects detected in the images of each place category was verified. When running Yolov3 in a set of 7500 images of kitchens, in 1753 images one or more refrigerators were detected, and in 272 images one or more forks were detected. This would lead to obtain $p(refrigerator|kitchen) = 0.233$ and $p(fork|kitchen) = 0.0364$. However, such rational turned out to be wrong as, even assuming Yolov3 an ideal object detector, that does not produce false positives or false negatives, it is impossible to obtain $p(\hat{o}_i|\hat{x}_k)$ with this strategy. Most Places205 images contains parts of places, i.e. an image does not represent the totality of that place, and for that reason it is impossible to know the full list of objects present in that place and obtain acceptable values for $p(\hat{o}_i|\hat{x}_k)$ (e.g. while it is very likely to found refrigerators in kitchens, we obtained $p(refrigerator|kitchen) = 23.3\%$, this is explained as a big portion of the kitchen images contained in the Places205 might not have represented the part of the kitchen where a refrigerator is located.) Some objects can also be occluded by others, this is what happened with the forks example, most forks might be located inside drawers and, for that reason, it is impossible to detect them.

6.1.2 Automated learning of objects-places relationships via ConceptNet

ConceptNet [59] is a knowledge graph that contains 36 different types of relations such as *IsA*, *AtLocation* and *Synonym*. Their ontology edges have a weight, a non probabilistic value between 1 and 10, and can be symmetric or asymmetric.

Furthermore, ConceptNet allows to analyze their ontology by performing requests to their server. A request has to be performed to a specific link which defines the concepts and relation types one wants to verify. A response is then sent by their server, in a *json* file, defining the edges found out given the executed request. For instance, in order to verify what is usually found in offices a request has to be made to the following link: `http://api.conceptnet.io/query?end=/c/en/office&rel=/r/AtLocation`. In this example, *AtLocation* is the edge's relation to be search, and 'office' is a concept which, in this relation, is always the end node (this relation is asymmetric). The response obtained from this request can then be analyzed and we find out that the following objects are usually found in offices: computer, chair, pen, cup of coffee, keyboard, fax machine, desk, etc.

An ontology was built in an automated manner by performing several queries, such as the previously explained one, and furthermore analyzing its responses. With a deep analysis of the obtained edges given the executed requests, it was noticeable that several places concepts (Places205 categories) where not related to the object concepts (MS-COCO categories) but to some synonyms of those (e.g., ConceptNet's ontology has an edge relating 'notebook' and 'office' but has no edge between 'laptop' and 'office'). Therefore, the developed ontology was also built by searching for object synonyms, using the *Synonym* relation, and verifying if any of those synonyms is linked to a place category or to its synonyms also via an *AtLocation* edge.

Several object-places relationships acquired with this procedure, and present in the built ontology, are shown in Table 6.1.

Table 6.1: Obtained object-places relationships example.

	Keyboard	**Mouse**	**Book**	**Chair**	**Fork**	**Wine Glass**	**Cup**	**Sink**	**Refrigerator**	**Bed**	**Laptop**	**Tv/Monitor**
Office	2.0			8.485							4.898	
Kitchen					5.291	2.828	2.0	6.928	7.211			
Living Room				2.828		3.464						2.937
Auditorium				1.0								
Classroom			1.0	6.324							1.0	
Bedroom			2.0	1.0								1.0
Waiting Room												

With a further analysis of Table 6.1, it is observed that the built knowledge graph still misses several important object-places relations. While there is a high likelihood of finding a bed in a bedroom, ConcepNet's knowledge graph still does not have that information. ConceptNet's knowledge graph still misses several important object-places relationships and we could not overpass that and built a proper ontology in a fully autonomous manner. Another problem would be of finding a good mapping function to translate the weighted founded connections to probability values as the obtained weights turned out to be noisy (e.g. while it is very likely to exist a chair in an auditorium, that edge's connection have the minimum weight value of 1.0.)

6.2 Objects registration screenshots

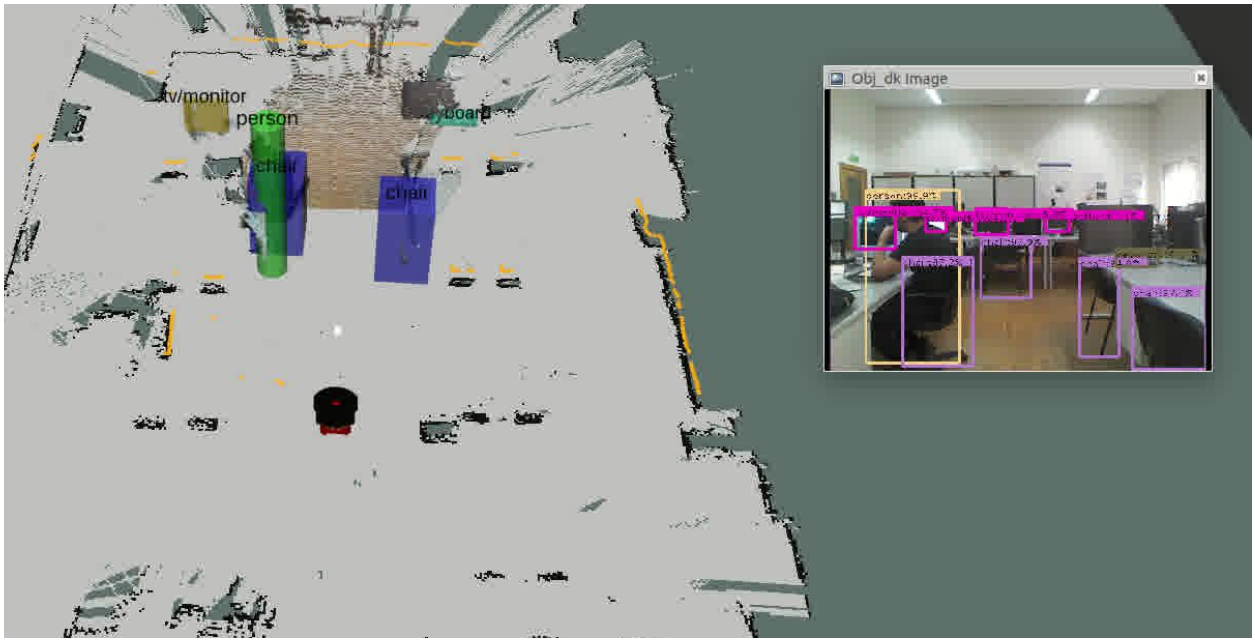


Figure 6.1: Semantic map at t=9.24s

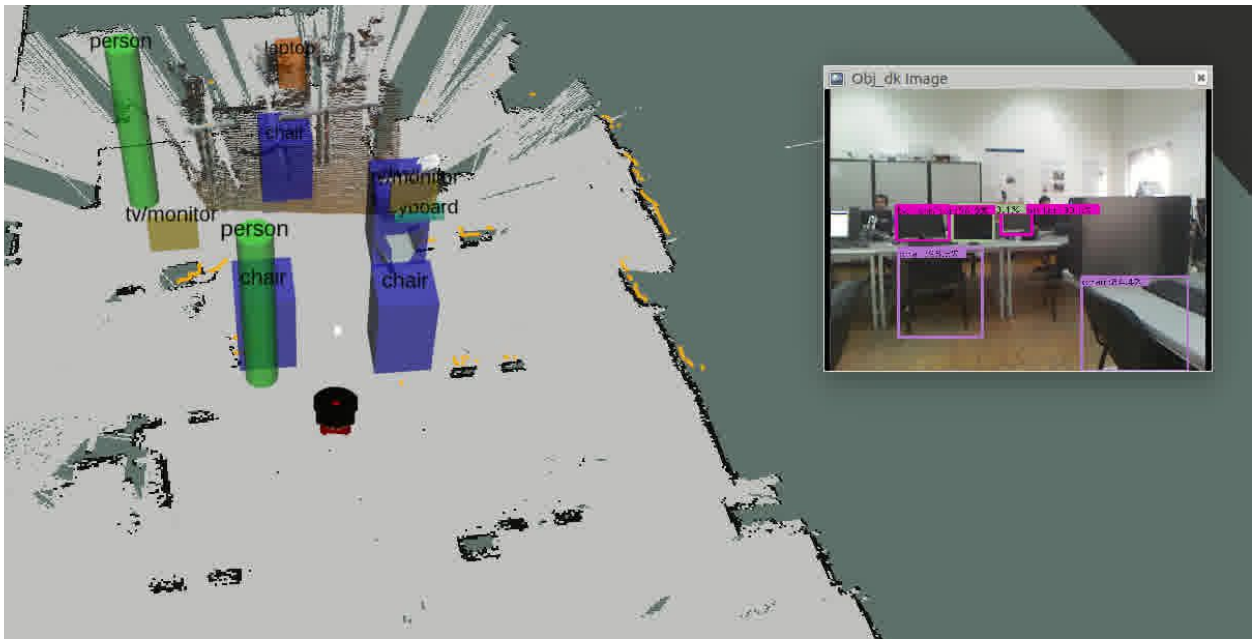


Figure 6.2: Semantic map at t=16.83s

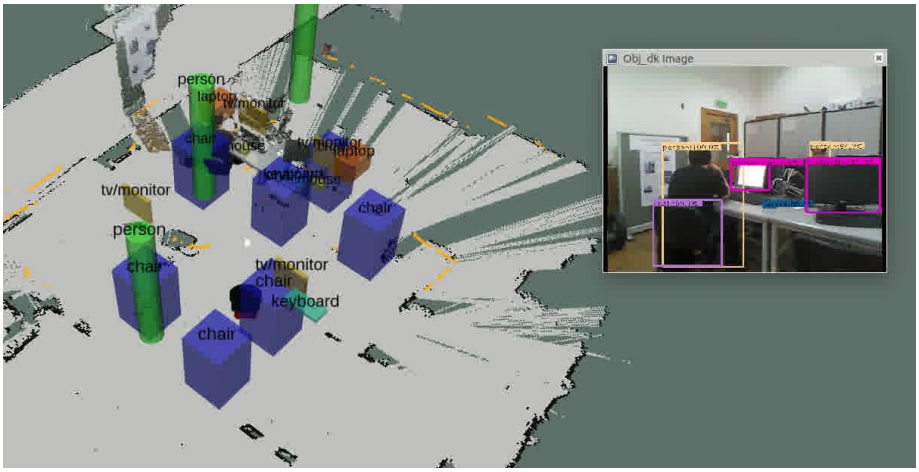


Figure 6.3: Semantic map at t=24.76s

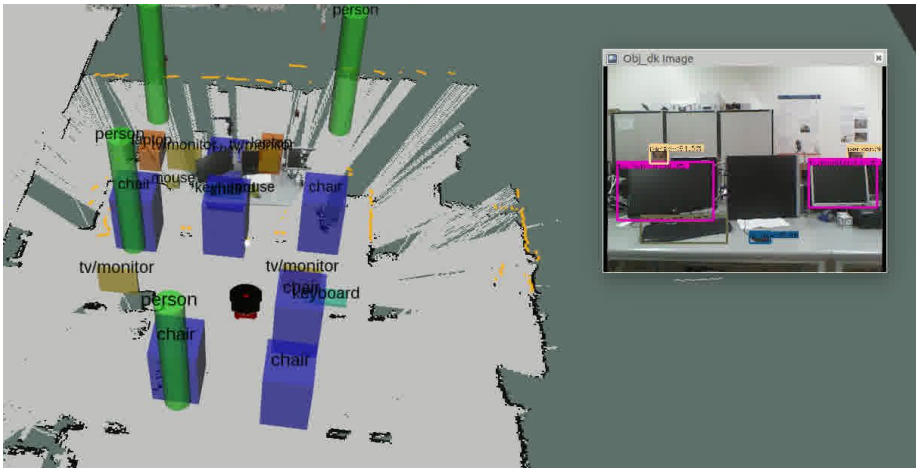


Figure 6.4: Semantic map at t=29.56s

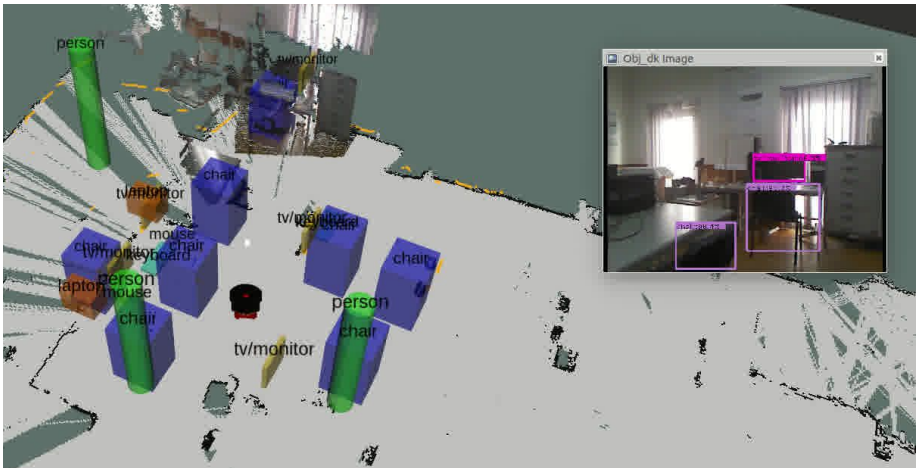


Figure 6.5: Semantic map at t=32.61s

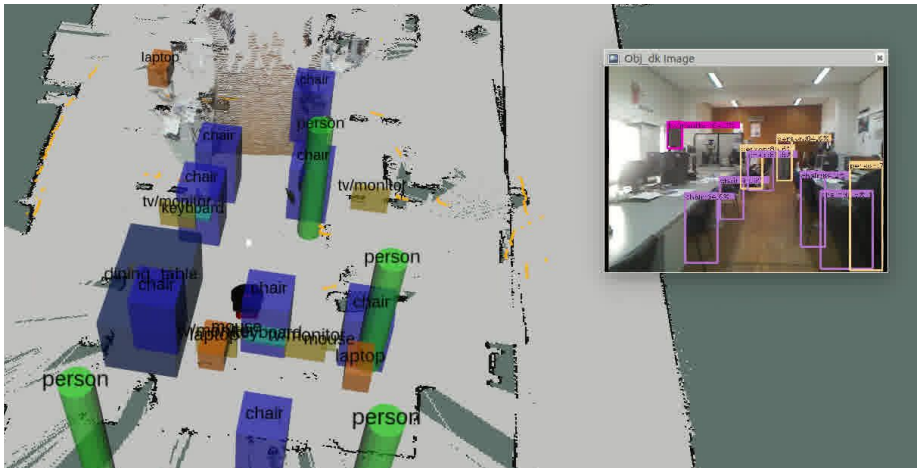


Figure 6.6: Semantic map at t=36.03s

Figure 6.7: Semantic map at t=36.59s

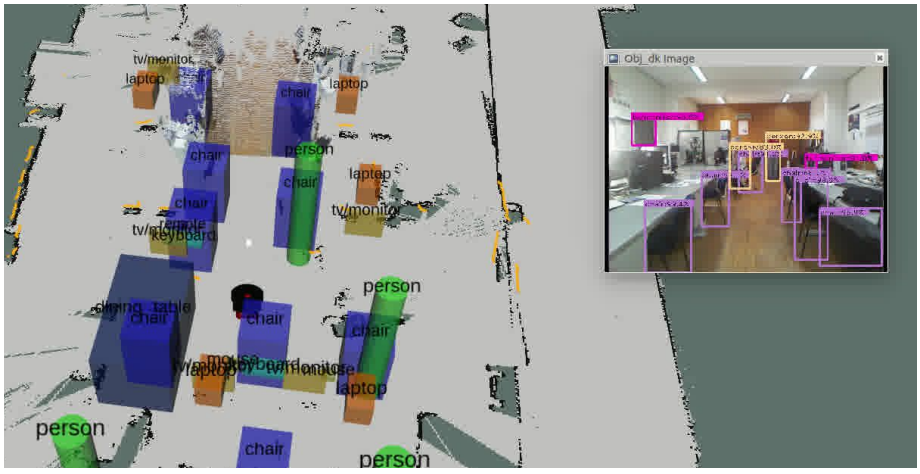


Figure 6.8: Semantic map at t=44.57s

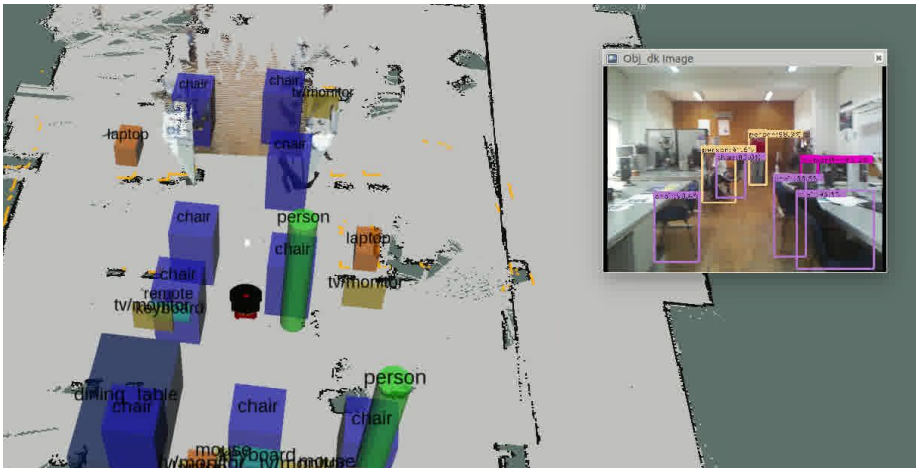


Figure 6.9: Semantic map at t=48.33s

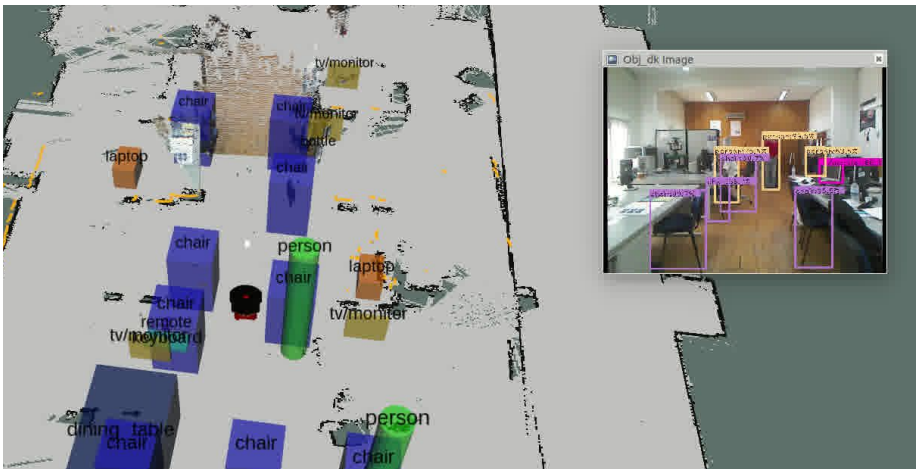


Figure 6.10: Semantic map at t=49.60s

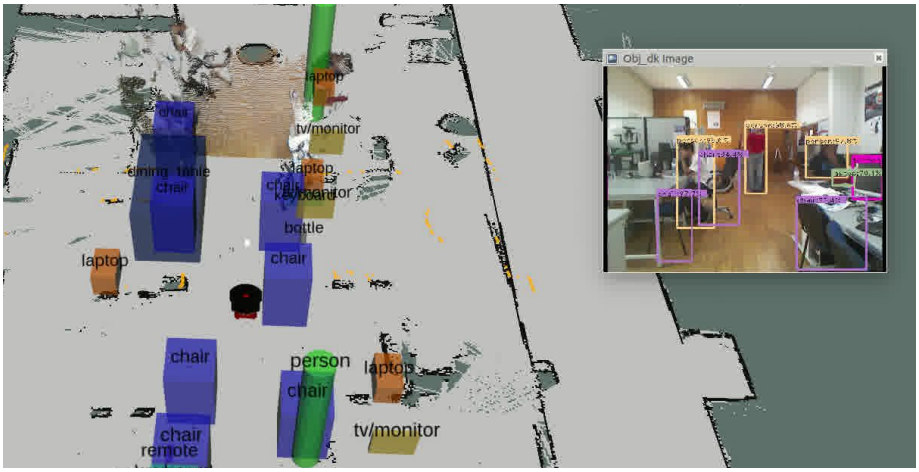


Figure 6.11: Semantic map at t=54.55s

Figure 6.12: Semantic map at t=57.82s

Figure 6.13: Semantic map at t=59.30s

Figure 6.14: Semantic map at t=64.04s

Figure 6.15: Semantic map at t=64.56s

Figure 6.16: Semantic map at t=68.49s

Figure 6.17: Semantic map at t=69.83s

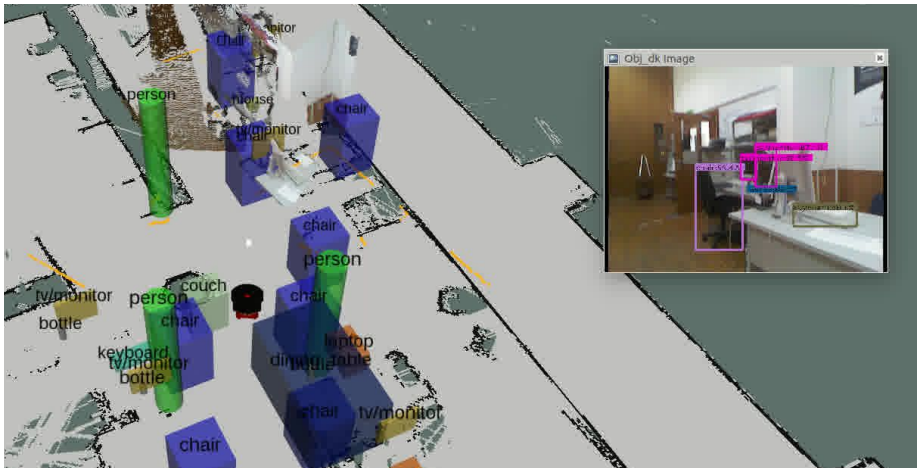


Figure 6.18: Semantic map at t=70.97s

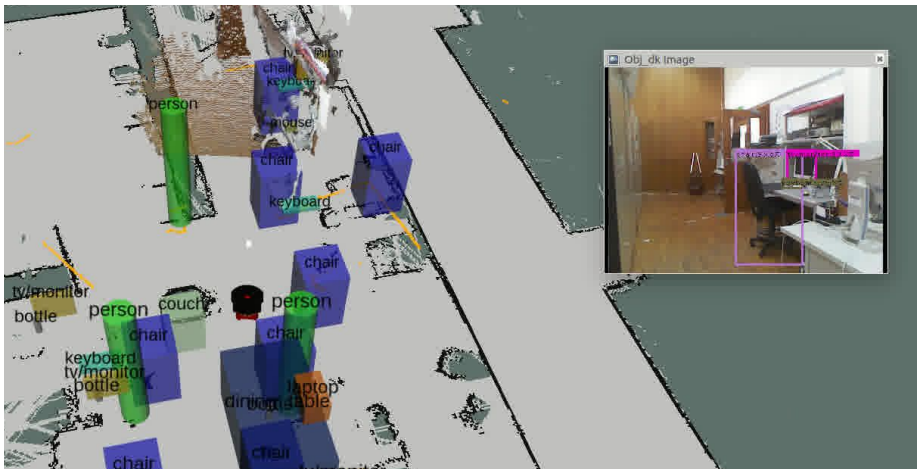


Figure 6.19: Semantic map at t=73.81s

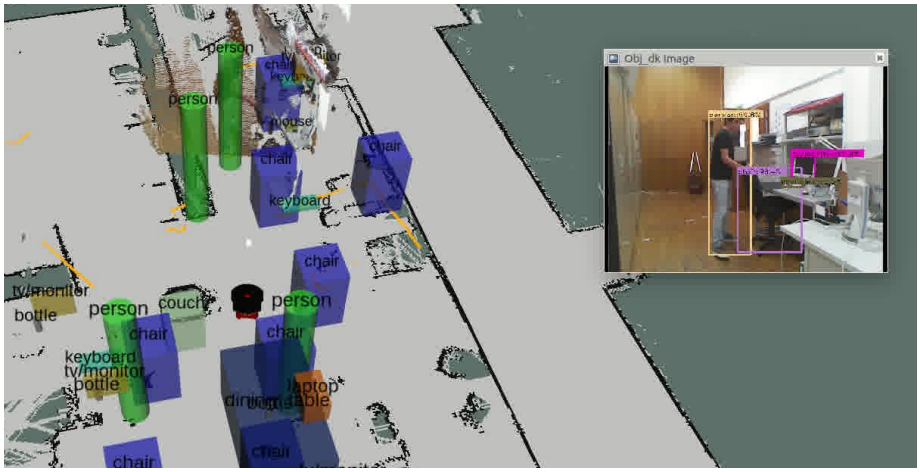


Figure 6.20: Semantic map at t=78.31s

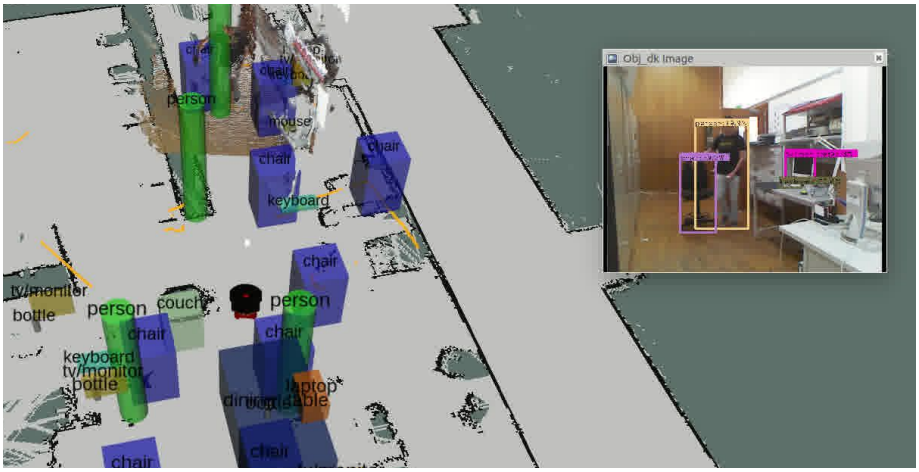


Figure 6.21: Semantic map at t=81.76s

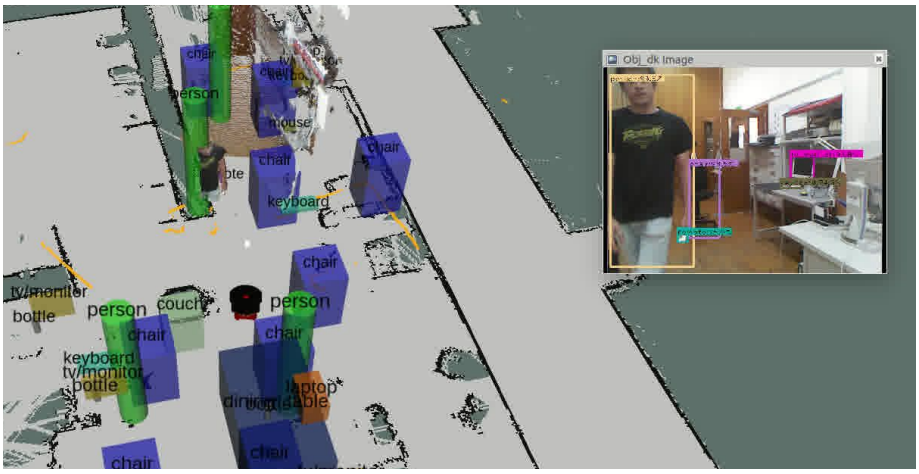


Figure 6.22: Semantic map at t=83.83s

Figure 6.23: Semantic map at t=87.56s

Figure 6.24: Semantic map at t=89.23s

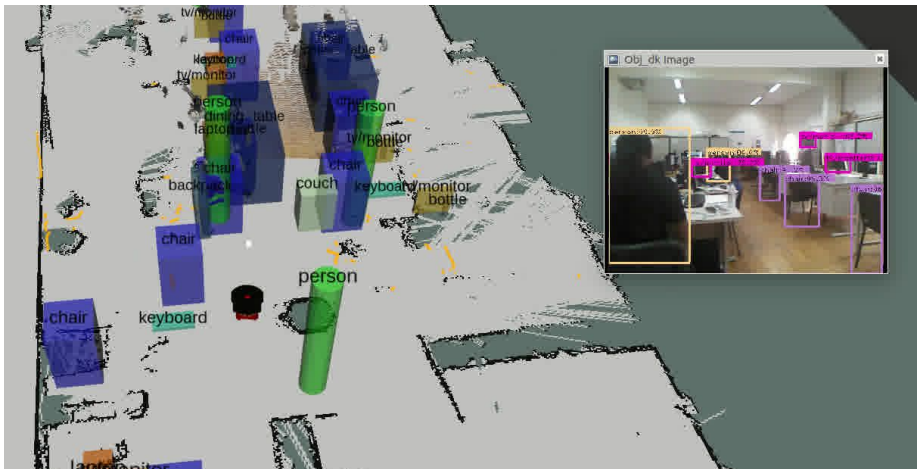


Figure 6.25: Semantic map at t=92.86s

Figure 6.26: Semantic map at t=95.83s

Figure 6.27: Semantic map at t=97.90s

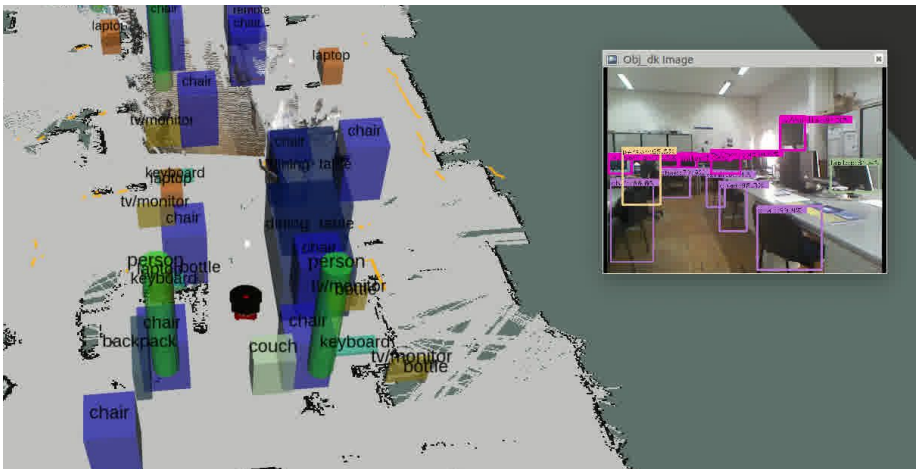


Figure 6.28: Semantic map at t=102.23s

Figure 6.29: Semantic map at t=106.01s

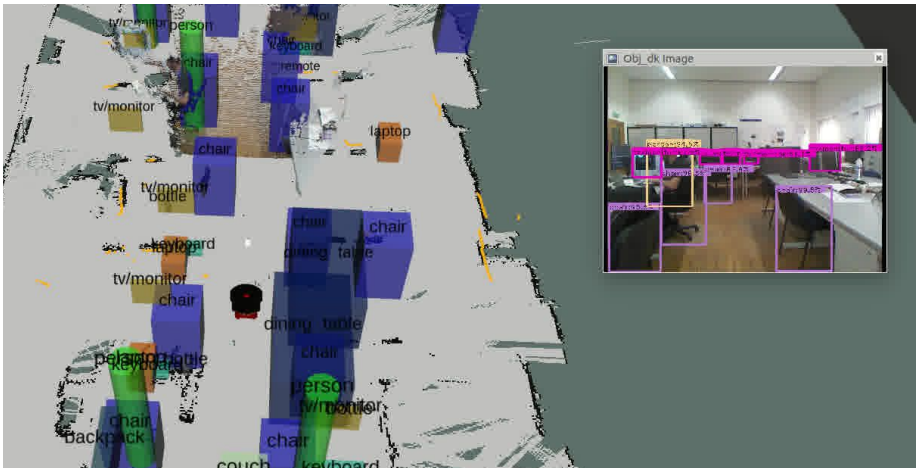


Figure 6.30: Semantic map at t=110.56s

6.3 Place Categorization Results

Table 6.2: Place Categorization Results: top-5 predicted categories

		Belief Updating Method			
Place	**Correct Label**	**Bayesian Filter**	**Bayesian Filter with prior knowledge**	**Mean Average Filter**	**Mean Average Filter with prior knowledge**
0	corridor	corridor - 1,89 Remaining 204 classes - 0,48 – – –	**corridor - 52,74** Remaining 20 classes - 2,36 – – –	**corridor - 78,65** desert sand - 2,83 basement - 2,02 highway - 1,92 lobby - 1,63	**corridor - 70,31** attic - 9,70 staircase - 4,81 bedroom - 2,84 lobby - 2,16
1	waiting room	**waiting room - 63,70** Remaining 204 classes - 0,18 – – –	**waiting room - 73,27** Remaining 20 classes - 1,34 – – –	**waiting room - 30,47** corridor - 9,04 lobby - 6,59 patio - 5,19 basement - 4,93	**waiting room - 35,91** corridor - 14,52 lobby - 10,97 patio - 8,79 office - 7,44
2	corridor	**corridor - 64,35** locker room - 0,34 Remaining 203 classes - 0,17 – –	**corridor - 82,22** Remaining 20 classes - 0,89 – – –	**corridor - 90,36** closet - 2,14 basement - 1,93 lobby - 0,72 hospital - 0,57	**corridor - 93,80** lobby - 1,25 pantry - 1,09 attic - 0,84 bedroom - 0,70
3	office	locker room - 5,33 office - 1,43 Remaining 203 classes - 0,46 – –	corridor - 67,35 office - 2,58 Remaining 19 classes - 1,58 – –	locker room - 37,91 **office - 17,99** home office - 6,78 shower - 4,14 corridor - 3,68	**office - 35,18** corridor - 23,78 home office - 12,53 kitchen - 7,85 pantry - 5,28
4	living room	shower - 3,43 kitchen - 1,84 kitchenette - 0,58 restaurant kitchen - 0,56 Remaining 201 classes - 0,47	kitchen - 76,29 Remaining 20 classes - 1,18 – – –	waiting room - 13,71 locker room - 13,59 office - 4,99 shower - 4,87 reception - 4,86	waiting room - 30,40 office - 15,82 kitchen - 8,78 lobby - 8,74 corridor - 5,89
5	office	classroom - 79,86 office - 0,17 Remaining 203 classes - 0,09 – –	classroom - 82,14 Remaining 20 classes - 0,89 – – –	**office - 62,49** classroom - 24,22 home office - 5,61 art studio - 1,50 reception - 1,38	**office - 66,52** classroom - 25,97 home office - 6,75 kitchen - 0,20 waiting room - 0,13
6	corridor	**corridor - 80,65** Remaining 204 classes - 0,094 – – –	**corridor - 82,22** Remaining 20 classes - 0,89 – – –	**corridor - 92,24** basement - 2,03 art gallery - 1,84 closet - 0,54 lobby - 0,45	**corridor - 95,17** attic - 0,89 lobby - 0,75 bedroom - 0,67 living room - 0,51
7	auditorium	conference room - 6,36 Remaining 204 classes - 0,46 –	**auditorium - 66,85** classroom - 8,79 Remaining 19 classes - 1,28 –	**auditorium - 23,81** classroom - 20,15 conference room - 19,26 waiting room - 5,84 conference center - 5,69	classroom - 35,00 **auditorium - 33,86** waiting room - 12,51 bedroom - 4,34 office - 2,48
8	corridor	**corridor - 80,65** Remaining 204 classes - 0,094 – – –	**corridor - 82,22** Remaining 20 classes - 0,89 – – –	**corridor - 98,67** hospital - 0,31 basement - 0,31 lobby - 0,23 jail cell - 0,08	**corridor - 99,15** lobby - 0,23 office - 0,07 veranda - 0,02 attic - 0,01

Bibliography

[1] G. Scott, "Gray Scott Quotes, BrainyQuote.com, BrainyMedia inc, 2019." last accessed on 07/02/2019. [Online]. Available: https://www.brainyquote.com/quotes/gray_scott_802410 1

[2] N. Sünderhauf, F. Dayoub, S. McMahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, "Place categorization and semantic mapping on a mobile robot," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 5729–5736. 1, 21, 39, 50

[3] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. A. Fernandez-Madrigal, and J. Gonzalez, "Multi-hierarchical semantic maps for mobile robotics," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug 2005, pp. 2278–2283. 1, 19, 20

[4] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009. 2, 23

[5] N. Buduma and N. Locascio, *Fundamentals of deep learning: Designing next-generation machine intelligence algorithms.* " O'Reilly Media, Inc.", 2017. 5, 8

[6] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1. 5, 6, 9

[7] X. Wang, "Deep learning in object recognition, detection, and segmentation," *Foundations and Trends® in Signal Processing*, vol. 8, no. 4, pp. 217–382, 2016. [Online]. Available: http://dx.doi.org/10.1561/2000000071 6

[8] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3025–3032. 6

[9] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017. 6

[10] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, "Advanced deep-learning techniques for salient and category-specific object detection: a survey," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018. 6, 9, 11, 13, 16, 17

[11] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285. 7

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. 7

[13] A. Saxena, "Convolutional neural networks (cnns): An illustrated explanation." [Online]. Available: https://xrds.acm.org/blog/2016/06/convolutional-neural-networks-cnns-illustrated-explanation/ 7

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105. 10, 12

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255. 10

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556 10, 19

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 10, 14

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 10

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. 12, 13

[20] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013. 12

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European conference on computer vision*. Springer, 2014, pp. 346–361. 12

[22] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. 13, 16

[23] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99. 12, 13, 16

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788. 14, 15

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37. 14, 16

[26] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242 15, 16, 17

[27] ——, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767 15

[28] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010. 15

[29] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015. 15

[30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755. 15

[31] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495. 17, 18, 19, 21

[32] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin, "Context-based vision system for place and object recognition," in *Proceedings Ninth IEEE International Conference on Computer Vision*, Oct 2003, pp. 273–280 vol.1. 17

[33] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *null*. IEEE, 2006, pp. 2169–2178. 18

[34] S. N. Parizi, J. G. Oberlin, and P. F. Felzenszwalb, "Reconfigurable models for scene recognition," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2775–2782. 18

[35] P. Sousa, R. Araújo, and U. Nunes, "Real-time labeling of places using support vector machines," in *Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on*. IEEE, 2007, pp. 2022–2027. 18

[36] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 413–420. 18

[37] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 73–86. 18

[38] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2017. 18, 19

[39] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 18

[40] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492. 18

[41] L. Wang, S. Guo, W. Huang, and Y. Qiao, "Places205-vggnet models for scene recognition," *CoRR*, vol. abs/1508.01667, 2015. [Online]. Available: http://arxiv.org/abs/1508.01667 19

[42] A. Gangopadhyay, S. M. Tripathi, I. Jindal, and S. Raman, "Dynamic scene classification using convolutional neural networks," in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, pp. 1255–1259. 19

[43] L. Wang, S. Guo, W. Huang, and Y. Qiao, "Places205-vggnet models for scene recognition." [Online]. Available: https://github.com/wanglimin/Places205-VGGNet 19

[44] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. Kruijff, and W. Burgard, "Conceptual spatial representations for indoor mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 6, pp. 493–502, 2008. 20, 21

[45] S. Vasudevan and R. Siegwart, "A Bayesian conceptualization of space for mobile robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2007, pp. 715–720. 20

[46] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 915–926, 2008. 20, 23

[47] C. Couprie, C. Farabet, L. Najman, and Y. LeCun, "Indoor semantic segmentation using depth information," *arXiv preprint arXiv:1301.3572*, 2013. 21

[48] A. Hermans, G. Floros, and B. Leibe, "Dense 3D semantic mapping of indoor scenes from rgb-d images," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 2631–2638. 21, 23

[49] C. Nieto-Granda, J. G. Rogers, A. J. B. Trevor, and H. I. Christensen, "Semantic map partitioning in indoor environments using regional analysis," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 1451–1456. 21

[50] A. Pronobis and P. Jensfelt, "Large-scale semantic mapping and reasoning with heterogeneous modalities," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 3515–3522. 21, 23

[51] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4628–4635. 22, 23

[52] T. Whelan, S. Leutenegger, R. Salas-Moreno, B. Glocker, and A. Davison, "Elasticfusion: Dense slam without a pose graph." Robotics: Science and Systems, 2015. 22

[53] Y.-T. Wang, C.-A. Shen, and J.-S. Yang, "Calibrated Kinect sensors for robot simultaneous localization and mapping," in *Methods and Models in Automation and Robotics (MMAR), 2014 19th International Conference On.* IEEE, 2014, pp. 560–565. 25

[54] K. Khoshelham and S. O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437–1454, 2012. 25

[55] M. Viager, "Analysis of kinect for mobile robots," in *Technical report.* Technical University of Denmark, 2011. 25

[56] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L 2 theory," *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, vol. 57, no. 4, pp. 453–476, 1981. 34

[57] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979. 34

[58] D. P. Doane, "Aesthetic frequency classifications," *The American Statistician*, vol. 30, no. 4, pp. 181–183, 1976. 34

[59] H. Liu and P. Singh, "Conceptnet—a practical commonsense reasoning tool-kit," *BT technology journal*, vol. 22, no. 4, pp. 211–226, 2004. 57