Tampere University

Christen Blom-Dahl Casanova

AUTONOMOUS ROBOT EXPLORATION WITH SELECTIVE OBJECT DISCRIMINATION BY USING DEEP LEARNING OBJECT DETECTION

Faculty of Engineering and
Natural Sciences
Master of Science Thesis
May 2019

ABSTRACT

Christen Blom-Dahl Casanova: “Autonomous Robot Exploration with Selective Object Discrimination by Using Deep Learning Object Detection”
Master of Science Thesis, 105 pages, 2 Appendix pages
Tampere University
Robotics and AI
May 2019

Since the geopolitical world is not polarized anymore, the market competitivity is increasing as never before so in order to survive as an industrial organization, it is key to be competitive. That is, reducing costs and production times among other needs. Mobile robots are resources that manage to get those needs relieved since they can substitute humans and perform better. This causes human issues casuistic drop, human resources re-allocation in more creative job positions which cannot replaced by robots, and more long-term efficiency.

The state-of-the-art of the use of mobile robots remains on the fact that we are talking about not just a single mobile robot but a fleet of them which performs in a smart and coordinated way. These devices can be integrated in the supply-chain so that can transport payloads without the need of any human intervention. In addition, such integration allows a huge flexibility since smart industrial mobile robots can adapt to new conditions, imposed parameters and obstacles that were not predicted. For any autonomous mobile robot, a prior knowledge about its environment is necessary before performing autonomous navigation, that is to have a previous map. Mapping usually is a human intervened task which takes time, especially for large facilities. This work proposes a way to map autonomously, in the most efficient way, an indoor 2D environment by using the Rapidly-exploring Random Trees approach since it is biased towards unknown regions.

In addition, this work proposes object discrimination during mapping. With the conventional approach, during the mapping process laser scanners read the presence of all the obstacles in the environment. This fact is undesired since some of such scanned obstacles are scanned just by causality during the exploration (e.g. personnel, industrial mobile equipment…). Such undesired registered data in the map suppose noise and does not represent the actual long-term environment. The implementation of removing such noise is managed by the combination of two modules. On one hand, by using state-of-the-art deep learning tools in order to achieve real-time object detection. On the other hand, a filter to the laser scanner so that it is blind towards such detections during the exploration, so they are never registered on the map.

The results show quite potential high-quality results which are intrinsically associated with the object detector module. Since such module is state-of-the-art, the technology involved is constantly developing and improving not just the performance but also flexibility and capabilities. This work is a potential new high-fidelity approach besides the conventional approach in order to perform mobile robot exploration.

Keywords: *Exploration, Rapidly-exploring Random Tree, RRT, Robot exploration, Laser Scanner, MiR, Industrial Mobile Robot, Non-Structural, Mapping, Robot Operating System, ROS, Computer Vision, Mapping, OpenCV, SLAM, Stereo Camera, Intel RealSense, Pointcloud, Deep Learning, Object Detection, Pretrained Models, Transfer Learning, 3D Tracking, 3D Localization, Machine Vision, Robot Perception,…*

PREFACE

Without the contribution of active members on ROS forums (like Thomas Delbaere), GitHub (especially Martin Günther), and guides on the Internet this would not have been possible at all. This demonstrates how powerful can be an open source altruistic community. Also, thanks to the deep learning community for providing the required datasets and pre-trained networks that makes possible transfer learning.

I would also like to thank:
- TUT-Robotics Slack members for their contributions
- Matti, awesome IT guy from FAST department in TUT, for providing me a suitable laptop for my implementation
- ROS Ignite Academy
- Prof. Jose Luis Martinez Lastra for the Master's thesis position
- Prof. Joni Kämäräinen for his time, kindness and advices

Finally, and the most important, I would like to thank my mother and father for all the support and advices along this tough but rewarding challenge.

This work contains research areas such as Autonomous Exploration, Mobile Robots, Mobile Robots applied to Logistics, Path Planning, SLAM, Image Processing, Neural Networks, Convolutional Neural Networks, Deep Learning, Machine Learning and Pattern Recognition, Computer Vision, Robot Perception, depth machine vision, and integration in ROS.

CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

3D	3-Dimension
AGV	Automated Guided Vehicle
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processor Unit
CUDA	Compute Unified Device Architecture
cuDNN	CUDA Deep Neural Network
DBN	Deep Belief Network
DLP	Data Level Parallelism
e.g.	exempli gratia (for example)
Exp.	Experiment
FAST	Factory Automation Systems and Technologies
FC	Fully Connected
FFD	Fast Frontier Detector
GPS	Global Positioning System
GPU	Graphic Processor Unit
i.e.	id est (that is)
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IMU	Inertial Measurement Unit
JSON	JavaScript Object Notation
LASER	Light Amplification by Stimulated Emission of Radiation
LiDAR	Light detection and ranging
LTS	Long Term Support
MiR	Mobile Industrial Robot
NLP	Natural Language Processing
NN	Neural Network
No.	Number
OpenCV	Open Computer Vision
PDF	Probability Density Function
RAM	Random-access memory
RBM	Restricted Boltzmann Machine
RBPF	Rao-Blackwellized Particle Filter
R-CNN	Region-Current Neural Network
ReLU	Rectified Linear Unit
REST	REpresentational State Transfer
RGB	Red Green Blue
ROC	Receiver Operating Characteristics
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RTAB	Real-Time Appearance-Based Mapping
RViz	ROS Visualizer
SGD	Stochastic Gradient Descent
SIMD	Single Instruction Multiple Data

SLAM	Simultaneous Localization and Mapping
SRT	Sensor-based Random Tree
SSD	Single Shot Detector
SVM	Support Vector Machine
UNIX	from uni- ‘one’ + a respelling of -ics, on the pattern of an earlier less compact system called Multics
USB	Universal Serial Bus
VOC	Visual Object Classes
WFD	Wave Frontier Detector
Wi-Fi	Wireless Fidelity
YOLO	You Only View Once

LIST OF SYMBOLS

IoU	Intersection over Union	
mAP	Mean Average Precision	
FPR	False Positive Rate	
AUC	Area Under Curve	
PPV	Positive Predictive Value	
Gbps	Gigabits per second	
GB	Gigabytes	
GHz	Gigahertz	
D	Laser beam length	m
c	Speed of light	m/s
t	Time	sec
x0	Projection of the observed point, X0, in the scanline left image	m
x0’	Projection of the observed point, X0, in the scanline right image	m
X0	Observed point in the 3D world	m
B	Baseline	m
B_1	Distance from 0 to the projection of X0 in the baseline	m
B_2	Distance from 0’ to the projection of X0 in the baseline	m
f	Focal length	m
z0	Normal distance from X0 to the baseline	
C	Pointcloud or set of p_i	m
p_i	Point in the 3D space belonging to C	m
i	i-th iteration over series or loops	
{x, y, z}	Coordinates in the 3D cartesian space	m
X	Map	
x	Point belonging to X	
x_{rand}	Random point belonging to X	
x_{init}	Initial point	
$x_{nearest}$	Nearest point between x_{rand} and the graph G	
x_{new}	Output point generated by the Steer function	
$x_{current}$	Current point during the iteration	
x_{free}	Free space in the domain of X	
V	Vertices vector which contains v elements	
v	Vertex generated by the RRT	
E	Edges vector which contains e elements	
e	Pair of vertices generated by the RRT	
G	Graph which is a set composed by V and E	
z	Output from steer function which minimizes $\|\|z - y\|\|$	
η	Tree growth rate	
Q	Vertex set	
u	u-th vertex	

$\emptyset$	Null symbol	
∞	Infinite symbol	
O	Total amount of obstacles	
m_i	i-th obstacle present in the mapped environment	
m	Set of m_n	
j	Robot pose	m
k	Observation (e.g. laser scanner reading)	m
l	Control input (e.g. odometry data)	m
p	p-th particle	
$\mathbb{R}^{dim}$	Real domain of dim-th dimension	
dim	dim-th dimension	
x1	Point remaining to a certain cluster	
N	Total amount of x1	
h	Bandwidth of the kernel	
x2	Input data of a perceptron	
b	Bias of a perceptron	
w	Weight of a neuron	
y_k	Output prediction of the neuron	
w_{ab}	The b-th weight of the a-th neuron	
ε	Learning rate of a neural network	
E	Error function	
c or C_i	Category of a detection	
γ	Sensitivity or threshold of a detector	
K	Thousand unit	
P_i	Probability of a detection	
mm	Millimetres	
º	Decimal degrees	
m	Meters	
m^2	Squared meters	
#	Number of / Amount	
$TPRP$	True Positive Ratio Performance	
$LMQR$	Local Map Quality Ratio	
$P.O.$	Permanent Object set	
$T.O.$	Temporal Object set	
M_f	Filtered map	
M_{gt}	Ground truth map	
p.u.	per unit (in this case, per unit of the average of times required to perform exploration with no obstacles)	
AVG.	Average	
#	Number (quantity)	

1. INTRODUCTION

The topic of robot navigation is getting more and more relevant nowadays since quite useful applications on industry had been found to be interesting tolls that fight costs and time wasting. A mobile robot in order to proceed with an efficient navigation must know its environment through a representative map a priori. And the procedure to obtain such map is what is known as map exploration, which is moving around an unknown region while recording data out of it on a map[1]. The targets of this procedure are to minimize the time of map building while maximizing the area to explore considering a behaviour of the mobile robot biased toward regions still unexplored avoiding the zones already mapped.

The navigation hierarchy can be represented in a simplified manner as in Figure 1. The hierarchical approach in SLAM starts with the lowest level, which is localization. This is for the robot knowing real-time position and orientation on the real world, that is pose, which is with respect a selected frame (global or relative). After this priority, the following level is the ability of moving autonomously of the robot which is performed by path planning which consist of the generation of targets and moving towards them considering obstacle avoidance, that is an improvisation ability. Once these both levels are operating, then it is possible for the highest level to be executed, that is exploration, which must be performed in parallel with the other levels of hierarchy while updating the map.

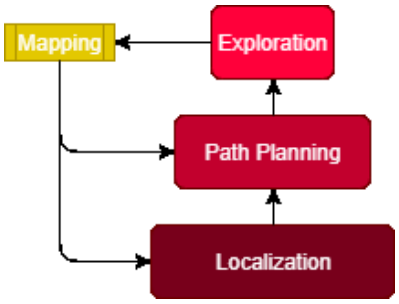


Figure 1. *Hierarchical approach in SLAM.*

In comparison with other exploration approached, the frontier-based ones have currently more attention from the research community[2]. Why? Because they are more efficient since there is no redundancy on the exploration, that is, not exploring again known regions. This is achieved as the mobile robot is biased towards frontiers which represent the borders between known from unknown areas of the map. Image processing tools for the frontier edge detection makes this approach possible. Other approaches are based on randomized path planning methods whose hypothesis relies on the inherent probabilistic feature of already being biased towards the frontiers[3].

This work proposes a frontier-based strategy based on the path planning method of Rapidly-exploring Random Tree, which can be extrapolated to the third dimensional space. The navigation under this implementation ensures that the path planning is independent from the growth of the tree so that makes able the tree to spread quicker increasing the efficiency of the algorithm. It is also considered the minimization of energy consumption in the way that the target assigner discards those targets which are too far away from the mobile robot. This is because in such cases the likelihood of exploration redundancy is high. So that the implementation has a target filter assigner module whose purpose is to consider relevant points to have as navigation goals.

In addition to the previous, once the exploration is done with the conventional approach, all material shapes present in the environment, regardless whether they are relevant or not to the map, have been recorded on it. That are, personnel walking around, temporal objects (such as industrial trucks) and other objects which can be assumed that are just transitory in the environment. The work pursues an implementation using object detection algorithms based in deep learning and a map filtering module. With the information provided by the object detector (bounding boxes encapsulating the detections), it is possible to omit such objects during the map scan. Since nowadays there are available quite powerful and flexible deep learning-based object detectors, it is possible to infer an object category despite its perspective, illumination variance, and bizarre shape. In addition, this set of tools allows to use pre-trained models which nowadays can detect dozens of different object categories (chair, person, car, …) and filter them in order to detect a certain class of objects.

1.1 Motivation

Nowadays, the industry is experiencing a transition to a next level never experienced before. Globalization is the catalyst for this situation as generates competitivity worldwide. So that, industrial businesses must update their systems in order to survive in the market. This transition is known as Industry 4.0[4] or The 4th Industrial Revolution. Which is the trend of automating as many manufacturing processes as possible via digital data exchange and monitoring. One application can be the management of a fleet of mobile robots in a highly automated factory. Mobile robots need previous information of their environment encoded in a map in order to perform path planning. The problem comes when such map does not represents the reality properly, so neither do the path plannings. The effects of this fact can be perceived in a long-term period, especially, this effect is exponential when such path plannings are performed by a fleet of robots that use the same map.

1.2 Problem definition

Mapping the indoors of industrial facilities takes a considerable amount of time as the areas to explore are usually large. The usual approach is done via teleoperation which involves human resources to commit such task. In addition, the environments to map in the industrial context are usually quite dynamic, i.e., there are constantly mobile elements moving around or certain types of objects, although ”static”, that are known that they will not be in the same position in the near future since they are considered temporal. The problem comes when during the mapping, these objects are recorded anyways without regard of their features which make them not suitable to figure on the map.

1.2.1 Justification of the work

The point of seeking for a solution is to upgrade the synergy of the processes to the next level. That is adopting the premises that Industry 4.0 postulates. In few words, to improve competitivity minimizing costs in terms of time and money by allocating correctly available resources) and enhancing all the processes involved in the supply-chain.

1.2.2 Problem Statement

Exploring autonomously an area with the possibility of discarding, from the generated map, certain selected categories of objects in the environment under the hypothesis that they are dynamic in the space (i.e. temporal or positionally non-static).

1.3 Research questions

- How can I perform autonomous navigation which involves path planning without a map?

  There exist exploration algorithms that are based on SLAM by targeting unknown areas

- How can I perform object detection with generalization capacity to know a certain type of objects?

  Deep learning has the required capability of generalization from the learning. That is, can recognize an object of a certain type even though has never seen it before (i.e. it is not in the training set).

- What sensors do I need for the implementation?

  - Laser scanner provides the readings for the mapping
  - Depth camera provides the RGB image required to feed the Object Detector Module

- What software environment do I choose to work with?

  Robot Operating System (ROS). Since this environment provides open source ready-to-use packages, high flexibility and high interchangeability, easy communication with the robot and other devices, and community support.

- Do I need a GPU for the implementation?

  Yes, the object detection algorithm requires a lot of tensorial computations that the CPU cannot handle properly. Modern GPUs are optimized for such applications.

1.4 Objectives

The aim of this work is to increase the automatization level in the previous stage of the integration of the mobile robots in the supply-chain and to increase the efficiency during the integration phase. The former goal is achieved by automating the usually teleoperated mapping phase before the mobile robot fleet deployment. The later goal, by making the mobile robots compute the path plannings according to real environments free of mapped objects that actually do not exist anymore in the position they were recorded. This actually means more efficient path planning, since the generated paths are more straightforward since they are not affected by re-routing due to objects that presumably do not exist anymore in such recorded positions. This is made under the hypothesis that a certain class of object is considered transitory. A case would be an industrial mobile equipment that was transporting goods in a facility was recorded while mapping with the mobile robot. Most likely, such object will not be anymore in that static position anymore in the future.

1.5 Thesis outline

After clearing the context of this work, now you will see the fundamentals and theory that makes the implementation work behind scenes. Under a scientific approach, this document makes clear any rational questions sort of “How does it work?”, “How come is this possible?”, “What is/was the context of this?”, “Is actually black magic what is behind all of this? Witchery may be?”. If you are not familiar with concepts such as SLAM, robot perception, neural networks, deep learning, object detection and so on, it is highly recommended to go through this point.

Once the concepts behind scenes are clear enough, the proposal of the work indicates what and why in this work certain approaches were considered as valid. Then in the implementation point, documentation provides a guide on how come this was managed to make the whole thing work.

Finally, the results of the synergy of every module of the system are provided as evidence that the proposed system works. Also, at the end, the document suggests enhancements and future developments from this work.

2. LITERATURE AND TECHNOLOGY REVIEW

2.1 Industrial mobile robots

Since the 3rd industrial revolution, the attention and use of industrial mobile robots has increased exponentially due to the constant evolution of electronics. The point of such fact is contributing to the increase of the automation degree promoted by that revolution. It is proven that despite these devices commit mistakes, they are less frequent than the ones made by human labours. In addition, they are capable of working in hazardous areas which for a human would be dangerous[4].

The use of mobile industrial robots is mainly intended for payload transportation and palletization.

Figure 2. *Palletizer mobile robot designed by Boston Dynamics. [5]*

Before the current autonomous mobile robots, the traditional Automated Guided Vehicles (AGV) had not path planning algorithms in order to perform navigation and they relied usually on guide tapes (magnetic or colored if robot posses primitive machine vision) printed on the floors so the robots could follow such manually and previously made paths. The main drawbacks are the lack of flexibility and use limitation for elementary fleet coordination and non-complex environments.

AGV navigation system based on guided tapes. [6]

Since technology evolved, smarter mobile robots now are capable of performing real-time path planning which enables the possibility of coordinate large fleets of mobile robots, improvise on-the-way if there are obstacles such as humans with obstacle avoidance protocols that communicate directly with path planning algorithms. The implementation of the technologies offers huge flexibility since paths can be modified and monitored from a computer which is connected wirelessly to the mobile robots[7].

A more robust and advanced robot perception makes the implementation possible. The mainly used sensors in order to perform autonomous navigation are laser scanners and depth cameras.

Figure 3. *Amazon robot fleet in warehouse. [8]*

For this it is needed previous information of the environment contained on the map of the environment where the mobile robots are going to perform and share space. Thus, the global planner is based on such map in order to compute path planning and proceed with SLAM. Note: all the robots of a fleet working in the same area use the same map.

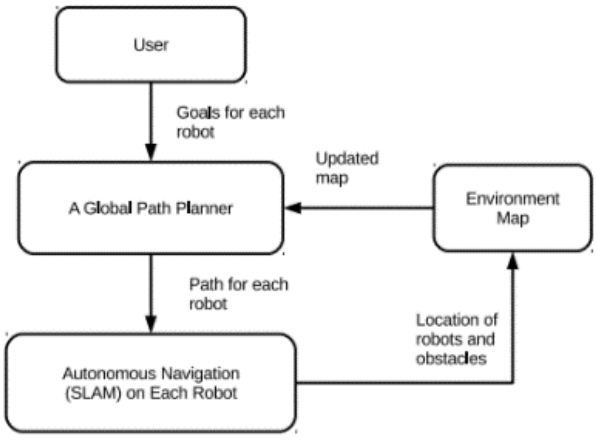


Figure 4. *Diagram of fleet management system for autonomous vehicles. [7]*

2.2 Robot perception

Robot perception is the ability of a computerized system to interpret the world environment through the data provided by sensors such as laser scanners or depth cameras. This feature is fundamental in order to make the mobile hardware perform in a coherent way with the real world.

2.2.1 LiDAR technology

An application of the LASER technology is LIDAR which is a surveying which quantifies the distance between the target and the sensor. It is robust perception approach since the illumination and noise are invariants. The parameters to compute such distance are wavelengths and return times of the reflected pulsed laser light. In order to compute the distance, the speed of light and the time of the returning photon coming from the laser beam is:

$$D = \frac{1}{2} \cdot c \cdot t \quad (1)$$

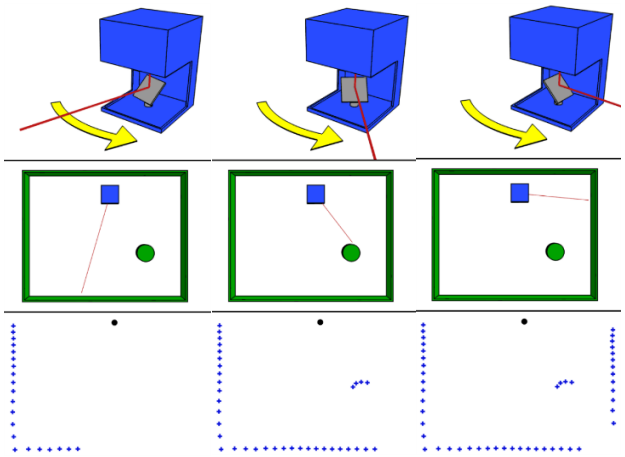


Figure 5. *Surveying process of a single laser beam.[9]*

Such rotating laser can be stacked with other beams so several scanned planes are retrieved forming a pointcloud.

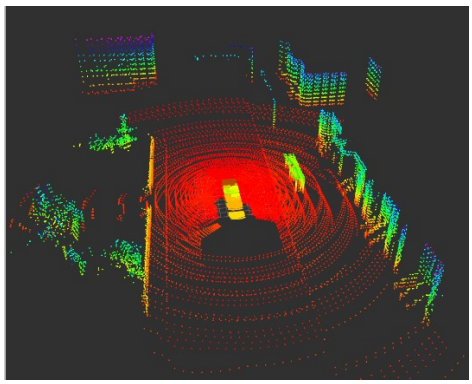


Figure 6. *Pointcloud formed by multi-laser scan layer horizontally stacked. [10]*

2.2.2 Stereo vision

A very important ability to have depth perception of the real world, that is why life evolution has provided animals and humans with two sensors (eyes) instead of one. In the field of computer vision, stereo vision is the extraction of the depth information from digital images. Such extraction is made in dense depth maps and pointclouds.

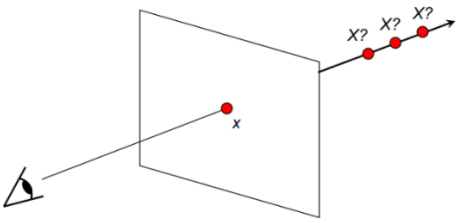


Figure 7. *The depth retrieval information from a single image is inherently ambiguous.*

The mathematical model of the stereo vision is based on the epipolar geometry which is described as follows:

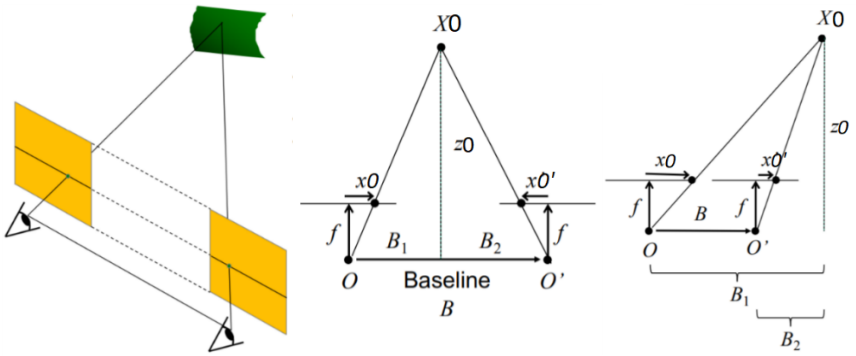


Figure 8. *Geometric representation of the mathematical model of the stereo vision. [11]*

The depth $z0$ is function of the disparity, which is defined as:

$$disparity = x0 - x0' = \frac{B \cdot f}{z0} \quad (2)$$

The baseline, which is the distance between the cameras, is modeled as:

$$\begin{cases} B = B_1 + B_2 \; if \; the \; projection \; of \; X0 \; is \; out \; of \; the \; baseline \\ B = B_1 - B_2 \; if \; the \; projection \; of \; X0 \; is \; inside \; the \; baseline \end{cases}$$

And finally f is an intrinsic parameter of the camera defined as the focal length.

2.2.3 Point Cloud

It is the representation of the scene and its objects, which are observed by a certain scanner sensor, in a set of points displayed in 3D Euclidian space[12]. The points are meant to be in the contours and surfaces visible by the aforementioned scanner which is a valuable information for depth perception.

The point cloud is mathematically defined as:

$$C = \{p_i\}_{i=1}^{n} \quad (3)$$

in which p_i means a point in such cloud:

$$p_i = \{x, y, z\} \quad (4)$$

which are the global coordinates with respect to the scanner sensor frame.

The information of a single point of the cloud usually is irrelevant, so it is the whole set which is useful for data analysis. Some applications[13] of point clouds are the generation of 3D maps (RTAB-Maps), extraction of keypoints in the 3D space for descriptors, outliers filtering from noisy data, robot perception, LiDARs, and so on.

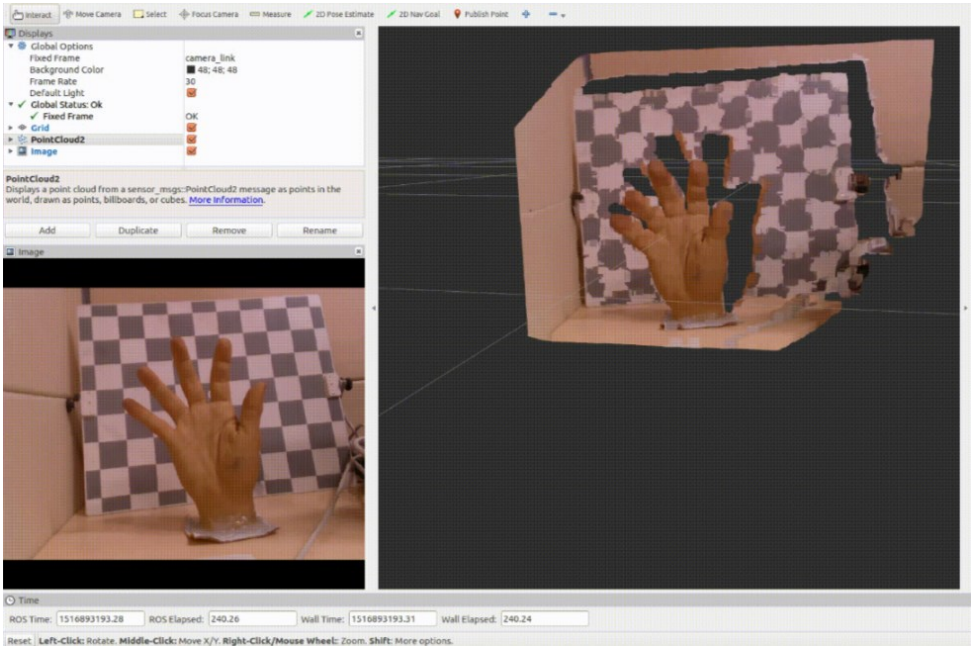


Figure 9. *Pointcloud with RGB data in Rviz.*

On the left side, an RGB image received by a depth camera. On the right side, the point cloud generated mixed with RGB data. [14]

2.3 Environment exploration

One of the godfathers of the frontier-based exploration is Yamauchi[1]. His work could make a mobile robot to explore autonomously a complex indoor environment with obstacles and map it. It was based on the principle that the algorithm is biased towards unexplored areas. After an iterative process, the whole area should be completely explored. Such implementation was tested using a mobile robot in a real office which contained furniture, objects and so on as obstacles.

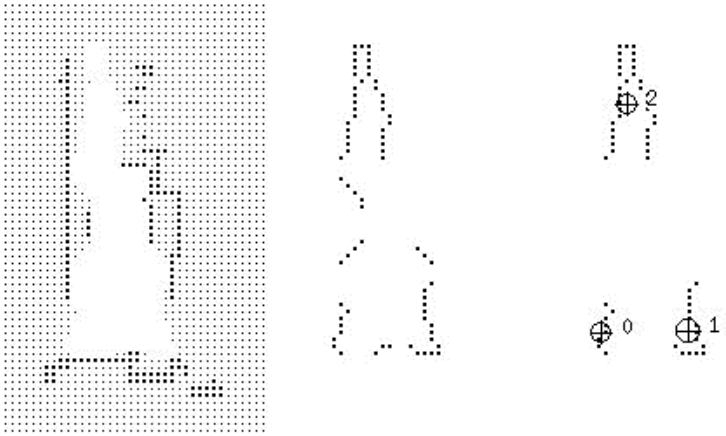


Figure 10. *Yamauchi's map obtainment process.*

a) Generated map b) Frontier edges c) Frontiers areas

The empty spaces (white areas) represents known space, the region with dots symbolizes unknown region and the plusses symbols circumscripted in circles are the frontier areas.

In his implementation, frontier detection is an iterative process that re-starts every time the mobile robot arrives the designated target position. This feature means a high computational cost[15] since the whole map has to be scanned completely every iteration. After Yamauchi's work, alternative and variants of exploration strategies based on the idea of frontier detection were discovered.

Not many years ago, researchers[2, pp. 113–120] proposed two more efficient variants of frontier detection strategy. FFD (Fast Frontier Detector) and WFD (Wave Frontier Detector). The first approach just considers the laser scans messages (which were converted in contours) in the frontier searching. The process consisted on the fact that when the detected frontier is not previously stored then it is created and stored. The second approach considers just the known areas of the map. The trigger of the extraction of the containing edge happens when a target is found on the frontier edge.

Other exploration approaches are based on randomized path planning methods. The purpose of frontier-based approaches is to avoid redundancy while exploring unknown maps, that is to avoid already explored regions. This is accomplished by exploiting the

inherent probabilistic properties of the RRT algorithms. So that, there is no need of detecting frontier regions in order to direct the mobile robot towards to unexplored areas. The resulting growth of the tree[3] during robot navigation is biased the frontiers so that the map is iteratively expanded.

There is a variation of RRT algorithm known as SRT (Sensor-based Random Tree) [16] in which navigation goals are randomly generated within a range limited surrounding space of the mobile robot laser scanner. On the other hand, RRT generates target positions along the entire map. This main difference makes this variant a depth-first exploration approach since the sequence of random targets will appear like a chain of nodes. Instead, in RRT, trees branches spread in alternative directions sub-growing in sub-branches (kind of fractals). Because of the divergence of both variants, the SRT implementation need of backtracking. This is the mobile robot requires to retrocede and track parented nodes of the ascending node when a branch stops spreading, i.g. this can happen if the robot arrives to a dead end for instance. The problem comes when backtracking involves visiting certain places more than once, since it an undesired feature. Nonetheless, some researches[17] propose enhanced treatments to this issue.

The exploration methods that cannot be related to the aforementioned approaches can be categorised into information-based exploration[18], [19]. That considers mapping and localization simultaneously with exploration in the way that target positions are selected so that increases the certainty of the mobile robot pose, thing that improves the information got in the map.

2.4 Path Planning

This is the core procedure in navigation. There is a starting position where the robot begins to compute the path planning and a goal position to which the robot has to arrive somehow. These two points are located in a map which contains the obstacles that should be avoided. Here it is covered an explanation and differences between Dijkstra’s algorithm and RRT algorithm. The main difference between both implementations is that the former one is used when the map is already built and the purpose is to minimize the distances between points. Meanwhile the later one is meant to be used to find exploration goals, that is, when the map is not made yet.

2.4.1 Common concepts in Dijkstra's and RRT algorithms

Map X: Represents the space containing the whole information.

FreeSpace X_{free}: It is the space containing nothing.

Vertices V: Points or nodes appearing on the map. The RRT algorithm generates them so they are interconnected forming tree branches. Each point is defined as vertex. The whole set of the different vertices are stored in a vector V.

Edge E: It is the line connecting pairs of vertices. Each edge is stored in terms of the spacial coordinates of both points. The edges are stored in the vector E.

Graph: Vertices and Edges generates a graph. G = (V, E).

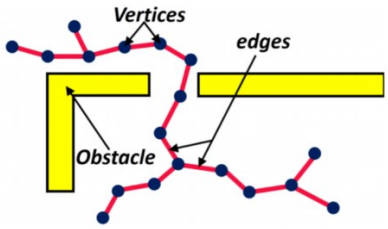


Figure 11. *RRT graph structure. [20]*

Nearest(G = (V, E), $x \subset X_{free}$): It takes a certain graph and a generic point in the free space as input. This function returns the closest vertex to a point $v \subset V$ in the way that Nearest(G = (V, E)) = argmin $_{v \subset V}$ ||x – v||.

Steer: This function takes a pair of points with x and y coordinates and outputs a z point where ||z – y|| is minimized, while ||z – x|| ≤ η, which η is the tree growth rate and η > 0.

ObstacleFree: The output of this function is the type of a boolean, in which a pair of points in the FreeSpace are the input. So it outputs False if there is any obstacle between them.

getFirstMinCost: It returns the first vertex $x \subset G$ with the lowest cost.

Neighbor(Q, u): It returns the vector of vertices that are directly connected to *u* vertex, $X_{neighbor} \subset Q$.

Cost(x_1, x_2): Represents the cost assigned to an edge connecting a pair of vertices. It can be an assigned cost or the edge length.

Parent: Each parent vertex can generate multiple child vertices and has a unique parent.

2.4.2 Dijkstra's algorithm

From a certain graph G(V, E), this algorithm is meant to find the shortest path between any pair of vertices that belong to G under the assumption of such path. In the implementation of the work proposal, this algorithm is used to coordinate the mobile robot toward goal nodes generated by the tree obtained via RRT algorithm. For a certain vertex R on the minimum distance (or cost) path between two other vertices (A and Z), seeking this path means finding the minimum path between A and R. This means that the optimum path which connects a pair of vertices is also the optimum paths connecting all the belonging vertices[21]. So that, the algorithm keeps seeking for the optimum paths to all vertices until the goal node is reached.

```
                        Dijkstra algorithm
1: Q ← Ø;                                   // Create empty vertex set
2: Parent ← Ø;                              // Create empty parent set
3: for x ⊂ V do
4:      C(x) ← ∞;                    // Initial cost for all vertices
5:      Q ← Q ∪ {x};
6: end for
7: C(x_start) ← 0;                          // Cost of starting vertex
8: while Q != Ø do
9:      C_min ← getFirstMinCost(Q);
10:     u ← x ⊂ Q s.t. C(x) = C_min;
                    // u gets x, so that cost of x is the minimum
11:     Q ← (Q \ u);     // Remove current visited vertex from Q set
12:     X_neighbor ← neighbor(Q, u);
13:     for each x_n ⊂ X_neighbor do
14:             cost ← C(u) + Cost(u, x_n);
15:             if cost < C(x_n) then
16:                     C(x_n) ← cost;          // Update vertex's cost
17:                     Parent(x_n) ← u;      // Update vertex's parent
18:             end if
19:     end for
20: end while
21: return Parent, C;
```

Figure 12. *Dijkstra's algorithm pseudocode[22].*

2.4.3 RRT algorithm

Introduced by Steven Lavalle[22], it is composed by a tree structure which starts from an initial vertex V = {X_{init}} and E = ∅. Every iteration generates a random point $x_{rand} \subset X_{free}$ which is sampled. Then, the nearest vertex $x_{nearest} \subset V$ belonging to the tree to this random coordinate x_{rand} is identified. Now the Steer function generates anew point x_{new} between x_{rand} and $x_{nearest}$. As long as there is no obstacle, both the vertex x_{new} and the edge {($x_{nearest}$, x_{new})} are incorporated to the tree. After successive iterations, the tree incrementally grows in the free space until the target coordinate is found, once that happens, the loop stops.

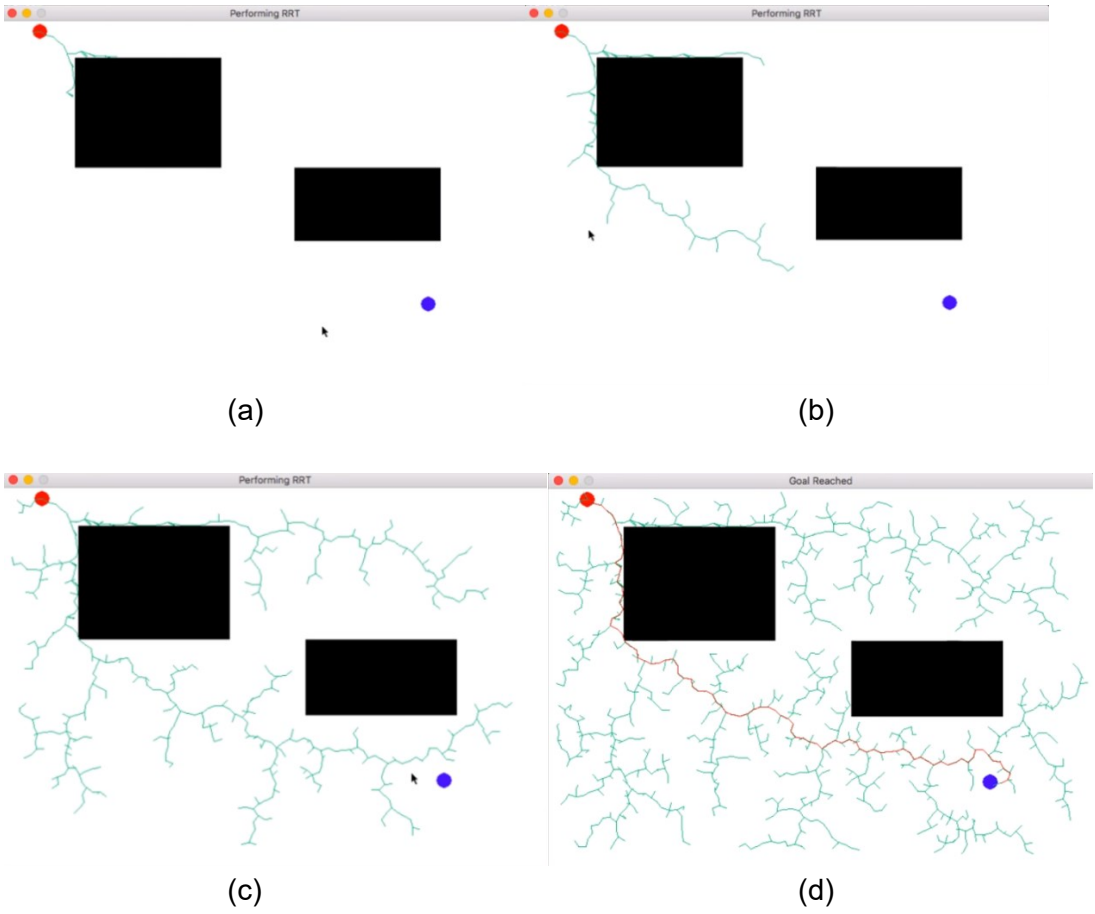


Figure 13. *Progress of the tree propagation during RRT execution.*

In (a) the tree begins to shape and the branches reaching obstacles stop. (b) and (c) show the spreading of the tree branches around the free space. Finally in (d) one of the branches reach the target point.

```
                    RRT algorithm
1     V ← {x_init}; E ← ∅;
2     for i = 1,..., n do
3           x_rand ← SampleFree_i;
4           x_nearest ← Nearest(G = (V, E), x_rand);
5           x_new ← Steer(x_nearest, x_rand);
6           if ObtacleFree(x_nearest, x_new) then
7                 V ← V ∪ {x_new};
8                 E ← E ∪ {(x_nearest, x_new)};
9     return G = (V, E)
```

Figure 14. *RRT pseudocode. [22]*

2.5 Map Representation

The core concept of this work is the map, which is the representation of the environment of a mobile robot. It is a list of obstacles present in the environment with features[23] attached:

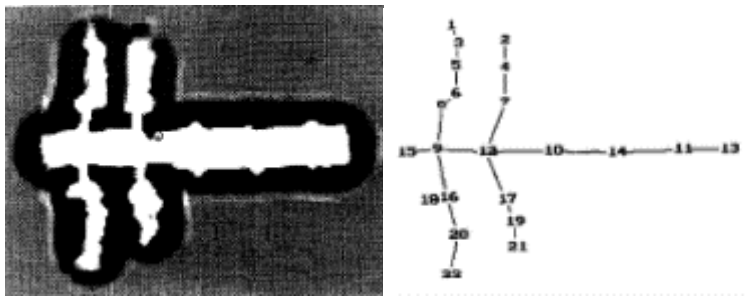
$$m = \{m_1, \dots, m_O\} \quad (5)$$

in which *O* is the total amount of obstacles.

Map data can be sorted in three classes: topological maps, volumetric maps and featured-based maps.

2.5.1 Topological maps

It is a simplified version that just contains essential information and unrelevant information is discarded. That is, only certain regions in the environment with the relationships between those areas are stored in the map. The graph of this sort of maps have nodes which represents the regions and edges that link nodes. Topological maps[24] can be extracted from occupancy grid (see concept in next section) maps.



(a) (b)

Figure 15. *Extraction of topological maps.*

In (a) it is visible the occupancy grid map and in (b) the topological extraction.

2.5.2 Volumetric maps

In this kind of maps, every item in the equation (5) belongs to a certain position in the environment so that is why it is also known as location-based map. So the result is an occupancy grid which objects, free space and unknown space is represented.

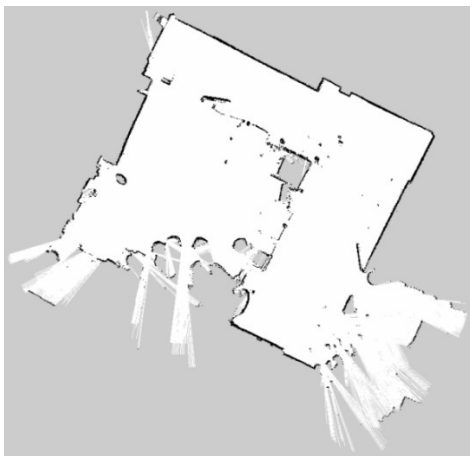


Figure 16. *An example of an occupancy grid map made in an office.*

This representation is a 2-D grid composed by cells. Those cells are the equivalent of pixels and their meaning are: whiter pixels are free, blacker pixels are occupied, and pixels in between are unknown. The occupancy grid mathematical model is:

$$p(m) = \prod_i p(m_i) \quad (6)$$

in which m_i are the cells in the grid that corresponds to a certain position in the mapped region. If the probability value $p(m_i)$ is null, then it means that the cell is free. Otherwise, if the value is $p(m_i) = 1$, then the cell is certainly occupied, while a $p(m_i) = 0.5$ means that there is a state of maximum uncertainty, so the status of such cell is unknown. The resolution of a map depends directly on the cell size.

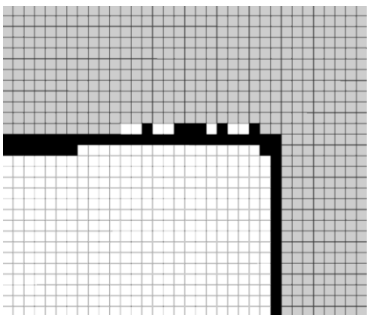


Figure 17. *A detailed view of an occupancy grid map showing the cells.*

2.5.3 Featured-based maps

This is a more elementary version than the previous type of maps. It is just intended to store certain features of the environment and not all locations. Every detected obstacle m_i in equation (5) is a feature which contains properties and localization. A property could be a visual landmark, but for this case it would be necessary to deploy vision sensors. It is recommended when memory size is a limitation.

2.6 SLAM: Rao-Blackwellized Particle Filters (RBPF)

In autonomous navigation, Simultaneous Localization and Mapping (SLAM) is concept that defines the robotic problem of building a map of an unknown environment while simultaneously keeping track of the robot's location on the map that is being built. A standard ROS package implementing this module is known as gmapping[25]. The contained SLAM algorithm makes use of Rao-Blackwellized Particle Filter (RBPF). The inputs of this package are the odometry and laser scanner data. And the outputs are the map generation as a occupancy grid and robot pose (position and orientation).

2.6.1 Rao-Blackwellization process

The point os SLAM is to estimate the robot pose within the simultaneously map generation. The problem is modelled as seeking the following joint probability:

$$p(j_{1:p}, m \mid k_{1:p}, l_{1:p}) \quad (7)$$

in which j is the robot pose, m is the map, k is the observation (e.g. laser scanner readings) and l is the control input (odometry data usually). This approach, known as Rao-Blackwellization is divided into two different subproblems:

- Estimating the map posterior $p(m \mid k_{1:p}, j_{1:p})$ for every $p - th$ particle
- Estimating the robot pose posterior $p(j_{1:p} \mid k_{1:p}, l_{1:p})$ using a particle filter

The former subproblem is also known as mapping with known poses. Then the SLAM problem can be simplified in:

$$p(j_{1:p}, m \mid k_{1:p}, l_{1:p}) = p(m \mid k_{1:p}, j_{1:p}) \cdot p(j_{1:p} \mid k_{1:p}, l_{1:p}) \quad (8)$$

2.6.2 The process of building a map with known pose

Having known the sensor observations and the robot path, the problem model is:

$$p(m \mid j_{1:p}, k_{1:p}) = \prod_i p(m_i \mid j_{1:p}, k_{1:p}) \quad (9)$$

where m is the occupancy grid map and m_i is a cell in such grid. Making use of the Bayesian filter, the probability $p(m_i)$ can be estimated[23] as:

$$l(m_i \mid j_{1:p}, k_{1:p}) = l(m_i \mid j_p, k_p) + l(m_i \mid j_{1:p}, k_{1:p-1}) - l(m_i) \quad (10)$$

where

$$l(x) = ln \frac{p(x)}{1-p(x)} \quad (11)$$

Summarizing, $l(m_i)$ is the prior which is computed from $p(m_i)$ in (11). Usually the cell is initialized as unknown, that is, $p(m_i) = 0.5$. The term $l(m_i \mid j_p, k_p)$ refers to the inverse sensor model which is updated from the sensor readings and $l(m_i \mid j_{1:p}, k_{1:p-1})$ is a recursive term of the equation (10).

2.7 Frontier-Based Autonomous Exploration

So far, this strategy is the most widely used for autonomous exploration. So that, this approach is compared to RRT-based exploration. In the proposed strategy, RRT is meant to find exploration targets while the robot task allocator assigns the detected exploration goals.

The frontier-based approach detects exploration targets by the extraction of frontier edges which, in an occupancy grid, are the lines that separate the known from unknown space. After such extraction, the center of every edge is targeted as exploration goals.

2.7.1 Frontier edge extraction

In the proposed strategy, frontier edge extractions are made by the use of computer vision tools, particularly OpenCV. The process consists of:

1. The occupancy grid map topic is converted into a image with a grey-scale format. This is necessary since OpenCV works with image files. The map topic message (nav_msgs/OccupancyGrid[26]) has this structure:

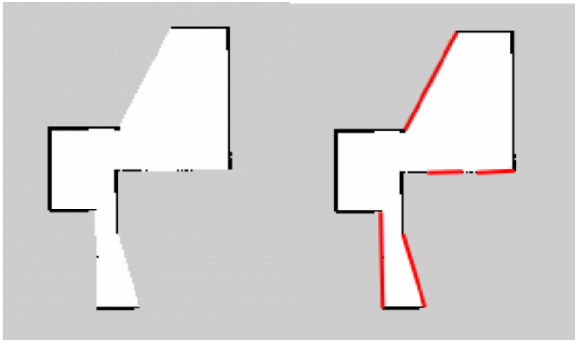
```
# This represents a 2-D grid map, in which each cell represents the probability of
# occupancy.
Header header

#MetaData for the map
MapMetaData info

# The map data, in row-major order, starting with (0,0).
# Occupancy probabilities are in the range [0,100].  Unknown is -1.
int8[] data
```

Here data is a 1-D array in which the elements are the values of every cell in the grid. So that, it is converted into a 2-D array which is the grey-scale image that will be fed to OpenCV. The conversion is made in the following way:

- Occupancy grid cell value of 0 (free space) → pixel value of 255 (white)
- Occupancy grid cell value of 100 (occupied) → pixel value of 0 (black)
- Occupancy grid cell value of -1 (unknown) → pixel value of 205 (grey)



(a) (b)

Figure 18. *Frontier edge extraction example. [20]*

(a) The occupancy grid map. (b) Detected frontier edges and extracted.

2. Then in order to keep just the pixels occupied by obstacles a threshold is applied on the image file (Figure Yb), after that, contours are marked (Figure Yc). Finally, a negative filter is applied to the image so the result is just occupied grid cells in the map marked in bold.

3. After previous step, a Canny edge detector is applied returning an image that contains all edges, occupied grid cells (walls or obstacles) and the frontier edges (Figure Ye).

4. Finally, occupied cells are subtracted from edges gathered in the prior step giving as result only frontier edges. This is achieved by the bitwise operation AND between the images gathered in the steps 2 and 3, giving as a result the filtered frontier edges image (Figure Yf).

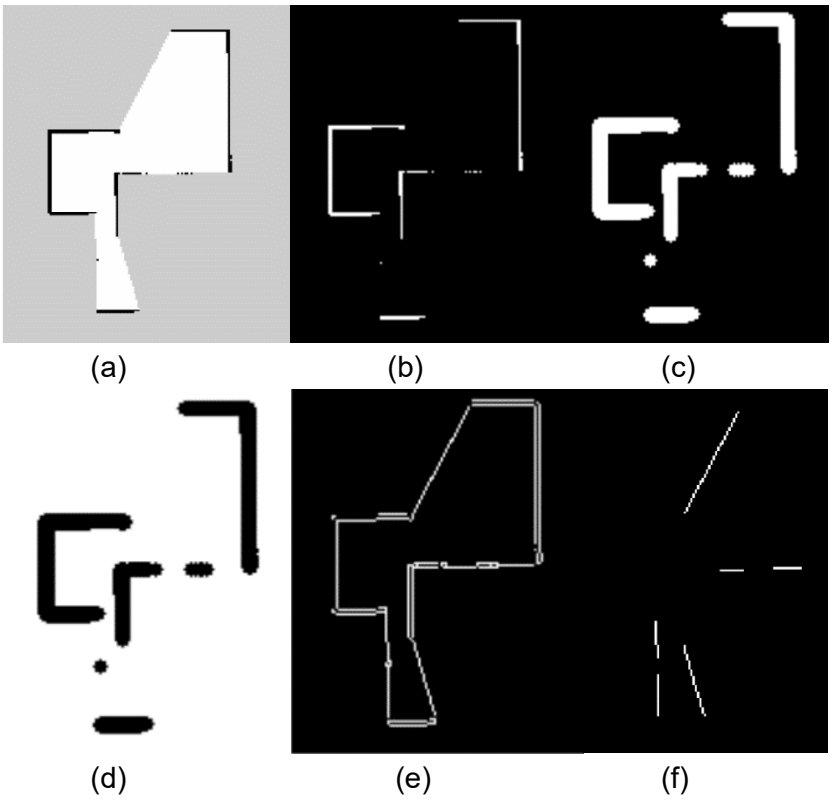


Figure 19. *Frontier edge extraction procedure. [20]*

Nevertheless, frontier-based algorithms have the limitation of relying too much on the map representation. That is, the map requires to be in an occupancy grid format which is meant for just 2-D spaces. For 3-D environments, this approach is not suitable[27].

2.8 Mean Shift Clustering

Mean shift clustering is integrated in the implementation and it is meant in order to increase computational efficiency by lowering the computational cost. The proposed exploration approach detects frontier targets in the map. Such amount of targets can be in certain cases considerably high which usually portions of them are located massively nearby. So that, those point groups should be clustered in order to remove redundancy.

The approach selects this algorithm to solve this problem as the number of clusters is not required as an input but the size of the cluster. These properties makes adequate this selection as part of the integration.

2.8.1 General concepts

The mean shift algorithm takes as input the samples taken from a PDF (Probability Density Function), $f_k(x1)$, where higher likelihoods correspond to dense areas (clusters) given a set of points $\{x1_i\}_{i=1}^{N} \in \mathbb{R}^{dim}$. The center of mass of every cluster corresponds to the local maxima of $f_k(x1)$. Iteratively, this algorithm shifts the points to their belonging local maxima.

The kernel density estimator provides $f_k(x1)$, by using a specific kernel K(x1).

$$f_k(x1) = \frac{1}{Nh^n}\sum_{i=1}^{N} K\frac{x1-x1_i}{h} \quad (12)$$

$$K(x1) = \frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2}|x1|^2} \quad (13)$$

where dim is the dimension of the data and h is known as the bandwidth, which for a Gaussian kernel h is the variance (width of the normal distribution). The PDF equation (12) states that is composed by the addition of individual kernel functions.

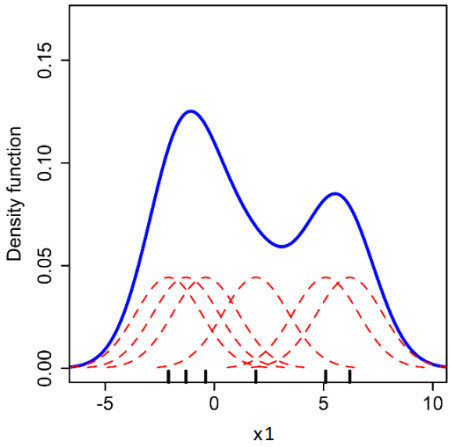


Figure 20. *PDF function composed by individual kernels.*

In this example, 6 individual kernels (red dashed curves) are shown. The Gaussian kernel density produces the blue curve. [20]

Tunning the bandwidth value will shape the PDF plot shape. So that, the smaller this parameter gets, the sharper $f_k(x1)$ curve shows (Figure 22.a). On the contrary, the larger it gets, the smoother it becomes (Figure 22.b). There is an actual optimal value for the bandwidth that will fit the best the cluster models (Figure 22.c). [21]

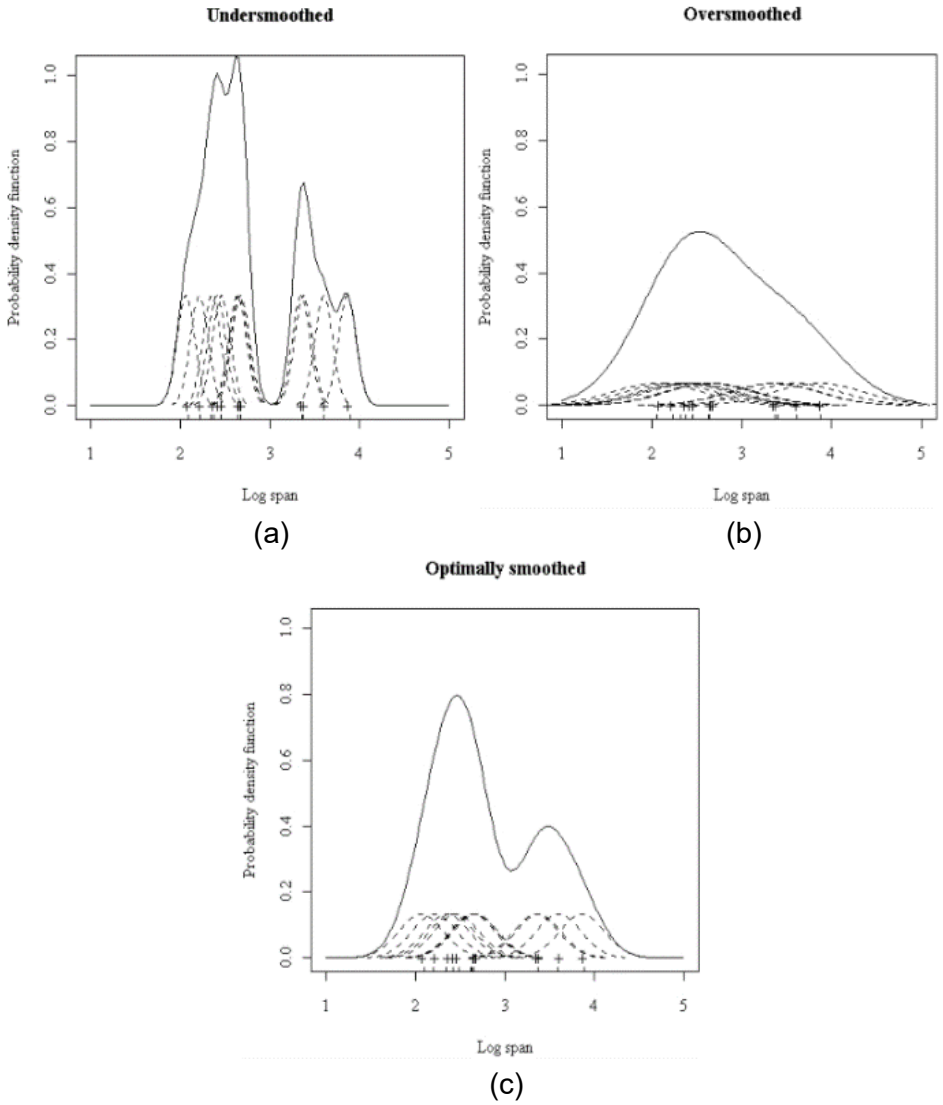


Figure 21. *Influence of the bandwidth value on the PDF curve.*

The number of local maxima of $f_k(x)$ is equal to the amount of obtained clusters after applying the kernel density estimator.

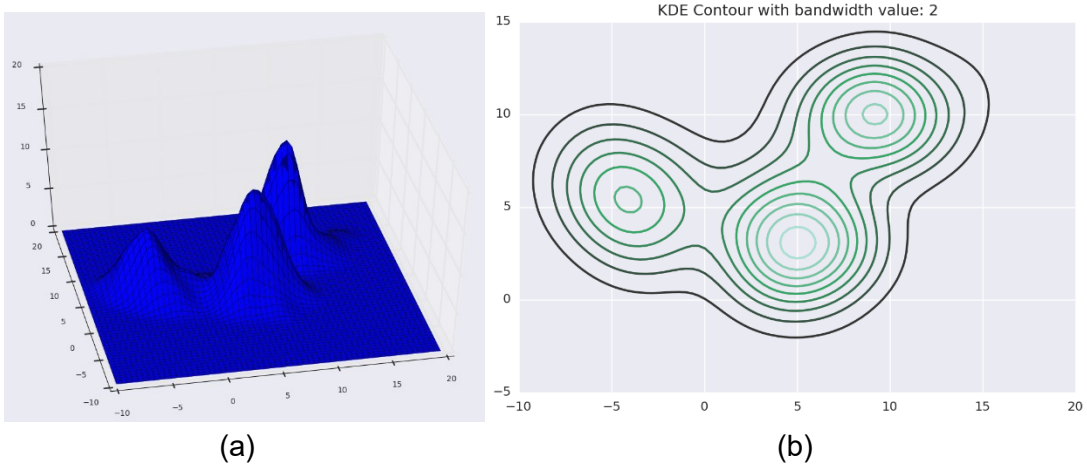


Figure 22. *Kernel density estimation.*

(a) shows the surface plot of the kernel density estimation of the data and (b) shows the contour plot of (a).

The mean shift algorithm shifts data towards the direction of the gradient $\Delta f_k(x)$. That is adding the mean shift vector to the data.

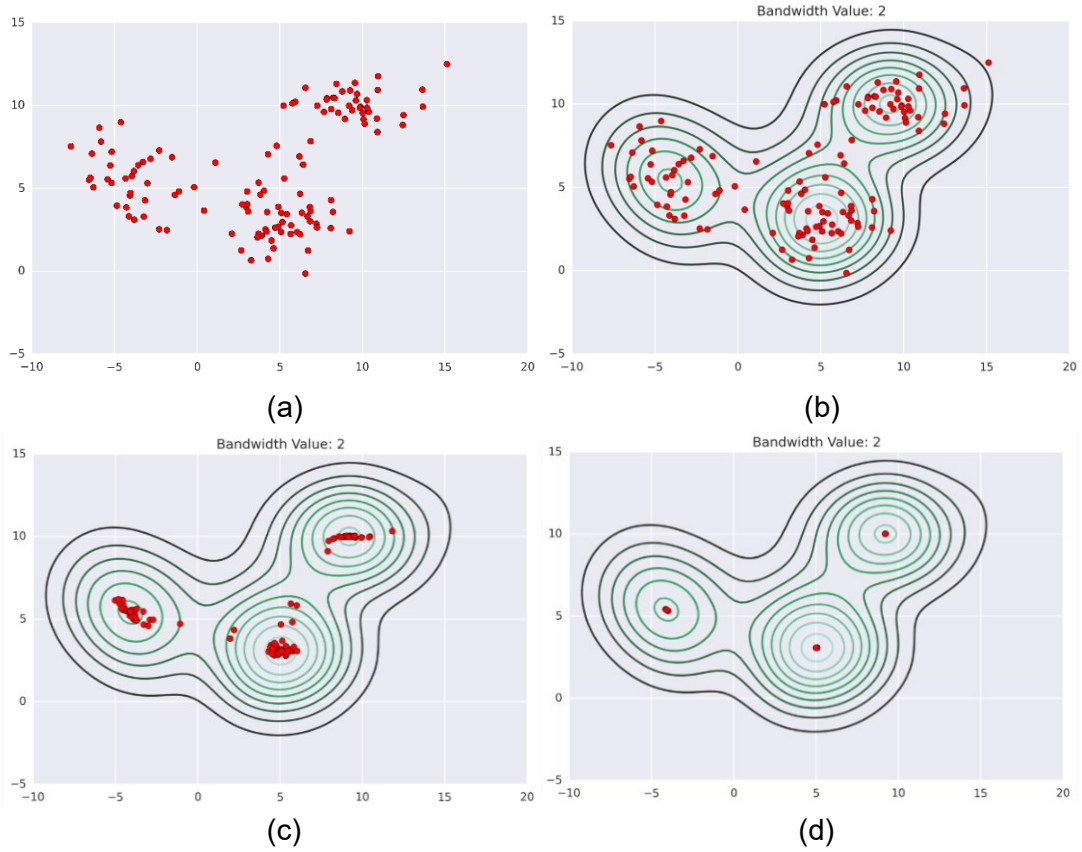


Figure 23. *Mean shift clustering using uniform kernels.*

(a) Shows clusters before being processed. In (b) a kernel is applied so the shifting of the detected clusters begins. (c) Clusters converge into their corresponding local maxima. (d) shows the processed clusters remaining only a singular value per cluster.

2.8.2 Mean shift with flat kernels

Flat kernels are defined as:

$$K(x) = \frac{1}{2}\begin{cases} 1, & if\ ||x1|| \leq h \\ 0, & if\ \left|\left|x1\right|\right| > h \end{cases} \quad (14)$$

Every point that belongs to a certain cluster data $x1 \in \{x1_i\}_{i=1}^{N}$ is shifted towards the mean $m(x)$ of such cluster data points[28]. After each iteration until convergence, such points are shifted step by step. Such cluster center is defined as:

$$m(x1) = \frac{\sum_{i=1}^{N} K(x1_i - x1)x1}{\sum_{i=1}^{N} K(x1_i - x1)} \quad (15)$$

in which $m(x1) - x1$ is the mean shift.

There is already a ready-to-use Scikit-learn library[29] implementing this algorithm, so it is used in the proposed exploration approach.

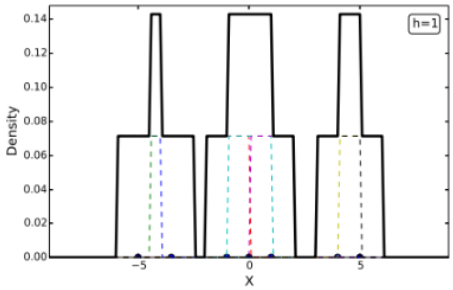


Figure 24. *Kernel density estimation curve by use of flat kernel.*

2.9 Neural networks and Deep Learning

Neural networks it is a computer science field which has been researched for decades. In the beginning, due to the inferior computational processing power, lack of databases and know-how, the order of categories was a few dozens and the input dimensions, the order of hundreds.

Nowadays, that is different, the input sizes feeding the neural networks can reach the order of 100k from different thousand of categories. Additionally, some decades ago the NN were fully connected (i.e. dense layers) networks with one up to three layers. Today, it is possible to make work a NN with a thousand of hidden layers.

2.9.1 Neural networks

A neural network is composed by neurons or perceptrons, which is a unit defined by the dot product between the inputs $x = (x_1, x_2, \dots, x_m)$ and the weights $w = (w_1, w_2, \dots, w_m)$. Then a non-linear (commonly logsig or tanh) function is applied to it. So that, a NN is a stack of logistic regression (or other nonlinearity) models.

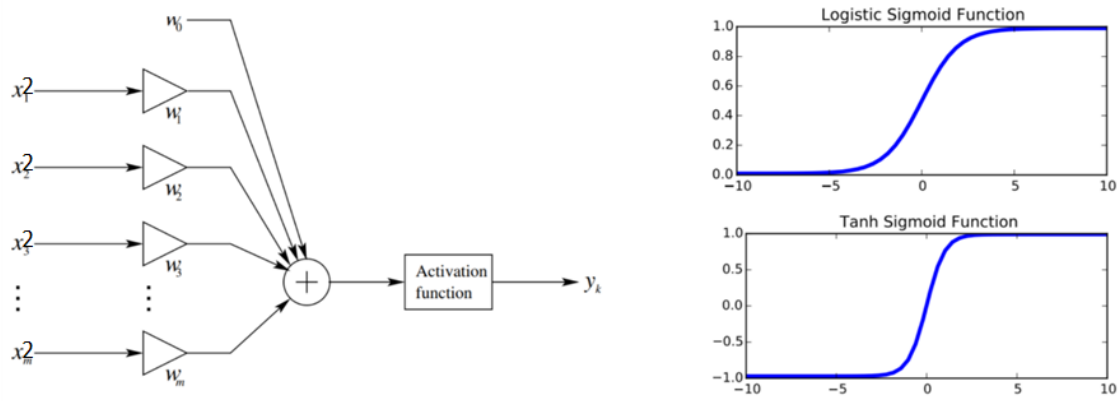


Figure 25. *Perceptron morphology and two common activation functions. [30]*

In addition, usually it is incorporated a bias as component of the weight vector by always including a feature with value set to 1. So that, for the case of the use of Logistic Regression function:

$$y_k = \sigma(z) = \sigma(w \cdot x2 + b) \quad (16)$$

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (17)$$

being $b = (b_1, b_2, \dots, b_m)$, with all components set to the same value. These variables are known as parameters of the NN.

2.9.1.1 Training a neural network

The training procedure is about adjusting the weights according to the partial derivatives:

$$w_{ab} \leftarrow w_{ab} - \varepsilon \frac{\partial E}{\partial w_{ab}} \quad (18)$$

Which means that the b-th weight of the a-th neuron steps ($\varepsilon > 0$, which is one of the hyperparameters of the NN) towards the negative gradient[31]. Note: E is the error function[32].

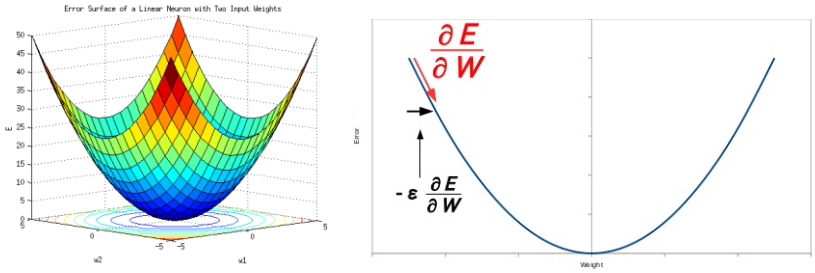


Figure 26. *Error function and updating of the weights during gradient propagation.*

So in each iteration of the training the forward and backpropagation of the gradients is computed. The first procedure feeds one or more samples to the NN and gradients are computed layer by layer using the chain rule. The second one proceeds to compute the resulting error and propagates back the gradients re-adjusting the weights one at a time. Once all samples are fed to the NN, an epoch occured (usually there are thousands of them)[30].

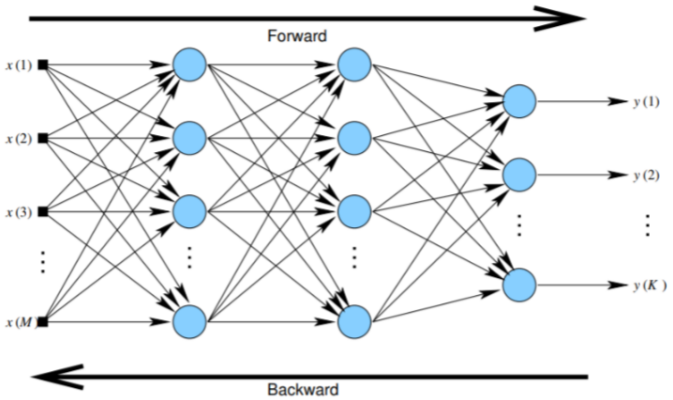


Figure 27. *Back and forward propagations in a neural network*

The database which feeds the neural network needs to be split randomly in:

- Training set: inputs for the forward propagation (usually 80% of the whole dataset).
- Validation set: hyperparameters tunned on this partition of the dataset to make optimal the training. (It is optional and when it is present, the proportion is around 10%)
- Test set: this part of the dataset evaluates the generalization capacity after the learning (20% if there is no validation set, 10% in afirmative case).

It is fundamental that the partitions of training and test sets have completely different data, otherwise, the test will not evaluate the actual learning but just the memory (i.e. it will not evaluate the generalization capabilities). Usually a NN requires of a quite large training dataset.

Figure 28. *Usual proportions of the partitions in the dataset for a neural network.*

2.9.1.2 Initial problems with deep neural networks and solutions

It was a huge problem with the NN deeper than a couple of hidden layers since:

- The deeper the NN becomes, the larger the local minima areas get. So the training stucks at one of these fake minima areas.

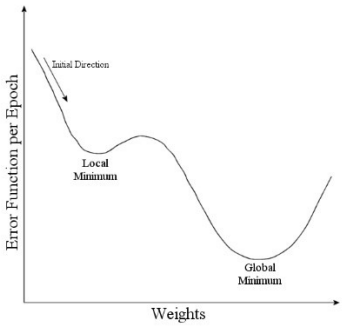


Figure 29. *Local and global minima areas.*

- The propagated gradient vanishes at the bottom layers, so that the nonlinearity tends to shrink its magnitude at each layer. At some point, the value is that small that the NN stops learning from the training.

These problems were solved and enabled a new era in the pattern recognition field, the deep learning field was born. So in few words, deep learning is the field that deals with deep neural networks. This was possible by unsupervised pre-training which consists on:

- Train layered models that learned to represent data without any classification and class labels
- Then, initialize the NN with the resulting weights coming from such unsupervised model.

for these, some of the tools used are the deep belief network (DBN), restricted Boltzmann machine (RBM), autoencoders, and so on.

However, new researchs found some ways to avoid that unsupervised pre-training procedure:

- Novel weight initialization techniques (e.g. Xavier initialization) adjusts the initial weight magnitudes layerwise[33].
- Dropout regularization to prevent overfitting by increasing randomness to the NN. This procedure consists on randomly shutting down a portion of the perceptrons during the training[34].
- Enhanced non-linearities (or activation functions) that preserves the gradient over layers (e.g. ReLU(z) = max(0,z))[35].

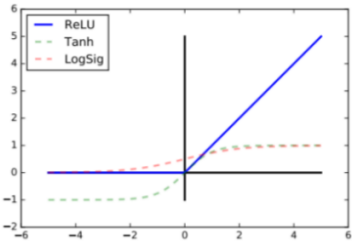


Figure 30. *Rectified Linear Unit activation function.*

2.9.2 Convolutional Neural Networks

This neural network architecture makes possible to preserve the topology of the input[36]. It is mainly intended for image processing but it is also possible to use CNNs for Natural Language Processing (NLP), recommendation systems and so on.

The structure usually is composed by:

1) Input image
2) Convolution filters the input with a number of convolutional kernels.

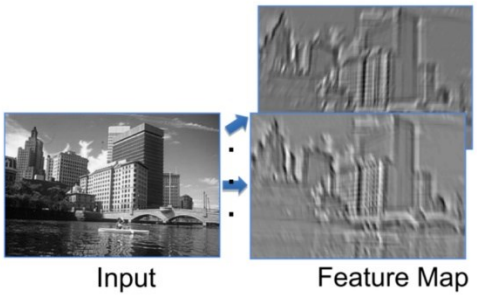


Figure 31. *Feature maps resulted from the convolution from a sample image.*

3) Non-linearity ReLu passes feature maps through a pixelwise Rectified Linear Unit.
4) Spatial pooling: subsampling shrinks the input dimensions by an integer factor.
    - Although it was a good practice using the average of each 2x2 block, nowadays it is taken the maximum value, i.e. maxpooling.
    - It reduces the data size without losing information along the pipeline and improves spatial invariance.

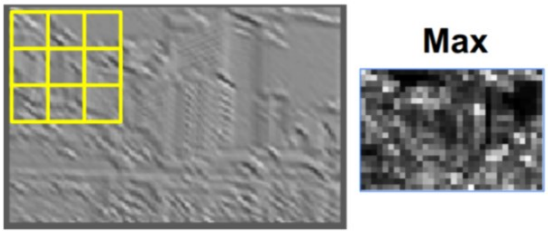


Figure 32. *Appearance of the filter after maxpooling.*

5) Normalization

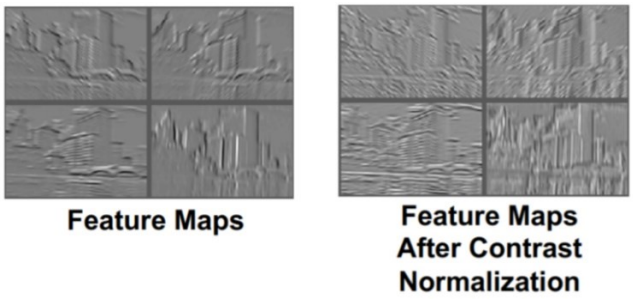


Figure 33. *Appearance of the feature maps after normalization.*

6) Feature maps

The feed for the CNNs in image recognition are usually RGB-based images, that means, that the file owns three channels: Red, Green and Blue. For this standard color format, the values of each cell is comprised between 0 and 255. If the image is in gray scale, it will have just a single channel.

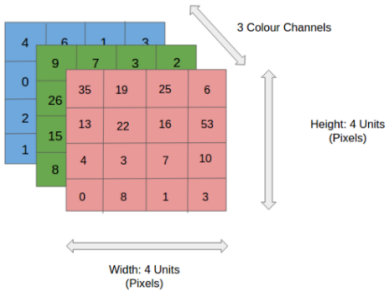


Figure 34. *RGB-based image structure.*

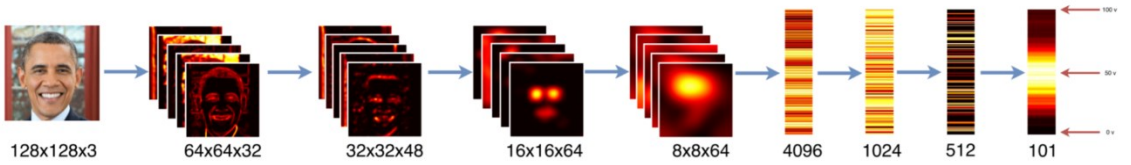


Figure 35. *Appearance of the learned filters in the CNN.*

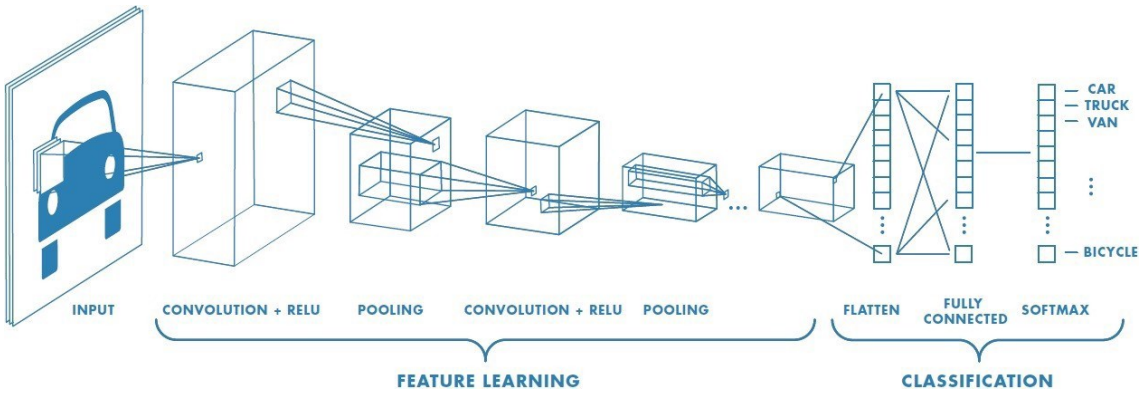


Figure 36. *CNN structure. [37]*

Once the feature maps are generated, the flatten module converts the tensor into a vector which will feed the standard neural network (a fully connected layer or more) which outputs the image classification. Basically, a deep convolutional neural network is a sequence of filters and non-linear functions. Such output can be binary if there is a single class label (e.g. a cat, not a cat) or can be multiple if there are several class labels (e.g. car, truck, van, bicycle, ...). In the last case, instead of a bottom hidden layer with a single perceptron, the softmax layer is used so that it outputs the probabilities of such classes appearing on the image (e.g. car (0.01), truck(0.04), van (0.94), bycicle (0.02), ...).

The softmax function is:

$$P(c \mid x) = \frac{\exp(w_c \cdot x2)}{\sum_{i=1}^{C} \exp(w_i \cdot x2)} \quad (19)$$

CNNs by themselves are only capable of output the class/es of the detected object/s and its/their probability but not the location/s.

2.9.2.1 Applications and state-of-the-art

In a matter of two decades the evolution of the field has been frenetic:

- The available database escalated from less than 50K images up to more than 1M
- The number of category labels in the open source databases increased from less than 100 up to more than 1000
- From small image samples of size 10x10 up to 256x256
- The achieved depths of the networks went from less than 4 layers up to more than 100

And all of this has been possible due to an altruistic community which shared know-how and labeled databases under the open source philosophy.

Some of the most famous CNN architectures which supposed a milestone in the deep learning field are[38]:

- LeNet
- AlexNet
- VGGNet
- Inception
- ResNet
- ZFNet

The most significant one was AlexNet[39] in 2012, since significally outperformed all the previous competitors in the ImageNet Large Scale Visual Recognition Competition (ILSVRC), achieving to reduce the top-5[1] error from 26% to 15.3%. The main features are it is deeper than LeNet with stacked convolutional layers and more filters per layer. It consisted of 3 convolutions, max poolings, dropout, data augmentation, ReLu activations after every convolution and fully connected (FC) layers, and SGD with momentum. It contains 60M parameters and was trained by using the open database of ImageNet.

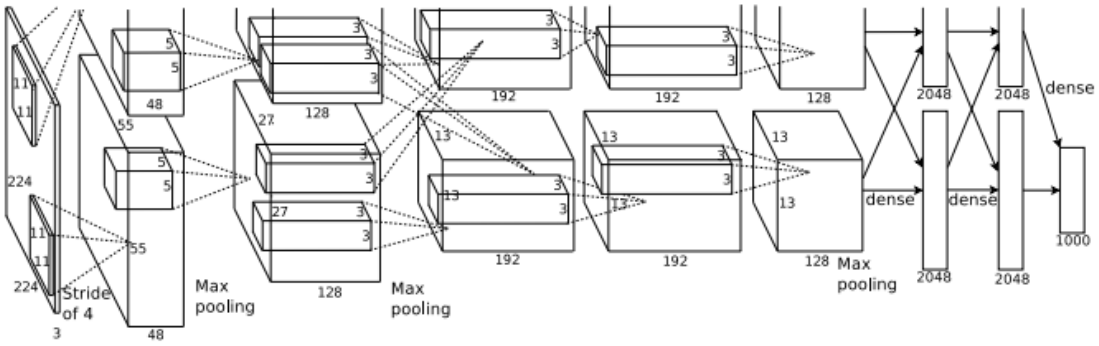


Figure 37. *AlexNet architecture. [38]*

Besides it is possible to train a network from scratch, it is possible to apply a pre-training or transfer learning which consists on using the same parameters of ready-working models such as VGG16 or ResNet by downloading the weights from a database. Then, it is either possible to use it directly for generic detections or it is possible to train with custom dataset in order to refine detections. The results of the training procedure by using one or the other method are different:

[1] Unlike top-1 error, the error is computed by taking in account the 5 detected objects with highest probability predicted by the network.

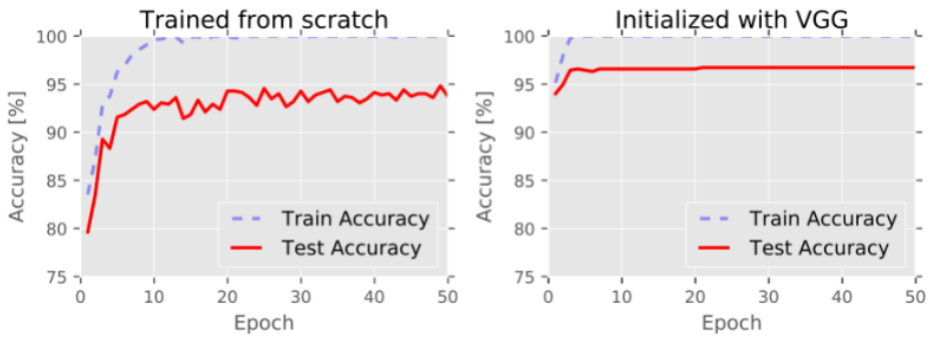


Figure 38. *Results of training from scratch or by initializing the weigths with a ready-to-use model.*

2.9.3 ROC and Precision-Recall curves

The Receiver Operating Characteristics (ROC) curve is an illustrating empirical tool to vizualise the detector performance. It describes the relationship between the probability of false alarm (P_{FA} which is the False Positive Rate (FPR) or Fall-out) and the probability of detection (P_D which is the True Positive Rate (TPR) or Recall) for all possible values of the threshold γ. It is used when there are approximately the same number of observations for each class.

		True condition			
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$
Predicted condition	Predicted condition positive	**True positive**, Power	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$
		True positive rate (TPR), Recall, Sensitivity, probability of detection = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Figure 39. *Table of contingency or confusion matrix. [40]*

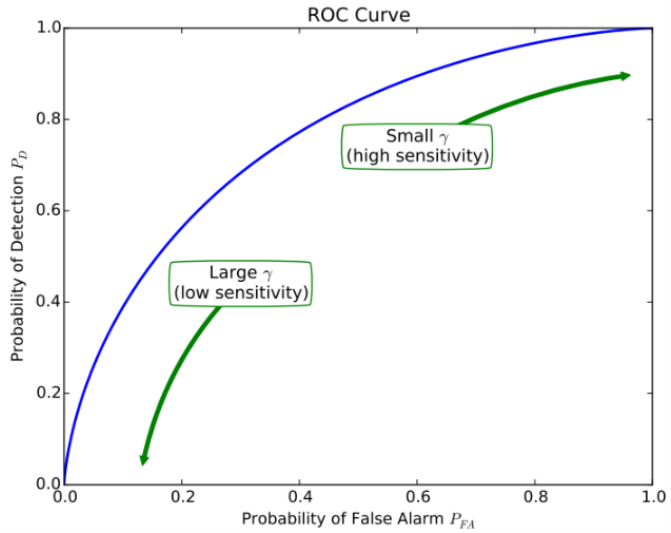


Figure 40. *A ROC curve from a certain detector. [41]*

When there is a moderate to large class imbalance, then Precision-Recall curve should be taken in consideration instead of the ROC curve. The reason is because ROC curves present an optimistic illustration of the model on datasets with category imbalance[42] which can lead to incorrect interpretations. The threshold (value of the detector sensitivity) is directly proportional to the precision and inversely proportional to the recall.

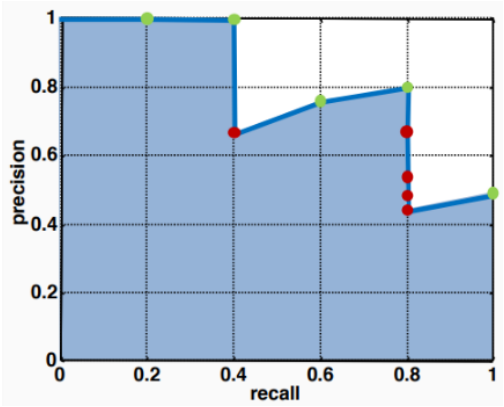


Figure 41. *Precision-Recall curve of a dataset size of 10 images and 5 relevant outputs of a detector. [43]*

The metric to measure the efficiency of the detector is the Area Under Curve (AUC) so that it is directly proportional to such efficiency.

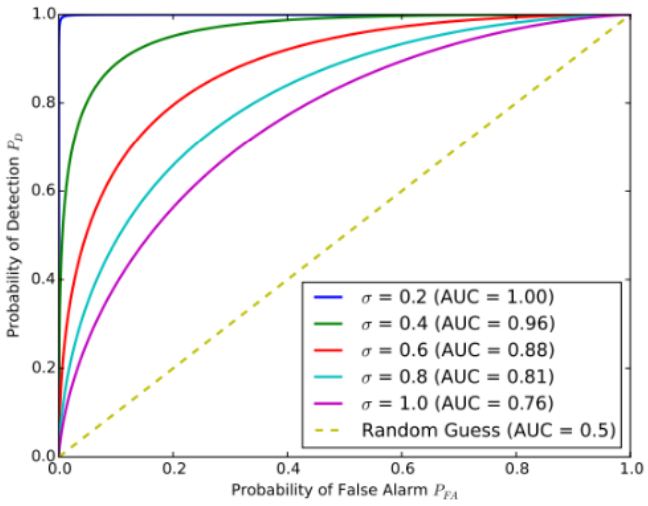


Figure 42. *The larger the AUC is, the better detector performance. [30]*

2.10 Deep learning object detection algorithms

Traditional object detection methods based on machine learning algorithms required of a pre-process of the image data so the feature extraction was handcrafted and then shallow trainable architectures such as SVMs would make the corresponding classification. The apparition of deep learning tools enable the implementation of new object detection approaches which are proven to be more powerful, capable of learning semantics, high-level and deeper features[44].

2.10.1 Two stage detectors

These detectors are the precursors of the single stage detectors and the architecture is based on:

1) Region proposal network (RPN)
2) Feature extraction rom regions for classification and regression of the proposed region

The two stage based object detector has gone through two iterations in order to achieve a good enough performance for real-time image sequence inference. The final version is capable to be used for real-time object detection.

- R-CNN → Fast R-CNN → Faster R-CNN

The problems with R-CNN[45] are that it takes a huge amount of time to be trained since it has to classify 2 K region proposals per image, so the real-time implementation is just impossible since it takes almost 50 sec to inference each image. With regard of Fast R-CNN[46], during testing time by including region proposals the algorithm is affected negatively since it is still slow. Thus, region proposals still suppose bottlenecks[47] in the performance and makes the inference time, 2 sec, to still be impractical.

Since selective search is slow and time-consuming, the removal of such module in the algorithm is achieved in Faster R-CNN[48] architecture. The image is fed to the CNN backbone based on a VGG16 pre-trained on ImageNet which provides a convolutional feature map which is used by an auxiliar network to predict the region proposals instead of using selective search algorithm directly on the feature map. Then, the predicted region proposals are reshaped by the RoI pooling layer that is afterwards used to classify the image within such proposed regions and predict the offset values for the bounding boxes. This architecture manages to inference in 0.2 sec per image, which makes it suitable for real-time purposes.

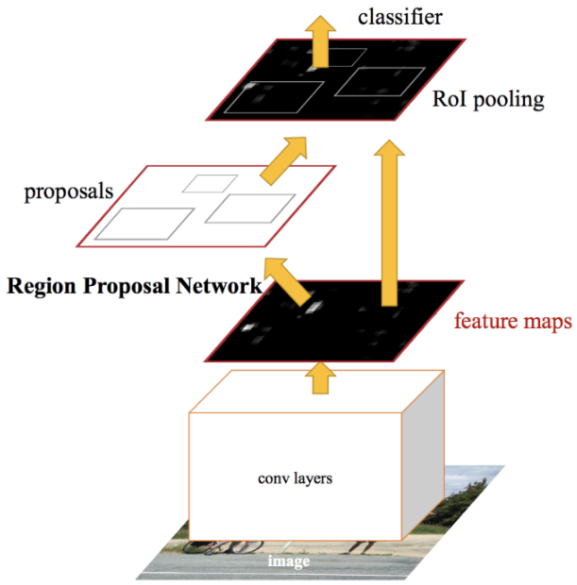


Figure 43. *Faster R-CNN architecture.*

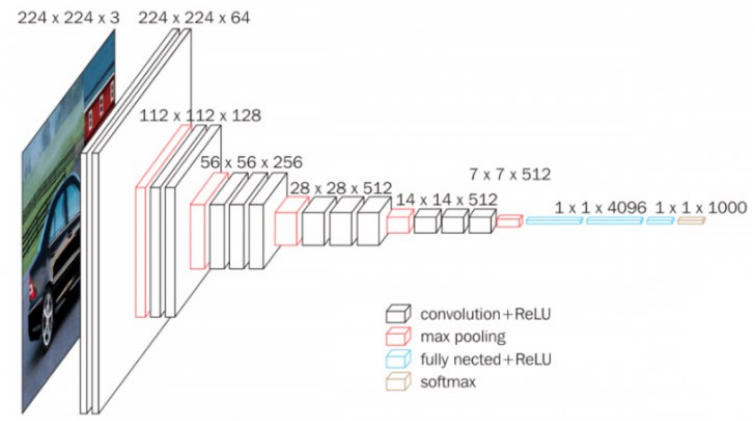


Figure 44. *VGG-16 which composes the backbone of the Faster R-CNN. [38]*

2.10.2 Single stage detectors

Unlike the two stage detectors, the single detectors do not have an explicit region proposal network (RPN) stage but they are built into the architecture, i.e. into the convolutional layers. In comparison, this kind of detectors are significantly faster in terms of inference time.

- SSD (Single Shot MultiBox Detector)
- YOLO (You Only Look Once)
- TinyFaces
- CornerNet
- RetinaNet
- RefineNet

Unlike the two stage based detectors, the network does not look at the whole image at once but instead the parts of the image with higher probabilities of containing the objects. A single convolutional network predicts the bounding boxes and their category probabilities.

A popular single stage object detector is SSD[49]:

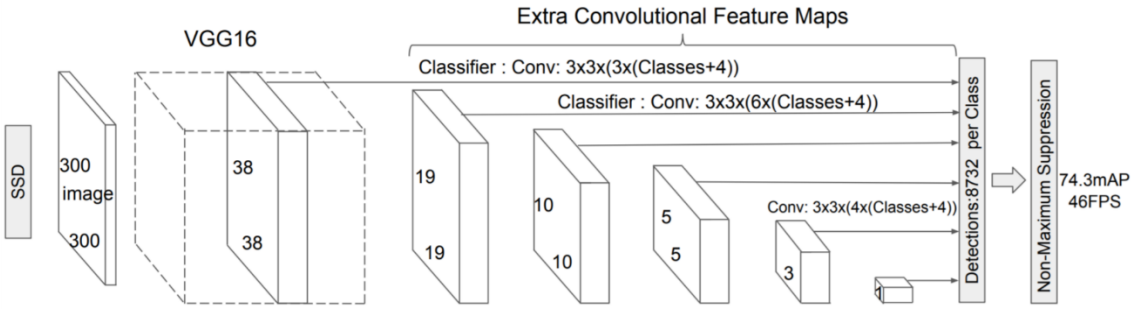


Figure 45. *SSD architecture.*

It just uses a single deep neural network and discretizes the output space of bounding boxes into a group of default bounding boxes over different aspect ratios and scales per feature map. During the inferencing, this detector generates scores of the different detected objects of each class per default bounding box and then such boxes are resized accordingly to match every object with the correct aspect ratio.

Besides, it combines predictions from several feature maps with different resolutions in order to manage objects of different sizes and scales in the same image. In conclusion, it encapsulates all computation in a single network and removes completely the proposal generation and subsequent feature resampling stage.

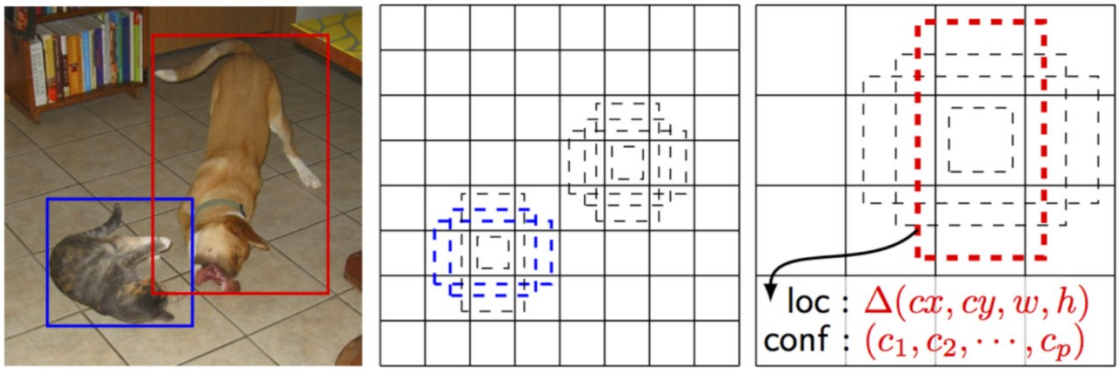


Figure 46. *SSD generating scores and adjusting the correct aspect ratios of the bounding boxes.*

2.10.3 Benchmarks

The quality of detections is based on two factors: accuracy and speed inference. There is a trade-off between them so that if the preferred accuracy is too high then the speed inference will be slower and viceversa. Besides the type of detector, there are other factors that affect the performance[50]:

- Training configuration (batch size included), learning rate, input image resize and other hyperparameters
- The deep learning platform used for implementation
- Output strides for the extraction
- Matching approach and IoU threshold (how predictions are excluded in calculating loss)
- Which feature map layers used for detection
- Localization loss function
- Non-max suppression IoU threshold
- Training dataset
- Data augmentation
- Boundary box encoding
- Number of proposals and predictions
- Hard example mining ratio (positive vs. Negative anchor ratio)
- Use of multi-scale images in training or testing (with cropping)

The two most used benchmarks are:

2.10.3.1 PASCAL VOC dataset

The dataset contains objects from 20 different categories: airplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining, table, dog, horse, motorbike, person, potted plant, sheep, train, TV.

It is obtained from real world images downloaded from Flickr[51]. The point is to contemplate complex scenes, different lighting conditions, occlusions, clutter, multiple scales... so it makes the network learning robust. There is approximately similar distribution between training and test sets and a minimum of 600 training objects per category.

Competitions have been arranged in order to impulse progress in the field. This fact had a huge impact in the development of the state-of-the-art algorithms for object detection.

Detection challenge comp4: train on own data

	mean	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv/ monitor	submission date
Faster RCNN, ResNet (VOC+COCO) [?]	83.8	92.1	88.4	84.8	75.9	71.4	86.3	87.8	94.2	66.8	89.4	69.2	93.9	91.9	90.9	89.6	67.9	88.2	76.8	90.3	80.0	10-Dec-2015
ION [?]	76.4	87.5	84.7	76.8	63.8	58.3	82.6	79.0	90.9	57.8	82.0	64.7	88.9	86.5	84.7	82.3	51.4	78.2	69.2	85.2	73.5	23-Nov-2015
MNC baseline [?]	75.9	86.4	81.1	76.4	64.3	57.8	81.1	80.3	92.0	55.2	82.6	61.0	89.9	86.4	84.6	85.4	53.1	79.8	66.1	84.7	69.9	15-Dec-2015
Faster RCNN baseline (VOC+COCO) [?]	75.9	87.4	83.6	76.8	62.9	59.6	81.9	82.0	91.3	54.9	82.6	59.0	89.0	85.5	84.7	84.1	52.2	78.9	65.5	85.4	70.2	24-Nov-2015
LocNet [?]	74.8	86.3	83.0	76.1	60.8	54.6	79.9	79.0	90.6	54.3	81.6	62.0	89.0	85.7	85.5	82.8	49.7	76.6	67.5	83.2	67.4	06-Nov-2015
** HRCNN ** [?]	74.6	85.9	83.9	75.5	60.9	54.5	81.4	79.1	90.6	53.3	79.7	61.6	89.9	86.2	85.8	78.2	49.1	75.1	68.6	86.1	67.7	13-Nov-2015
MR_CNN_S_CNN_MORE_DATA [?]	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0	06-Jun-2015
HyperNet_VGG [?]	71.4	84.2	78.5	73.6	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7	12-Oct-2015
HyperNet_SP [?]	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6	28-Oct-2015

Figure 47. *PASCAL VOC Leaderboard of December 2015. The winner was the Faster R-CNN architecture with a precision of 83.8%.*

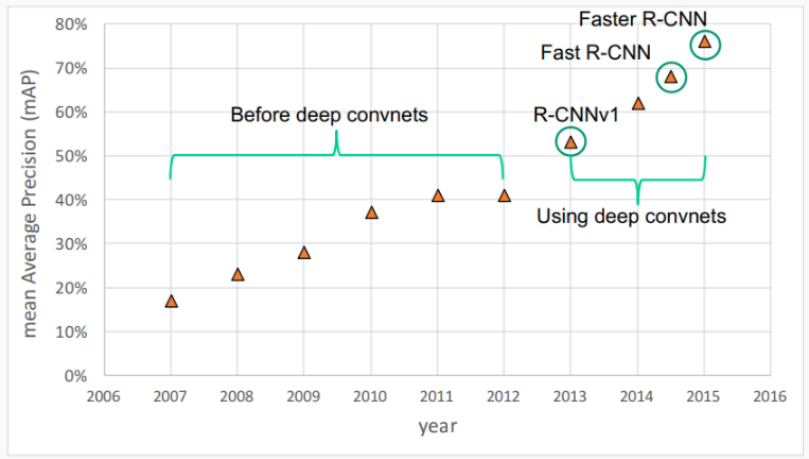


Figure 48. *Progress in the object detection field.*

2.10.3.2 MS COCO

Microsoft COCO (Common Objects in Context)[52] is a large-scale object detection, segmentation, and captioning dataset. This dataset is larger than PASCAL VOC, it contains 80 object categories and more than 200K labeled images. In addition, COCO dataset also provides additional labeled dataset intended for object segmentation[53].

With the COCO dataset, it has been evaluated the performance of the different modern convolutional detectors. As mentioned beforehand, it is observed a speed / accuracy trade-off[54] for this algorithms.

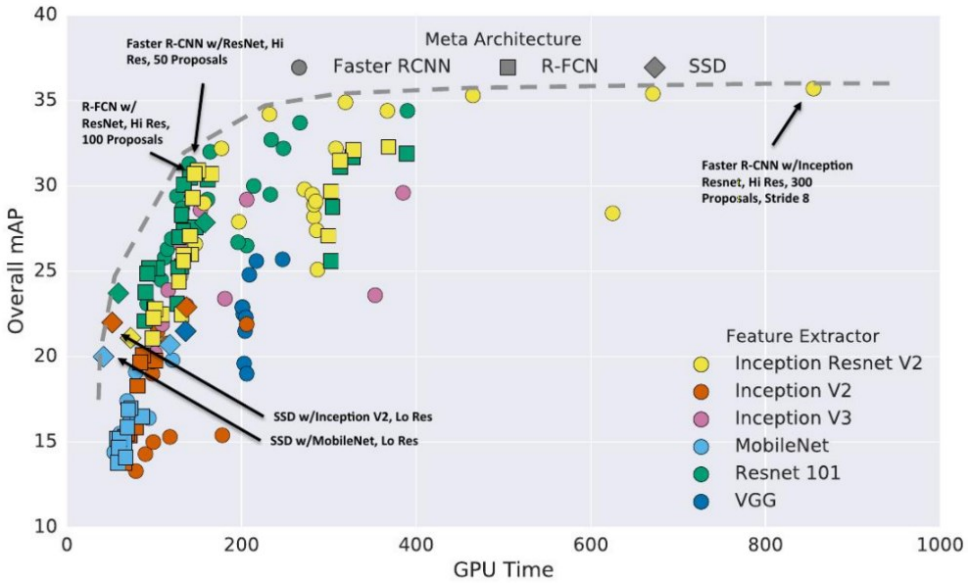


Figure 49. *Inverse relationship between precision vs. Inference time.*

2.10.4 Bounding boxes

Unlike image classification models which only provides as output the category label of the detected object in the image, object detection models provides in addition the detection localizations. This can be done by encapsulating the detections in bounding boxes defined as:

$$Bounding\ Box_i = \{C_i, P_i, xmin_i, ymin_i, xmax_i, ymax_i\} \quad (20)$$

In which $xmin_i, ymin_i$ define the coordinates of the upper left corner and $xmax_i, ymax_i$ define the coordinates of the bottom right corner.

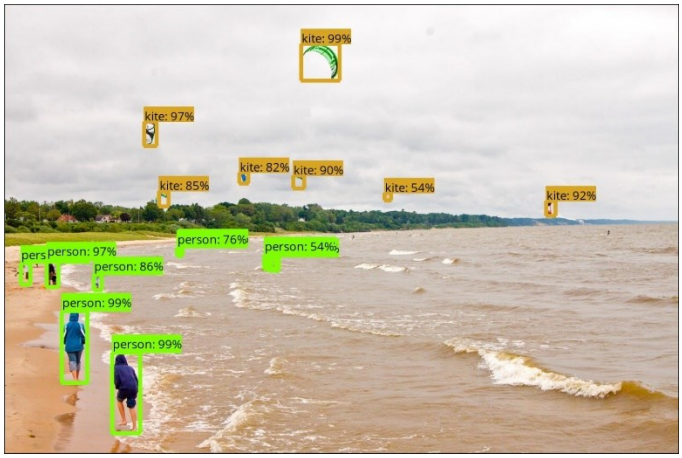


Figure 50. *Example of bounding boxes.*

Or alternatively as

$$Bounding\ Box_i = \{C_i, P_i, bx_i, by_i, bw_i, bh_i\} \quad (21)$$

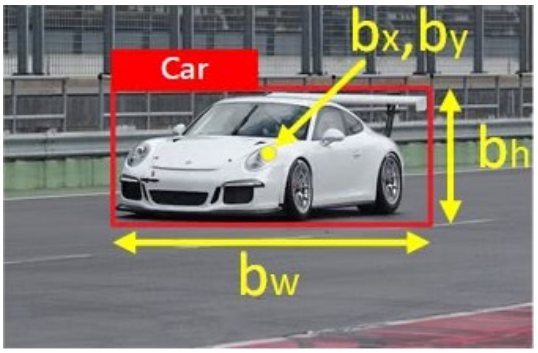


Figure 51. *Bounding box in cental coordinate, width and height format.*

In which bx_i, by_i are the coordinates of the geometric center of the bounding box and bw_i, bh_i are the width and height respectively.

Commonly to both representation styles, C_i, P_i are respectively the class label of the i-th detection and P its probability.

2.11 ROS ecosystem

ROS (Robot Operating System) is the most popular open-source platform for robots software development. It provides tools and libraries that allows splitting code into modular and reusable packages. The purpose of this framework is to provide high reusability of code so that developers can create hardware-independent content. The libraries and tools aforementioned include:

- Device Drivers: Provides an already built-in compatibility feature among many different robot platforms and sensors, so the user does not need to waste time coding software to enable the utilities. In addition, provides an standardization of the format messages transmitted by the sensors.

- Hardware abstraction: One of the cornerstones of ROS, it allows independence between software and hardware. This is achieved as long as each actuator or sensor is operating with ROS drivers. Those are the elements that exchange data at a low level with the hardware, coded in ROS standard messages and communicated with the software.

- Libraries and Community Support: As an open-source platform, it counts with many people worldwide which contribute with their content, commits fixes and enhancements of packages, help other users with their issues. Some of the contributions are such powerful that they got standardized as official ROS packages, such as SLAM (Simultaneous Localization and Mapping) features. Since it is not limited by any organization, it ensures a long term operativity.

- The distribution of computation and message passing: It allows message passing between processes which can be on different robot platforms at the same time to the same local network.

- ROS file system: The content is arranged in modules called packages which contain pieces of code known as nodes. They can also contain configuration parameters, message formats, and more complex data structures such as services and action servers. Every package, after being compiled (catkin_make), a file known as package manifest which has general information about the package is created. Packages can also form conglomerates, also known as stacks or metapackages (i.e. navigation stack which contains multiple packages global_planner, map_server, move_base, ... that are interdependent).

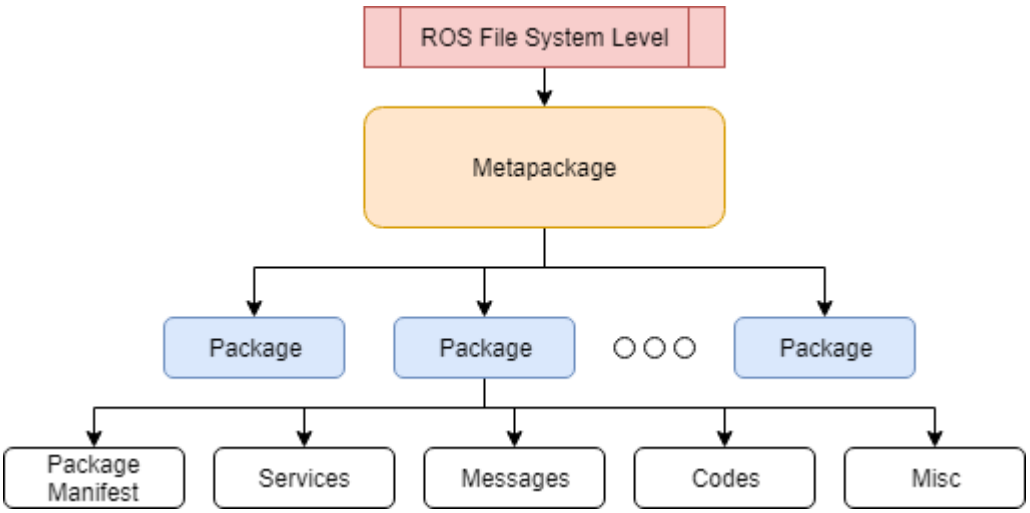


Figure 52. *ROS File system*

ROS ecosystem consists in 3 different levels

2.11.1 ROS basics

ROS ecosystem consists in 3 different levels[55]: the file system level, the computational graph level and the community level. The computational graph level is the one that processes and shares data via peer-to-peer network. The main concepts at this level are:

- Nodes: they are processes that performs computation, that is an executable script (.py, .cpp, ...). It is usual that a package runs several nodes simultaneously.

- Subscriber: a node which receives data.

- Publisher: a node which sends data.

- Topics: content of the messages that are either published or subscribed. This feature provides independency from the need of having third party nodes running.

- Messages: represent the data structure, format and type of the topics (integer, boolean, floating point, ...). They are stored in text files which are used during the compilation of a package so that ROS translates their content into the source code on the programming language implemented. For instance, a ROS message used in this work is **sensor_msgs/LaserScan.msg** which is used to input the data gathered by the laser scanner to the system. The **LaserScan** message is actually a text file with the ".msg" extension which is located within **sensor_msgs** native ROS folder. Here there is the structure of this particular message:

```
# Single scan from a planar laser range-finder
# If you have another ranging device with different behavior (e.g. a sonar
# array), please find or create a different message, since applications
# will make fairly laser-specific assumptions about this data

Header header           # timestamp in the header is the acquisition time of
                        # the first ray in the scan.
                        # in frame frame_id, angles are measured around
                        # the positive Z axis (counterclockwise, if Z is up)
                        # with zero angle being forward along the x axis

float32 angle_min       # start angle of the scan [rad]
float32 angle_max       # end angle of the scan [rad]
float32 angle_increment # angular distance between measurements [rad]

float32 time_increment  # time between measurements [seconds] - if your scanner
                        # is moving, this will be used in interpolating position
                        # of 3d points

float32 scan_time       # time between scans [seconds]

float32 range_min       # minimum range value [m]
float32 range_max       # maximum range value [m]

float32[] ranges        # range data [m] (Note: values < range_min or > range_max
should be discarded)
float32[] intensities   # intensity data [device-specific units].  If your
                        # device does not provide intensities, please leave
                        # the array empty.
```

Notice that *float32* is a ROS native type that in C++ is compiled as *double* type.

- Parameter Server: it is where parameters such as configuration parameters are stored by the master. All nodes can become clients of this server so they can retrieve any needed parameter at anytime.

- Master: it is the nexus and coordinator module as responds for tracking of topics, node registration, services and action servers.

Finally, the community level is about all related to repositories, ROS distributions and forums and so on.

2.11.2 ROS coordinate systems

By convention[56] x-axis points forward, y-axis points to the left and z-axis points upwards of the mobile robot. As long as there are more than one frame, then it is required transformations among them, that is, the relationship among coordinate systems in terms of position and orientation. In ROS there is a package which does the relevant computations, this is *tf* package[57]. It allows to do the mentioned computations and store them in a tree format over the time, that means that their pose history is stored. Usually the transformations are published by a node in charge of that. Regarding mobile robots, the standard[58] frames are:

- base_link: rigidly attached to the mobile robot base. It is possible that can be attached to any arbitrary pose of such base. This frame is the head in the *tf* tree which contains all sensors and mobile parts of the robot.

- odom: is a global fixed frame which represents the pose obtained by the odometry data (visual or mechanic) or IMU (inertial measurement unit). The main feature is that is continuous which means that the pose of the mobile robot always evolves in a smooth way without discrete jumps. It is an accurate short-term reference but it drifts over time so it makes it not suitable for long-term reference.

- map: this is the global fixed frame. It is not continuous which means that discrete jumps over time can happen but drifts over time is limited which makes it suitable for long-term use. This frame is estimated using localization algorithms by using laser scanners or other sensors that allows the mobile robot understand where is it. Notice that is not recommended for local actuation and sensing due to the discrete nature.

- base_footprint: is the projection of base_link to the ground where roll and pitch orientations are null but yaw is the same as base_link value.

- laser_link: is the frame corresponding to the pose of the laser scanner and it has no relative motion with the mobile platform.

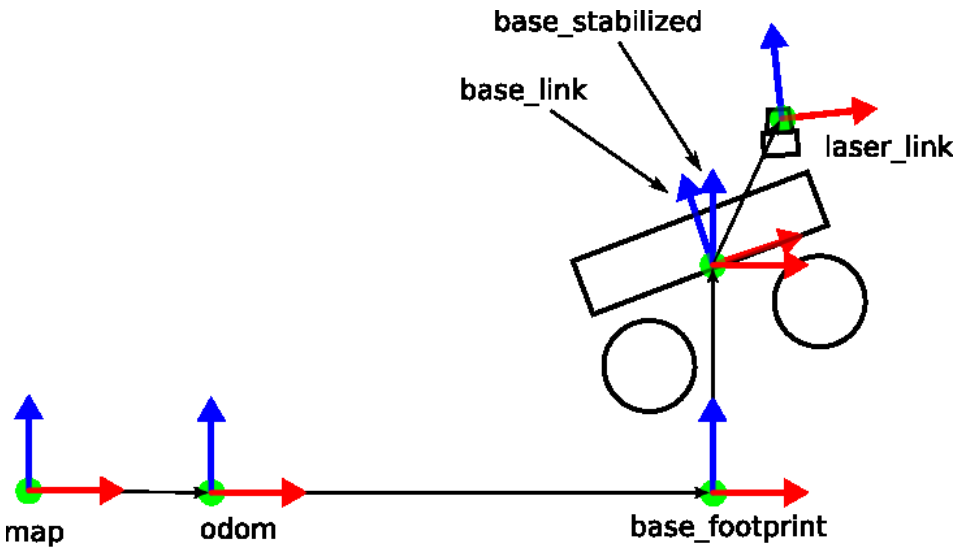


Figure 53. *Usual frames in mobile robots. [59]*

2.11.3 ROS launchers

In the UNIX command line the launch files can be executed by the *roslaunch* command. This type of files have the purpose of remapping[2] variables, setting parameters and running nodes. They allow to execute multiple nodes simultaneously which can be the same (with other parameters) or different. This feature is needed for complex projects containing multiple instances. A launch file can also call another launch files.

2.11.4 ROS visualizer

Also known as RVIZ, it is a tool that allows you to visualize Images, PointClouds, Lasers, Kinematic Transformations, RobotModels... It is fundamental in order to get interpretable information about what the robot is perceiving, i.e., the message topics in a graphic representation.

[2] Change of a ROS variable name which allows high interchangeability of topics and allows running simultaneous instances of the same node with different configurations.

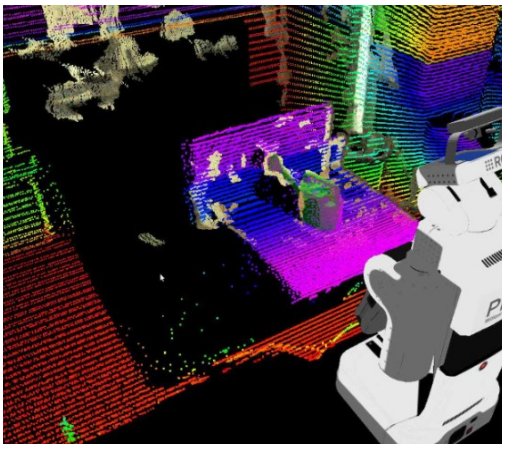


Figure 54. *Point cloud data visible in 3-D space in Rviz.*

2.12 Conclusions

In conclusion, after assessing the usability of the concepts for the work in both of the Methodology and Literature and technology review sections, the work is based on certain theoretical foundations (marked on green color) and other concepts are not required for the implementation but for analysis explanation for the solutions. This can be summarized in the following table:

Robot perception	LiDAR	Stereo Vision	Point clouds	RGB images
Exploration approach	Randomized	Frontier-based		
Exploration strategies	FFD	WFD	RRT	SRT
Map representation	Topological	Volumetric	Featured-based	
Mean Shift Clustering	Gaussian kernel	Flat kernel		
Neural network training approach	From scratch	Pre-trained		
Object detection foundations	Machine Learning based	Deep learning based		
Object detector types	Two stage	Single stage		
Benchmarks	Pascal VOC	MS COCO		

Figure 55. *Table showing what concepts are implemented in the work.*

3. PROPOSED SOLUTION

3.1 Sensors

In the field of mobile robots, there is a wide range of sensors that can be utilized in order to be source for the input data of the navigation module. So the automated navigation can be based on some of the following examples:

- Wires
- Guide tape
- Laser scanners
- Thermal cameras
- GPS
- Sonar
- Vision guidance

For this work, the navigation is performed by the input of the laser scanner data which is quite a robust source[60], that much that is implemented in autonomous vehicles. It is invariant to the illumination factor or outdoor conditions unlike vision guidance or thermal cameras. In addition, a industrial mobile robot usually performs indoors, so the use of GPS is not feasible. Neither it is the use of sonar, since it requires clean ambient sound and industry environments are usually noisy. On the other hand, wires and guide tapes need to be embedded in the environment previously to the robot navigation and they work in sort of the same way as trains in rails. For instance, robots based on guide tapes follow the lines with a steering control and a guidance system.

Besides, the MiR100 has by default embedded basic sensors for navigation such as an odometer and an Inertial Measurement Unit (IMU). The first device measures the variation of the pose along the time (orientation and position). The second one, measures among other parameters, linear and angular accelerations. In addition, it counts with ultrasonic sensors as redundant security measures. For instance, laser scanners might have problems to detect translucid or transparent objects while the ultrasonic sensors do not have such drawback.

For the object detector module, since convolutional neural networks take as input a 2D image, the already embedded depth camera can provide a channel with RGB streamed live video. For this, the object detector takes a constant sequence of image that has to process in real-time.

3.2 Navigation for autonomous exploration module

This module of the work is based on the single robot case of "Multi-Robot Map Exploration Based on Multiple Rapidly-Exploring Randomized Trees" work[20], [61].

The algorithm is divided in two modules: the RRT-based frontier detector module and the filter module. The former is in charge of the frontier targets detection and returning those to the filter module. The later uses the mean shift clustering algorithm in order to cluster the frontier points, filter the incompatible and outdated ones, and store the valid frontier targets[62]. In addition, it is required also the path planning and SLAM modules which are implicit in this exploration strategy[21], [63], [64].

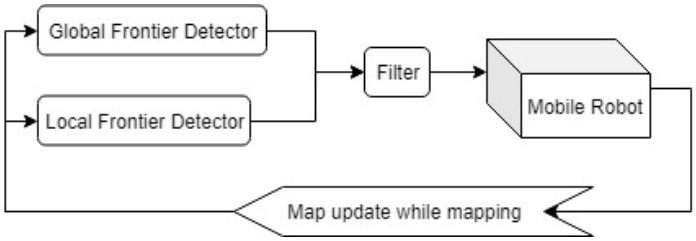


Figure 56. *General diagram of the exploration system.*

Such configuration structure provides the capability of having several instances of frontier detectors running in parallel for quicker performance. In addition, it provides the possibility of executing different types of frontier detectors simultaneously, feature which is required when comparing different sorts of frontier detectors.

3.2.1 Good to know

Before continue, you should know what these terms mean:

PublishPoint: This is the function in charge of sending detected frontier points to the filter module.

Invalid frontier point: The mobile robot cannot reach it in the real world which means that there is no valid path possible between such point and the mobile robot position.

Old frontier point: Detected in earlier iterations and no longer belonging to the unknown area of the map.

GridCheck: A function whose input are the map and two points. It returns 1 if the points are in the known space. It returns 0 if there is any obstacle between the points. And finally, returns -1 if there is unknown area between such points.

3.2.2 Motivation of RRT

Since RRT is highly biased[3] to unknown areas of the map, the generated trees tend to grow towards such directions which is an desired feature for the proposal. So that it makes this approach more interesting than just the use of frontier-based implementations. This means that it is more efficient in terms of time, thus in terms of energy and derived costs.

To explain the biased property of the RRT-based algorithm, the voronoi diagram is a proper math tool to clarify this fact. Such diagram consists of the division of the space into discrete areas created by circles growing from every vertex filling all the possible available space until neighbor circles boundaries collide, then a linear frontier is created separating non-intersected regions. An area is a set of points that are the most close to the corresponding vertex. So that, such diagram is useful in practice to know in a visual way which vertex is closer to any certain point in the map. The Voronoi area associated with a certain vertex is larger the closer the vertex is to the frontier with the unknown space. The tree tends to spread more in the bigger regions as vertices selection is made bylooking for the closest neighbor.

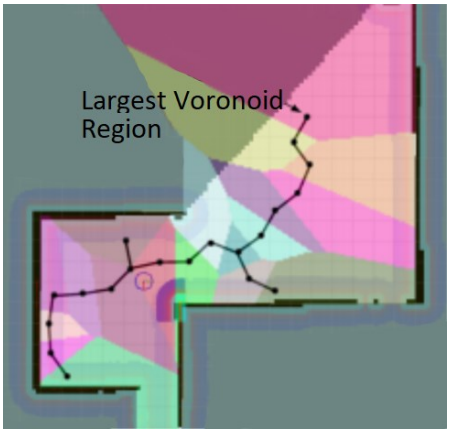


Figure 57. *Voronoi diagram resulted from RRT exploration.*

Another useful feature of RRT approach is that can be extrapolated to the 3-D exploration[65], [66] unlike frontier-based techniques. In addition, it is probabilistically robust[67], so that, it is assured that the environment sooner or later will be completely mapped.

3.2.3 Frontier detector

This module is actually a combination of two parallel sub-modules: the local frontier detector and the global frontier detector. Any point reached by the growing generated tree is a frontier point as long as it belongs to the unknown area of the environment. The execution of extra instances of the frontier detectors enhances the frontier point detection but with a higher computational consumption as trade-off.

3.2.3.1 Local frontier detector

The tree generated in this sub-module begins from an initial vertex $V = \{x_{init}\}$ and $E = \emptyset$. Every each iteration a random point is stored $x_{rand} \subset X_{free}$. Such initial vertex is the nearest to x_{rand} xnearest $\subset V$.

```
                    Local Frontier Detector
1     V ← x_init; E ← ∅;
2     while True do
3             x_rand ← SampleFree;
4             x_nearest ← Nearest(G = (V, E), x_rand);
5             x_new ← Steer(x_nearest, x_rand);
6             if GridCheck(x_nearest, x_new)= -1 then
              // Unknown area
7                     PublishPoint(x_new);
8                     V ← x_current; E ← ∅;
                      // Tree reset
9             else if GridCheck(x_nearest, x_new) = 1 then
              // Free known area
10                    V ← V U {x_new}; E ← E U {(x_nearest, x_new)};
11            end if
12    end while
```

Figure 58. *Local frontier detector pseudocode. [20]*

3.2.3.2 Global frontier detector

This detector sub-module is the same as the previous one with the difference that the tree does not get reset and keeps on growing indefinitely during the exploration. This makes it similar to the RRT algorithm. It is intended for the detection of frontier points through the entire map considering areas far away from the mobile robot.

```
                    Global Frontier Detector
1     V ← x_init; E ← Ø;
2     while True do
3             x_rand ← SampleFree;
4             x_nearest ← Nearest(G = (V, E), x_rand);
5             x_new ← Steer(x_nearest, x_rand);
6             if GridCheck(x_nearest, x_new)= -1 then
              // Unknown area
7                     PublishPoint(x_new);
8             else if GridCheck(x_nearest, x_new) = 1 then
              // Free known area
9                     V ← V ∪ {x_new}; E ← E ∪ {(x_nearest, x_new)};
10            end if
11    end while
```

Figure 59. *Global frontier detector pseudocode. [20]*

3.2.3.3 Their combination purpose

The local tree resets when a frontier point is detected and starts growing again from the robot current position. Thus, in order to avoid the mobile robot from not exploring small corners in the environment and to assure that the far away frontier targets are also explored, the global frontier detector is required. The combination synergy provides a faster detection of target points since the tree begins to grow from the prior detected frontier point so that the next point selected from the RRT in the unknown area has a higher likelihood.

Nevertheless, the spreader the tree becomes in the global frontier detector, the slower the growth gets. By observing the Voronoi diagram of RRT, this fact can be explained: the more amount of vertices, the more decomposed in smaller regions the map becomes; so that the steer function will generate smaller edges, thus, the target point detection gets slower. This is the reason why local frontier detector complements the global frontier detector so that the performance is quicker.

3.2.4 Filter module

The input of this module is the output of the frontier detector module which is the set of the different possible frontier points. Every detection is stored in a frontier points array which will be filtered by using the mean shift clustering algorithm[29], [62]. Then the array is cleaned from the discarded points, the ones that do not belong to a center of a certain cluster. This module is needed since the generated points of the frontier detector can be many and this suppose redundancy and uncertainty which increases computational consumption. In addition, it gets rid of non-valid and outdated frontier points for every iteration. Afterwards, a votation system selects the most convenient and efficient target out of all the possible candidates.

3.3 Object detection module

Briefly is worth to mention that another taken approach could have been the use of object detectors based on machine learning methods. The problem comes with the poor generalization capabilities since feature extraction requires to be handcrafted for every class and the shallow classification method is not robust against constant object state changes (i.e. noise, scale, illumination, position, angle, ...). For all of this, the enhanced object detection technology based on deep learning is taken as the valid approach for the work.

This module generates the bounding boxes encapsulating the detections which are the inputs of the Map filtering module.

3.3.1 Transfer learning

Since CNN-based object detection algorithms requires huge amount of training dataset, it is out of the scope of this work to train a customized model either from a pre-trained model or from a model from scratch. Fortunately, there are open source[68] pre-trained models with the parameters (weights) already tuned which makes them suitable to be ready-to-use without any posterior training procedure. The procedure of obtaining the parameters of an already trained network and applying to a vanilla model is known as transfer learning.

Such open source available pre-trained models can be fed from several datasets[68]:

- COCO
- PASCAL VOC
- Kitti
- Open Images
- iNaturalist Species

Among all of the available bases, the pre-trained models based on the COCO dataset is the most generic and with more detectable categories (80 different classes). So this is the one chosen for the work.

3.3.2 Selective object discrimination

Since this work intends for selective object detection it is possible to do so by doing some modifications which filter the label map or the corresponding configuration file for the pre-trained model. For instance, in the context of Tensorflow Object Detection API with a COCO based pre-trained model there is a file called `mscoco_label_map.pbtxt` which contains the 80 different category labels from the COCO dataset[68]:

```
item {
  name: "/m/01g317"
  id: 1
  display_name: "person"
}
item {
  name: "/m/0199g"
  id: 2
  display_name: "bicycle"
}
item {
  name: "/m/0k4j"
  id: 3
  display_name: "car"
}
item {
  name: "/m/04_sv"
  id: 4
  display_name: "motorcycle"
}
```

Figure 60. *A small piece of code of mscoco_label_map.pbtxt. [68]*

3.3.3 Object detection model

Since this module has to work in the context of SLAM, which is a real-time process, it is fundamental the inference time. Since the most efficient state-of-the-art of the real-time object detection nowadays is YOLOv3[69], this model has been selected for the work.

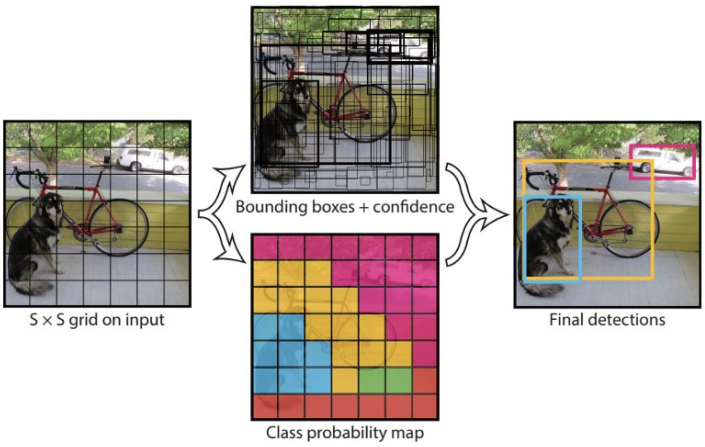


Figure 61. *YOLO architecture.*

It works in the way that the input image is split into an S x S grid, within each of the grid cells, n bounding boxes are retrieved. For each of the generated bounding boxes, the network outputs a category probability and offset values for the bounding box. As long as the candidate bounding boxes have a class probability above a certain threshold, they are selected and used to localize the object in the image. The main limitation is that small objects are hardly detected due to the spatial constraints.

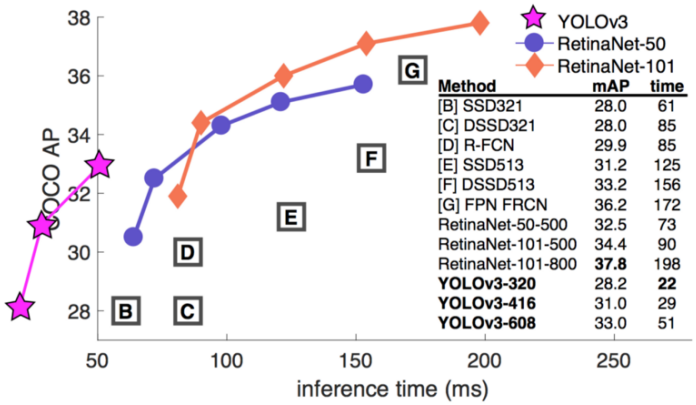


Figure 62. *Comparative plot of the current best real-time object detectors available. [69]*

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 63. *Comparative table of the current best real-time object detectors available. [69]*

3.4 Map filtering module

Since ROS allows huge flexibility it is possible to do the proper modifications of the laser scan or the live generated map during the process, so that, post-processing is actually not needed. Actually, it is undesired since by doing such approach, it is less straightforward and it would require to store the history of the positions of the mobile robot and the detected objects along the time which it would increase the complexity of the module.

3.4.1 Object discrimination procedure

For this module, two approaches were found feasible. For simplifying the problem, let's call them RGB+PointCloud input and RGB+LaserScan input approaches:

3.4.1.1 RGB+PointCloud input approach

This approach takes RGB and pointcloud streamings as inputs. The RGB feeds the object detector and then there is an auxiliar process in charge of retrieving the depth of the bounding boxes in the space via the pointcloud input. Such retrieval is possible to do it in different ways, some of them can be by taking the depth of the point in the pointcloud corresponding to the pixel which is at the same time corresponding to the center of the bounding box. A more refined way is it take the average of the positions of the neighboring points from the pointcloud so that the depth retrieval is more robust to noise.

Once the position in the 3D space is retrieved, it is saved in a frame whose parent frame is the camera frame. Since the orientation is irrelevant, all the generated frames have the same default orientation.

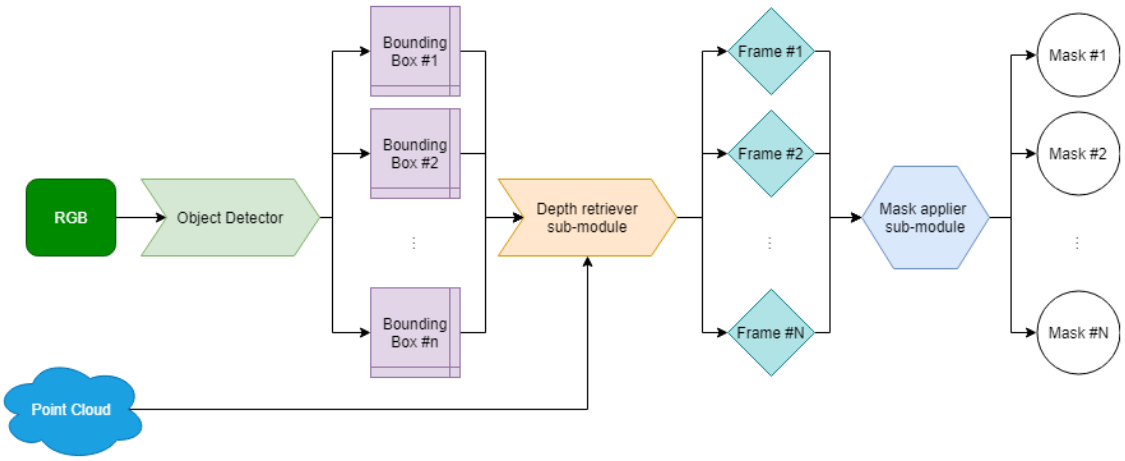


Figure 64. *Workflow chart of the RGB+PointCloud input approach.*

When defining the masks, that is, the polygon areas that will override the occupied grid cell values with free values, there can be different approaches. One approach (option A) is to gather all the neighboring and connected occupied grid cells, and also the ones surrounded by them. Then, their grid cell values are set to free, so that, such detected object will no longer be part of the map. Let's see an example with a scanned paper bin in an office:

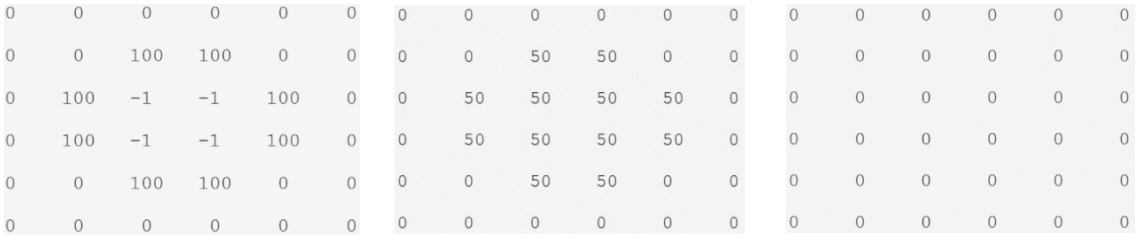


Figure 65. *Occupancy grid processed values by the RGB+PointCloud input approach from the map filtering module.*

The left image shows the occupancy of the example after being scanned. In the center image, the neighboring and surrounded grid cells gathered in common values are shown. The right image displays all the grid cells a priori composing the detection set to free values.

The other approach (option B) takes also as input the width of the bounding boxes of the detected objects and then apply a polygon under the assumption that the detected object has a certain width / depth ratio. That ratio can be generalized to 1 since many objects frequently share such similar ratio value. The model can be much more refined since the bounding boxes are labeled with their corresponding category. A certain category always will hold a similar width / depth ratio value among the different corresponding samples.

Another core feature of this approach is to correct the position of the frame corresponding to the detection. That is because it is by default located in the contour of the object and it should be in its center of gravity which is obtained under the assumption of a certain depth dimension value. So that when the polygon area is applied to the detection, it is made sure that it is centered and matched with the object projection to the map.

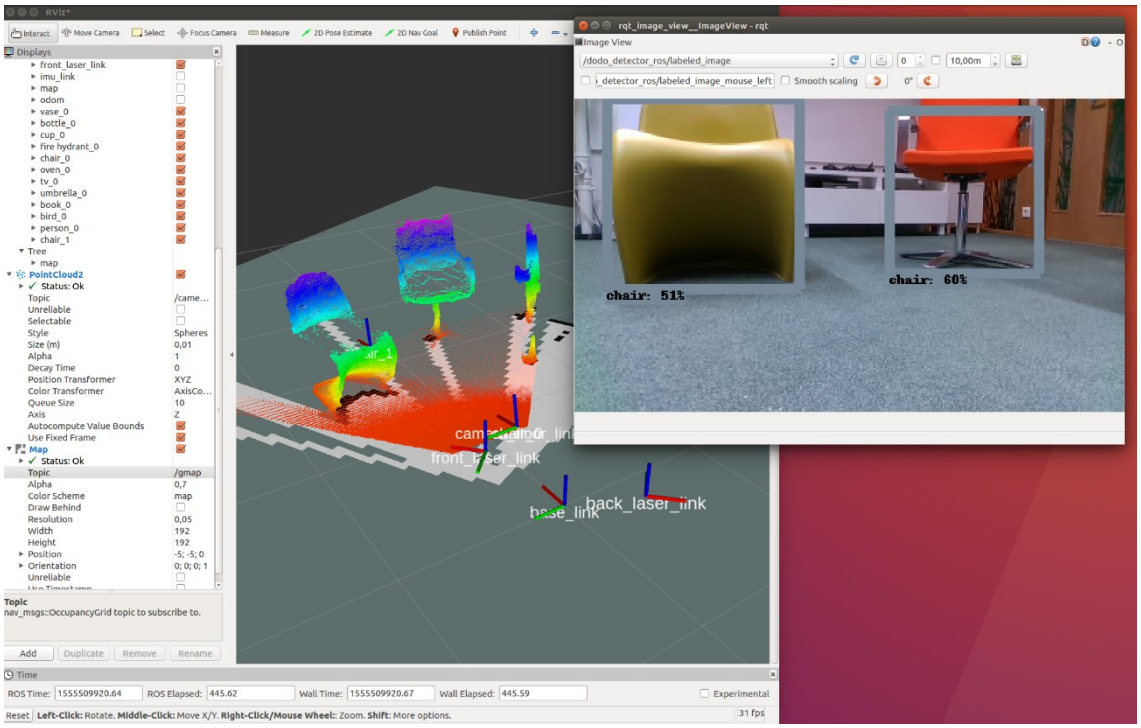


Figure 66. *Backbone testing of this approach showing the detections and their location in 3-D space using the generated pointcloud by the depth camera.*

3.4.1.2 RGB+LaserScan input approach

This approach aims for the RGB feeding for the object detector and then apply a filter to the laser scanner input data so it crops the range in wich the detected object/s are found. This means, that such objects will be blind to the sensors during the entire exploration procedure.

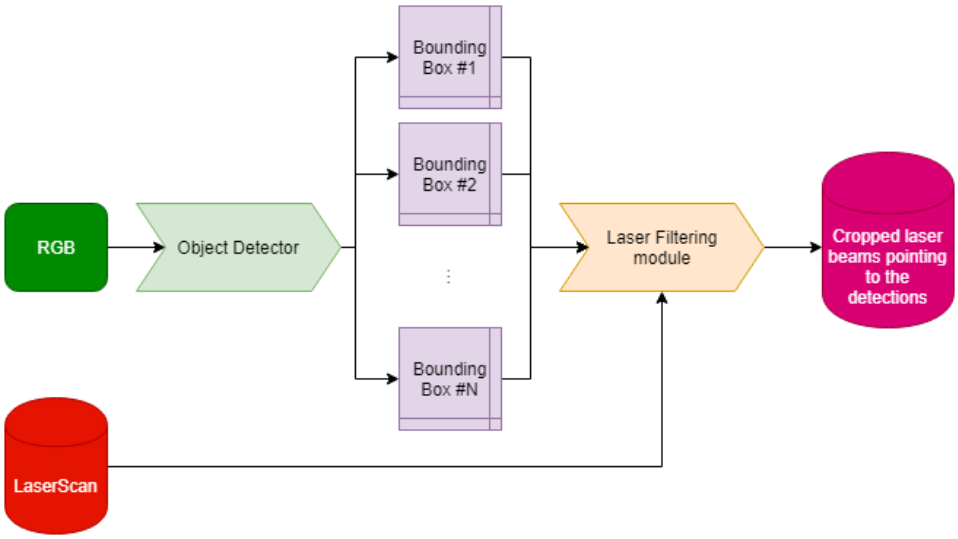


Figure 67. *Workflow chart of the RGB+LaserScan input approach.*

The algorithm used for range data discrimination in the LaserScan stream works as:

```
                    Laser Filtering module
1    S ← getLaserScanPoints();
2    B ← getBoundingBoxes();
3    StructScan ← Ø;
4    NonStructScan ← Ø;
5    PClass ← StructPoint;
6    for each Sn in S do
7       Pn ← convertToCartesianPoint(Sn);
8       for each Bn in B do
9            if isInsideBoundingBox(Pn, Bn) then
10                PClass ← NonStructPoint;
11                break;
12           end if
13      end for
14      if PClass == StructPoint then
15           StructScan += Sn;
16      else
17           NonStructScan += Sn;
18      end if
19   end for;
```

Figure 68. *Laser Filtering module pseudocode.*

3.4.2 Motivation of selecting the RGB+LaserScan input approach

There is an actual trade-off of using one approach or the other. On one hand, the RGB+PointCloud input approach tends to be more sensitive when discriminating detected objects from the map because, even though there is a *False Alarm* (i.e. False Positive) in the detection, such object will be removed from the map. On the other hand, the RGB+LaserScan input approach has the opposite effect, it is less strict when it comes discriminating objects since a non-detection of an actual object in the real world (a False Negative) will include it on the map.

The point of removing the detections from the map is to increase the long-term global path plannings of the mobile robots using such map. So there is a seek in maximizing the amount of discriminated objects as long as it is done in a reasonable way. It is considerably more undesirable to have a False Positive which is part of the structural environment since such structural item will stay there permanently. For example, a column which is falsely inferred by the object detector as a paper bin.

On the other hand, if there is a False Negative of a temporal object, i.e. it is not detected and included on the map, it is a less critical issue, since the navigation is more efficient if it is performed by just relying on the global path planning and not the local path

planning since re-routing would require more time if unpredicted obstacles are encountered in the real world and are not documented on the map. Thus, precision is ponderately more relevant than recall.

In addition, the RGB+LaserScan input approach is computationally more efficient. This is especially important for SLAM since it is undesirable to overload the computer system operating the robot as it requires higher hardware capabilities and more energy. Besides, it is way less complex to implement since it does not need to deal with any type of depth information such as depth images or pointclouds. This means, we keep the scope of the work in the 2-D dimension so does the output, i.e. the filtered map, so we do not have to care about the technical specifications of the depth resolution and ranges of the depth camera.

Lastly, the drawbacks when applying masks with the RGB+PointCloud input approach are serious:

- Option A: By gathering all the neighboring and connected occupied grid cells and also the ones surrounded by them, there is not certainty when is enough to stop gathering. For instance, let's imagine a detected paper bin contiguous to a wall, without any post processing, the wall will also be included and then deleted from the map. So that, an additional process would be necessary in order to control this problem.
- Option B: By assuming a certain width / depth ratio of a detection is quite a subjective hypothesis. It makes the system biased and unreliable. On the contrary, the RGB+LaserScan input approach is absolutely objective since any assumption was made when determining the parameters of the system.

3.5 The system as a whole

So that the system runs in the way that the navigation for autonomous exploration module runs in independently in parallel with the sequence: map filtering module which depends on the object detection module. All of this can be visualized in a simple diagram:

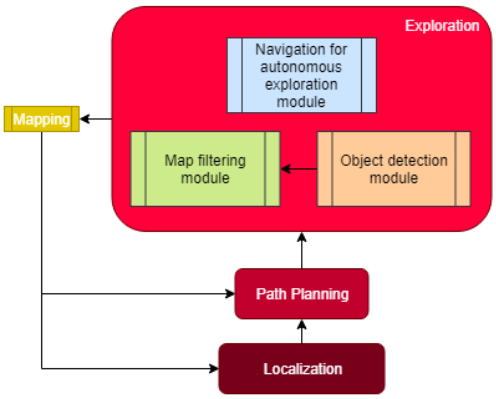


Figure 69. *Diagram of the whole system.*

4. IMPLEMENTATION

4.1 Navigation for exploration module

The autonomous exploration approach is composed of the SLAM module, the path planning, the frontier detector and the filter modules.

For the implementation, the first two aforementioned modules are standard ROS open source packages available to use for path planning and planning. The other modules are integrated by the rrt_exploration package which is composed by the frontier detector and the filter nodes.

4.1.1 SLAM module

The inputs are the laser scan readings, the transformations between the robot base frame and the laser scanner sensor frame, and the odometry. It provides the map building implementation and the localization of the mobile robot in the environment simultaneously, so the outputs are the occupancy grid and the robot pose. For this purpose, gmapping package[25] which implements Rao-Blackwellized particle filtering is used[63], [64].

4.1.2 Path planning module

It takes the outputs of the previous module and a target position so that this module publishes velocity commands to the mobile robot. In order to carry on this task, the move_base node[70] which is part of the navigation stack[71] is required. This node generates the local and global costmaps which are needed for navigation. The costmaps are similar to maps with the difference that the cell values of the occupancy grids range from 0 to 100 (not binary like maps whose cells are either 100 or 0). Such values are the costs derived of inflating obstacles so that the closer the cells are to a certain obstacle the higher the cost value is. The local planner takes the local costmap as input which is in charge of directing the mobile robot to obey the orders of the global path generated by the global planner which takes as input the global costplan.

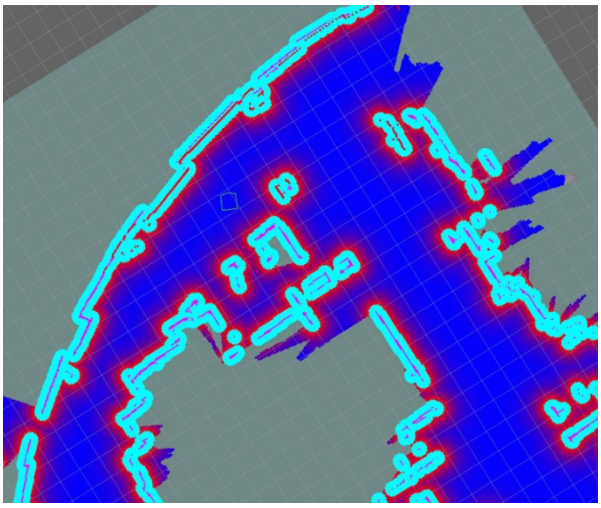


Figure 70. *Visualization of the generated costmaps. [72]*

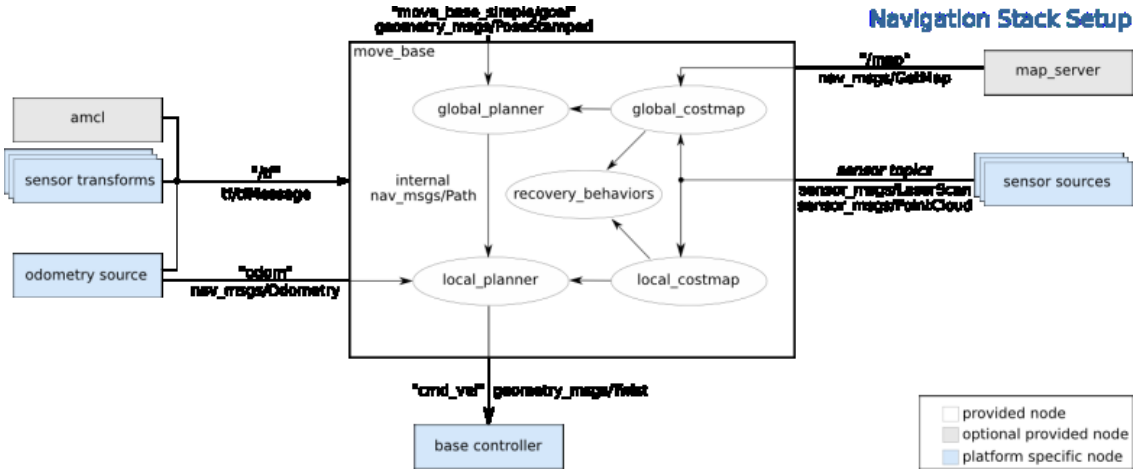


Figure 71. *Move base internal structure graph. [70]*

4.1.3 Frontier detector module

The two nodes are written in C++ to make the detection procedure the quickest possible. They are subscribed to the map topic and publish the identified frontier points in the topic /detected_points (of PointStamped type).

The related nodes have a couple of parameters that set the performance. The map topic name and the growth rate η of the generated tree. As different instances of global/local detector modules can be run simultaneously, they should have different topic names so that there is no information conflict (topic overlapping).

4.1.4 Filter module

This ROS node is written in Python programming language. It is subscribed to the ROS topic /detected_points which is the message (PointStamped type) containing the detected frontier points. Such topic contains point localization shown in coordinate frames.

The global costmap is used for the invalid frontier points deletion. The procedure consists of the fact that when a point is got by the node, its coordinate frame associated is read. Then, with the global costmap information, such frame will be analyzed to check the validity of such point. Thus, if the value of the grid cell of the global costmap in the position of the frame is above a certain threshold (defined by the parameters) then it will be considered as invalid and, as result, removed. Also this module computes the information gain of such frontier points so that if it is null then it they are discarded as outdated.

Once the frontier points are filtered, this node publishes clean data as /filtered_points topic which is the type of PointArray (default ROS message).

The ROS parameters that can be customized in this node are:

- map_topic: sets the map topic processed by the filter node to remove outdated frontier points (from this occupancy grid, the information gain of the frontier points is computed). By default is /map.
- info_radius: it is the information gain radius. By default is 1 meter.
- costmap_clearing_threshold: it is the threshold considered for the invalid frontier point filtering. If the cost value of a certain frontier point is higher than such threshold, then it will be discarded. The higher costs correspond to the grid cells closer to the obstacles. This is because the mobile robot cannot reach them in the real world because it would collide. By default it is 70.
- goal_topic: received frontier points are published in this topic. By default is /detected_points.
- rate: the node performs with this frequency. By default is 100 Hz.

4.2 Object detection Module

The implementation of this module is not trivial since it requires of specific software drivers in order to operate correctly and a compatible powerful GPU.

4.2.1 Software drivers

Modern object detection implementations based in deep learning for real-time applications require of a considerable computational power since they have to deal with inferencing a constant stream of images. In addition, the best accuracy and a fast invariant function taking in consideration constant object state changes (i.e. noise, scale, illumination, position, angle, ...) cannot be computed by a CPU. In order to deal with such intensive calculations and obtain a good performance, a Graphic Processing Unit (GPU) based on parallel processing technique, Data Level Parallelism (DLP), and single instruction multiple data (SIMD) operations with Compute Unified Device Architecture (CUDA) are required[73]. This device is optimized for tensorial calculations unlike CPU.

CUDA is a platform based on parallel computing and a programming model created by nVidia for general computing purposes on GPUs. With this tool it is possible to significantly speed up computing applications[74].

In addition, it is also used NVIDIA CUDA Deep Neural Network library (cuDNN) [75] which provides highly optimized implementations for standard routines such as forward and backward convolution, pooling, normalization and activation layers. It allows to avoid spending time on low-level GPU performance tuning.

4.2.2 Deep Learning framework

For the implementation, DarkNet[76] has been selected as the proper deep learning framework for YOLO:Real-Time Object Detection[77] whose current YOLO version is YOLOv3. It is nicely wrapped-up and the installation is much more simpler and straightforward than TensorFlow Object Detection API[68].

It is open source and efficient since it is written in C and CUDA which is intended for parallel computing.

4.3 Map filtering module

The filtering process should follow this sequence:

1) The laser scanner readings are appended in a cache until the object detector outputs the bounding boxes related data since the output rate publishing differs. In such instant, within such stored data, the laser scan timestamp reading is matched with the nearest timestamp of the object detector output.

For this piece of the implementation, it is used a message filtering protocol integrated in ROS under the name of *message_filters/ApproximateTime*[78].

2) The selected laser scan readings are converted into pointcloud data.

3) The pointcloud initial coordinate reference is translated to the robot's camera frame so that there is a perfect correlation. Thus, the misaligned problem is by-passed. For this piece of the implementation, it is used the library:

image_geometry/ProjectTfFrameToImage[79].

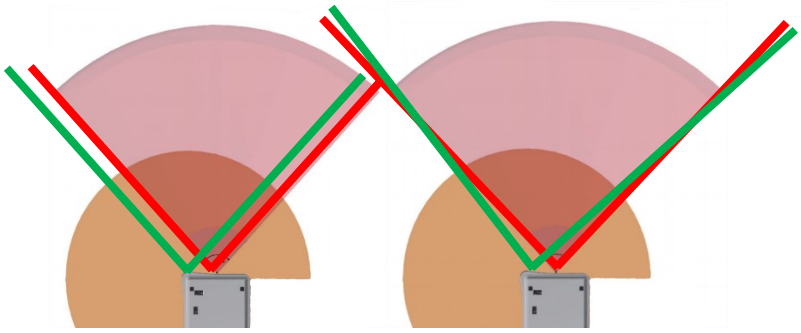


Figure 72. *Vertical axes offset between camera and laser scanner frame leads to some issues.*

Without the step 3), in the left figure we would face some non-intersected field of views small areas. In the right figure, a non-clean solution would be to spin slightly the laser scanner field of view towards the field of view of the camera just valid for short distances.

4) All points outside the image frame are discarded.

5) All the points that lay inside any of the detected published bounding boxes are classified and discarded from the structural LaserScan stream topic.

6) All the points that lay inside any of the published bounding boxes are classified and included in a secondary non-structural LaserScan stream topic.

7) As a result, two LaserScan topics are published, one containing all the LaserScan data that lay outside any of the published bounding boxes (/struct_scan topic) and another one with the data inside any of such bounding boxes (/nonstruct_scan topic).

Both LaserScan stream topics (struct_scan and nonstruct_scan) are fed in the modified slam_gmapping module.

Figure 73. *Resulting included (in green) and excluded (in red) pointcloud data printed directly to the camera data from the laser scanner.*

4.3.1 Additional required features

4.3.1.1 Laser scanner angular range cropping

Since the mobile robot has a frontal laser scanner with a field of view of 270º and also a another one in the rear (see section 5.1.3), it is needed to use just the frontal one and crop it so that there are not blind spots for the camera. That is, to match the field of views of both the camera and the frontal laser scanner.

The reason for this is that, the camera cannot make detections to objects which are out of the field of view. In the case, the laser scanner is recording such ”non-visible” objects, they will be recorded on the map despite being discarded in the case they were ”visible” for the camera. Proceeding in this way, this taken measure allows to discard a considerable amount of False Positives during the exploration.

4.3.1.2 darknet_ros package modification

Since the published topics rely on a constant subscription to the bounding boxes topic, despite of not having detections, such topic should still be publishing. In order to achieve such thing the source code has been just modified with a character in a single line. From the YoloObjectDetector.cpp in line 588:

Instead of

```
if (num > 0 && num <= 100) {
```

replace with

```
if (num >= 0 && num <= 100) {
```

so that this issue is solved.

4.3.1.3 gmapping package modification

The published bounding boxes of certain detections are not constant due to a constant change in illumination, scale, and perspective since the mobile robot is moving so that the camera does.

Figure 74. *Screenshots of a sequence which shows the limitations of the object detector while publishing constant bounding boxes.*

From left to right and from top to bottom, representative frames of the mobile robot spinning clockwise motion.

Since the implementation relies entirely on the constant bounding boxes published detections streaming, if there is a frame during the real-time sequence that there is a misdetection (false positive), the laser scanner will record such object on the map regardless the fact it was previously detected and discarded.

To overcome this problem, a voting system is included in the algorithm that records information for a period of time and classifies the grid cells by using a statistical model.

The standard gmapping algorithm stores the number of visits and the number of detections in each grid cell of the map

1) The visits count value is incremented each time a detection lays inside a grid cell and each time a laser line runs through a cell.
2) The number of detections is incremented each time a detection lays inside a grid cell.

At a periodic rate, a new map is published. The grid cells, as mentioned in previous sections can hold three states:

- Empty cell: with values of 0.
- Occupied cell: with values of 100.
- Unknown cell: with values of -1.

The decision to classify each cell is defined by the following procedure:

- Non-visited cells, i.e. grid cells with visits count null, are classified as unknown cells.
- The occupancy of each cell is calculated with the ratio of the number of detections and the number of visits.
- If the occupancy is greater than a certain threshold, such grid cells are classified as occupied cells, otherwise, as empty or free cells.

This algorithm is modified so that it can use two different counter for the detected laser scans: One for the structural detections and another for the non-structural detections. When a detection lays inside a cell, if such detection is from a structural point, the structural counter is incremented, otherwise, the non-structural counter is incremented. In order to calculate the occupancy of the modified cell, the sum of the structural and non-structural counters is divided by the number of visits.

Thus, the modified map data includes a new cell value: 55 which is meant for the non-structural cells belonging to the detected objects in the environment. The non-structural occupancy is calculated as a simple ratio of non-struct / (struct + non-struct). A grid cell is classified as non-structural if it has an occupied cell with a non-structural occupancy greater than a certain threshold.

In addition, in order to reduce noise in the data, a neighboring filtering is included which converts an occupied cell to a non-structural cell if some of its neighbors are non-structural cells.

5. TESTS AND RESULTS

5.1 Hardware setup

For the experimental work, the components that have been used for the implementation have been the MiR100 robot and a computer connected wirelessly.

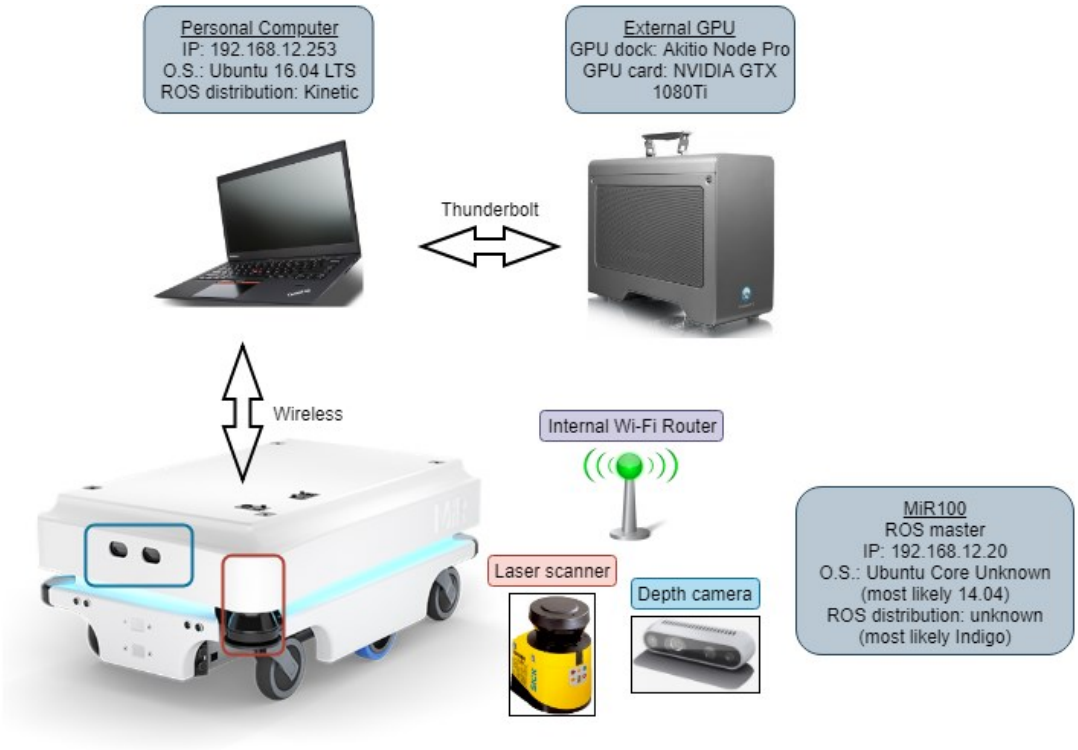


Figure 75. *Implementation components diagram.*

5.1.1 External computer

An external computer with Ubuntu 16.04 LTS as 64-bit OS with ROS Kinetic installed is connected wirelessly and directly to the MiR100 router. The implementation of all the modules and sub-modules runs in this device. The model used for testing has an i5 CPU @ 2.50 GHz and 12 GB RAM memory.

5.1.2 External Graphic Card

Due to the exigent required computational power demanded by the object detector module, an external GPU is connected to the external computer via Thunderbolt 3 connection[80] which uses the same connector format as the USB-C type. This connector provides a maximum transfer data speed of 40 Gbps which makes it suitable for the implementation. However, this cannot be done directly but by using a GPU dock which is the interface between the external computer and the GPU. The device used is an Akitio Node Pro[81], in which a NVIDIA GTX 1080 Ti[82] is attached.

This GPU model is designed to perform for virtual reality which is a very demanding computational task, so it is assured that it is a competent device for deep learning based object detectors. The memory speed is 11 Gbps and it is designed to support CUDA and cuDNN libraries.

5.1.3 Robot platform

The MiR100 (Mobile Industrial Robot) is the mobile robot used to experiment the implementation. The main purpose of this model is to transport autonomously payloads in indoor environments. It comes with ready-to-use autonomous navigation algorithms and teleoperating features via virtual joystick for manual map elaboration through a pre-build browser web interface based on REST services. The implementation of such algorithms is based in ROS packages which are installed in a mini computer which runs the ROS master with Ubuntu as OS (operative system). The MiR100 includes also a Wi-Fi router so that any device can be directly connected to the mobile robot.

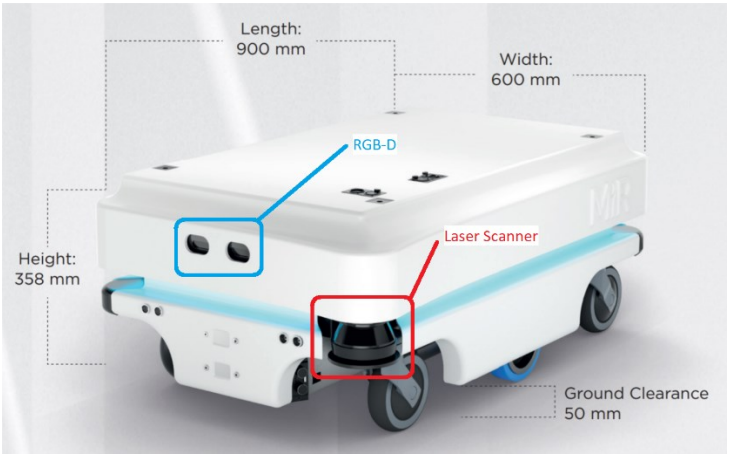


Figure 76. *MiR100 as robot platform with embedded sensors.*

5.1.3.1 Laser scanner

The MiR100 comes with two embedded laser scanners located in the front-left corner and back-right corner, diagonally opposed so that they cover a full range surrounding entirely the mobile robot. Each sensor individually scans an angle range of 270º but when fusing the data of both sensors, the angle range is full 360º so that there are no blind spots. Both models are SICK S300[83] located 200 mm above the ground with a maximum laser scan range of 8 m.

5.1.3.2 Intel RealSense Depth Camera

The depth camera model D435[84] which is embedded in the MiR100 takes two simultaneous images so that the distance of the pixels, with respect to the camera frame, is possible to be retrieved. It detects objects from ground level up to 995 mm above the ground with a maximum distance range of 1950 mm and 86º of angle range. The minimum distance from which the camera sees the floor is 370 mm.

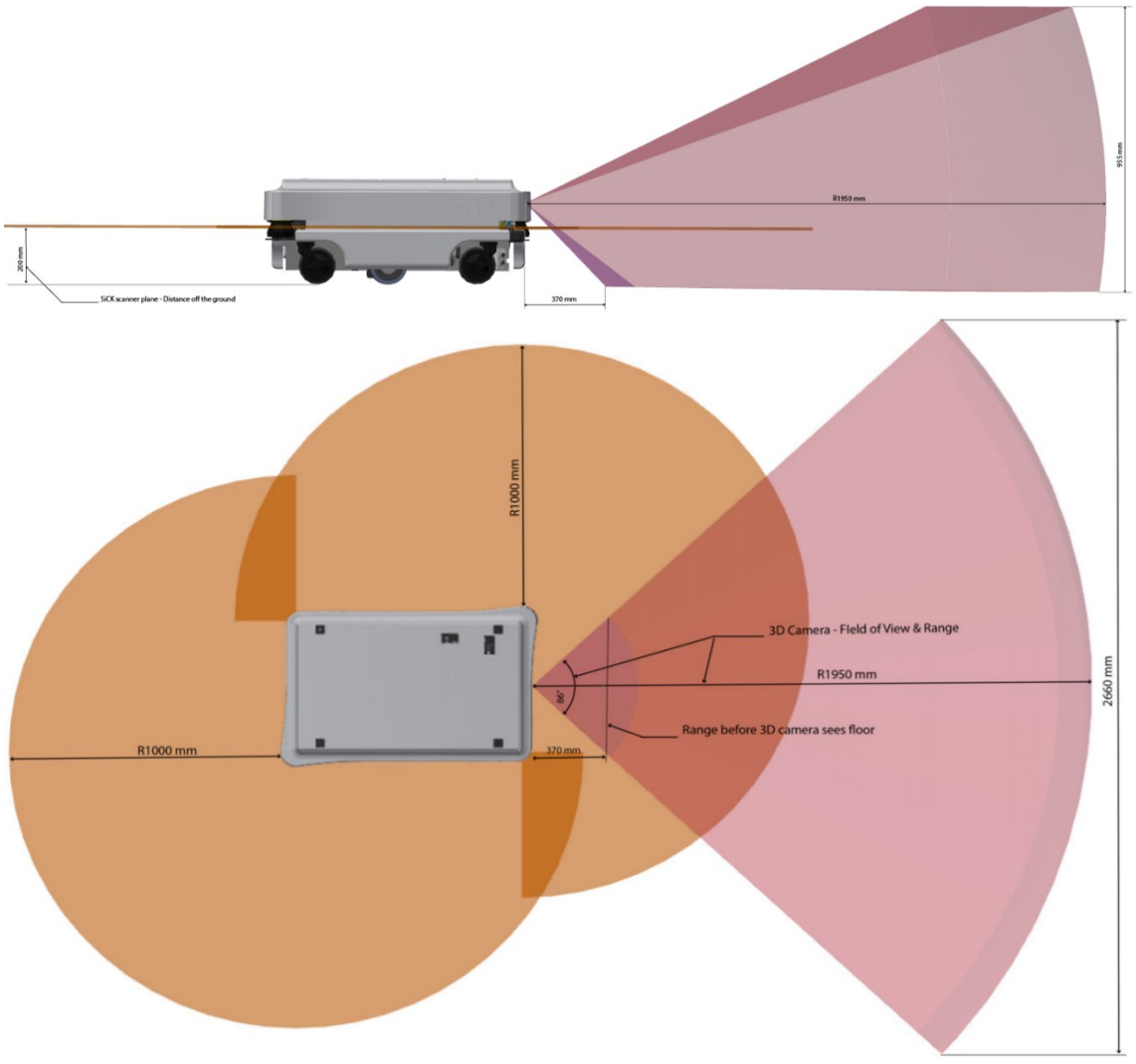
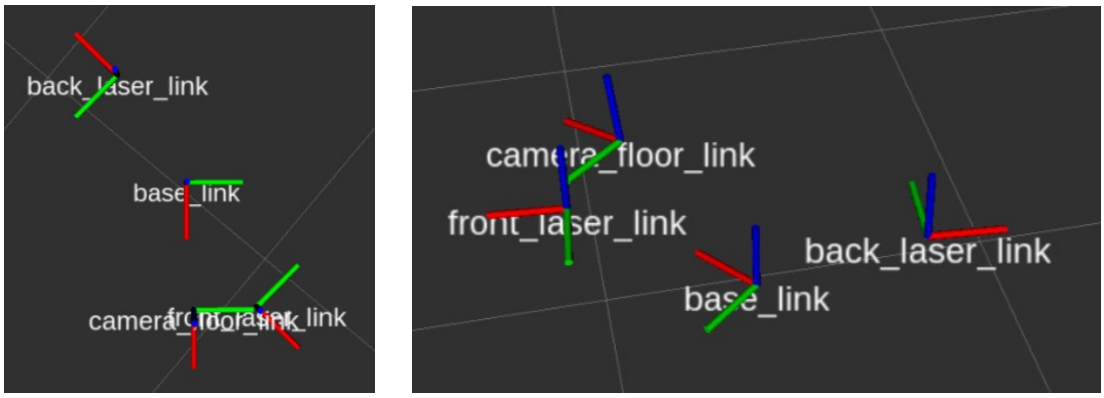


Figure 77. *MiR100 laser scanners and front depth camera fields and ranges of views.*



(a) (b)

Figure 78. *Coordinate frames of the sensors with respect to the mobile robot in 3-D space.*

Seen from above. (b) Isometric view.

5.2 Network setup

By executing the ROS master in a machine (with /roscore command), a ROS network can be built. The MiR100 does this by default every time is turned on. Besides, it executes automatically the required launchers in order to initialize all the features provided by the MiR100 and its components (e.g. the initialization of the depth camera with the corresponding calibration parameters).

Since it is a straightforward connection, in order to connect to the MiR100 it is necessary to just run two commands:

```
export ROS_MASTER_URI=http://192.168.12.20:11311
export ROS_IP=192.168.12.253
```

However, there are some important details to have in account. First of all, the clocks of the machines connected to a ROS network must be synchronized since many ROS messages are time stamped, and considerable time offsets will definitely cause errors.

Secondly, since MiR100 is an industrial robot which is not designed to be modified (not flexible for research / academic purposes), its unaccesible computer is protected with unknown username and password, so it is not possible to figure out of the version details of the embedded packages. While running the move_base node in the external computer an error occurs due to incompatibility between some topic versions:

```
Client [/mir_auto_bagger] wants topic /move_base/goal to have
datatype/md5sum
```

In order to by-pass this issue, a ROS bridge[85] is required for all the terminals which are related to the move_base goal. This is achieved by communicating indirectly to the topics of the MiR100 via this interface which is launched with the following command after having installed the referenced packages:

```
roslaunch mir_driver mir.launch
```

On the other hand, a ROS bridge cannot deal with large and high-frequency topics such as the ones coming from the pointcloud or image streaming data. This is because internally uses JSON to transfer messages in the network which blows up the message size by a factor of 5 and adds a significant computational overhead compared to directly subscribing to the ROS messages. The result is a huge lag in the network which makes unsustainable any SLAM procedure.

The solution to this is to use ROS bridge, omitting such large and high-frequency topics, in the terminals that move_base node and dependent are run. On the rest of the terminals, the other approach is to subscribe directly to the ROS topics by the two aforementioned commands.

5.3 Limitations

In practice, the implementation of every real engineering system always has some weaknesses. Those are defined by the allocated resources, the environments and conditions to operate.

5.3.1 Limited camera vertical field of view

The camera of the mobile robot is approximately as the same heigh as the laser scanner sensor. It is a relevant position since the pre-built camera is intended for navigation purposes. The problem comes when, during exploration, the mobile robot approaches too much to a tall object so that the most representative part of such object is out of the field of view of the camera. For some cases, the object detector is such powerful that is capable of inferring and encapsulating the object even though just part of it is viewed by the camera. But in other cases this is not possible to make a proper detection.

Figure 79. *Limited camera vertical field of view.*

In the left image, the object detector would inference the tables since the whole picture of the object offers a clear semantic of the case. In the right image, just the poles of the tables are observed, but there is not enough information to infere that the poles corresponds to tables.

5.3.2 Map filtering module limitation

As any approach, this one has its own drawbacks yet makes the system still work objectively for the intended purpose of discriminating detected objects from the map.

On one hand, for the cases in which the mobile robot can maneuver around the detected objects, the expected result is a blind spot in the map in which such object is supposed to be located. In terms of occupancy grid cell values this spot will be defined by UNKNOWN values, but not OCCUPIED since this is undesired value.

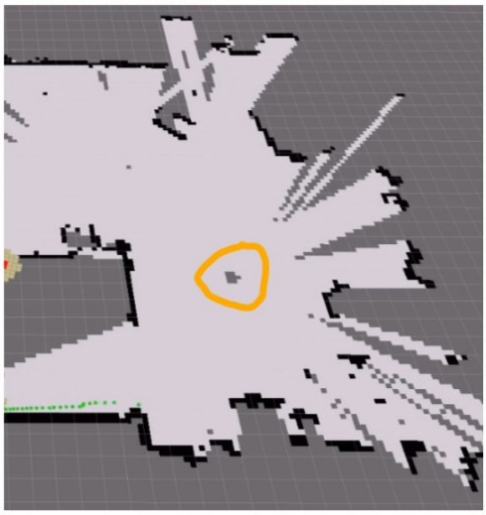


Figure 80. *Filtered map with a non-adjacent detected obstacle.*

On the other hand, for such cases in which the mobile robot cannot maneuver around the detected objects such as the ones contiguous to another objects or walls, there will be a blind area on the map caused by the occlusion product of the detection. This is due to the fact that the laser scanner is blind towards such detections so that everything behind such detection is occluded.

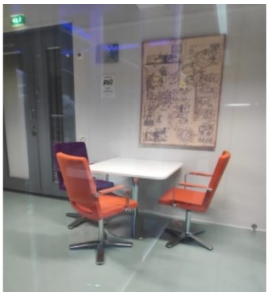


Figure 81. *Case of detections with contiguous wall and other obstacles.*

5.4 Testings

The scenarios for the testing cases are replicated with the same conditions so that the variability of the external conditions does not affect the behaviour of the result. Thus, the initial pose of the MiR is the same for all the experiments so does illumination conditions.

Even though the drawback consisting on not mapping occlusions behind the detections (explained in section 5.3.2), such collateral effect will not be considered as a metric. This is due to the fact that this effect is not related to the efficiency of the system, but mostly related to the layout of the environment and distribution of the objects.

Thus the metrics to evaluate the performance of the system are mainly:

a) Time spent to generate a good enough representative map during exploration.

b) Proportion of grid cell values set to FREE belonging to detections which should be removed since they are the considered temporal objects → True Positives.

c) Proportion of grid cell values with UNKNOWN value which should be KNOWN since they belong to the exploration area. It is a metric which measures the quality of the generated map.

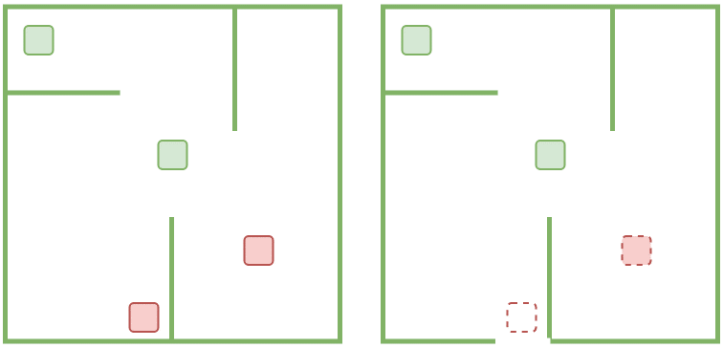


Figure 82. *Ground truth map M_{gt} (on the left) with permanent (green) and temporal objects (red) and filtered map M_f (on the right) with red fill figure representing a False Positive and an empty red figure a True Positive.*

Thus, for expressing quantitatively the second metric, the parameter TPRP (True Positive Removal Performance) is defined as:

$$TPRP = 1 - \frac{\#cells \in (M_f \cap TP)}{\#cells \in M_{gt}} \in T.O. \quad (22)$$

in which the grid cells belongs to the T.O. (Temporal Object) subset. The ratio is defined by the grid cells contained in the filtered map which corresponds to detections

and the ones contained in the original map (or ground truth map). This parameter defines the desired effect in the map filtering since it considers the temporal objects that are removed, which is the purpose of the filtering module.

The third metric is defined by the parameter LMQR (Local Map Quality Ratio):

$$LMQR = 1 - \frac{\#cells \in UNKNOWN}{\#cells} \in M_f, Local\ Exploration\ Area \quad (23)$$

in which the grid cells belongs filtered map and the local exploration area is defined as a control area just for testing purposes in order to quantify and evaluate a common region among all the experiments. The ratio is defined by the grid cells which corresponds to UNKNOWN values and the total number of grid cells. This parameter defines the proportion of UNKNOWN grid cells which should have FREE values but they are not most likely due to imperfections of the mapping module and occlusions of the detections which are not properly explored during the autonomous exploration.

The testings are performed in the FAST laboratory of TUT which is a quite dynamic environment which constantly evolves over the time, i.e. there is a constant change in the layout given by transitory objects, e.g. chairs, tables, people and other temporal stuff. The calculated efective[3] area for exploration is about 70 m2.

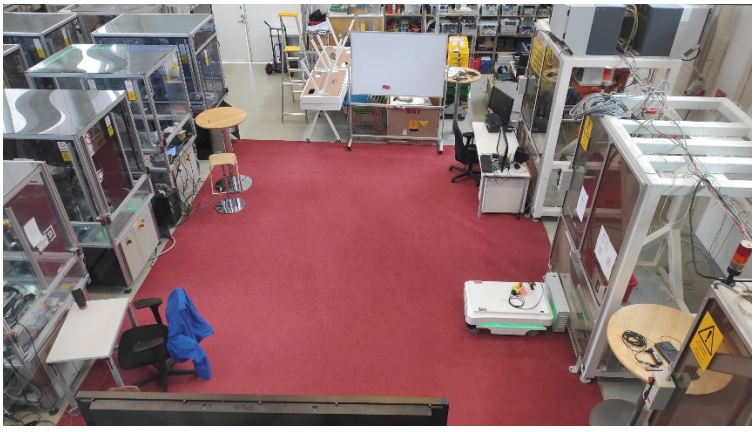


Figure 83. *Testing area for the implementation of the system.*

The local exploration area mentioned beforehand is 23.4 m^2 and it is defined by 4 manually allocated corners.

[3] The main and nearby explorable area which is not occluded by large structural objects such as the robot cages which are part of the environment.

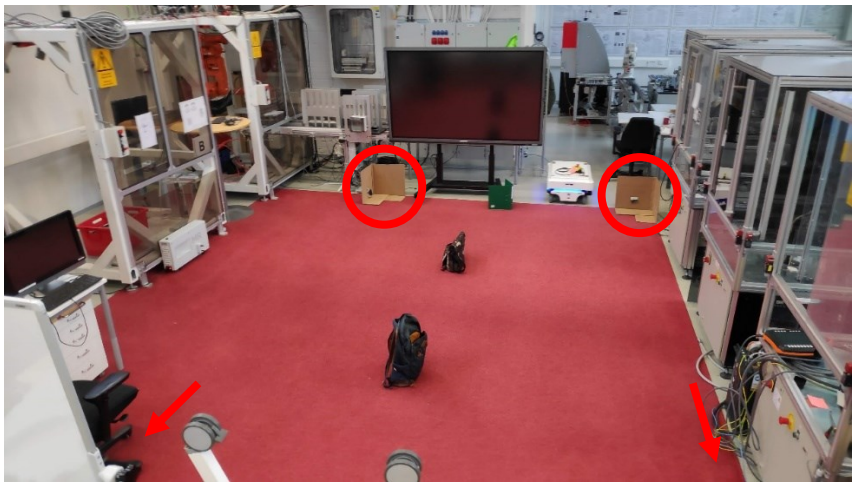
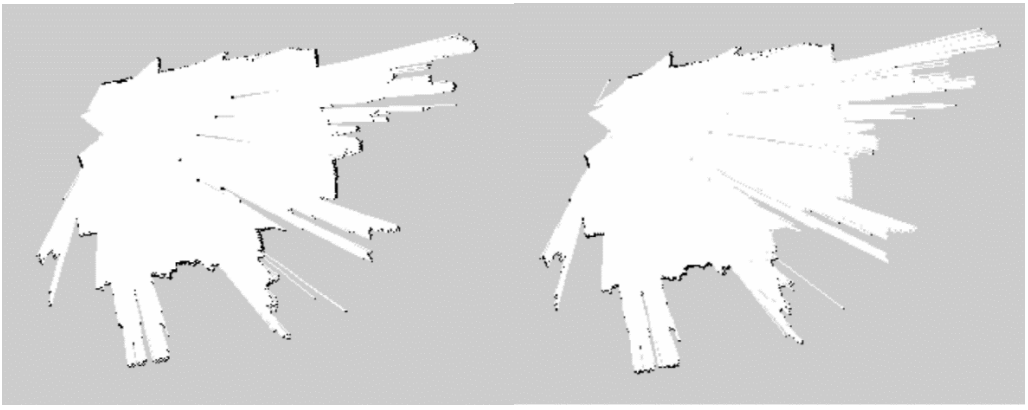


Figure 84. *Corners strategically allocated to compose the local exploration area.*

5.4.1 Case: Degree of crowdedness in the map

For this case scenario, the experiments are recreated by using chairs of similar type as obstacles in the environment since in the office there are plenty of them as resource. The point of this case scenario is to analyze the impact in the quality of the filtered map of the crowdedness of the objects present in the environment as study factor. The distribution of such objects is random and the layout is refreshed at each iteration during experimentation.

(a)



(b) (c)

Figure 85. *Experiment No. 5 in the case scenario of 7 chairs in the area.*

In (a) the shot of the environment. The figure (b) shows the raw map and (c) the filtered map.

It has been observed that the degree of crowdedness in a certain environment affects the required time to obtain a decent map as expected. The apparent correlation is non-linear, it reminds of the logarithmic function. And as expected, directly proportional.

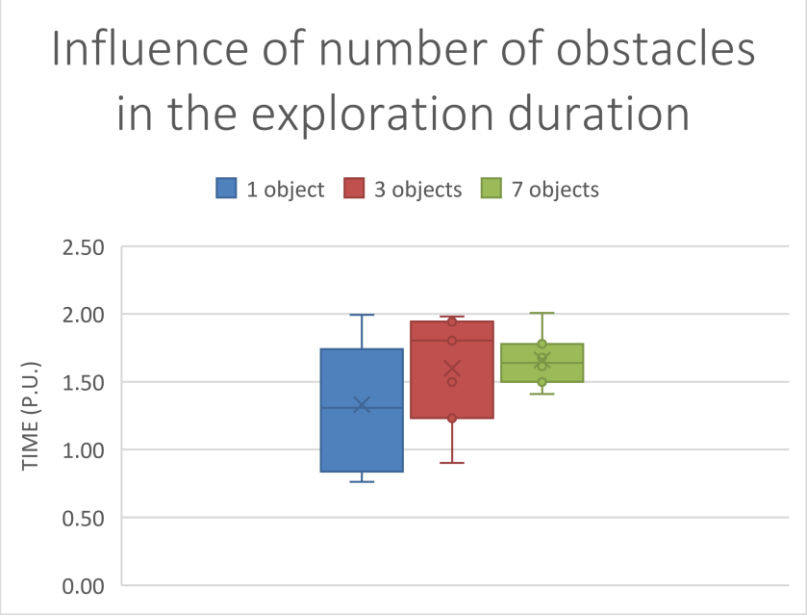
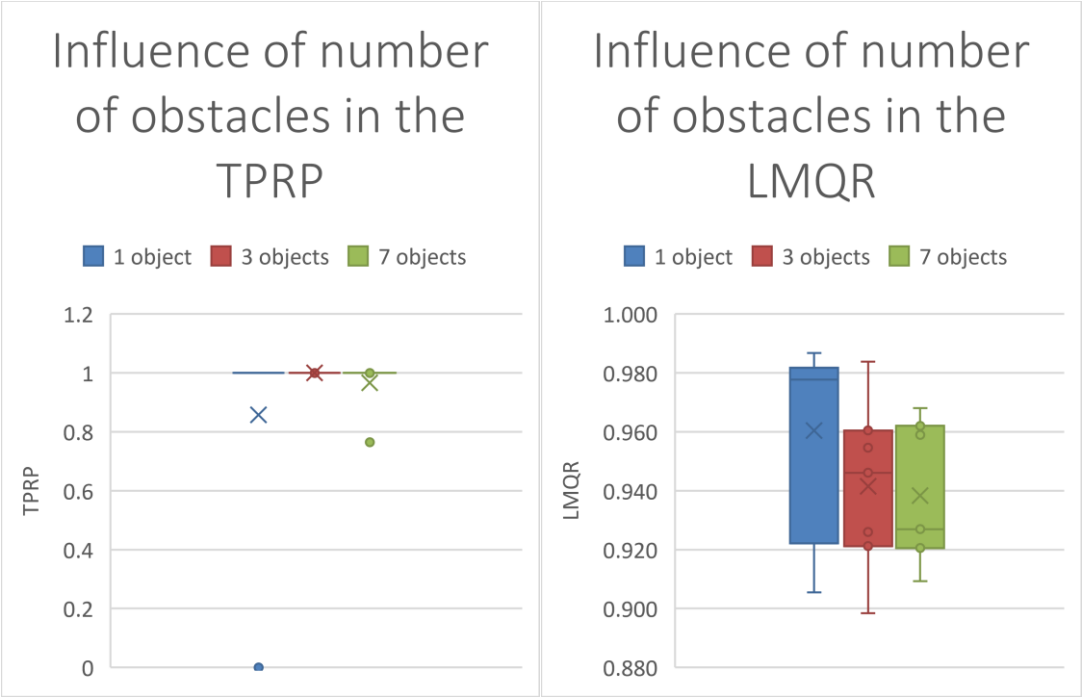


Figure 86. *Box and whiskers of the influence of number of obstacles in the exploration duration.*

On the other hand, there is no evidence that TPRP is correlated with the level of crowdedness in the area.

And finally, there is evidence that LMQR is affected by the amount of obstacles in the area and it is inversely proportional.



(a) (b)

Figure 87. *Box and whiskers of the influence of the obstacles amount in (a) the TPRP and (b) the LMQR.*

5.4.2 Case: Degree of objects size in the map

For this case scenario, the experiments are recreated by using two objects of the same category, one of them adjacent to a wall or permanent object and the other one non-adjacent to anything. The point of this case scenario is to analyze the impact in the quality of the filtered map with the different sizes of the objects present in the environment as study factor.

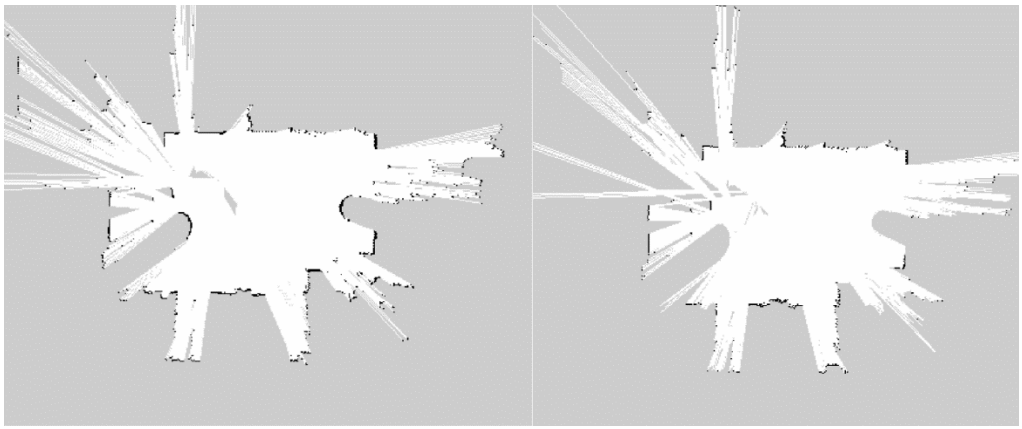
The sizes of the obstacles are quantified by the diameter of the circumcircle encapsulating the coplanar section of any object with the laser scanner height.

Figure 88. *Circumcircle of the coplanar section with the laser scanner range of a certain object.*

There are three considered size types for the testings:

- Small: Office chairs (circumcircle diameter of 7 cm approximately)
- Medium: Backpacks (circumcircle diameter of 35 cm approximately)
- Large: Bean bags (circumcircle diameter of 120 cm approximately)

(a)



(b) (c)

Figure 89. *Experiment No. 5 in the case scenario with large objects (bean bags).*

In (a) the shot of the environment. The figure (b) shows the raw map and (c) the filtered map.

It has been observed that the degree of objects size in a certain environment affects the required time to obtain a decent map as expected. The apparent correlation linear and as expected, directly proportional.

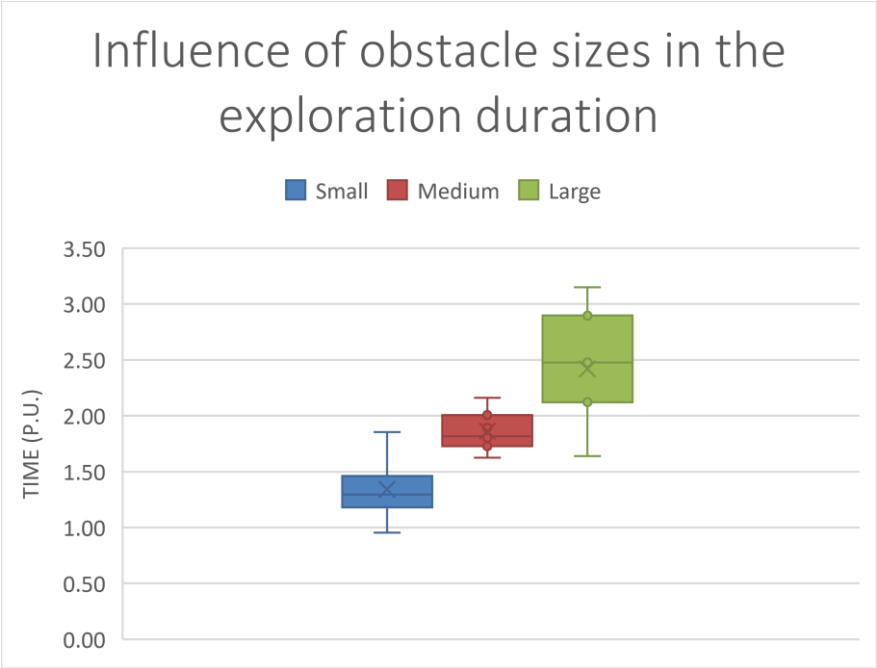


Figure 90. *Box and whiskers of the influence of obstacle sizes in the exploration duration.*

On the other hand, there is an evidence that TPRP is correlated with the level of crowdedness in the area in an inversely proportional way.

And finally, there is evidence that LMQR is affected by the amount of obstacles in the area in an inversely proportional way too.

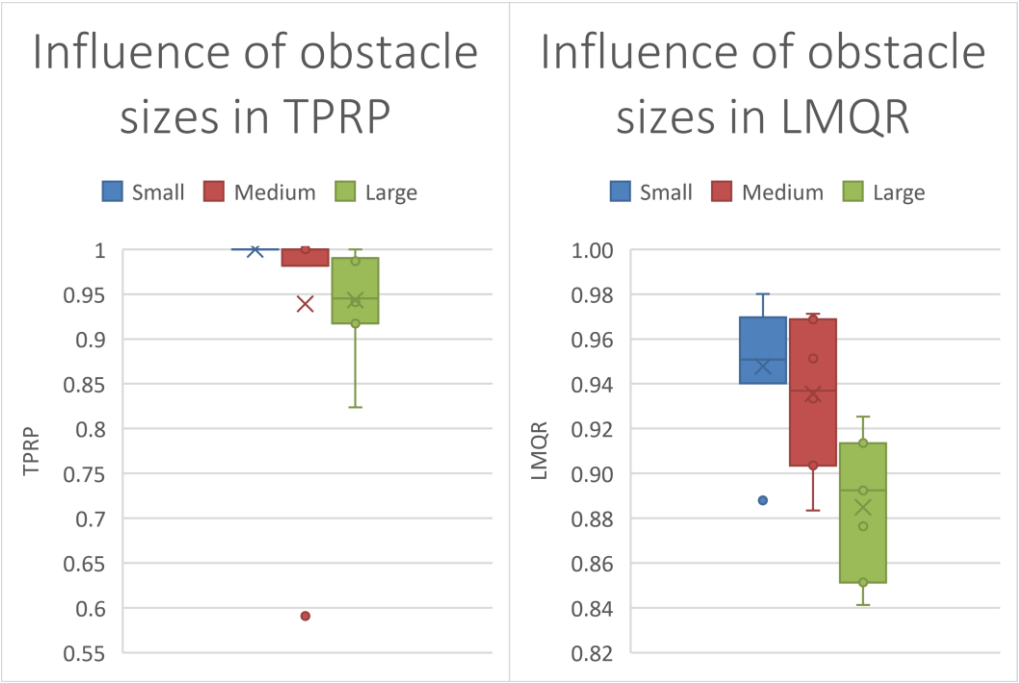


Figure 91. *Box and whiskers of the influence of obstacles sizes in (a) the TPRP and (b) the LMQR.*

5.4.3 Discussion of the results

It has been observed that the results for the quantity of false positives and false negatives are influenced by the motion of the mobile robot. So the quickness of the motion is a parameter to consider since it affects the system performance, however it has not been evaluated since it is out of the scope of this work. This is due to the fact, the camera has a certain shutter speed by default which would lead to blurry images if the motion of the camera is too fast. Naturally, the object detection module has a significantly harder task when dealing with blurry images in order to make proper detections.

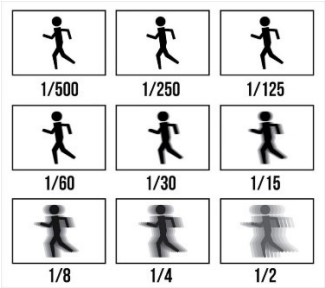


Figure 92. *Motion blur visual concept function of the camera shutter speed.*

Due to this fact, among others, the stream of bounding boxes might not be constant and that is why the modification in the gmapping package (section 4.3.1.3) was necessary in order to have a memory effect of the detections.

With regards of the TPRP, the value of this parameter might not be unity due to the fact that the bounding boxes are not encapsulating totally the obstacles due to bizarre shapes. Also another cause could be due to the fact that the camera of the mobile robot does just see the part of the detected object during exploration so some portions are blind to the filtering.

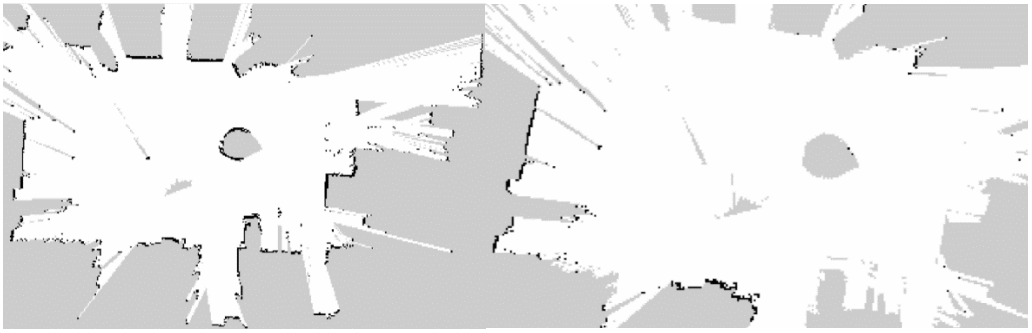
It has been observed that the required times to perform autonomous exploration increases the more crowded the exploration area is of obstacles and the bigger the objects are within it. This is something obvious since the mobile robot needs to perform obstacle avoidance during the mapping and do some re-routings in order to cover the blind spots caused by such obstacles. In addition to this, it has been detected a direct correlation with the hardware of the mobile robot, the computer capabilities and performance which is operating the implemented system, the wireless connection quality and the gmapping parameters.

Regarding the TPRP parameter, the quality of filtering is impacted in a directly proportional way by the size of the detections in the scene. This is evidently occurring since the bigger the detections are, the more prone to error the system is when filtering. Actually, the more surface area of such objects are scanned, the more likely it is to miss some portion from it with the encapsulating bounding boxes. In addition, bizarre shapes

of objects which are non-homogeneously looking are more difficult to be recognized and properly encapsulated by the object detection module.

And finally, the LMQR variable also is inversely proportional to the complexity of the scene. Besides, larger objects will cause larger hollows (UNKNOWN grid cells) which means lower LMQR.

(a)



(b) (c)

Figure 93. *A sample of the testings in which the object detector is enabled for all object categories with several objects of different categories in the scene.*

Notice the occlusions as grey cells. It is physically impossible to know what is beyond a scanned surface, so that objects with section larger than a parameter given by the resolution of the map are mapped as hollow.

6. CONCLUSIONS AND FUTURE WORK

6.1 Contributions

Despite of a remarkable progress in the field of SLAM and autonomous exploration the last decades, yet it is needed an approach in order to make the process of mapping more reliable so that is more coherent with the real world mapped scenario. That is not to just assume that the raw map is good enough for representing the environment. But a processed map which gets rid of objects which are inherently irrelevant to the map since they are not actual obstacles since they were at the moment of mapping just by casuality as they are considered temporal. The point of having more coherent maps for mobile robot navigation is to increase path planning efficiency since mobile robots do not need to avoid presumably non-existing obstacles, since temporal objects won't be there anymore in the near future. A potential of this implementation is that the filtered map generated by autonomous mobile exploration on this approach performed by a single robot can be used by a fleet of mobile robots which considers also the removal of such temporal objects.

6.2 Lessons Learned

It was learned that it is possible to make use of deep learning object detection tools for real-time applications in order to build a customizable filtered map which makes it more coherent to the long-term environment layout.

The initial thoughts were that it was only possible to do the implementation in a post-processing way, that is, to gather the coordinates of the detections on the map and then apply a removal once the map was finished. On the other hand, it was considered to use the depth information of the detections out of the pointcloud generated by the camera. Despite all of these ideas, a computationally more efficient and more straight-forward approach is just making the laser scanners blind towards the detections so we keep the scope in the 2-D dimension avoiding way more complicated approaches.

6.3 Future Work and Research Directions

The implementation of this algorithm has demonstrated to work properly on static detections. Due to time limitations and scope of this thesis, this implementation has not been tested on detections with motion during the exploration (e.g. persons moving around while doing the mapping). However, since the potential is considerable, it is expected to also work in a dynamic environment with objects moving on the scene, so it is a future work to verify this capability and, if required, to tune the algorithm of the package in order to adapt to this needs. Currently, the bottleneck is the Wi-Fi connectivity, since the camera video streaming speed rate is much slower than the laser

scanner data. This means that with the current setup, the system is verified to work for static and objects with very slow motion.

Since the MiR allows the possibility of adding a top camera just above the floor camera (the frontal one by default), it is wise to improve the performance of the system by feeding the object detector with a combination of two simultaneous cameras so that the vertical field of view is larger. This is a quite desirable feature especially for cluttered and tight environments in which the robot does not have enough distance to incorporate tall objects to the camera field of view.

It is possible also to add extra cameras to the corners and laterals, and then merging the images into a single one in order to feed the object detector. It is not worth to run parallel object detectors for every camera since the computational cost would be huge. In this case, by stitching[4] images, it would be possible to extend the laser scanner angular range to the fullest. Even should be possible to fuse the data of both laser scanners for the exploration in the case a 360º panoramic image would be generated by multiple cameras surrounding the mobile robot.

Due to the fact that the Map Filtering Module cannot take in consideration what is behind the objected detections while mapping, an approach dealing with this problem would post-process the map to infer missing parts of the walls that were occluded by such detections and re-fill the hollows caused by the filtering of the objects by changing such UNKNOWN grid cell values to FREE.

And finally, it is also a future work to generate the datasets with custom data for training the Object detector module in order to detect specific objects that are not part of the MS COCO dataset in which is based the current Object detector module.

[4] It is the procedure of combining multiple images with overlapping fields of view so that a panoramic image is generated.

REFERENCES

[1] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97. “Towards New Computational Principles for Robotics and Automation,”* 1997, pp. 146–151.

[2] International Foundation for Autonomous Agents and Multiagent Systems, *The 11th International Conference on Autonomous Agents and Multiagent Systems: Conference Proceedings - Volume III.* Ann Arbor: IFAAMAS, 2012.

[3] S. M. Lavalle, “Rapidly-Exploring Random Trees: A New Tool for Path Planning,” 1998.

[4] G. s. Virk, “Industrial mobile robots: the future,” *Ind. Robot Int. J. Robot. Res. Appl.*, vol. 24, no. 2, pp. 102–105, Apr. 1997.

[5] “Handle | Boston Dynamics.” [Online]. Available: https://www.bostondynamics.com/handle. [Accessed: 16-Apr-2019].

[6] “Building a Magnetic Track Guided AGV.” [Online]. Available: https://www.roboteq.com/index.php/applications/100-how-to/278-building-a-magnetic-track-guided-agv. [Accessed: 16-Apr-2019].

[7] A. Singhal, N. Kejriwal, P. Pallav, S. Choudhury, R. Sinha, and S. Kumar, “Managing a Fleet of Autonomous Mobile Robots (AMR) using Cloud Robotics Platform,” *ArXiv170608931 Cs*, Jun. 2017.

[8] “Amazon Warehouse Order Picking Robots - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=Ox05Bks2Q3s. [Accessed: 16-Apr-2019].

[9] “Lidar,” *Wikipedia*. 14-Apr-2019.

[10] “Incorrect visualization of LiDAR · Issue #32 · carla-simulator/ros-bridge,” *GitHub*. [Online]. Available: https://github.com/carla-simulator/ros-bridge/issues/32. [Accessed: 16-Apr-2019].

[11] E. Rahtu, “Fundamentals of Robot Vision slides from Two-View geometry and stereo vision.” Lab of Signal Processing, TUNI, 2019.

[12] T. Volodine, “Point Clout Processing Using Linear Algebra And Graph Theory.” KATHOLIEKE UNIVERSITEIT LEUVEN, 2007.

[13] “About - Point Cloud Library (PCL).” [Online]. Available: http://pointclouds.org/about/. [Accessed: 10-Apr-2019].

[14] *Intel(R) RealSense(TM) ROS Wrapper for D400 series and SR300 Camera: intel-ros/realsense*. Intel ROS, 2019.

[15] P. G. C. N. Senarathne, D. Wang, Z. Wang, and Q. Chen, “Efficient frontier detection and management for robot exploration,” in *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, 2013, pp. 114–119.

[16] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, “The SRT method: randomized strategies for exploration,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, 2004, vol. 5, pp. 4688-4694 Vol.5.

[17] H. El-Hussieny, S. F. M. Assal, and M. Abdellatif, “Improved Backtracking Algorithm for Efficient Sensor-Based Random Tree Exploration,” in *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, 2013, pp. 19–24.

[18] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using Rao-Blackwellized particle filters,” in *In RSS*, 2005, pp. 65–72.

[19] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte, “Information based adaptive robotic exploration,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, vol. 1, pp. 540–545 vol.1.

[20] H. Umari and S. Mukhopadhyay, “Autonomous robotic exploration based on multiple rapidly-exploring randomized trees,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1396–1402.
[21] E. W. Dijkstra, “A Note on Two Problems in Connexion with Graphs,” *Numer Math*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[22] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.
[23] S. Thrun, W. Burgard, and D. Fox, “Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).” The MIT Press, 2005.
[24] S. Thrun, “Learning metric-topological maps for indoor mobile robot navigation,” *Artif. Intell.*, vol. 99, no. 1, pp. 21–71, Feb. 1998.
[25] “gmapping - ROS Wiki.” [Online]. Available: http://wiki.ros.org/gmapping. [Accessed: 08-Apr-2019].
[26] “nav_msgs/OccupancyGrid Documentation.” [Online]. Available: http://docs.ros.org/kinetic/api/nav_msgs/html/msg/OccupancyGrid.html. [Accessed: 08-Apr-2019].
[27] C. Zhu, R. Ding, M. Lin, and Y. Wu, “A 3D Frontier-Based Exploration Tool for MAVs,” in *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2015, pp. 348–352.
[28] “Mean shift, mode seeking, and clustering,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 790–799, Aug. 1995.
[29] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, no. Oct, pp. 2825–2830, 2011.
[30] H. Huttunen, “SGN-41007 Pattern Recognition and Machine Learning. Slideset 6: Neural Networks and Deep Learning,” 2019. [Online]. Available: http://www.cs.tut.fi/courses/SGN-41007/. [Accessed: 14-Apr-2019].
[31] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
[32] “Backpropagation | Brilliant Math & Science Wiki.” [Online]. Available: https://brilliant.org/wiki/backpropagation/. [Accessed: 14-Apr-2019].
[33] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249–256.
[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J Mach Learn Res*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[35] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.
[36] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1–6.
[37] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way,” *Towards Data Science*, 15-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 14-Apr-2019].
[38] S. Das, “CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more ….,” *Medium*, 16-Nov-2017. .
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, USA, 2012, pp. 1097–1105.
[40] “Receiver operating characteristic,” *Wikipedia*. 20-Mar-2019.

[41] H. Huttunen, “SGN-41007 Pattern Recognition and Machine Learning. Slideset 3: Detection Theory,” 2019. [Online]. Available: http://www.cs.tut.fi/courses/SGN-41007/. [Accessed: 14-Apr-2019].
[42] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” in *Proceedings of the 23rd International Conference on Machine Learning*, New York, NY, USA, 2006, pp. 233–240.
[43] E. Rahtu, “Fundamentals of Robot Vision slides from Image Retrieval.” Lab of Signal Processing, TUNI, 2019.
[44] Z.-Q. Zhao, P. Zheng, S. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *ArXiv180705511 Cs*, Jul. 2018.
[45] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective Search for Object Recognition,” *Int. J. Comput. Vis.*, vol. 104, 2013.
[46] R. Girshick, “Fast R-CNN,” *ArXiv150408083 Cs*, Apr. 2015.
[47] R. Gandhi, “R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms,” *Towards Data Science*, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 15-Apr-2019].
[48] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *ArXiv150601497 Cs*, Jun. 2015.
[49] W. Liu *et al.*, “SSD: Single Shot MultiBox Detector,” *ArXiv151202325 Cs*, vol. 9905, pp. 21–37, 2016.
[50] J. Hui, “Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and…,” *Medium*, 27-Mar-2018. .
[51] “The PASCAL Visual Object Classes Homepage.” [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/. [Accessed: 15-Apr-2019].
[52] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *ArXiv14050312 Cs*, May 2014.
[53] J. Cheng, Y.-H. Tsai, W.-C. Hung, S. Wang, and M.-H. Yang, “Fast and Accurate Online Video Object Segmentation via Tracking Parts,” *ArXiv180602323 Cs*, Jun. 2018.
[54] J. Huang *et al.*, “Speed/accuracy trade-offs for modern convolutional object detectors,” *ArXiv161110012 Cs*, Nov. 2016.
[55] “ROS/Concepts - ROS Wiki.” [Online]. Available: http://wiki.ros.org/ROS/Concepts. [Accessed: 08-Apr-2019].
[56] “REP 103 -- Standard Units of Measure and Coordinate Conventions (ROS.org).” [Online]. Available: http://www.ros.org/reps/rep-0103.html. [Accessed: 08-Apr-2019].
[57] “tf - ROS Wiki.” [Online]. Available: http://wiki.ros.org/tf. [Accessed: 08-Apr-2019].
[58] “REP 105 -- Coordinate Frames for Mobile Platforms (ROS.org).” [Online]. Available: http://www.ros.org/reps/rep-0105.html. [Accessed: 08-Apr-2019].
[59] “hector_slam/Tutorials/SettingUpForYourRobot - ROS Wiki.” [Online]. Available: http://wiki.ros.org/hector_slam/Tutorials/SettingUpForYourRobot. [Accessed: 08-Apr-2019].
[60] V. Potó, J. Á. Somogyi, T. Lovas, and Á. Barsi, “Laser scanned point clouds to support autonomous vehicles,” *Transp. Res. Procedia*, vol. 27, pp. 531–537, Jan. 2017.
[61] H. Umari, *A ROS package that implements a multi-robot RRT-based map exploration algorithm. It also has the image-based frontier detection that uses image processing to extract frontier points.: hasauino/rrt.*. 2019.
[62] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[63] G. Grisettiyz, C. Stachniss, and W. Burgard, “Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective

Resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 2432–2437.
[64] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.
[65] S. Mukhopadhyay and F. Zhang, "A path planning approach to compute the smallest robust forward invariant sets," in *2014 American Control Conference*, 2014, pp. 1845–1850.
[66] P. Varnell, S. Mukhopadhyay, and F. Zhang, "Discretized boundary methods for computing smallest forward invariant sets," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 6518–6524.
[67] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *ArXiv11051186 Cs*, May 2011.
[68] *Models and examples built with TensorFlow. Contribute to tensorflow/models development by creating an account on GitHub*. tensorflow, 2019.
[69] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv180402767 Cs*, Apr. 2018.
[70] "move_base - ROS Wiki." [Online]. Available: http://wiki.ros.org/move_base. [Accessed: 08-Apr-2019].
[71] "navigation - ROS Wiki." [Online]. Available: http://wiki.ros.org/navigation. [Accessed: 18-Apr-2019].
[72] "Global costmap cleared from time to time · Issue #783 · ros-planning/navigation," *GitHub*. [Online]. Available: https://github.com/ros-planning/navigation/issues/783. [Accessed: 18-Apr-2019].
[73] S. A. Dawwd and U. T. Salim, "GPU acceleration of object detection on video stream using CUDA," in *2013 International Conference on Electrical Communication, Computer, Power, and Control Engineering (ICECCPCE)*, 2013, pp. 198–203.
[74] "CUDA Zone," *NVIDIA Developer*, 18-Jul-2017. [Online]. Available: https://developer.nvidia.com/cuda-zone. [Accessed: 16-Apr-2019].
[75] "NVIDIA cuDNN," *NVIDIA Developer*, 02-Sep-2014. [Online]. Available: https://developer.nvidia.com/cudnn. [Accessed: 16-Apr-2019].
[76] J. Redmon, "Darknet: Open Source Neural Networks in C," 2016-2013. [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 15-Apr-2019].
[77] "YOLO: Real-Time Object Detection." [Online]. Available: https://pjreddie.com/darknet/yolo/. [Accessed: 18-Apr-2019].
[78] "message_filters/ApproximateTime - ROS Wiki." [Online]. Available: http://wiki.ros.org/message_filters/ApproximateTime. [Accessed: 29-Apr-2019].
[79] "image_geometry/Tutorials/ProjectTfFrameToImage - ROS Wiki." [Online]. Available: http://wiki.ros.org/image_geometry/Tutorials/ProjectTfFrameToImage#CA-c6a3f2753eb8c6c346819f225dd63e566b352f98_12. [Accessed: 29-Apr-2019].
[80] D. Ngo, "Here's everything you need to know about USB-C and Thunderbolt 3," *CNET*. [Online]. Available: https://www.cnet.com/how-to/usb-type-c-thunderbolt-3-one-cable-to-connect-them-all/. [Accessed: 21-Apr-2019].
[81] "Node Pro - Thunderbolt 3 PCIe Expansion Chassis with PD," *AKiTiO*. [Online]. Available: https://www.akitio.com/expansion/node-pro. [Accessed: 21-Apr-2019].
[82] "It's Here: The New GeForce GTX 1080Ti Graphics Card," *NVIDIA*. [Online]. Available: https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/. [Accessed: 21-Apr-2019].
[83] "S300 Standard | SICK." [Online]. Available: https://www.sick.com/fi/en/opto-electronic-protective-devices/safety-laser-scanners/s300-standard/c/g187239. [Accessed: 21-Apr-2019].

[84] “Intel® RealSense™ Depth Camera D435.” [Online]. Available: https://store.intelrealsense.com/buy-intel-realsense-depth-camera-d435.html. [Accessed: 21-Apr-2019].
[85] *ROS support for the MiR100 Robot. Contribute to dfki-ric/mir_robot development by creating an account on GitHub*. DFKI Robotics Innovation Center, 2019.

APPENDIX A: DEGREE OF CROWDEDNESS DATA

None	Number of objects in the scene	1 object							
				Object A					
Time (sec)	Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR
61	1	137	1.74	2	2	0	19140	427	0.978
81	2	157	1.99	2	0	1	18454	337	0.982
103	3	66	0.84	4	0	1	19051	253	0.987
47	4	60	0.76	3	0	1	18592	363	0.980
85	5	77	0.98	1	0	1	18688	584	0.969
70	6	103	1.31	2	0	1	19439	1513	0.922
104	7	134	1.70	2	0	1	19307	1824	0.906
78.7	AVG.	104.9	1.33			0.857			0.960

Number of objects in the scene	3 objects													
			Object A		Object B		Object C		Global					
Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR
1	144	1.83	4	0	1	0	2	0	7	0	1	18951	1494	0.921
2	97	1.23	2	0	5	0	2	0	9	0	1	18651	846	0.955
3	153	1.94	4	0	2	0	4	0	10	0	1	18679	739	0.960
4	118	1.50	2	0	2	0	4	0	8	0	1	19453	1440	0.926
5	71	0.90	5	0	3	0	3	0	11	0	1	19033	1934	0.898
6	142	1.80	5	0	2	0	1	0	8	0	1	18703	302	0.984
7	156	1.98	3	0	4	0	2	0	9	0	1	19481	1050	0.946
AVG.	125.9	1.60									1.000			0.942

Number of objects in the scene	7 objects													
			Object A		Object B		Object C		Object D		Object E		Object F	
Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells
1	129	1.64	1	0	1	0	4	0	4	0	4	4	2	0
2	158	2.01	3	0	2	0	1	0	1	0	2	0	3	0
3	118	1.50	4	0	3	0	2	0	2	0	1	0	NULL	NULL
4	111	1.41	3	0	3	0	1	0	3	0	4	0	1	0
5	132	1.68	2	0	4	0	4	0	1	0	2	0	2	0
6	127	1.61	4	0	3	0	2	0	3	0	4	0	5	0
7	140	1.78	2	0	2	0	3	0	3	0	4	0	1	0
AVG.	130.7	1.66												

Number of objects in the scene	7 objects							
	Object G		Global					
Exp. No.	#Occ. Cells	Filt. Occ. Cel	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR
1	1	0	17	4	0.7647059	18107	743	0.959
2	3	0	15	0	1	19269	1408	0.927
3	2	0	14	0	1	19125	1520	0.921
4	3	0	18	0	1	17963	575	0.968
5	1	0	16	0	1	18946	1719	0.909
6	2	0	23	0	1	18932	1462	0.923
7	3	0	18	0	1	18546	704	0.962
AVG.					0.966			0.938

Figure 94. *Raw data results for the degree of crowdedness in the map experiments.*

APPENDIX B: DEGREE OF OBJECTS SIZE DATA

None	Circumcircle order of the object to be filtered	Small														
				Object A		Object B		Global								
Time (sec)	Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR			
61	1	108	1.37	1	0	1	0	2	0	1	18548	1110	0.94			
81	2	100	1.27	NULL	NULL	4	0	4	0	1	19286	2160	0.89			
103	3	93	1.18	1	0	1	0	2	0	1	19250	583	0.97			
47	4	115	1.46	5	0	3	0	8	0	1	18358	733	0.96			
85	5	146	1.85	1	0	3	0	4	0	1	18738	374	0.98			
70	6	75	0.95	4	0	3	0	7	0	1	18352	903	0.95			
104	7	102	1.30	NULL	NULL	4	0	4	0	1	19655	1072	0.95			
78.7	AVG.	105.57	1.34							1.000			0.948			
	Circumcircle order of the object to be filtered	Medium														
				Object A		Object B		Global								
	Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR			
	1	158	2.01	27	1	28	0	55	1	0.98	18256	571	0.97			
	2	170	2.16	26	0	18	18	44	18	0.59	19714	2298	0.88			
	3	136	1.73	23	0	25	0	48	0	1	18742	914	0.95			
	4	149	1.89	34	0	20	0	54	0	1	18364	528	0.97			
	5	128	1.63	33	0	20	0	53	0	1	19579	1890	0.90			
	6	142	1.80	33	0	20	0	53	0	1	19738	1246	0.94			
	7	143	1.82	13	0	26	0	39	0	1	18963	1264	0.93			
	AVG.	146.57	1.86							0.939			0.935			
	Circumcircle order of the object to be filtered	Large														
				Object A		Object B		Global								
	Exp. No.	Time (sec)	Time (p.u.)	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	#Occ. Cells	#Filt. Occ. Cells	TPRP	#TotalLocCells	#UNK. Cells	LMQR			
	1	170	2.16	38	0	81	21	119	21	0.82	19280	2384	0.88			
	2	195	2.48	74	0	78	2	152	2	0.99	18769	2791	0.85			
	3	195	2.48	42	6	94	2	136	8	0.94	19245	2070	0.89			
	4	248	3.15	42	1	59	0	101	1	0.99	18358	2914	0.84			
	5	129	1.64	43	0	46	0	89	0	1	17949	1553	0.91			
	6	228	2.90	83	10	62	2	145	12	0.92	18494	1968	0.89			
	7	167	2.12	40	2	51	3	91	5	0.95	19628	1466	0.93			
	AVG.	190.29	2.42							0.943			0.885			

Figure 95. *Raw data results for the degree of objects size in the map experiments.*