Universidade de Aveiro

Ano 2018

Departamento de Eletrónica
Telecomunicações e Informática

**José Manuel
Miranda Ferrão**

Study of SLAM algorithms for autonomous navigation in unstructured environments.

Estudo de algoritmos SLAM para navegação autónoma em ambientes não estruturados.

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizado sob a orientação científica do Doutor António José Ribeiro Neves e Doutor Paulo Miguel de Jesus Dias, Professores auxiliares do Departamento de Eletrónica Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president	**Prof. Doutor José Luis Costa Pinto de Azevedo** Professor Auxiliar, Universidade de Aveiro
vogais / examiners committee	**Prof. Doutor Paulo José Cerqueira Gomes da Costa** Professor Auxiliar, Faculdade de Engenharia, Universidade do Porto
	Prof. Doutor, António José Ribeiro Neves Professor Auxiliar, Universidade de Aveiro

agradecimentos

Gostaria de agradecer à empresa Follow Inspiration por ter possibilitado este trabalho, tendo disponibilizado a plataforma WiiGo e a todos os colegas na empresa que pelo ambiente que proporcionaram.

Quero agradecer ao Professor António Neves e ao Professor Paulo Dias pelo trabalho que fizeram enquanto orientadores deste projeto e por todos os ensinamentos que me passaram ao longo dos últimos anos.

Por fim, um agradecimento especial para a minha família e para a Mafalda por me terem acompanhado e incentivado durante todo este percurso académico.

keywords Mapping, Navigation, Localization, ROS, SLAM, Robotics

abstract

Robotics is one of the most exciting areas that has been through constant innovation and evolution over the years.

Robots have become more and more a part of our lives and are no longer a vision for the future but a reality of the present. Nowadays we have robots cleaning our home, vacuuming our floors, playing soccer or even exploring the unknown outside of our planet.

Robots are a major theme in research projects with special attention given to mobile robots since they have the capability to navigate the environment and interact more easily with humans.

In the last couple of years, we have observed a big growth in the market of service robots. A service robot is dedicated to help humans in their everyday tasks.

While reactive or pre-programmed behaviors are sufficient to let a robot appear intelligent, to be truly be intelligent a robot must learn and adapt to its environment.

SLAM is the computational problem of learning an environment by constructing its map while simultaneously keeping track of the robot location inside it.

Follow Inspiration is a company focused on the development of robotic systems. The autonomous shopping cart WiiGo was its first product, it is an autonomous service robot designed to help people carry their purchases in supermarkets.

In this document we describe the testing and integration of SLAM algorithms, development of a marker-based solution to detect interest-points and the development of visualization tools for the WiiGo robot.

The results presented in this document allowed the WiiGo robot to become capable of autonomous navigation in human occupied environments independently, without resourcing to external localization systems.

palavras-chave Mapeamento, Navegação, Localização, SLAM, ROS, Robotica

resumo

A robótica é uma das áreas mas excitantes e dinâmicas que tem apresentado um elevado crescimento ao longo dos últimos anos.

Robôs tornaram-se parte da nossa vida e não são mais uma visão do futuro. Atualmente temos robôs a limpar as nossas casas, aspirar o chão, jogar futebol e até a explorar o desconhecido fora do nosso planeta.

A robótica é um dos maiores temas de investigação atualmente com especial foco em robôs móveis. Estes robôs são capazes de se movimentar e interagir com o ambiente abrindo caminho para novas possibilidades possibilitando novas formas de interação com humanos.

Nos últimos anos foi possível observar um grande crescimento do mercado de robôs de serviço. Estes robôs têm como objetivo auxiliar humanos na execução de tarefas diárias.

Comportamentos reativos ou pré programados são suficientes para fazer um robô parecer inteligente, mas para um robô ser realmente inteligente deve aprender e adaptar-se ao seu ambiente.

SLAM é o problema computacional de aprender um ambiente criando um mapa do mesmo enquanto simultaneamente se estima a localização do robô dentro do ambiente aprendido.

A Follow Inspiration é uma empresa focada no desenvolvimento e produção de sistemas robóticos. O robô WiiGo foi o primeiro produto da empresa, é um robô de serviço que tem como objectivo auxiliar clientes de supermercados a carregar as suas compras.

Neste documento apresentamos testes e integração de algoritmos de SLAM, desenvolvimento de um sistema baseado em marcadores visuais para detecção de pontos de interesse e desenvolvimento de ferramentas de visualização de dados para o robô WiiGo.

Os resultados apresentado possibilitaram que o robô WiiGo se torna-se capaz de navegação autónoma em ambientes ocupados por humanos sem recurso a sistemas de localização externos.

Table of Contents

Index of Tables

Table of figures

1 Introduction

The work described in this document was developed in association with the company Follow Inspiration as a master thesis for the University of Aveiro in the Computer and Telematics Engineering course.

Robotics is one of the most exciting areas that has been through constant innovation and evolution over the years. Robots have become more and more a part of our lives and are no longer a vision for the future but a reality of the present.

With the rapid growth of technology robots are everywhere and as time passes they will be more and more present. Nowadays we have robots cleaning our home, vacuuming our floors, playing soccer or even exploring the unknown outside of our planet.

In the last couple of years, we have observed a big growth in the market of service robots. A service robot is dedicated to help humans in their everyday tasks and are not targeted for industrial applications. Service robots have become a major theme in research projects with special attention given to mobile robots since they have the capability to navigate the environment and interact more easily with humans.

Today service robots are rapidly becoming more intelligent and are already capable of helping humans in a wide range of tasks being in some cases able to replace them completely. But while simple reactive or pre-programmed behaviours are sufficient to let a robot appear intelligent, to truly be intelligent a robot must learn its environment and become capable of dynamically adapt to it.

Automated Guided Vehicles (AVG) are robots capable of autonomous navigation. These robots use external guidelines (e.g. magnetic tape, visual beacons, reflective markers) to define their path. To detect obstacles in their path these robots can be equipped with sensors and can avoid collision with obstacles only by stopping their movement. Figure 1.1 presents an example of these type of autonomous robots.

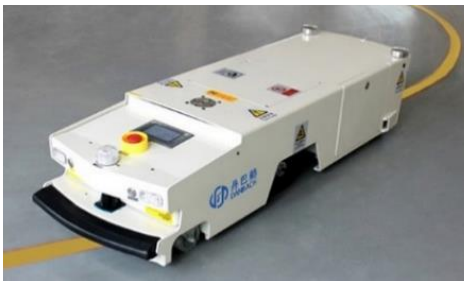


Figure 1.1 - Line following AGV system.

Today AVG technology is widely used for industrial applications. However, the problem with these robots is that they cannot adapt dynamically to the environment. Constant maintenance is required to keep them working properly. When a change to their path is required a new set of guidelines need to be installed.

A solution to overcome these limitations is the usage of Simultaneous Localization and Mapping

(SLAM) algorithms. SLAM can be described as the computational problem of constructing a map of an unstructured environment while simultaneously keeping track of an agent location inside of it.

An unstructured environment is an unknown environment for which no previous knowledge exists. SLAM is an environment independent approach to the navigation problem. Instead of using landmarks as guides, the SLAM algorithm uses the robot sensors (e.g. LIDAR (Light Detection and Ranging), IMU (Inertial Measurement Unit)) to map the unstructured environment while simultaneously keeps track of the robot localization.

Even though SLAM has been frequently used in research for several years it has not been widely accepted in industrial applications.

While AGV systems can use their physical guideline to mark points of interest. In SLAM systems since no previous knowledge of the environment exists, to identify interest points it is necessary to use external solutions (e.g. object detection, beacons, visual markers).

1.1 WiiGo Robot

Follow Inspiration is a Portuguese company founded in 2011. This company focuses on the development and production of robotic systems. The autonomous shopping cart WiiGo was the first product developed by Follow Inspiration it is an autonomous service robot targeting the clients of supermarkets.

The WiiGo robot (Figure 1.2) is an autonomous self-driven shopping chart, designed to help people carry their purchases. WiiGo follows its user around the supermarket carrying all the goods inside.

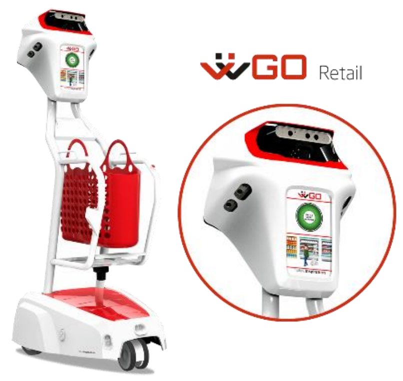


Figure 1.2 - WiiGo retail robot. [1]

During its development there was a big focus on ease to use, user interaction and security. To start the robot the user presses a button and, in a matter of seconds the robot recognizes the user and starts following him automatically. To stop the robot the user can press the button again. The robot is equipped with multiple sensors that allow it to see and avoid any type of obstacles while following its the user.

The company also produces other two robots based on the original user following WiiGo technology, the WiiGo logistics and the WiiGo curator.

The WiiGo Logistics presented in Figure 1.3 is a robot used in industrial environments for the transport objects or as mobile workstation. The robot can follow operators automatically around the environment or be controlled manually using a joystick. The embedded touchscreen allows operators to access management software that can be directly installed on the robot computer.

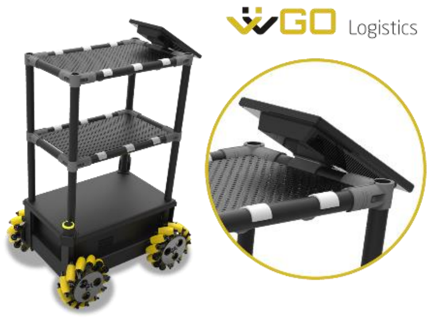


Figure 1.3 - WiiGo Logistics robot. [1]

The WiiGo Curator presented in

Figure 1.4 is an assistant robot focused on human interaction. The principal function of this robot is to welcome and provide information to visitors in public spaces. It is equipped with two big touchscreen screens making it a versatile solution for various scenarios (e.g. user interactable advertising, telepresence for video chat, mobile information spot).

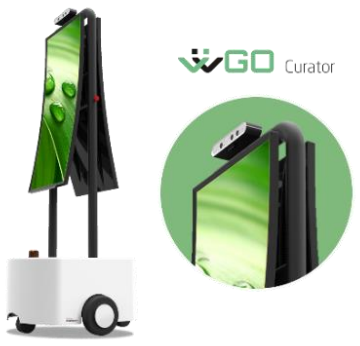


Figure 1.4 - WiiGo Curator robot. [1]

1.2 Objectives and motivation

The first objective of this project is to test SLAM algorithms and select one to be integrated in the WiiGo robot. This objective was motivated by the need to have a robot capable of autonomous navigation in unstructured environments where the use of external localization systems is not feasible. The WiiGo robot software is already capable of point to point navigation and obstacle avoidance. It is only required a solution for mapping and localization.

The second objective is the development of a visual marker-based solution for interest-point detection motivated by the need for solution to identify the charging station in development for the WiiGo robots. Visual markers were chosen because they can be easily detected using the already existing camera of the robot.

The third objective is the development of a new Graphical User Interface (GUI) and a set of 3D visualizations tools for the WiiGo robot. It was motivated by the need to include visualization and interaction tools with the SLAM system integrated into to the robot and the existence of some problems with the old GUI for example the lack of responsiveness to multiple screen resolutions or the fact that there was no support for multiple languages. For the new user interface, a set of requirements were established based on limitation found in the current solution.

- Responsive user interface elements, adaptable to multiple resolutions.
- Modular component system to allows the user interface to be further extended;
- Multi-language support;
- Remote access from a mobile-phone, tablet or computer;
- Access to the robot mapping, localization and navigation system.
- Create of paths for autonomous navigation.

1.3 Document structure

The document is divided into seven chapters, each chapter presents a phase of the work developed. The chapters are organized as:

Chapter 1 (this chapter), introduces the theme, presents Follow Inspiration and the WiiGo robot. It also explains the motivation and objectives defined for this work that are discussed in the next chapters.

Chapter 2 presents the current state of the art for autonomous robots capable of navigation in human occupied unstructured environments.

In Chapter 3 we present the SLAM problem and explore some of the challenges with these types of solutions. We also present SLAM, localization, graph-optimization and feature extraction algorithms that were tested using the WiiGo robot. Most modern SLAM algorithms like the ones we chosen for testing are based on graphs and integrate graph-optimization solutions, some SLAM algorithms also use visual features to overcome the “kidnapped” robot problem.

Chapter 4 presents a detailed study of the WiiGo robot software and hardware architectures and the results obtained from tests performed with the SLAM and localization algorithms presented in Chapter 3 using the WiiGo retail robot.

Chapter 5 presents the development of a visual marker detection solution for interest points identification using Aruco markers. The results obtained with the proposed solutions showed an improvement when compared to other state of the art solutions. The work presented in this chapter was written as an article and submitted for the “15th International Conference on Image Analysis and Recognition” with the title “Detection of Aruco markers using the quadrilateral sum conjuncture.”. At the time of this document it is still not known if the article has been approved for the conference.

Chapter 6 describes the implementation of a new GUI for the WiiGo robot and the development of 3D visualization tools.

Chapter 7 presents conclusions obtained, explains the results obtained and proposes future work to further improve the solutions presented in this document.

2 Autonomous robots

In this chapter we present and explore currently available robots capable of autonomous mapping and localization in unstructured environments. Five robots were selected for analysis.

The robots selected are capable of autonomous navigation in human occupied environments without the relying on external systems for localization. These robots presented have similar sensor configurations to the WiiGo robot and face similar challenges to the ones found during the development of this work.

2.1 Gita Robot

Gita is a robot developed by the Piaggio Group [2], it is intended to be a personal cargo carrying robot, that has two modes of operation, a follow mode and autonomous mode. In the follow mode, Gita learns how to navigate spaces following a human carrying a wearable, in the autonomous Gita can move between defined waypoints detecting and avoiding obstacles.

The Gita robot presented in Figure 2.1. Instead of using more expensive sensors such as LIDAR, the Gita robot maps its environment using cameras. It a uses a stereoscopic camera and several other fisheye cameras to provide a 360-degree view around the robot. An advantage of this approach is possibility to use the robot outdoors.

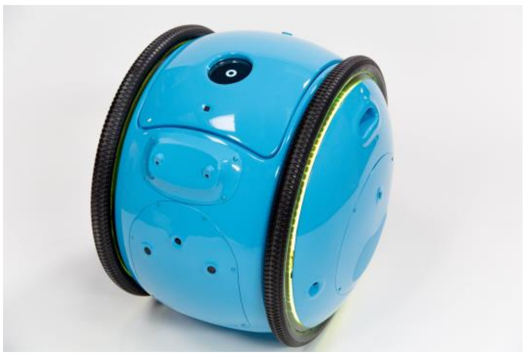


Figure 2.1 - The Gita robot. [2]

The Gita robot is designed to carry up to 40 pounds (18,14kg) of cargo and 2000 cubic inches (32.77L) of volume and can move at a speed of 22mph (9.83 m/s) and has a battery life of eight hours that can be recharged in three hours

For user interaction a small round touchscreen is used, and an array of RGB LED is installed around its wheel for easily visualization of its status.

2.2 OTTO 100 Robot

ClearPath [3] is a Canadian company founded in 2009 dedicated to the development and manufacturing of robots for research and self-driving vehicles for industrial environments. In 2016 they announced the OTTO motors [4] brand dedicated to the development and production of autonomous vehicles.

OTTO has a wide range of robotics solutions, they sell mainly robotics platforms and hardware. We will focus on their OTTO 100 robot presented in Figure 2.2 self-driving vehicle [5].

The OTTO 100 robot is intended to be a direct replacement of typical AGV technology in industrial environments. Providing an environment independent navigation using SLAM algorithms it is designed to move boxes, carts, bins, and other human-scale payloads through unstructured environments.

Figure 2.2 - OTTO 100 robot. [4]

OTTO 100 can carry 100Kg of cargo, move at a speed of 2 m/s and measures 750x500x304mm. The robot is equipped with LIDAR sensors for fully autonomous mapping and navigation, has Wi-Fi support for communication, and is equipped with LED to display its status to its users.

The robot software is based on ROS (Robotics Operative System). For navigation the robot uses a proprietary SLAM solution named Autonomy Research Kit Mapper (ARK Mapper) [6] developed by ClearPath.

User interaction with the robot is achieved using their centralized OTTO M control software. Figure 2.3 presents a screenshot of the OTTO M software. It can be used to interact with the robot mapping and localization system, set paths for autonomous navigation or manually control the robot. The software runs in a tablet and allows to control and check the status of multiple robots simultaneously.

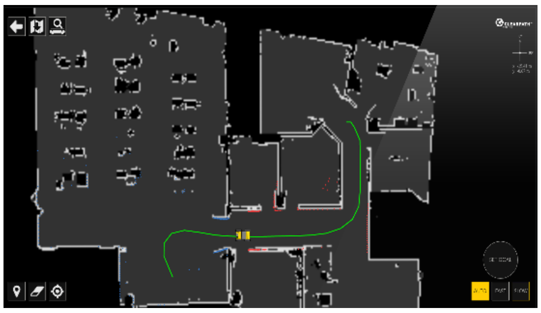


Figure 2.3 - OTTO M control software.

2.3 Personal Robot 2

Personal Robot 2 (PR2) presented in Figure 2.4 is an open research platform developed by the company Willow Garage as a successor to the PR1 that was a Stanford university project [7].

Willow Garage is a robotics company created in 2006 dedicated to the development and manufacturing of hardware and was the home for creation of the open source framework ROS that was been widely used to power every type robotics applications.

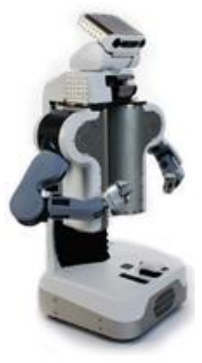


Figure 2.4 - PR2 robot. [7]

The PR2 robot has four cameras (a Microsoft Kinect and three RGB cameras), two Hokuyo UTM-30LX LIDAR sensors and is equipped with two robotic arms allowing it to interact with other objects.

The robot has two onboard computers and its software is based on ROS. As a research platform multiple SLAM algorithm were developed using the PR2 robot.

2.4 StockBot

StockBot presented in Figure 2.5 is an autonomous robot developed by PAL Robotics [8]. Its purpose is to take account of inventory in retail stores and warehouses using RFID technology.

StockBot is autonomous and its capable of mapping the environment automatically and moves around it without any help, detecting obstacles and avoid them. It can work at night without supervision, runs through corridors while detecting products, and crafts a report with not only a list of the items, but also their location. Its report includes a 3D map with the products location, which also gives a deeper understanding about customers behaviour. [9]

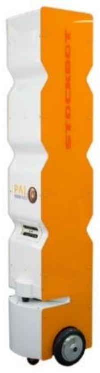


Figure 2.5 - StockBot robot. [8]

The StockBot robot software is based on ROS and uses a SLAM based solution for mapping and navigation. Users can interact with the robot using a tablet with REEMote application developed by PAL robotics presented in Figure 2.6.

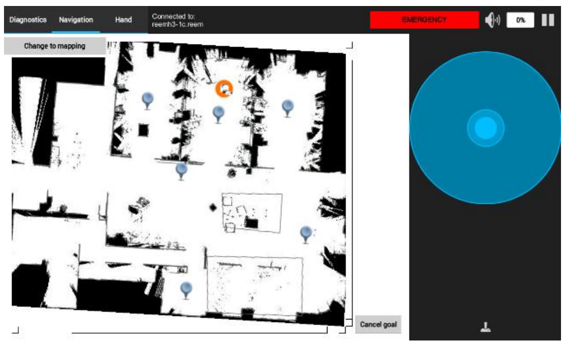


Figure 2.6 - PAL Robotics REEMote mobile application. [10]

2.5 Mi Robot Cleaner 2

The Mi Robot Cleaner 2 [11] presented in Figure 2.7 is an autonomous vacuum cleaning robot developed and manufactured by Xiaomi. Xiaomi is a Chinese electronics and software company founded in 2010 that has recently expanded its marker into the development of smart home robotic devices.

The Mi Robot 2 is equipped with a LIDAR sensor, multiple ultra-sonic sensors for collision and cliff detection and an IMU unit. It uses an ARM based computer to process data collected from its sensors and its equipped with Wi-Fi connectivity.

Figure 2.7 - Xiaomi Mi Robot Cleaner 2. [11]

The robot uses a proprietary SLAM solution for mapping and localization in the environment in real-time. To detect and locate its docking station an infrared based system is used, the robot automatically returns to its charging station after cleaning or when its battery is low (resuming the cleaning task after its recharged).

Users can interact with the robot using a mobile application presented in Figure 2.8. The mobile application allows the users to interact with the robot mapping system, define cleaning schedules, and check to robot status in real-time remotely.

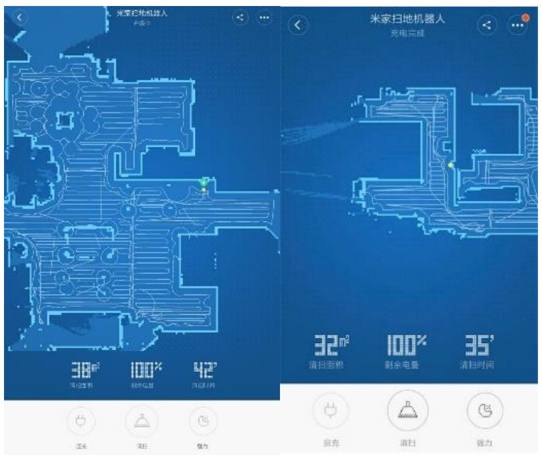


Figure 2.8 - Xiaomi Mi Robot 2 mobile application. [11]

3 Algorithms

In this chapter we describe theoretical SLAM concepts and present SLAM, localization, graph-optimization and visual features extraction algorithms. The SLAM algorithms described in this chapter were tested using the WiiGo robot and the results obtained are presented in Chapter 4.

3.1 SLAM

In this section we will, present theoretical concepts related to SLAM and analyze SLAM algorithms. SLAM can be described as the computational problem of constructing a map of unknown environment while simultaneously keeping track of an agent location inside of it. *"SLAM is a chicken-or-egg problem: a map is needed for localization and a pose estimate is needed for mapping"* [12].

A SLAM algorithm merges data from multiple rangefinder and positioning sensors to create a map and locate the robot. The pose estimation error and mapping error are correlated, if one of them fails the other one will fail as well. (e.g. bad alignment due to error may result in a bad pose estimative). Error propagation is one of hardest tasks to handle in a SLAM algorithm.

A map is a topological model representing an environment and there are multiple ways to represent maps. The type of map used is chosen based on system requirements and on the type of data available (e.g. grid map, 3D points clouds), but not all types of maps represent directly the environment geometry (e.g. landmark mapping). Figure 3.1 presents a comparison of different types of maps commonly used in SLAM systems.

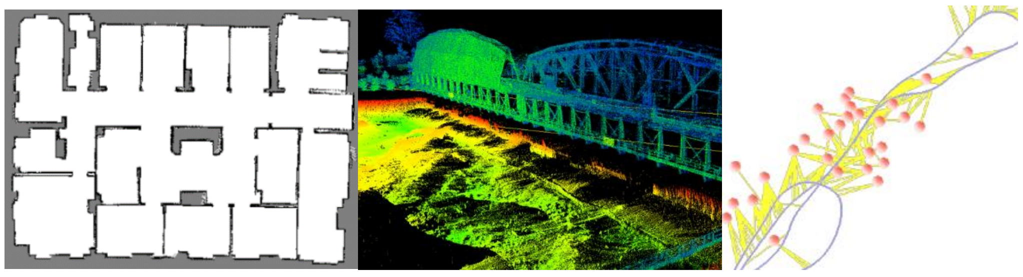


Figure 3.1 - Comparison of map types: grid map, point cloud and landmark based map.

State of the art SLAM systems are based on statistical models, they solve the mapping and localization problem as a correlated optimization problem. Most of them are based on graphs and are divided into two parts, the front-end and the back-end as presented in Figure 3.2.

The front-end is responsible for sensor data processing and correlation and storing data collected in a graph. The back-end is responsible for optimization. The optimization problem consists in reorganizing the data structure and data node relations to minimize the map error.

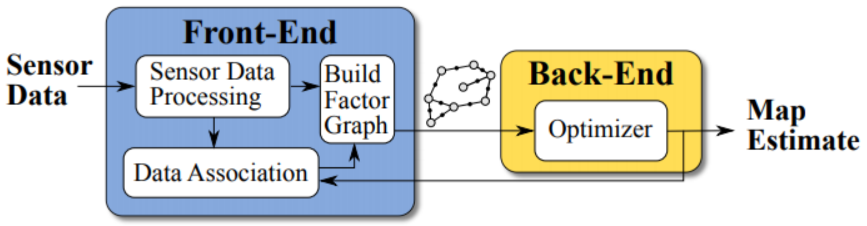


Figure 3.2 - State-of-the art SLAM systems architecture. [13]

Loop closure detection is the process involved in trying to find a match between the current and a previously visited locations in a SLAM system by recognizing an already mapped area. Figure 3.3 represents a case where loop closure detection can be used to correct odometry drift.

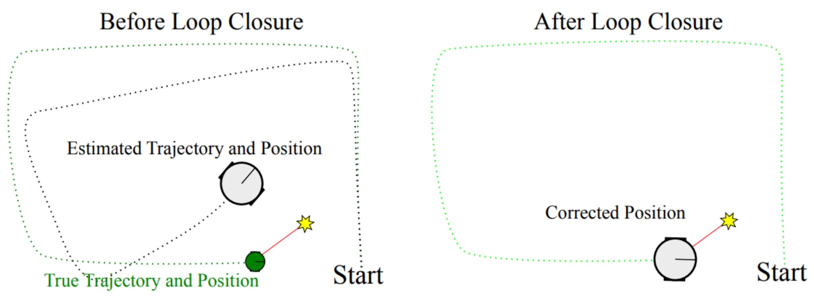


Figure 3.3 - Pose error correction using loop closure. [13]

When the map has multiple locations with similar appearance, it might be hard for the loop closure system to properly identify the correct matching place. If the system fails and a wrong loop closure is found the robot is transported to a wrong position. This is known as the “kidnapped robot” problem.

3.1.1 GMapping

GMapping [14] is a SLAM algorithm proposed by Giorgio Grisetti, Cyrill Stachniss and Wolfram Burgard that uses Rao-Blackwellized particle filters (RBPF) [15] to solve the SLAM problem.

Particle filters are a way to efficiently represent non-gaussian distributions. A particle represents a hypothesis. Each particle is tested, and a weight is attributed to it. Particles with a weight bellow a defined threshold are discarded. The hypothesis represented by a higher density of particles is accepted.

A RBPF is used to represent a potential trajectory of the robot and a map possibility. Each particle represents a new robot localization and new map possibility. The particle weight is calculated from the alignment of its own map with the current map status. Figure 3.4 represents visually how particles (in red) can be used to determine new possible positions for the robot.

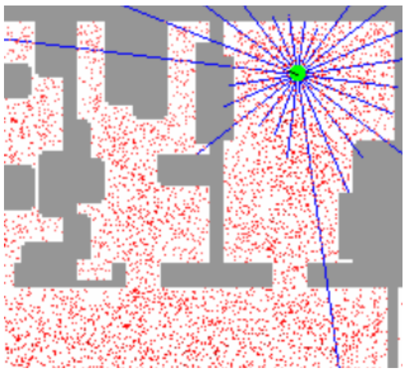


Figure 3.4 – SLAM problem using particle filters. [14]

One problem with this approach, is that since each particle carries its own map, the number of particles used must be limited to keep good computing performance. To solve this problem resampling can be applied, discarding particle with low weight and replacing them with particles with higher weight.

GMapping implements a solution that uses an adaptive resampling technique. To maintain a reasonable variety of particles and reduces the risk of particle depletion [16] GMapping uses data from previous distribution to propose more refined sets of new particles. Some examples of these distributions are represented in Figure 3.5.

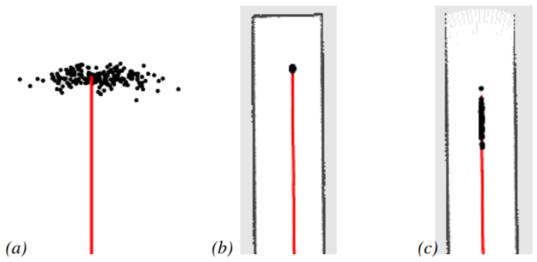


Figure 3.5 - Proposal distributions typically observed during mapping. [17]

3.1.2 Hector SLAM

Hector SLAM [18] is a SLAM algorithm developed by the team HECTOR (Heterogeneous Cooperating Team of Robots) in the University of Darmstadt to be used on Unmanned Ground Robots, Unmanned Surface Vehicles, handheld Mapping Devices and log data from quadrotor UAVs.

Hector SLAM contrary to other SLAM solutions is not divided into front-end and back-end. It works only as a front-end. It is not based on graphs and does not provide any global optimization technique. The approach used for scan matching is based on optimization of the alignment of laser points with current grid map using a Gauss-Newton based optimizer. New data from laser is written directly to the grid map after it is accepted.

Due to the discrete nature of grid maps there is limited precision. It does not allow the direct computation of interpolated values or derivatives for the optimization process. For this reason, an interpolation scheme allowing sub-grid cell accuracy through bilinear filtering is employed, as represented in Figure 3.6.

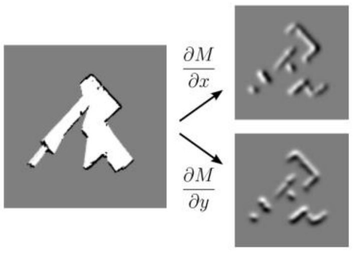

$\frac{\partial M}{\partial x}$
$\frac{\partial M}{\partial y}$

Figure 3.6 - Occupancy grid map and spatial derivatives. [18]

Hector SLAM does not provide any explicit loop closing ability. It relies on continuous data alignment and optimization.

To reduce the risk of getting stuck in a local minimum during its gradient ascend optimizer, it uses a multi-resolution map approach. Multiple maps are created with different resolutions organized in pyramid each level of the pyramid has half the resolution of the previous level. Figure 3.7 presents the results of multiresolution simultaneous mapping.

These maps are not obtained from down sampling, instead they are created updated independently. The multiple maps are compared to check their consistency across multiple scales.

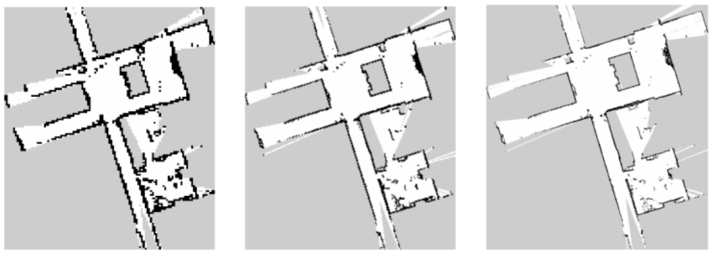


Figure 3.7 - Multiresolution representation of the map grid cells in Hector SLAM. [18]

3.1.3 Cartographer

Cartographer [19] is a graph-based SLAM solution developed by Wolfgang Hess, Damon Kohler, Holger Rapp and Daniel Andor at Google.

It provides real-time SLAM in 2D and 3D across multiple platforms and sensor configurations and organizes its data in submaps. Each submap corresponds to a node, multiple submaps are connected in a graph to obtain a map.

A submap is created from consecutive laser scans. A pose optimization algorithm is used to avoid error accumulation in these submaps. Each consecutive scan is matched against the submap grid using a non-linear optimization library called Ceres [20].

In Cartographer scans are stored into a probability grid. The main difference between a probability grid and the usual grid maps found in SLAM systems is that they store the probability of each cell being free. This allows the algorithm to have per cell probability values instead of having a value for the whole submap.

When a submap is finished no new scans will be inserted into it anymore. A scan matching is run to find loop closure and insert that new submap into the graph. All finished submaps and scans are considered for loop closure detection.

Loop closure optimization, like scan matching, is also formulated as a nonlinear least squares problem. It allows the addition of residual data previously discarded to improve the mapping result.

To guarantee that the system is capable of optimization in real-time a branch-and-bound approach is used. Various submap with multiple resolutions visible in Figure 3.8 are kept in memory to avoid recalculating every time optimization is performed.

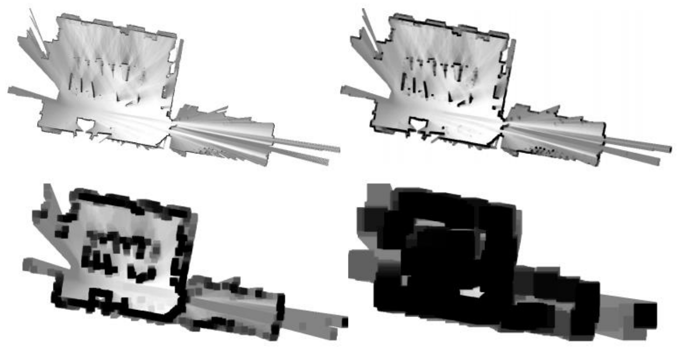


Figure 3.8 - Precomputed grids of size 1, 4, 16 and 64, in Cartographer. [19]

Figure 3.9 represents graphically the algorithm used by cartographer, including the steps applied for local and global matching. There are three main blocks in the algorithm. The input sensor data block that is responsible to receive data from the multiple sensors available. The local SLAM block is the front-end part of cartographer, it correlates close data and creates nodes to be added to the graph. The global SLAM block is the back-end of cartographer and it is used for to the solve graph optimization problem.

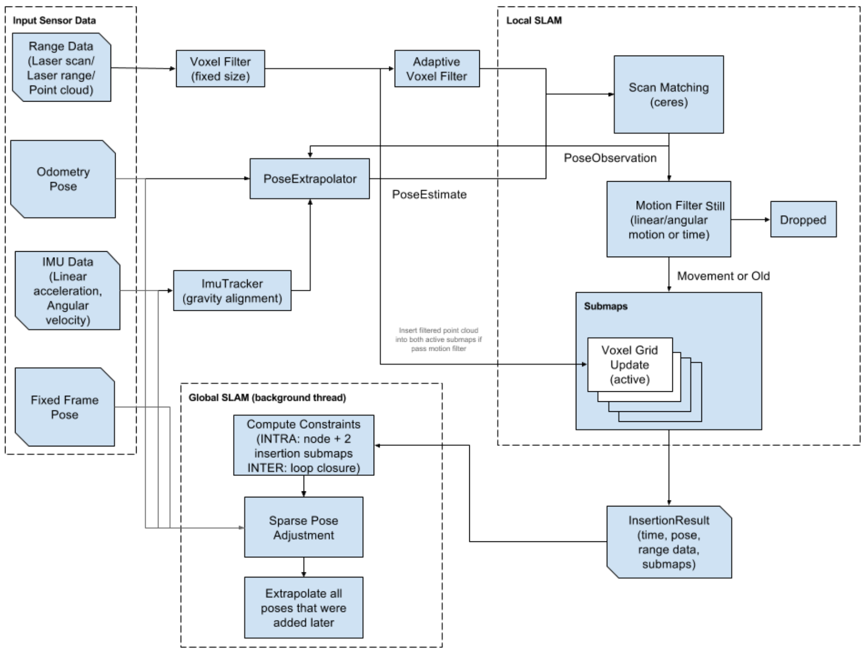

Input Sensor Data
Range Data (Laser scan/ Laser range/ Point cloud)
Odometry Pose
IMU Data (Linear acceleration, Angular velocity)
Fixed Frame Pose
Voxel Filter (fixed size)
Adaptive Voxel Filter
PoseExtrapolator
ImuTracker (gravity alignment)
PoseEstimate
Local SLAM
Scan Matching (ceres)
PoseObservation
Motion Filter Still (linear/angular motion or time)
Dropped
Movement or Old
Submaps
Voxel Grid Update (active)
Insert filtered point cloud into both active submaps if pass motion filter
Global SLAM (background thread)
Compute Constraints (INTRA: node + 2 insertion submaps INTER: loop closure)
Sparse Pose Adjustment
Extrapolate all poses that were added later
InsertionResult (time, pose, range data, submaps)

Figure 3.9 - Google cartographer system architecture. [21]

3.1.4 RTABMap

RTABMap (RealTime Appearance-Based Mapping) [22] is a graph-based SLAM framework developed by Mathieu Labbé and François Michaud in the IntroLab at the Université de Sherbrooke.

RTABMap uses an appearance-based approach for loop closure detection. It uses visual features extracted from RGB images to recognize already visited places. This combined with point-cloud or laser data allows the system to overcome the “kidnapping” problem more easily.

The loop closure detector uses a bag-of-words approach to create a signature of an image acquired to determinate how likely a new image comes from a previous location or a new location. When a loop closure hypothesis is accepted, a new constraint is added to the map graph. Then a graph optimizer minimizes the errors in the map by repositioning and realigning the graph nodes.

Figure 3.10 presents an overview RTABMap algorithm. The algorithm is composed of three main blocks. The sensor block receives and processes sensor data. The front-end applies odometry filtering and uses ICP algorithms to align sensor data and create graph nodes. The back end is responsible to add new nodes to the graph and apply graph optimization.

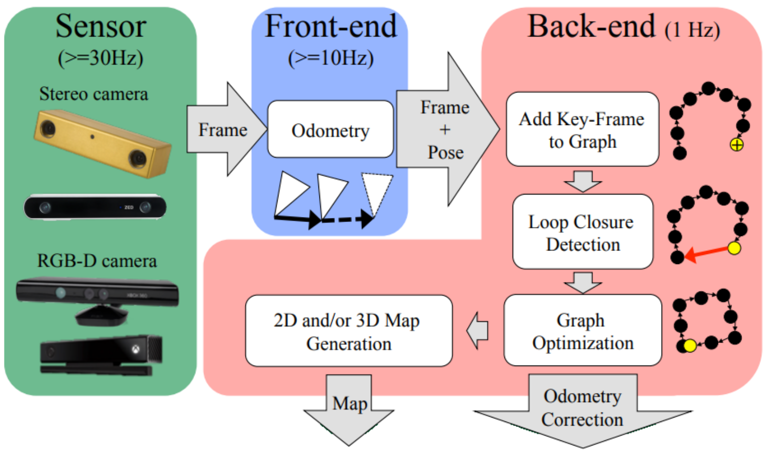


Figure 3.10 - RTABMap data processing pipeline. [23]

The amount of time required to process new observations increases with the size of the internal map which may affect real-time processing. RTABMap uses a memory management solution to limit the number of locations used for loop closure detection and graph optimization, so that real-time constraints on large-scale environments are always respected.

The memory management approach used (presented in Figure 3.11), consists of keeping the most recent and frequently observed locations in the robot Working Memory (WM), and transferring old ones into a Long-Term Memory (LTM). When a match is found between the current location and one location stored in WM, the associated locations (related locations) stored in LTM can be loaded and updated. To avoid loop closure detection on locations that have just been visited there is a Short-Term Memory (STM) that stores the most recent data. Data stored in the STM will eventually be moved into the WM.

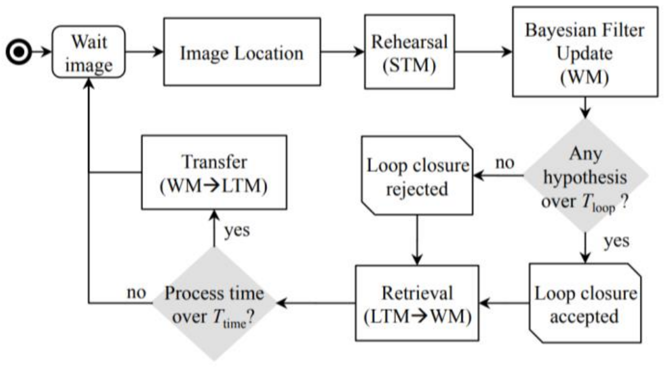


Figure 3.11 - Memory management in RTABMap. [22]

RTABMap uses an abstract approach over its software architecture. It defines interfaces for its front-end (e.g. ICP alignment, visual feature extraction) and back-end (e.g. graph optimization) and adapts existing solutions (e.g. G2O, GTSAM, TORO) to be integrated using these interfaces. This approach allows for the usage of different solutions over the same data structure making it possible to easily extended the framework to use new algorithms.

During the SLAM process, RTABMap stores all information and graph structure into a persistent database that can be reloaded later for multi-session mapping. Figure 3.12 represents a bigger map build from five smaller maps. It was created by finding loop closures to connect the maps.

"When turned on, a robot does not know its relative position to a map previously created, making it impossible to plan a path to a previously visited location. A solution is to have the robot localize itself in a previously-built map before initiating mapping. This solution has the advantage of always using the same referential, resulting in only one map is created across the sessions. However, the robot must start in a portion already mapped of the environment." [24]

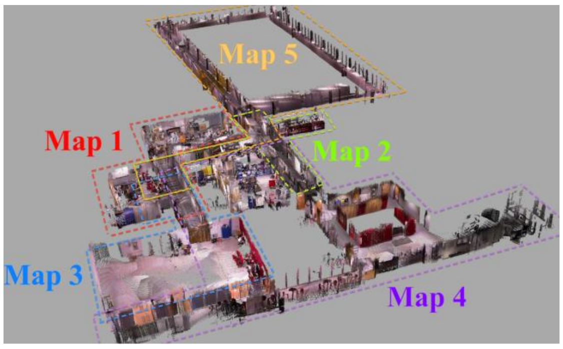


Figure 3.12 - Five online mapping sessions merged together automatically. [25]

3.2 Localization

In this section we will present a localization algorithm. Contrary to SLAM algorithms that need to constantly estimate a map and a position from received data, the localization algorithm already knows the map (it is provided to the localization algorithm) and only needs to estimate a localization.

One advantage of localization algorithms over the use of the SLAM solutions for localization is that due to the simpler nature of the task they generally require lower computing resources.

Most SLAM implementations do not consider the possibility of reloading data from previous sessions for localization, making the use of a localization algorithm the only solution possible to localize the robot after obtaining a map.

3.2.1 AMCL

AMCL (Adaptive Monte Carlo Localization) [26] is a ROS library for LIDAR based robot localization. It implements the adaptive Monte Carlo localization algorithm [27], which uses a particle filter to estimate the pose of a robot against a known map.

Given the map of the environment and a current visualization of the environment obtained using a LIDAR sensor the algorithm estimates a localization.

AMCL implementation uses an adaptive technique. It statically uses previous accepted particles to limit its proposal to a more refined set of new particles [16]. In Figure 3.13 it is possible to observe in red pose hypothesis (particles) proposed by AMCL and the position estimated.

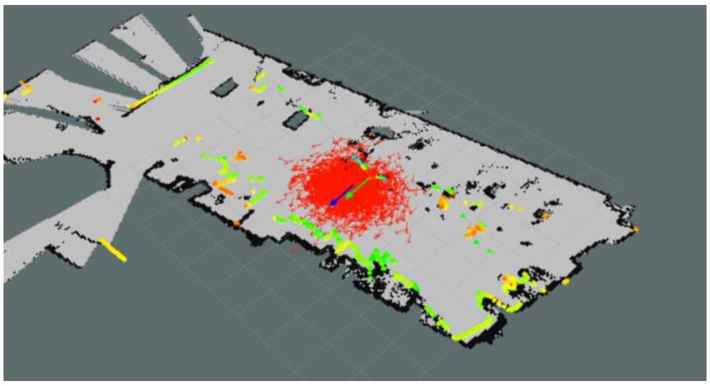


Figure 3.13 - AMCL localization. [28]

3.3 Graph optimization

Modern SLAM algorithms are based on graphs, the front-end of the algorithm collects data and generated nodes that represent a portion of the map. The back-end is responsible for optimizing globally the map by fixing misalignment and detecting loop closures.

The front-end can correlate close data and apply corrections to received data. In most cases the usage of the front-end algorithm could be used to optimize maps using all available data. However, this would result in a slow system. The use of dedicated global graph optimization solutions allows for faster results.

In this section we will present graph optimization algorithms. The algorithms presented in this section were tested with RTABMap and the results obtained are presented in Chapter 4.

3.3.1 TORO

TORO is an optimization algorithm for graph networks [29]. It is an improvement over the Olson's algorithm [30], that uses stochastic gradient descent optimization on an alternative state-space representation.

TORO applies a tree parameterization of the nodes in the graph that significantly improves the performance and enables a robot to cope with arbitrary network topologies.

Changes made to the Olson's algorithm allow TORO performance to be related the size of the mapped area and not to the length of the trajectory. Figure 3.14 shows an example of this optimization solution with a graph that has 1000 nodes, the runtime was bellow one second.

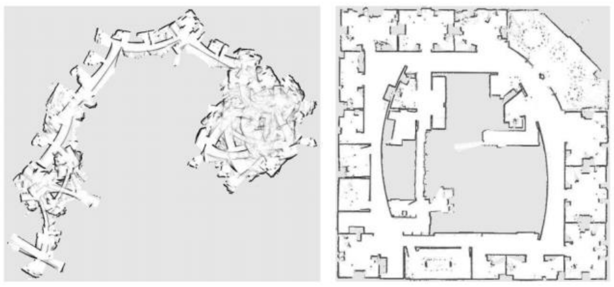


Figure 3.14 - Intel research lab map, before and after optimization with TORO. [29]

3.3.2 GTSAM

GTSAM [31] (Georgia tech smoothing and mapping library) is a library for optimization of nonlinear least squares problems. GTSAM is based on the Square Root SAM [32] technique and is an incremental version of the Dogleg [33] method.

Dogleg has excellent global convergence properties and is known to be considerably faster than both Gauss-Newton and Levenberg-Marquardt when applied to sparse least-squares problems. GTSAM approach, maintains the speed while providing superior robustness to objective function non-linearities.

3.3.3 G2O

g2o (general graph optimization) [34] is a framework for optimization of nonlinear least squares problems. It can be used not only for SLAM applications but also for other cases (e.g. bundle adjustment).

g2o has been designed to be easily extensible to a wide range of problems. A new problem can be specified by defining the error function and a procedure for applying a perturbation to the current solution.

Figure 3.15 represents the algorithm used by g2o. Only the boxes in gray need to be defined to integrate this solution. Structure and linear solver steps are abstracted, multiple solutions can be used.

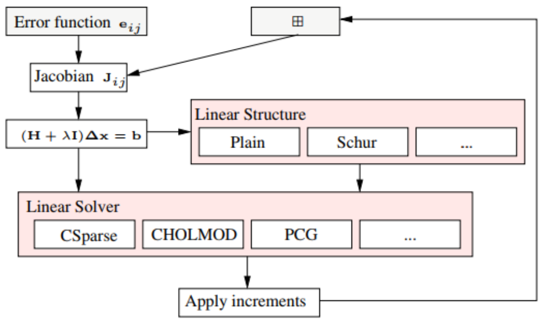

Error function $\mathbf{e}_{ij}$
$\boxplus$
Jacobian $\mathbf{J}_{ij}$
$(\mathbf{H} + \lambda\mathbf{I})\Delta\mathbf{x} = \mathbf{b}$
Linear Structure
Plain
Schur
...
Linear Solver
CSparse
CHOLMOD
PCG
...
Apply increments

Figure 3.15 - Overview of g2o framework algorithms. [34]

3.3.4 Vertigo

Vertigo [35] is an extension for G2O and GTSAM. It provides an implementation of switchable constraints and allows to solve pose graph SLAM problems despite the presence of false positive loop closure constraints.

G2O and GTSAM are based on least squares optimization and thus are not robust against outliers like data association errors and false positive loop closure detections.

Robot pose nodes are connected by odometry nodes when a loop closure is found. If a bad loop closure is found g2O and GTSAM do not implement a system to remove the bad connection between pose nodes.

Vertigo adds a new type of node, a switchable node represented in yellow in Figure 3.16. Depending on the value assigned to the switch node, the loop closure factor is switched on or off. It is activated or deactivated as part of the optimization process.

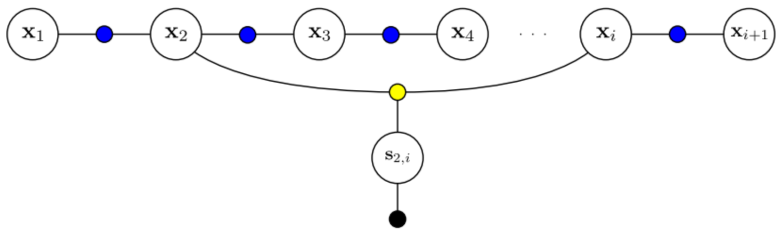


Figure 3.16 - Usage of switchable nodes to detect false positives in Vertigo. [35]

Every time the graph is optimized the switchable nodes are turned on and off to verify if the loop closure constrains were added correctly. This results in a more robust system capable of detecting false positive loop closures.

3.4 Visual features

In this section we will present multiple feature extraction algorithms. A feature is a vectorial pattern, based on which we can describe what we see on an image, in other words, a feature can be described as a fingerprint of a portion of an image.

A good feature should be invariant to distortion and lightning conditions. Multiple features extracted from an image can be compared to features from another image to determine if both images represent the same object or location.

Features can be used to visually identify already visited places in a SLAM algorithm and propose loop closures. Combined with classical SLAM loop closure detection, visual features can be used as a protection against false positives.

The algorithms presented in this section were tested in an object-detection scenario and the results are presented in Appendix A.

3.4.1 SIFT

SIFT [36] (Scale-Invariant Keypoints Features) features are represented by vectors and are invariant to image scale and rotation. They provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination.

SIFT features are extracted from locations at maxima and minima of the difference-of-Gaussian function. To reduce the number of variant features a threshold on minimum contrast is applied followed by an additional threshold on ratio of principal curvatures for each feature.

This simple approach results in a very fast and robust feature extraction algorithm. Figure 3.17 presents a result of SIFT feature extraction.

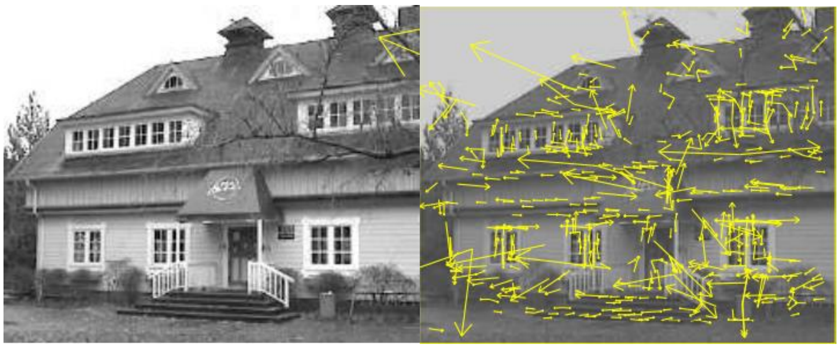


Figure 3.17 - Visual representation of SIFT features extracted from image. [36]

3.4.2 SURF

SURF [37] (Speeded Up Robust Features) is a feature descriptor algorithm that is an improvement over the SIFT algorithm.

SURF improves speed over SIFT by approximating its difference-of-Gaussian function with a Box Filter. One big advantage of this approximation is that, convolution with box filter can be easily calculated with the help of integral images. And it can be done in parallel for different scales. SURF relies on determinant of Hessian matrix for both scale and location.

To determine feature orientation SURF uses differential analysis in the neighborhood if the feature. This results in a faster algorithm with similar results to the SIFT algorithm. Figure 3.18 represents the result of SURF feature extraction.

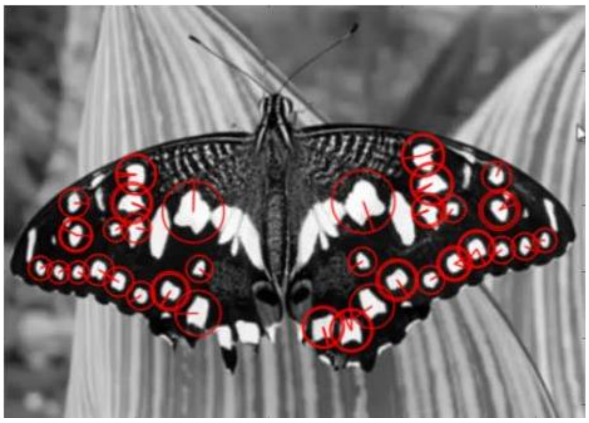


Figure 3.18 - Visual representation of SURF features extracted from image. [38]

3.4.3 ORB

ORB (Orientated FAST [39] and Rotated BRIEF [40]) [41] is a feature extraction algorithm that combines FAST corner detection and BRIEF features.

BRIEF (Binary Robust Independent Elementary Features) features uses binary strings as an efficient feature descriptor. It is highly discriminative even when using relatively few bits and can be computed using simple intensity difference tests using the Hamming distance, which is very efficient to compute, instead of the L2 norm as is usually done. As a result, BRIEF is very fast both to build and to match.

FAST and BRIEF are booth fast algorithms that complement one another. BRIEF is variant to rotation this is complement by a fast and accurate orientation component added in ORB using FAST.

This combination results in a much faster approach than the ones used in SURF and SIFT. Figure 3.19 shows the result of extraction and matching of ORB features between two images.

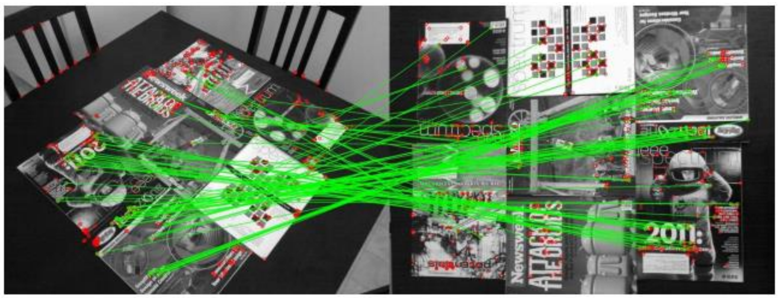


Figure 3.19 – Matching of ORB features extracted from images. [41]

3.4.4 KAZE

KAZE [42] features in contrast to other approaches that rely on the Gaussian scale space, is based on nonlinear scale spaces using additive operator splitting (AOS) techniques and variable conductance diffusion.

Similarly to SURF in order to obtain rotation invariant descriptors KAZE estimates the dominant orientation in a local neighbourhood centred at the key point location in a circular area.

AKAZE [43] (Accelerated KAZE) features are a more performant variant of KAZE. Instead of creating the nonlinear scale space it uses numerical schemes called Fast Explicit Diffusion (FED) that is faster to calculate.

3.4.5 BRISK

BRISK [44] (Binary Robust Invariant Scalable Keypoints) features rely on a FAST based detector in combination with the assembly of a bit-string descriptor from intensity comparisons retrieved by sampling each keypoint in the neighborhood.

The BRISK descriptor is composed as a binary string by concatenating the results of simple brightness comparison tests. Similarly to BRIEF it is possible to obtain descriptors matches using Hamming distance between descriptors.

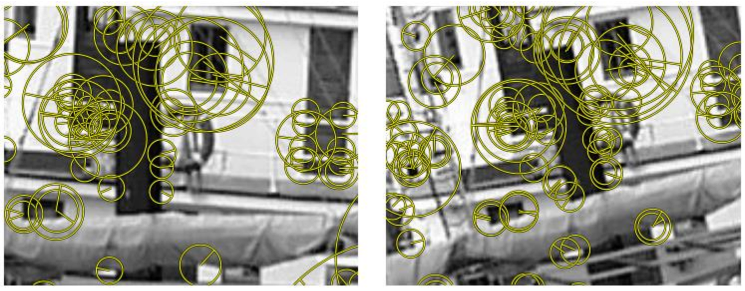


Figure 3.20 - Visual representation of BRISK features extracted from two images. [44]

4 Case study and Results

In this chapter we will present study of the current WiiGo robot hardware and software architecture, analyze two datasets recorded using the robot and present results obtained using the algorithms discussed in Chapter 3.

4.1 Hardware

The WiiGo robot presented in Figure 4.1 is 1,59m tall, the base is 0,59m x 0,75m and has a height of approximately 60kg. Its structure is made of metal and the outside covers are made of fiber glass.

The robot is divided in two main parts. The base where the computer, batteries, LIDAR, sonars and wheels are located. The head has the screen and three cameras. These two parts can be separated, making the robot more compact for shipping or maintenance when required.

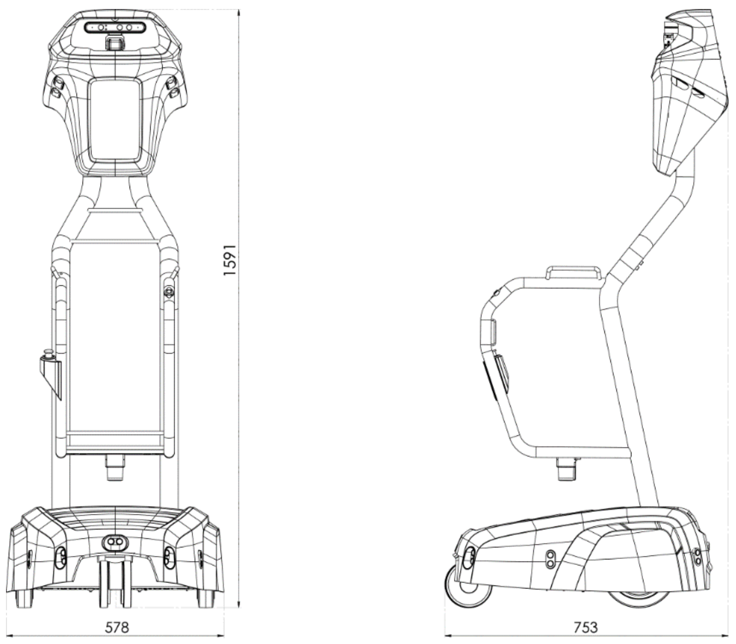


Figure 4.1 - WiiGo robot lateral and frontal views.

Figure 4.2 presents a diagram with all the hardware components present in the robot and all connections between sensors and the robot computer.

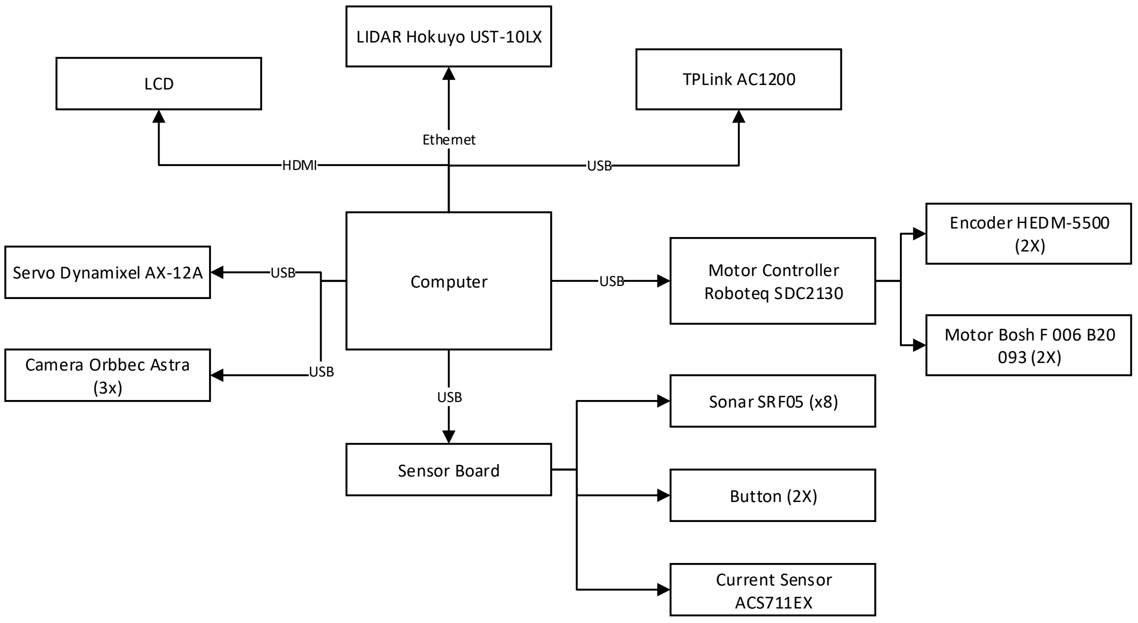


Figure 4.2 - WiiGo robot hardware diagram.

A complete detailed list of the all the sensors and electronics used in the robot is presented below:

- Computer
  - Intel Core i5 4460
  - ASUS H81I-PLUS
  - 8GB RAM DDR3
  - 120GB SSD
- Camera Orbbec Astra (3X)
- LIDAR Hokuyo UST-10LX
- Motor controller Roboteq SDC2130
- Encoder HEDM-5500 (2x)
- Motor Bosch F 006 B20 093 (2x)
- Sonar SRF05 (8x)
- Current sensor ACS711EX
- Batteries CSB EVX 12300 (2x)
- Servo motor Dynamixel AX-12A (2x)
- 10 Inch 1280x800 LCD
- TPLink 1200AC Wifi

4.1.1 Orbbec Astra camera

WiiGo is equipped with three RGBD cameras Orbbec Astra [45]. These cameras are capable of RGBD capture using an active depth system composed by a laser and a camera with an IR filter on top

These cameras have a dedicated RGB sensor and feature a pair of microphones for stereo audio capture. Figure 4.3 represents all the camera sensors.

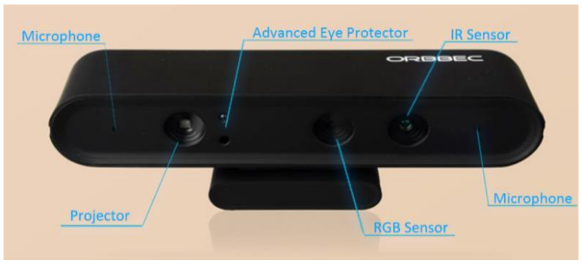


Figure 4.3 - Astra S camera hardware. [45]

The main camera mounted in the robot head and is mainly used for user detection and recognition this camera is attached to a stepper motor that can rotate.

The other two cameras and mounted on the robot head sides and are used for obstacle detection. Having these cameras mounted at a higher height allows them detect obstacles near the base of the robot where the laser and sonar sensors cannot detect them.

4.1.2 Hokuyo UST-10LX LIDAR

Hokuyo UST-10LX [46] presented in Figure 4.4 is a LIDAR made by Hokuyo. It is mounted in the robot based and is used to collect 2D information about the its surroundings.

This sensor can detect obstacles from up to 30m away, has an accuracy of 40mm and a scan angle of 270º with an angular resolution of 0.25º (1080 points).

Figure 4.4 - Hokuyo UST-10/20LX LIDAR. [46]

4.1.3 Roboteq SDC2130

The Roboteq SDC2130 board [47] presented Figure 4.5 is responsible for the control of the robot DC motors and reading of the encoders used to calculate odometry.

The board has 2 control channels and is capable of outputting up to 30V and 20A per channel. The controller's two motor channels can either be operated independently or mixed to set the direction and rotation of a vehicle by coordinating the motion of each motor.

Figure 4.5 - Roboteq SDC2130 board. [47]

4.1.4 HEDM-5500 Encoder

The HEDM-5500 [48] presented in Figure 4.6 is a 2-channel Optical Encoder, each encoder contains a lensed LED source, an integrated circuit with detectors and output circuitry and a code wheel which rotates between the emitter and detector IC, it uses a film code wheel allowing for resolution of 1024 CPR.

Figure 4.6 - HEDM-5500 Encoder. [48]

4.2 Software

In this section we will analyze the robot software and explore the libraries used in its implementation. The WiiGo robot software is based on ROS. The WiiGo software architecture is divided in six modules. Each module is responsible for a function and composed of multiple ROS nodes, and are organized as:

- Hardware
    - Robot sensors data acquisition, processes robot odometry, controls wheel motors and cameras servo motors.
    - Sensor fusion algorithms, to merge and align data from multiple sensors (e.g. LIDAR and camera depth information).
- Autonomous
    - Robot autonomous navigation system, path finding algorithms, path planning for obstacle avoidance.
- Mapping
    - Module responsible for environment mapping and obstacle detection.
- Following
    - User detection and recognition system, responsible for keeping track of the user estimating and estimating its position.
- Common
    - This is the main robot module responsible to taking decisions and controls all the other modules of the robot.
- Interface
    - Responsible for graphical user interface, and user interaction.

Figure 4.7 represents the robot software data flow. Each module is responsible of a specific function in the robot. The modular design allows for the robot software to be easily expanded, by adding modules with new functionalities.

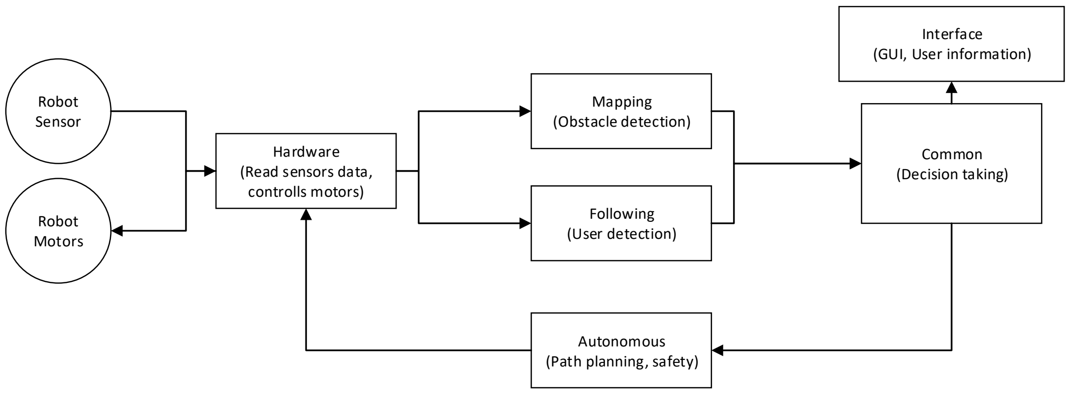


Figure 4.7 - Robot software modules data flow diagram.

4.2.1 ROS

ROS (Robot Operating System) [49] is a robotics open source framework focused on robotics applications. Developed initially by Willow Garage and currently maintained by the Open Source Robotics Foundation.

ROS is not an actual operating system, it runs on top of Linux based operative systems and provides tools for hardware abstraction, low-level device control, message-based communication, and package management.

It is being heavily adopted by the industry, the first commercially available robot was the PR2, released in 2010 by Willow Garage.

ROS uses a node-based approach, nodes are processes that perform computation. A system is typically comprised of multiple nodes that communicate with each other by passing messages. Node sends a message by publishing it to a given topic (identified by a string), another node that is interested in a certain kind of data will subscribe to the appropriate topic.

There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. Nodes connect to a master, the master is responsible for managing running nodes, announce existing topics and services.

Figure 4.8 presents an example of how the communication model used in ROS allow multiple nodes to access data published by another node.

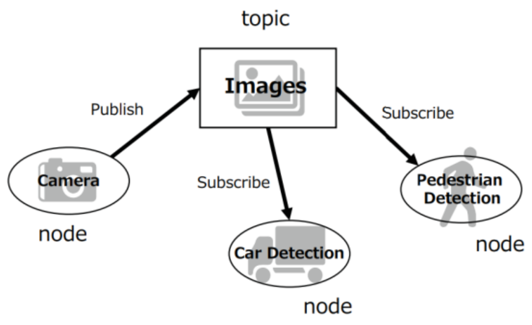


Figure 4.8 - Example of ROS publish/subscribe model. [49]

ROS uses a language agnostic approach, the ROS framework provides building tools that allow it to use a simple language-neutral interface definition language (IDL) to describe the messages sent between nodes declarations that are transformed into specific language specific declarations (e.g. C++, python, JavaScript) used into the development of nodes.

Messages (topics and services) are exchanged in IP layer [50] allowing easily for clustered computing. A single robot can use multiple computers for different tasks. This approach also allows easy development of networked solutions (e.g. collaborating robots, remote debugging systems, cloud processing systems), this approach also introduces some for local node communication.

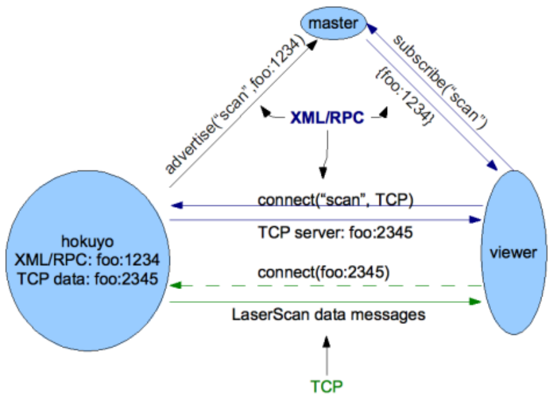


Figure 4.9 - ROS topic connection process. [50]

One of the biggest challenges in robotics applications is dealing with sensor fusion. To make this task easier the tf library was introduced in ROS. *"The tf library was designed to provide a standard way to keep track of coordinate frames and transform data within an entire system such that individual component users can be confident that the data is in the coordinate frame that they want without requiring knowledge of all the coordinate frames in the system."* [51].

Figure 4.10 shows the tf configuration used in the WiiGo robot, it is possible to observe that every sensor in the robot is represented by a tf.

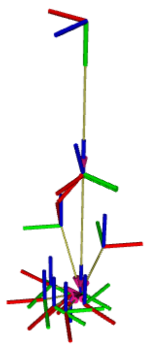


Figure 4.10 - A view of all the tf frames in WiiGo robot.

ROS is not a real-time development solution. But it is possible to use it alongside other real-time frameworks [52] (e.g. Xenomai).

In December 13th, 2017 ROS 2 [53] was released with support for Linux, Windows and MacOS, easier integration with real-time solutions, modular DDS (Data distribution system) support, and other improvements to the ROS ecosystem making it more modular and easier to integrate into different scenarios. Figure 4.11 presents a comparison between the system architecture used in ROS and ROS2.

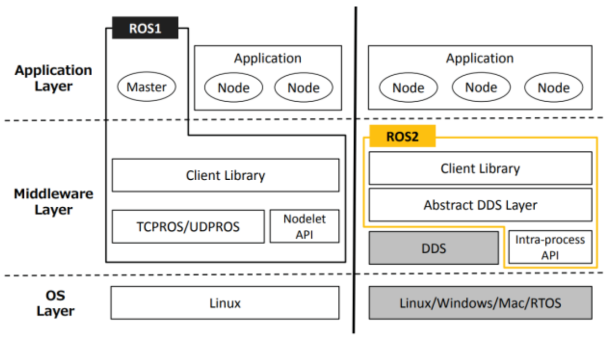


Figure 4.11 - Comparison between ROS1/ROS2 architecture. [53]

4.2.2 RViz

Rviz (ROS Visualization) is a visual debug tool for displaying sensor data and state information published from ROS nodes. It provides real-time visualizations of sensor values coming over ROS topics including camera data, sonar data, points clouds, etc. Figure 4.12 presents a screenshot of the graphical user interface of Rviz.

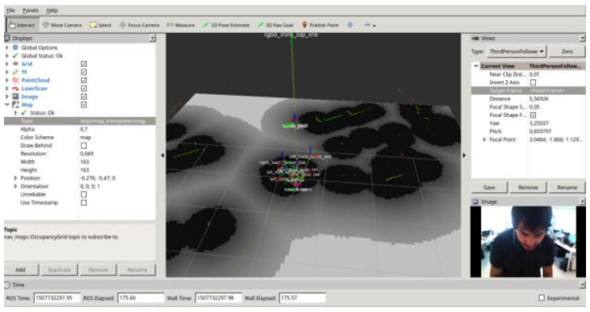


Figure 4.12 - RViz graphical user interface.

4.3 Datasets

In this section we will present the datasets created and used to test and evaluate the algorithms presented in Chapter 3. During the analysis of the robot software and hardware multiple datasets were collected. Two datasets were selected to be used for testing.

The first dataset represents a simple controlled environment inside the laboratory for witch a reference is known (presented in section 4.4). The second dataset represents a real word environment inside a supermarket where the WiiGo retail robot is typically used.

Each dataset is composed of two passes (the dataset was recorded two times). The objective is to have two different sets of data representing the same environment one used for training (mapping) and the other used for validation (localization).

4.3.1 Laboratory

The laboratory dataset was collected in the CEIIA building in Matosinhos, Portugal, were the Follow Inspiration company is located. It covers the 3rd floor of the building.

The environment in this dataset is mainly comprised of corridors and the laboratory where the WiiGo robot is developed. It was recorded using a gamepad, connected to the robot via Bluetooth. Figure 4.13 present an image of the WiiGo robot during the recording of this dataset.

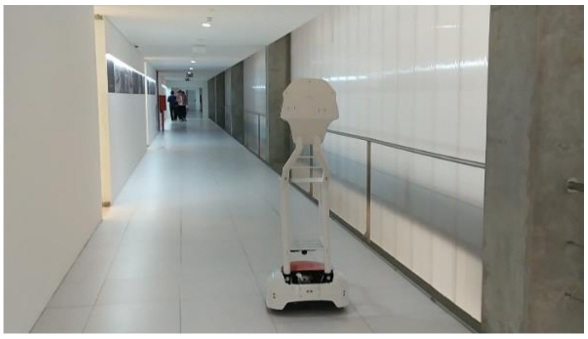


Figure 4.13 - WiiGo robot during laboratory dataset creation.

Figure 4.14 and Figure 4.15 represent the robot position estimated using odometry values. In the first pass good odometry values were obtained. In the second pass is possible to observe some drift in the robot odometry. This will allow to compare the mapping performance under bad odometry conditions.

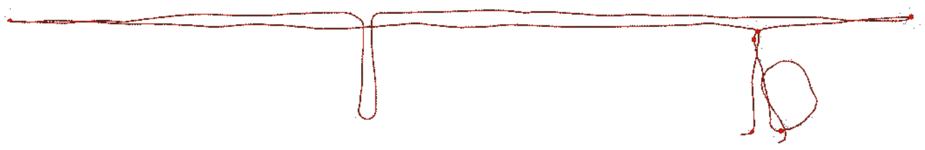


Figure 4.14 - Laboratory dataset first pass odometry.

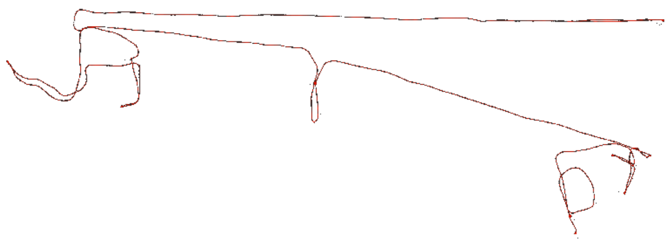


Figure 4.15 - Laboratory dataset second pass odometry.

4.3.2 Supermarket

This dataset was recorded in a Intermarche supermarket in Matosinhos, Portugal during a WiiGo testing session in a real environment.

In this dataset the two passes were recorded using different methods. The first pass covers most of the supermarket area and was recorded with the robot following a user. Figure 4.16 presents a visualization of the environment from the robot camera and LIDAR. It possible to observe that the presence of the user occupies a considerable porting of the camera image and is visible on the LIDAR sensor constantly. The presence of the user might impose a challenge for the SLAM algorithms.

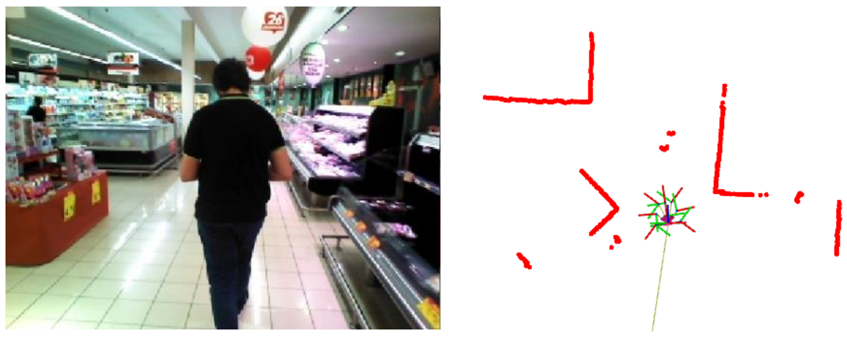


Figure 4.16 - Supermarket dataset first pass, camera and LIDAR.

The second pass was recorded using a gamepad to control the robot. Due to lack of memory during these tests the second pass only covers a smaller portion of the supermarket area.

The objective with the two passes in the supermarket data set is to validate if the presence of a user has any impact in the SLAM process.

Figure 4.17 and Figure 4.18 represent the robot position estimated using odometry values. It is possible to observe some drift in the robot odometry for both passes.

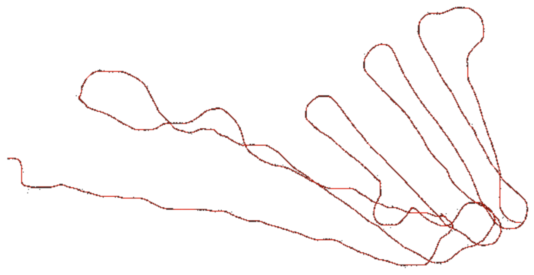


Figure 4.17 - Supermarket dataset first pass odometry.

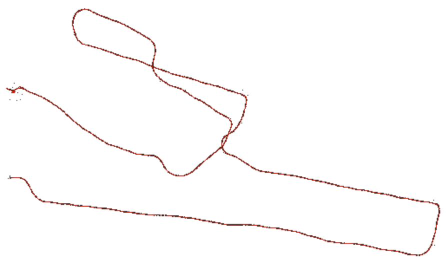


Figure 4.18 - Supermarket dataset second pass odometry.

4.4 SLAM Results

In this section we will test SLAM algorithms using the WiiGo robot and compare results obtained using the two datasets created. Results were evaluated based on their scale accuracy and alignment. Map scale was measured relatively to a ground-truth reference of the Laboratory dataset. Alignment was analyzed visually on the Supermarket dataset to verify if the algorithms were able to properly detect loop closures.

The maps obtained from each algorithm were saved into image files. A tool to capture the maps and store them in an image file was developed, it receives the map published by the SLAM solutions and stores them in image files.

A reference of the Laboratory was created from CAD drawings of the building. Since the CAD drawings do not include some element due to changes made to the building after its construction, one of the mapping results (obtained using Cartographer) was used to obtain the position of these missing elements.

Figure 4.19 represents the reference created. All elements inside rooms are not present. Walls made of glass were removed, because they are not visible by the robot LIDAR used by the SLAM algorithms.

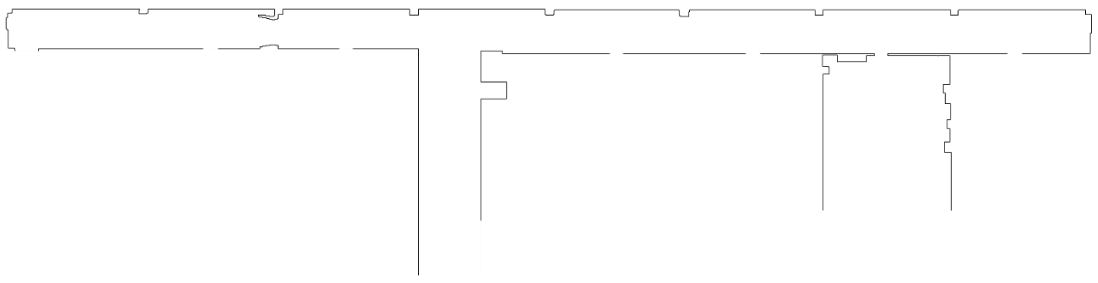


Figure 4.19 - Laboratory dataset reference, for mapping evaluation.

To compare the results obtained with the reference a comparison algorithm was created. It measures for each occupied pixel in the reference map the distance to the closest occupied pixel in the map result. The error is calculated from the average of all these distances.

Since different results of maps don't have the same size rotation and origin point it is required to first align the maps. The algorithm iteratively tries different position and rotation configurations.

Figure 4.20 represents the algorithm developed. It starts with the map as it is provided. The algorithm moves the map horizontally and vertically and rotates it in both directions testing the error for each configuration. The configuration with less error is used as starting point for the next iteration. The algorithm stops when it is not possible to improve the error result.

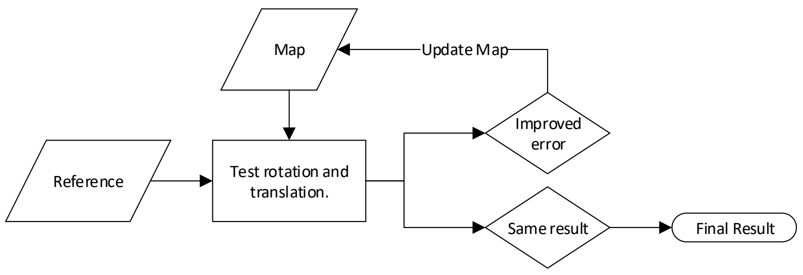


Figure 4.20 - Map comparison algorithm diagram.

Since the algorithm used is computationally expensive each map result was first manually aligned with the reference to reduce the number of iteration performed by the comparison program.

All tests were performed in a computer with a Core i7 4558U CPU, 8GB of RAM and a 250GB SATA 3 SSD. Datasets were read from the SSD to avoid any data access bottleneck.

To test performance of each solution tested environment was prepared. All programs were closed and only the solution in test and base components of the system were left running.

System Monitor installed by default in Linux Mint 18 was used to monitor CPU usage. Figure 4.21 shows the 60 seconds of CPU usage, it is possible to observe some minor fluctuation in the CPU usage, but it should not affect the measurements performed. The same dataset was used to test performance of all solutions.

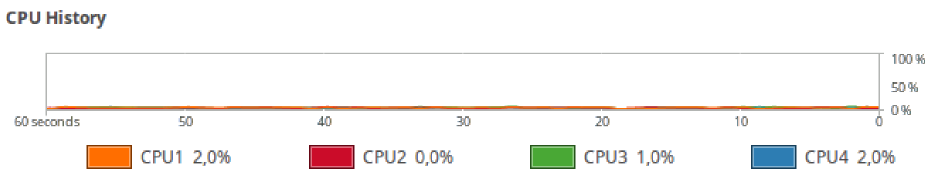


Figure 4.21 - System monitor baseline CPU usage for test environment.

4.4.1 GMapping

GMapping was installed, and a launch file was created using the default configuration values. Figure 4.22 represents the first test performed with this library, it is possible to observe bad alignment in the map.

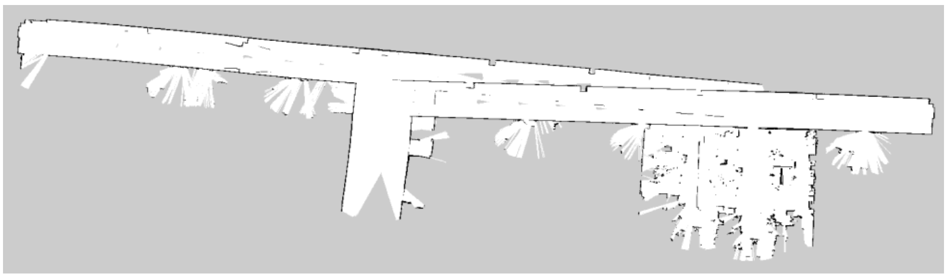


Figure 4.22 - Initial map result using Gmapping with default values.

After performing multiple tests, a final configuration was obtained. Table 4.1 contains the configuration values used for the final tests. These parameters were obtained empirically, default values were omitted from this table.

Parameter	Value
map_update_interval	1.0
maxUrange	10.0
maxRange	10.0
linearUpdate	0.1
angularUpdate	0.05

sigma	0.05
lstep	0.05
astep	0.05
ogain	3.0
iterations	5
delta	0.05

Table 4.1 - Final testing configuration for Gmapping.

Figure 4.23 and Figure 4.24 represent the results obtained for the Laboratory dataset where it is possible to observe that the misalignment present on initial test was fixed.

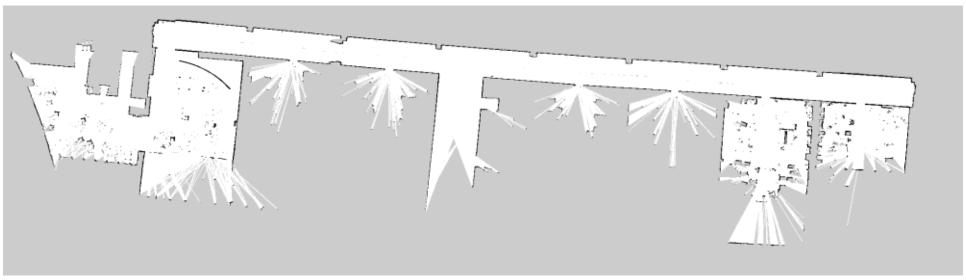


Figure 4.23 - Laboratory dataset first pass mapping result with Gmapping.

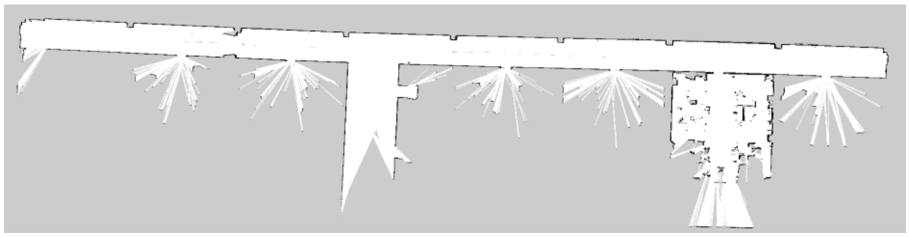


Figure 4.24 - Laboratory dataset second pass mapping result with Gmapping.

In the first pass (Figure 4.25) the solution was not able to properly alignment the initial section of the map but was able to correct most of the error present in the robot odometry and align most of the corridors. It is possible to observe that the presence of the user had no impact in the final map quality.

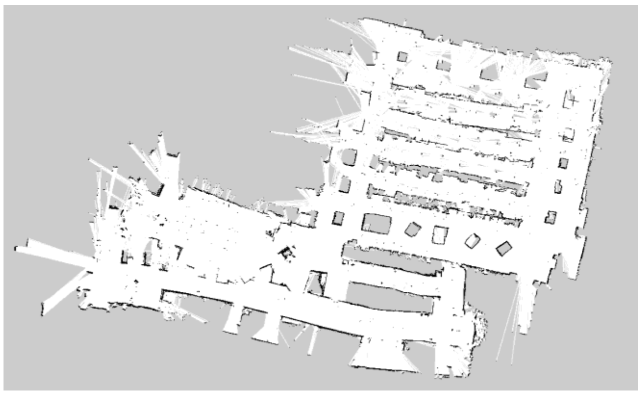


Figure 4.25 - Supermarket dataset first pass mapping result with Gmapping.

In the second pass represented in Figure 4.26. It was possible to obtain good map alignment but a bad loop closure was detected during the mapping process causing part of the map to be wrongly rotated 90 degrees.

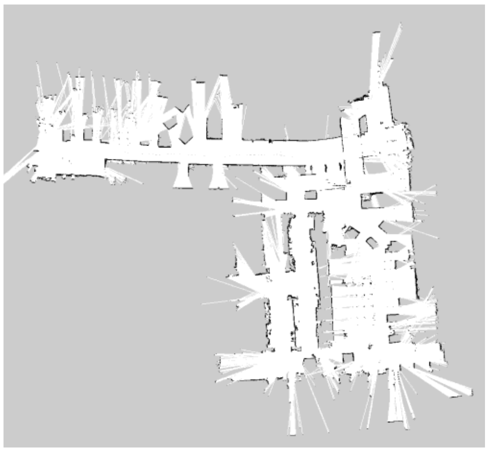


Figure 4.26 - Supermarket dataset second pass mapping result with Gmapping.

Figure 4.27, presents the CPU usage with Gmapping running. It is possible to observe that GMapping focuses most of its work on a single thread at a time.

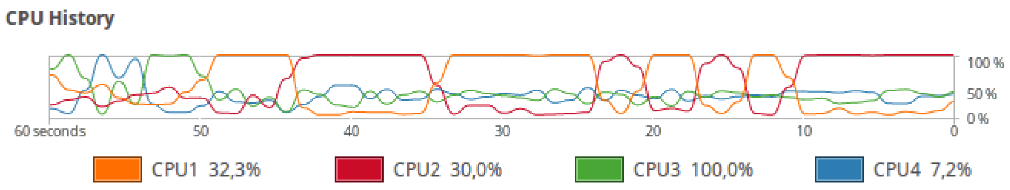


Figure 4.27 - System monitor CPU measurement with GMapping running.

4.4.2 RTABMap

The last version of the RTABMap framework was installed (V15.0.0), and configured with the G2O, GTSAM, TORO libraries and all visual feature extraction methods. Before testing any configuration, RTABMap was run using the default values to get a baseline measurement. In its default configuration RTABMap uses data from both the LIDAR and the RGBD camera, allow us to obtain not only a 2D grid map but also a 3D point cloud.

RTABMap was run using default configuration values to establish a baseline for comparison. In its default configuration RTABMap relies on depth information from the camera, uses TORO as a graph optimizer to create the grid map and GFTT/BRIEF features for loop closure detection. Figure 4.28 shows the result obtained it is possible to observe bad map alignment and lack of loop closure detection.

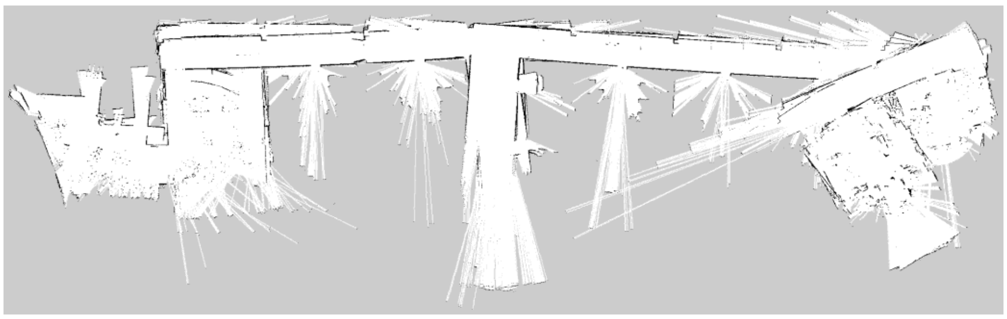


Figure 4.28 - Initial map result using RTABMap with default values.

When testing the default configuration with data from the first pass in the Supermarket dataset it was possible to observe that a trail of the user presence was appearing in the map, as it is possible to see in Figure 4.29.

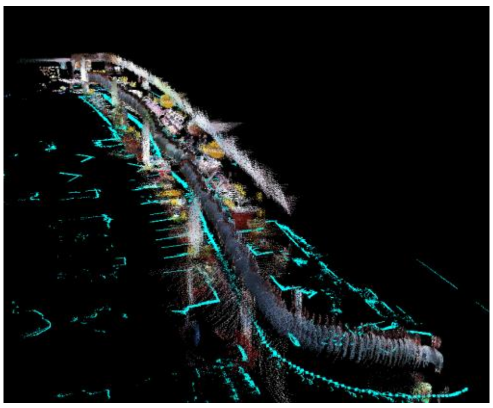


Figure 4.29 - User presence in map result obtained with RTABMap.

By analyzing the log published by RTABMap it was also possible to conclude that due to the presence of the user the system was always trying to calculate loop closure due to visual feature matching between frames were the user was present. Those loop closure hypotheses were rejected because the system could not find laser matches using ICP.

To remove the presence of the user from the data processed in RTABMap a ROS node was developed. The node receives data from the user detection system present in the WiiGo robot and the camera data, removes the area of image where the user is (by filling it with black pixels), and publishes a new version of the image without the area where the user was present.

User are removed from both the RGB and depth image. Figure 4.30 presents the result obtained.

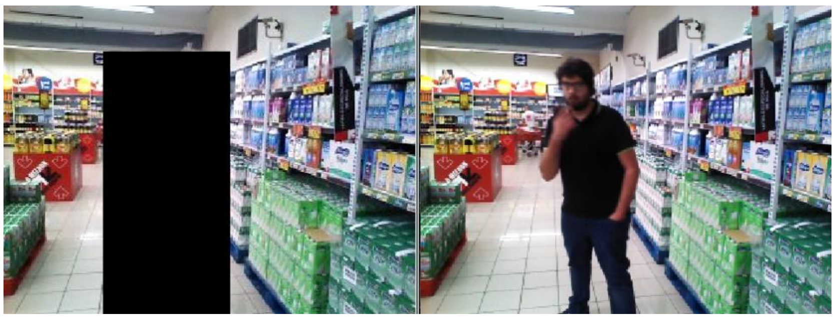


Figure 4.30 - Result of user removal from the camera image.

After removing the user, the system was retested. It is possible to observe still some presence of the user in the point cloud data. Loop closure detection system was working correctly after applying this change.

Figure 4.31 represents a close-up of the point cloud data processed by RTABMap, is possible to observe still some presence from the user.

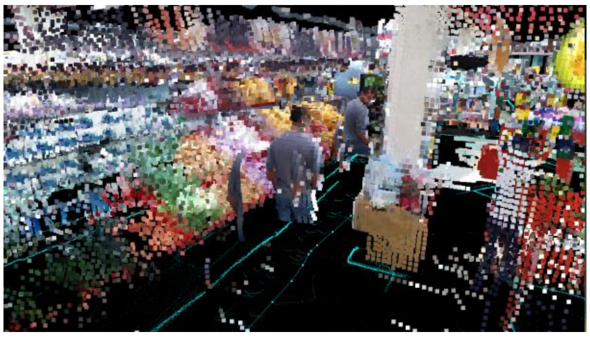


Figure 4.31 - Close-up of point cloud data obtained with RTABMap.

There are available on the ROS wiki page sample configurations for RTABMap. We tested the "Kinect + 2D laser + Odometry" configuration that was used as a base for the next tests. Figure 4.32 represent the result obtained, it is possible to observe some improvement over the first test.

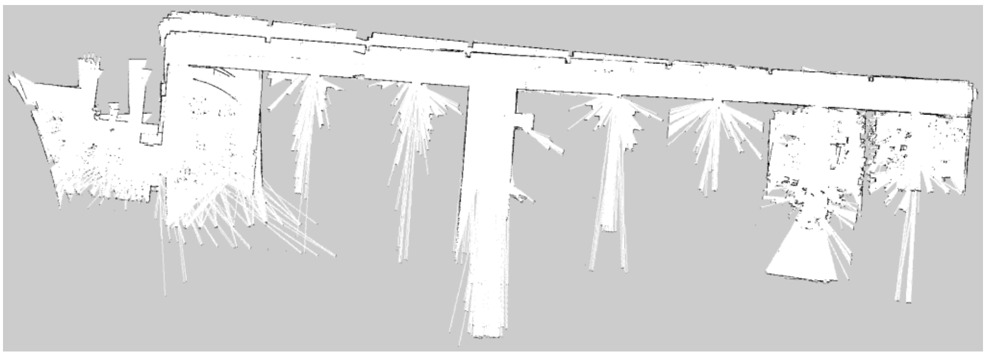


Figure 4.32 – Mapping sample configuration with RTABMap.

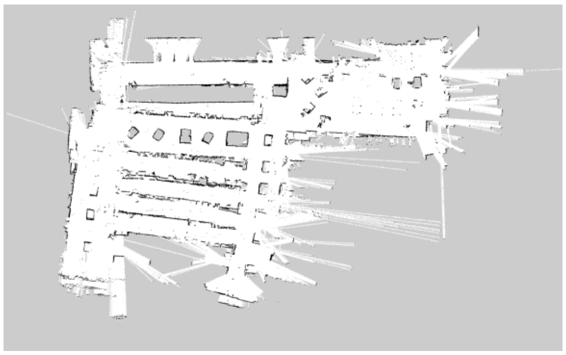
RTABMap uses visual features extracted from the RGB image to detect loop closures. There are multiple visual feature extraction algorithms available to use. To determine the best type of features for our use case, a test environment was created. The results obtained are described in Appendix A.

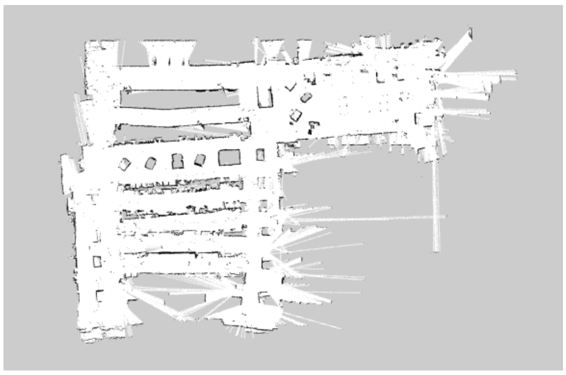
For the final configuration, SURF features were used. The results obtained showed that they provide the best results and were the best performing algorithm.

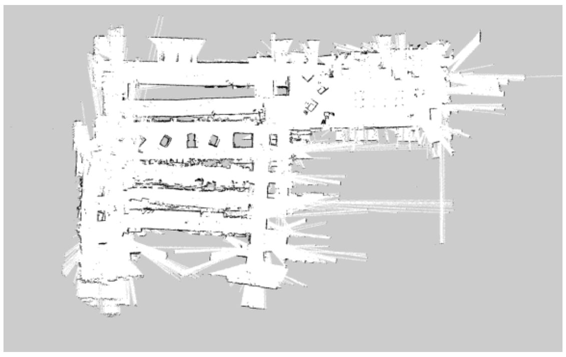
RTABMap integrates multiple optimization algorithms (TORO, G2O, GTSAM and VERTIGO). To determine the best graph optimization solution a set of tests were performed.

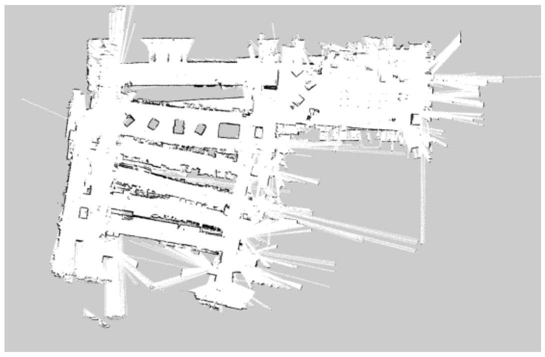
The graph optimization solution only affects global graph quality. The Supermarket first pass was used since it is the biggest dataset and it is the one that presents more loop closure problems.

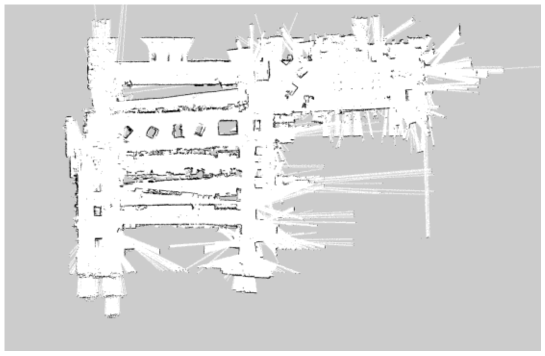
Figure 4.33 presents the results obtained for different configurations. It is possible to observe that G2O with Robust optimization obtained the best results all eight corridors of the supermarket can be distinguished without any noticeable overlapping between them.


a) G2O


b) G2O + Vertigo


c) G2O + Vertigo (Priors Ignored)


d) GTSAM


e) GTSAM + Vertigo

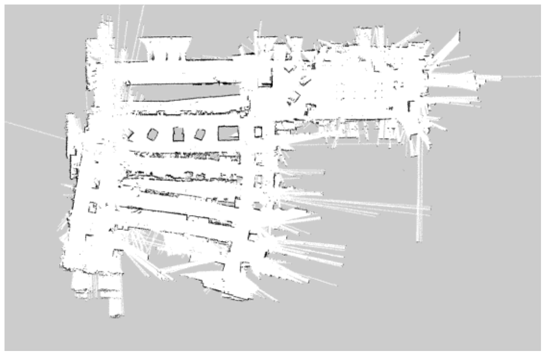

f) TORO

Figure 4.33 - Comparison of TORO, GTSAM, G2O and Vertigo in RTABMap.

Table 4.2 presents the final configuration used for final testing. These parameters were obtained empirically, default values were omitted from this table.

Category	Parameter	Value
RTABMap	DetectionRate	10
	TimeThr	0
	MemoryThr	0
RGBD	ProximityBySpace	true
	OptimizeFromGraphEnd	false
	OptimizeMaxError	0
Optimizer	Strategy	1 (G2O)
	Iterations	60
	Robust	false
	PriorsIgnored	false
	VarianceIgnored	false
Reg	Strategy	1 (ICP Only)
	Force3DoF	true
Vis	BundleAdjustment	0 (Disabled)
	FeatureType	0 (SURF)
	CorNNType	1 (kNNFlannKdTree)
	InlierDistance	0.1
	MinInliers	4
SURF	HessianThreshold	80
	GpuVersion	false
	Extended	false
ICP	MaxRotation	1.0
	MaxTranslation	1.0
	Iterations	80
	CorrespondenceRatio	0.1
	MaxCorrespondenceDistance	0.05
Odom	FilteringStrategy	1 (Kalman Filter)
	AlignWithGround	false
GridGlobal	FullUpdate	true
Mem	STMSize	10
	RehearsalSimilarity	0.45

Table 4.2 - Parameters used for testing with RTABMap.

Figure 4.34 and Figure 4.35 show the results obtained for the Laboratory dataset. The results obtained for these datasets where reasonable, the maps are well constructed. But it is possible to observe inferior laser data alignment when compared to previous solutions tested.

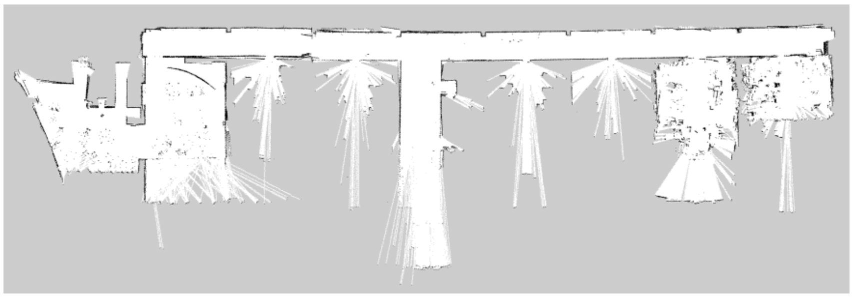


Figure 4.34 - Laboratory dataset first pass mapping result with RTABMap.

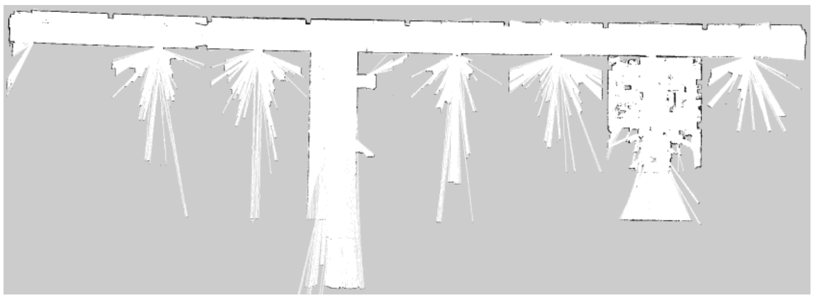


Figure 4.35 - Laboratory dataset second pass mapping result with RTABMap.

Figure 4.36 and Figure 4.37 show the results obtained for the Supermarket dataset. It is possible to observe that this framework was able to completely recover from drift in the robot odometry. It is the first solution tested to be able to overcome this problem.

Similarly, to the results obtained with the Laboratory dataset it is possible to observe some noise in the map and some misalignment in parts of the map.

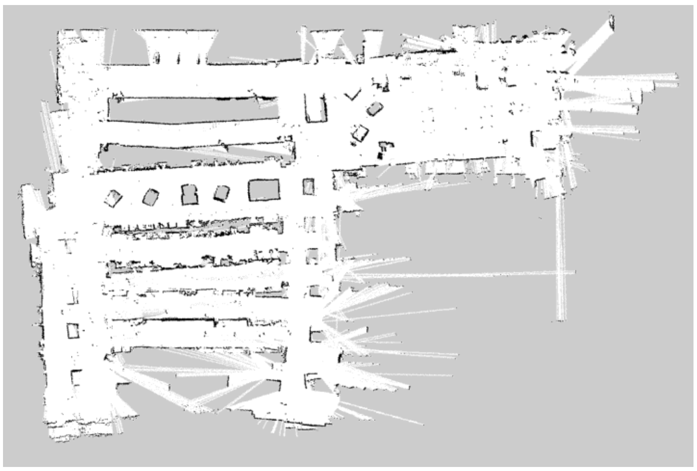


Figure 4.36 - Supermarket dataset first pass mapping result with RTABMap.

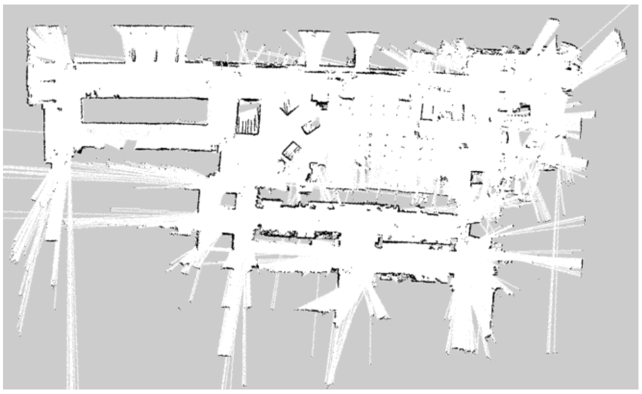


Figure 4.37 - Supermarket dataset second pass mapping result with RTABMap.

Figure 4.38 presents the CPU usage with RTABMap. It is possible to observe that this solution can leverage multi-core computing, it is possible to observe load being distributed by all cores. The average CPU usage with RTABMap is higher than other solutions tested.

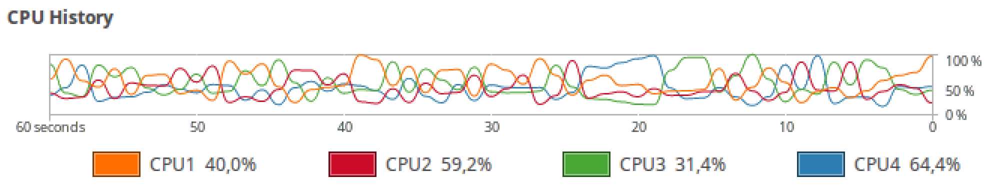


Figure 4.38 - System monitor CPU measurement with RTABMap running.

4.4.3 Cartographer

The last version of Cartographer was installed (V2.0) alongside with the corresponding ROS integration node and protocol-buffer based, map serialization tools.

A launch file was created using the default configuration values for 2D slam. Unlike other ROS packages instead of node parameters, cartographer uses a LUA program as a configuration file.

Figure 4.39 represents the results obtained in the Laboratory dataset with default configuration. It was possible to obtain excellent alignment results and good loop closure detection.

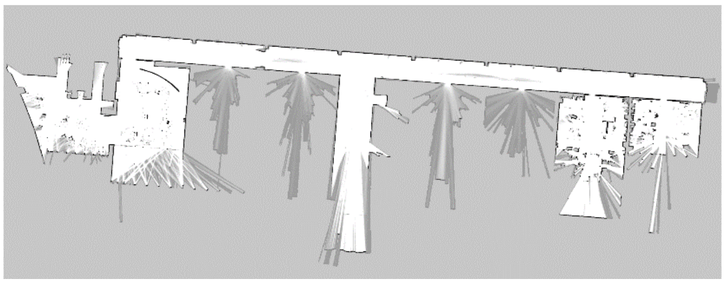


Figure 4.39 – Cartographer default configuration result on Laboratory dataset.

Figure 4.40 shows the results obtained in the Supermarket dataset with default configuration. It is possible to observe that bad loop closure detection occurred. All corridors of the supermarket were merged.

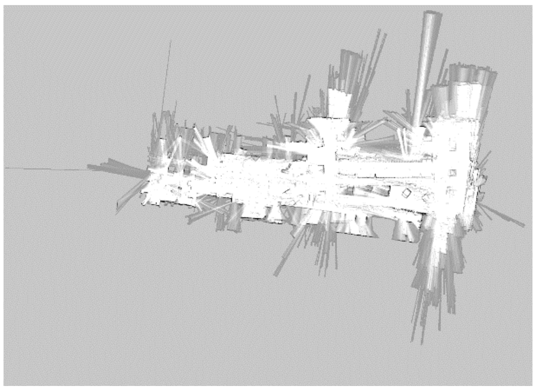


Figure 4.40 - Cartographer default configuration result on Supermarket.

Table 4.3 presents the configuration used with cartographer for the final tests. These parameters were obtained empirically, default values were omitted from this table.

Parameter	**Value**
pose_graph/optimize_every_n_nodes	20.0
pose_graph/constraint_builder/sampling_ratio	0.5
pose_graph/ constraint_builder/max_constraint_distance	20.0
pose_graph/min_score	0.6
pose_graph/ constraint_builder/global_localization_min_score	0.7
pose_graph/ constraint_builder/optimization_problem/huber_scale	10.0
pose_graph/global_sampling_ratio	0.1
submaps/range_data_inserter/ hit_probability	0.7
submaps/range_data_inserter/ miss_probability	0.4

Table 4.3 - Final testing configuration for Cartographer.

Figure 4.41 and Figure 4.42 present the results obtained for the Laboratory dataset. It is possible to observe good map alignment and no problems related to loop closure detection were found.

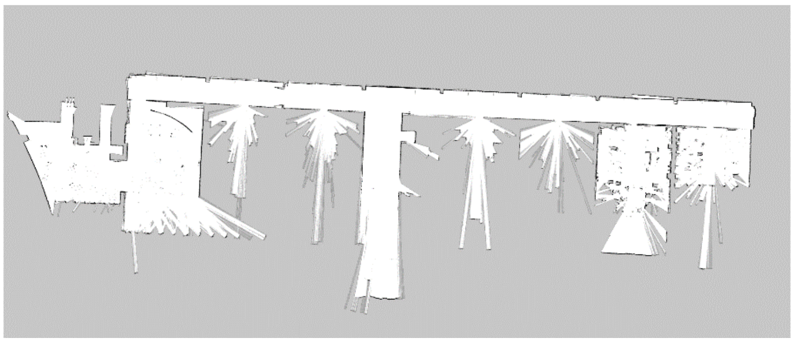


Figure 4.41 - Laboratory dataset first pass mapping result with Cartographer

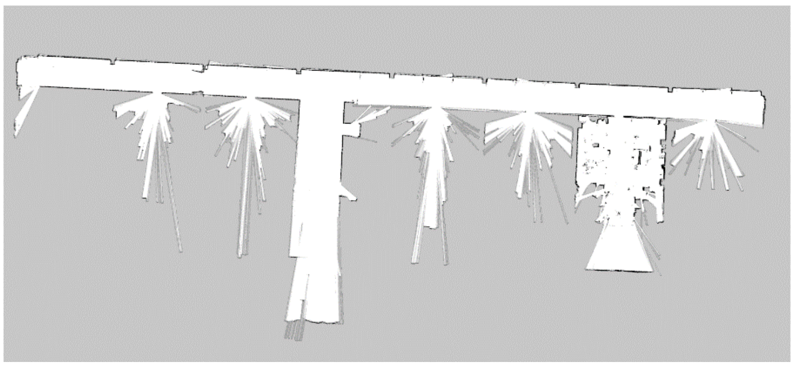


Figure 4.42 - Laboratory dataset second pass mapping result with Cartographer.

Figure 4.43 and Figure 4.44 represent the results obtained for the Supermarket dataset. Similarly to the results obtained for the Laboratory dataset it is possible to observe good laser alignment, but there are some problems with bad loop closure in both the first and second passes.

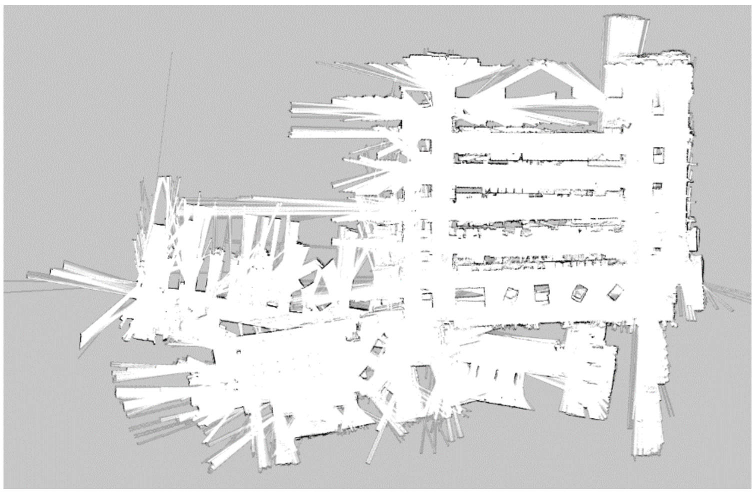


Figure 4.43 - Supermarket dataset first pass mapping result with Cartographer.

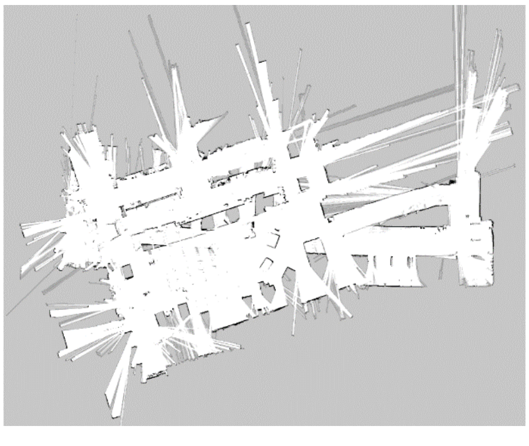


Figure 4.44 - Supermarket dataset second pass mapping result with Cartographer.

Figure 4.45 shows the CPU usage for cartographer. It is possible to observe that cartographer can distribute its processing by all computing cores available. It is also possible to observe very high CPU usage at the same frequency as the global map optimization is set on the configuration.

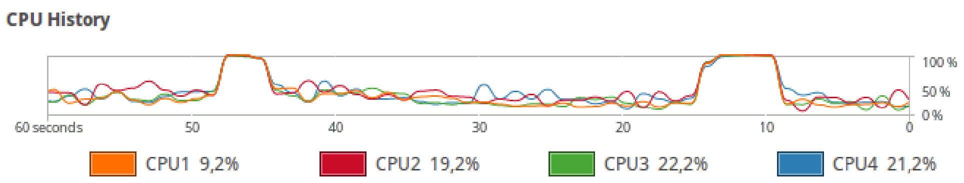


Figure 4.45 - System monitor CPU measurement with Cartographer running.

4.4.4 Hector SLAM

The last version of Hector SLAM was installed (V0.3.5), it uses information from the robot laser. An initial launch file was created using default values for everything except the map size that was set to 8192 pixels and resolution was kept at 0.05 meter/pixel. This values allow the map to store up to 409.6 square meters of area, which is enough for our datasets.

Figure 4.46 represents the first mapping result obtained where it is possible to observe good point alignment results.

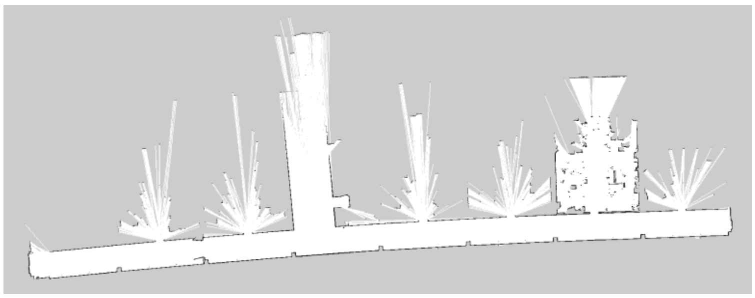


Figure 4.46 - Initial mapping result obtained with Hector SLAM.

It was possible to observe a situation during mapping in which Hector SLAM applied bad correction to the robot pose (represented in Figure 4.47). The robot was moving through a corridor and the pose was corrected as if the robot was static in the beginning of the corridor.

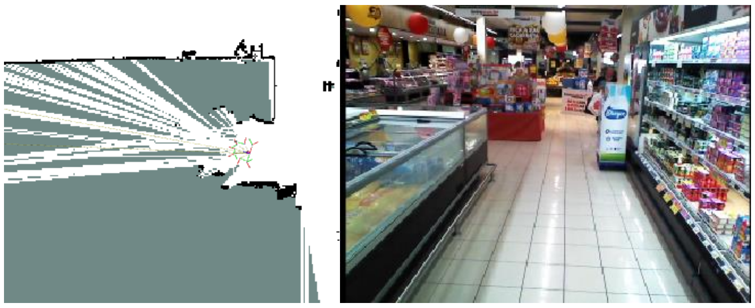


Figure 4.47 - Bad pose correction from similar laser information, in Hector SLAM.

Since Hector SLAM does not have any loop closure system it is not capable of recovering from drift after mapping an isolated section (e.g. room or corridor), this makes Hector SLAM dependent on good pose estimation from the robot sensors.

Figure 4.48 represents two cases where is clear the lack of loop closure detection, corridors are well aligned but on rotation points were the robot position has more drift it is possible to observe a breakage in the map.

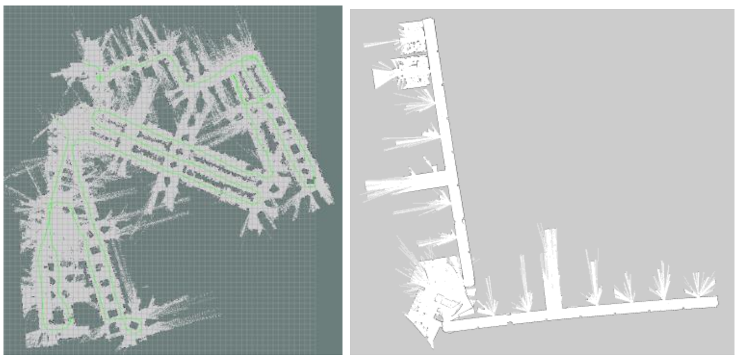


Figure 4.48 – Bad results with Hector SLAM, due to lack of loop closure detection.

After testing multiple times Hector SLAM with different configurations value, a final configuration was created. Table 4.4 presents the values used for the final tests. These parameters were obtained empirically, default values were omitted from this table.

Parameter	**Value**
map_pub_period	1.0
map_resolution	0.03
map_size	8192
map_multi_res_levels	3
update_factor_free	0.3
update_factor_occupied	0.9
map_update_distance_thresh	0.05
map_update_angle_thresh	0.05

Table 4.4 - Final testing configuration for Hector SLAM.

Figure 4.49 and Figure 4.50 show the final results obtained for the Laboratory dataset show an improvement over the tests initially performed. It is possible to observe an improvement in laser alignment in the first pass. But it is possible to observer that it was not able to merge different sections of the map properly in the second pass.

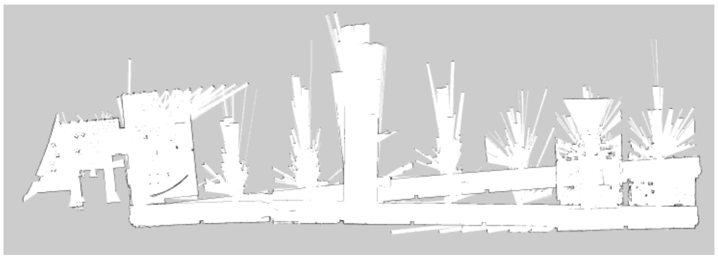


Figure 4.49 - Laboratory dataset first pass mapping result with Hector SLAM.

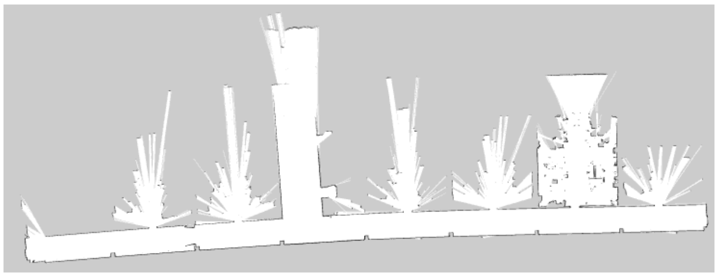


Figure 4.50 - Laboratory dataset second pass mapping result with Hector SLAM.

Figure 4.51 and Figure 4.52 show the final results obtained for the Supermarket dataset, it is possible to observe that the lack of loop closure leads to some breaks in the map that were not possible to fix for these datasets. The laser alignment for this solution was good and it is possible to visualize and distinguish small details like product boxes in the shelves of the corridors.

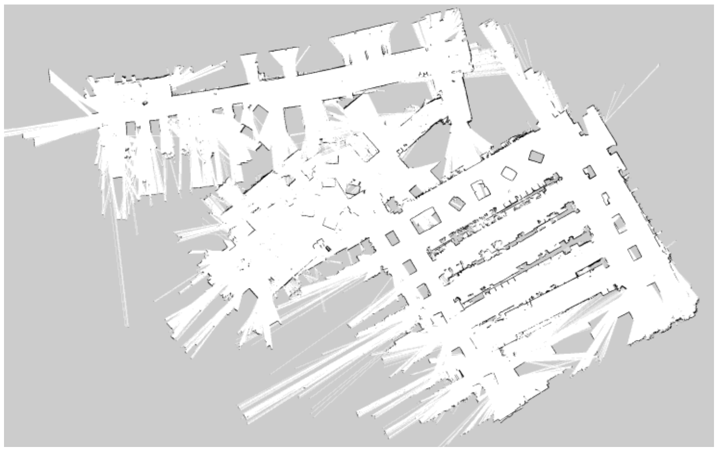


Figure 4.51 - Supermarket dataset first pass mapping result with Hector SLAM.

For the second pass, hector slam breaks the map into two separate maps. This happens when no correspondence is found.

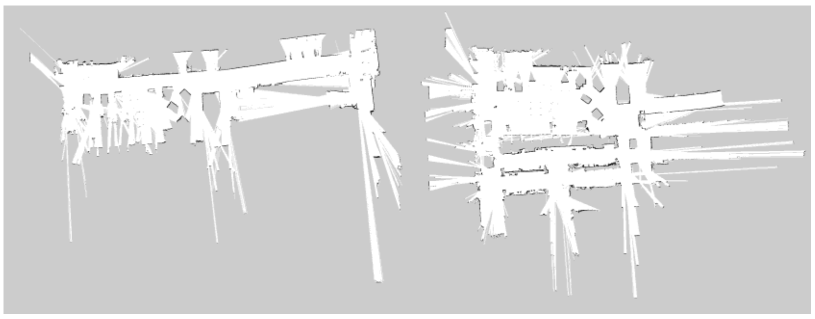


Figure 4.52 - Supermarket dataset second pass mapping result with Hector SLAM.

Figure 4.53 presents the CPU usage with Hector SLAM. It is possible to observe that it can use multiple threads to divide its work. The average CPU usage is low, and, since it uses a fixed map size. This can be an advantage for low memory devices since the map size is always limited to a defined level.

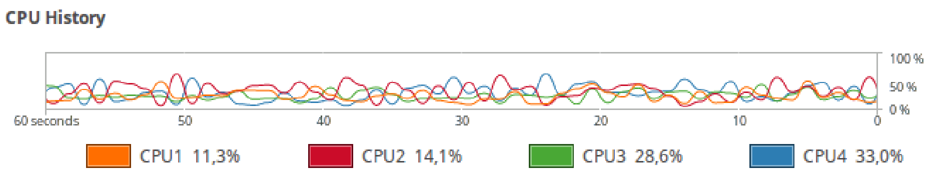


Figure 4.53 - System monitor CPU measurement with Hector SLAM running

4.4.5 Comparison

In this section we will compare results obtained for the multiple solutions tested based on their mapping results. We will compare booth map alignment results and loop closure detection performance.

To compare alignment quality the second pass of the Laboratory dataset was used. It was chosen because all solutions were able to obtain good results without loop closure problems.

Table 4.5 presents the results obtained. It is possible to observe that cartographer obtained the best average error value followed by RTABMap, and Hector SLAM obtained the worst result. These values only represent map scale accuracy and do not consider the amount of noise present in the map.

Algorithm	**Error (m)**
RTABMap	0.16
Hector SLAM	0.44
Cartographer	0.32
GMapping	0.37

Table 4.5 - Average mapping error obtained on the Laboratory dataset

Figure 4.54 represents the alignments obtained by the map comparison program relatively to the Laboratory reference. The green lines represent the correspondences that were measured to obtain the results presented. It possible to observe that cartographer and RTABMap obtained good results. Hector SLAM presents some distortion in the right side of the map. Gmapping presents some scale problems horizontally more noticeable in the end of corridors.

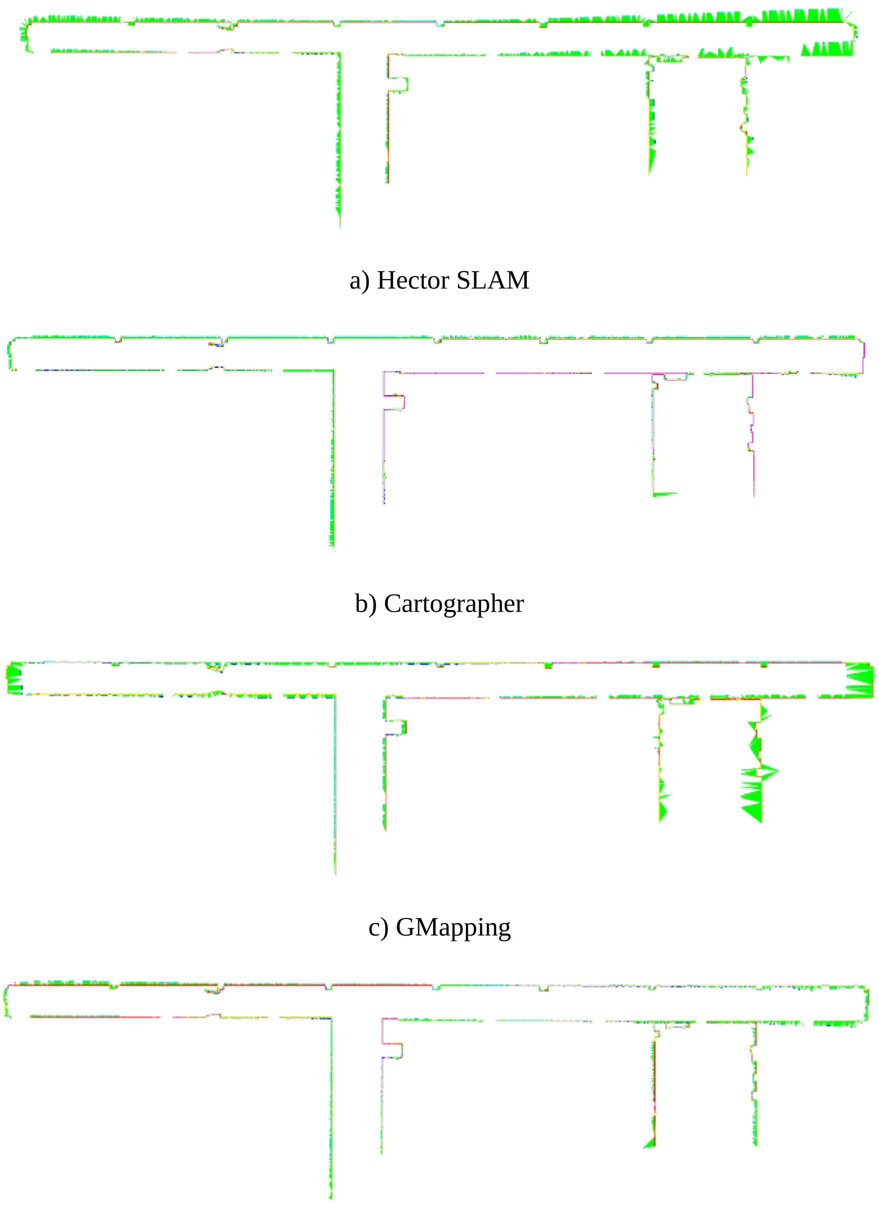
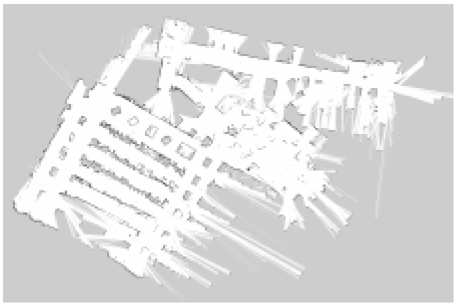
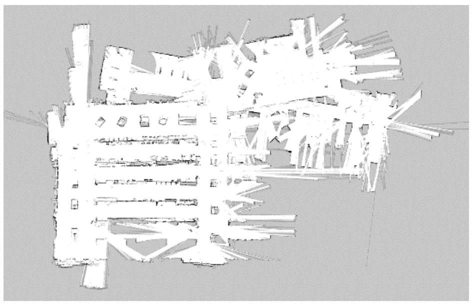


Figure 4.54 - Laboratory dataset mapping results alignment with reference for comparison.

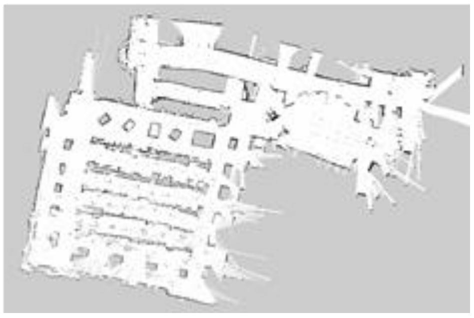
To compare loop closure performance the supermarket dataset was used. Figure 4.55 presents the results obtained for all tested solutions. Only RTABMap was able to obtain a complete map in a usable state and recover completely from odometry drift. All the other solutions failed in this dataset to properly detect and close all loop closures.



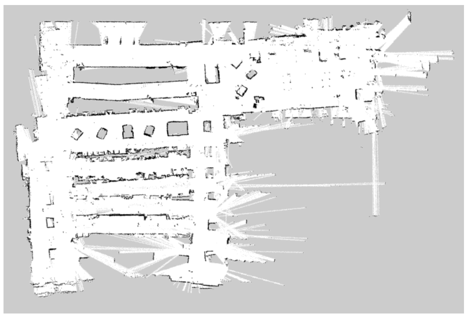
a) Hector SLAM



b) Cartographer



c) GMapping



d) RTABMap

Figure 4.55 - Loop close detection performance comparison for SLAM solutions tested.

4.5 Localization Results

In this section we will present the results obtained for localization. To evaluate the localization performance of each algorithm the maps from the first pass of each algorithm obtained using the SLAM algorithms were saved.

The second pass of each dataset was then loaded. Time was measured until the robot was able to obtain a correct localization estimative.

RTABMap already serializes its data into a database file and features a localization only mode. The files were reloaded and RTABMap was launched. For the other algorithms the grid maps stored were published using the map server tool available in ROS and AMCL was used to estimate the robot position in these maps.

AMCL uses the map origin as starting pose hypothesis. Due to low matching results with this initial pose, particle hypothesis spread out to all map. After some time, the solution converges to a position of higher hypothesis acceptance. Figure 4.56 demonstrates the particles used for hypothesis tests spreading in the laboratory dataset.

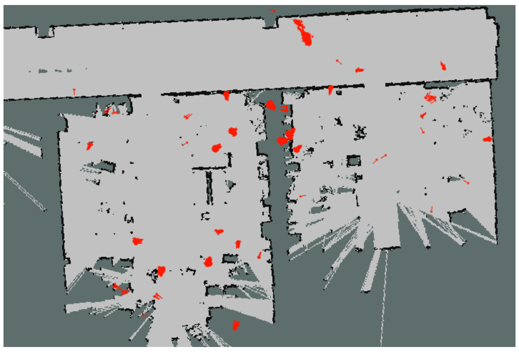


Figure 4.56 - AMCL particle vectors spreading.

During our tests AMCL was not able to estimate correct poses in any test scenario and converged to bad localization. Figure 4.57 presents one test scenario in the Supermarket dataset where is possible to observe bad particle convergence.

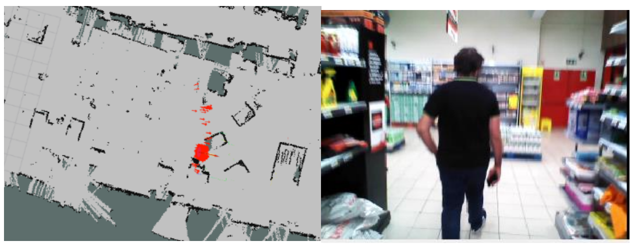


Figure 4.57 - Bad pose estimation with AMCL.

Contrary to AMCL, RTABMap only presents a pose estimation after obtaining a good match between the camera data and the visual feature data stored in the database. Sometimes this process can take some time.

Table 4.6 presents the results obtained, it is possible to observe that the algorithm can obtain an initial pose estimation in seconds. After obtaining an initial pose the algorithm keeps updating it in real time. During our tests it was not possible to observe any bad pose estimation.

Supermarket (s)		Laboratory (s)	
First Pass	Second Pass	First pass	Second Pass
1.57	4.24	3.83	34.95

Table 4.6 - Time to obtain localization with RTABMap.

The results obtained with RTABMap were possible due to the use of a feature-based loop closure detection system. Using this additional info extracted from RGB image it is possible to better recall the robot position in more complex environments like the supermarket environment tested.

5 Visual Markers

In this chapter we will present and explain a vision algorithm to detect, read and estimate the pose of marker relatively to the camera position. We propose an algorithm for the detection of aruco markers. Our approach uses the quadrilateral sum conjecture and analyzes the sum of the cosine of the internal angles to detect squares at larger distances.

Absolute pose estimation is a common problem in robotics. One possible solution to tackle this problem is the use of odometry information (coming from encoders and Inertial Units). However, this solution suffers significantly from drift error.

An alternative commonly used not only in robotics but also in areas like augmented reality is the usage of markers to estimate an absolute position of the camera. Fiducial Markers are heavily used for pose estimation in many applications from robotics to augmented reality.

A visual marker is something that can be easily distinguished from the rest of the ambient and have characteristics that allow it to be easily detected and identified.

One of the main problems found when experimenting already existing solutions (ROS Aruco [54] and Alvar [55]) was that they are not able to identify the markers from far away or under hard perspective distortion making it hard to discover the charging station.

5.1 Aruco markers

Aruco markers are geometrically square, they have a black border and an inner grid that is used to store a numeric identifier in binary code.

To identify the marker a dictionary is used [56]. The dictionary defines a set of rules used to calculate the marker identifier, perform validation and apply error correction.

We use the original aruco dictionary [54], that uses bits from the marker 2nd and 4th columns to store the marker identifier in natural binary code, the remaining bits are used for parity checking, Figure 5.1 represents the first four markers in this dictionary.

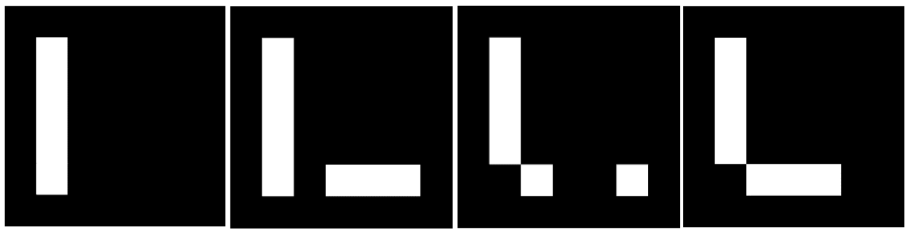


Figure 5.1 - Aruco markers with id's 0, 1, 2 and 3 respectively.

To validate the marker a signature matrix is used, each row of this matrix encodes a possibility of 2 bits. An aruco marker is valid only each its rows are equal to one of the rows of the signature matrix. This forces the marker to only have one valid rotation. Table 5.1 represents the signature matrix used in this dictionary.

Value	Data				
0	1	**0**	0	**0**	0
1	1	**0**	1	**1**	1
2	0	**1**	0	**0**	1
3	0	**1**	1	**1**	0

Table 5.1 - Signature matrix, used to validate aruco markers.

By analysing the signature matrix, its possibly to verify that it is not enough to guarantee that there is only one possible rotation for each marker, in the Figure 5.2 we can see the marker 1023 that is horizontally symmetric.

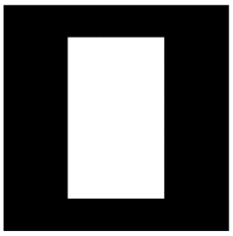


Figure 5.2 - Aruco Marker 1023.

5.2 Detection algorithm

The detection algorithm was implemented using the OpenCV library, since it provides a large set of image processing algorithms. Figure 5.3 shows the steps applied to detect and identify markers.

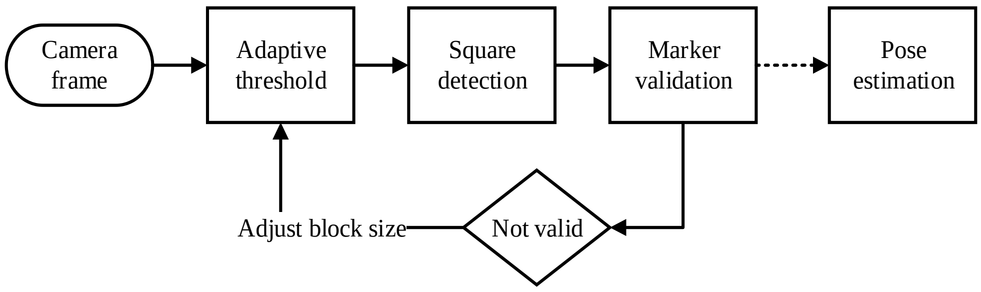


Figure 5.3 - Aruco detection algorithm diagram.

The algorithm starts by applying adaptive threshold [57] to the image, this algorithm consists in calculating for each pixel a threshold value using the histogram of its neighbourhood, its indicated for situations where it is possible to observe multiple lighting conditions. Figure 5.4 represents the results obtained from adaptive thresholding.

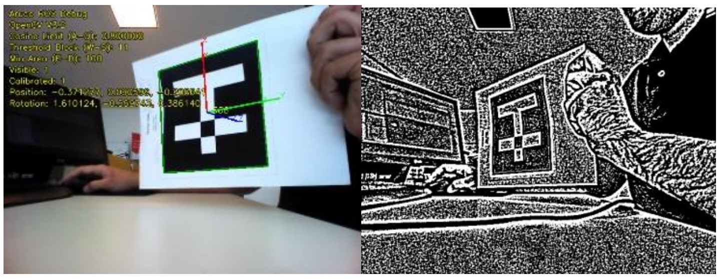

```
Aruco [illegible] Debug
OpenCV [illegible]
Cosine limit (Q-A): [illegible]
Threshold Block (W-S): 11
Min Area (F-D): 100
Visible: 1
Calibrated: 1
Position: -0.37[illegible], [illegible], -0.[illegible]
Rotation: 1.610124, -0.[illegible], [illegible].386140
```

Figure 5.4 - Adaptive threshold result.

To determine the threshold block (neighbourhood size), one block size is tested on each frame, the block size chosen is the medium size from all block sizes were the maximum number of markers were found, the block size is retested when there are no markers visible. In Figure 5.5 it is possible to observe two cases where different block sizes were chosen allowing the algorithm to adapt and detect marker with different size.

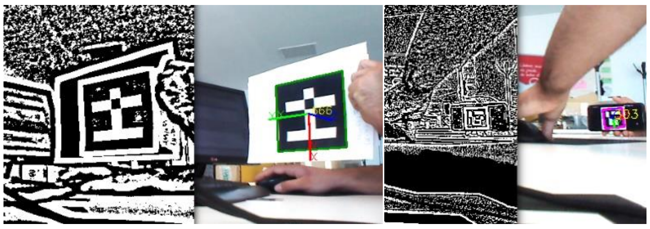


Figure 5.5 - Comparison of different threshold block sizes.

After threshold is applied to the image, we perform square detection, the algorithm starts by obtaining contours using a border-following algorithm [58], after obtaining contours the Douglas-Peucker algorithm [59] is used for contour simplification.

Using the previously detected contours, we use the Quadrilateral Sum Conjecture as a criterion to detect squares, the Quadrilateral Sum Conjecture tells us the sum of the angles in any convex quadrilateral is 360 degrees (Figure 5.6).

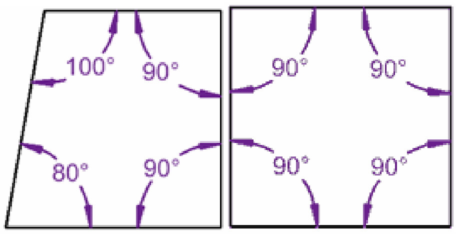


Figure 5.6 - Internal angles of convex quadrilaterals.

Even under perspective distortion a square is always a convex quadrilateral. This will be our first criteria, it is also possible to verify that the sum of the cosine of all inner angles equals in a convex quadrilateral is close to zero. Considering this our second criteria will be to make sure that the sum of inner angle between all corners is under a defined threshold (close to zero).

To filter noise a third criterion was added, all contours composing a geometry with an area bellow a defined threshold will be discarded.

These three criteria allow to properly filter squares even under heavy distortion from the contour list, the Figure 5.7 represents the obtained result for a maximum sum of cosine of 0.25 and a minimum area of 100px.

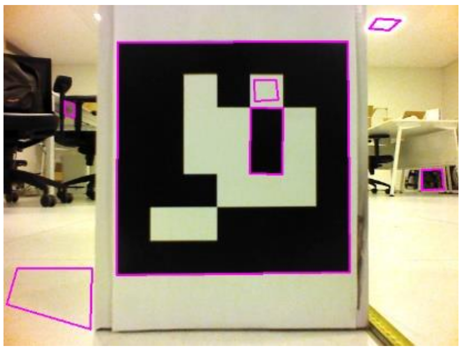


Figure 5.7 - Result of square detection algorithm.

Perspective distortion is removed from the squares found, then they are to a 7x7 matrix using linear interpolation and threshold is applied using the Otsu's Binarization algorithm [60], at this point we obtain a matrix with the marker data in it. Figure 5.8 represents the matrix obtained after the binarization process.

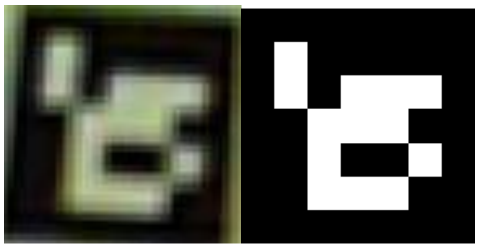


Figure 5.8 - Aruco marker reading result.

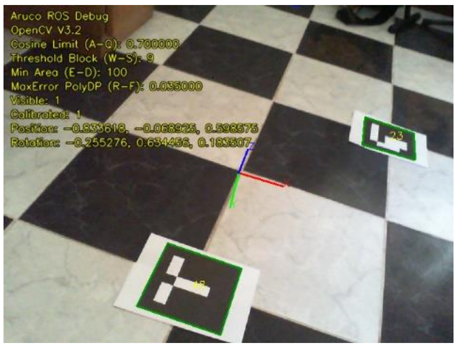
The marker data is now validated using the signature matrix, if the data is not valid that means that it does not correspond to an aruco marker. Markers might be detected in any orientation. The algorithm tests the data with different rotations (90º, 180º, 270º), if the marker is not recognized for any rotation it is then discarded.

5.3 Pose estimation

For pose estimation the method solvePnp from OpenCV was used, in iterative mode using Levenberg-Marquardt [61] optimization.

To obtain the camera position, markers need to be registered into the program, a marker is represented by its identifier and a real-world pose (position and rotation).

Corners obtained from all visible known markers are used to estimate the camera pose. In Figure 5.9 we can observe the origin referential and camera pose estimated using the corners detected from two visible makers.



```
Aruco ROS Debug
OpenCV V3.2
Cosine Limit (A-Q): 0.700000
Threshold Block (W-S): 9
Min Area (E-D): 100
MaxError PolyDP (R-F): 0.035000
Visible: 1
Calibrated: 1
Position: -0.833618, -0.068925, 0.598575
Rotation: -0.255276, 0.634456, 0.183507
```

Figure 5.9 - Aruco marker detection result.

5.4 Corner refinement

After getting a fully function algorithm to refine it further a corner refinement step was added using the sobel derivative operator was. First the section around the detected corners was isolated. The sobel derivative was calculated in x and y. After calculating the derivative, the maximum intensity points are used as a refined corner.

In the Figure 5.10 it is possible to observe a case where the quadrilateral detector obtained close but not precise corner results. The sobel operator allow to get a more refine position of the corners, the red dot represents the initially detected corner and the greed dot represents the refined corner, in the left we have the marker as detected originally without corner refinement and on the right, we have the sobel operator result.

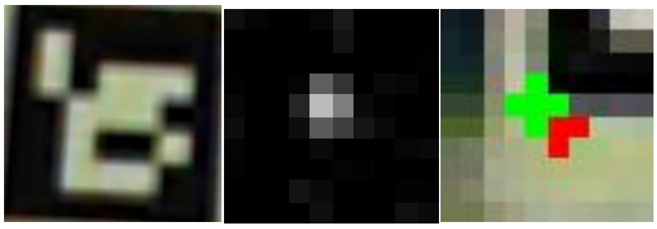


Figure 5.10 - Result of corner refinement using sobel operator.

This algorithm assumes that there is only one corner in the picture, if by the block size used is too big sometimes another corner from the marker are visible resulting in a wrong corner position.

Figure 5.11 shows an example where that happens. In red is the initial corner obtained from quadrilateral detector and in green is the bad corner, this can be easily fixed by automatically calculating the block size relatively to the marker resolution, the block size should be 1/7 of the marker resolution to guarantee that the marker inside corners are not visible.

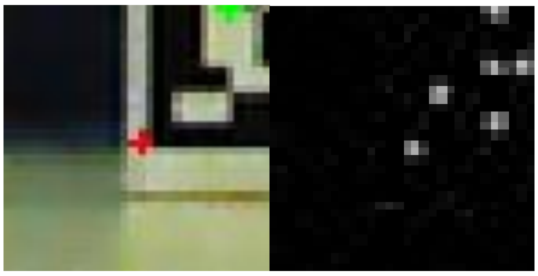


Figure 5.11 - Bad corner refinement due to big block size.

5.5 Algorithm evaluation

To compare the developed solution with the ones already existing a testing environment was created, two test markers were printed, one Aruco maker and one ARTag maker. Both markers had exactly 20cm in size, the camera was placed on top a box and the marker was aligned with the camera, the marker was then held on a box using transparent tape.

A measuring tape with a millimetric scale was used to measure the distance between the box and the markers, an image was taken for each distance tested, the markers were moved 30cm each time until none of the algorithms was able to detect the marker. Figure 5.12 represents some samples of the testing data used.

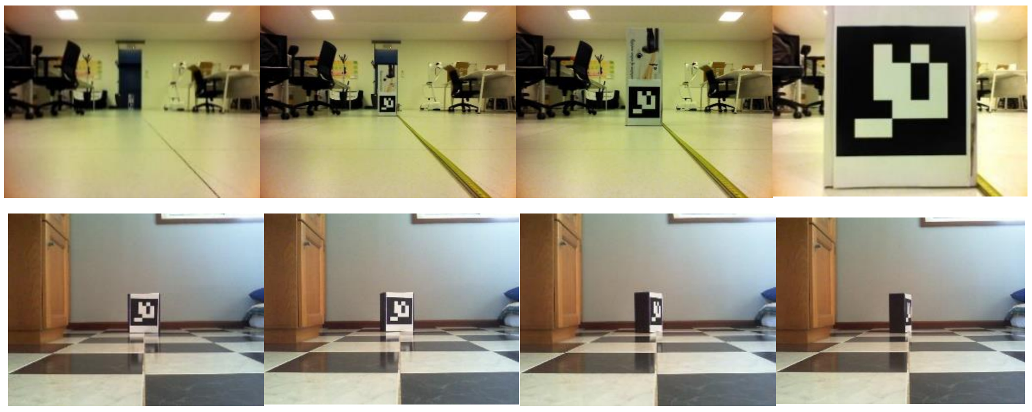


Figure 5.12 - Samples of the testing data.

To measure the tolerance of the detector to perspective distortion a second testing environment was created, a marker was placed on a box and the camera was positioned 2.0 meters away from the marker. The marker was rotated around itself in steps of 10º from 0º to 80º.

Camera was calibrated was performed, a chessboard pattern was used for calibration, the values obtained were stored and used for the tests.

Table 5.2 presents the results obtained for marker detection distance. It is possible to observe that the proposed solution obtained better detection distance results, being capable of detecting aruco markers up to 9 meters away from the camera (using a resolution of 640x480).

Distance (m)	Proposed solution		ROS Aruco		Alvar	
	Distance (m)	Error (%)	Distance (m)	Error (%)	Distance (m)	Error (%)
0,3	0.309	3.004	0.315	4.79	0.303	0.838
0,5	0.501	0.136	0.505	1.013	0.509	1.832
0,7	0.699	0.101	0.705	0.663	0.707	0.974
0,9	0.895	0.61	0.904	0.486	0.908	0.862
1,1	1.094	0.544	1.111	0.976	1.11	0.94
1,3	1.287	0.974	1.312	0.944	1.307	0.524
1,5	1.479	1.425	1.515	0.973	1.505	0.352
1,7	1.679	1.264	1.707	0.414	1.708	0.472
1,9	1.864	1.91	1.926	1.348	1.913	0.655
2,1	2.054	2.229	2.109	0.433	2.107	0.334
2,3	2.283	0.757	2.336	1.544	2.332	1.355
2,5	2.467	1.33	2.539	1.545	2.514	0.548
2,7	2.679	0.787	2.786	3.073	2.726	0.944
2,9	2.84	2.097	2.95	1.682	2.9	0.016
3,1	3.032	2.229	3.142	1.349	3.16	1.907
3,3	3.237	1.961	3.343	1.299	3.358	1.713
3,5	3.459	1.181	3.581	2.263	3.534	0.963
3,7	3.647	1.446	3.791	2.413	3.741	1.104
3,9	3.859	1.058	4.013	2.82	4.014	2.849
4,1	4.091	0.218			4.212	2.662
4,3	4.249	1.19			4.393	2.108
4,5	4.452	1.077			4.595	2.059
4,7	4.643	1.236			5.017	6.322
4,9	4.808	1.923			5.054	3.04
5,1	5.116	0.31				
5,3	5.154	2.836				
5,5	5.566	1.18				
6	5.712	5.049				
6,6	6.304	4.703				
7,2	6.922	4.021				
7,8	7.695	1.367				
8,4	8.618	2.534				
9	8.639	4.181				

Table 5.2 - Results obtained for marker detection.

Figure 5.13 presents the results obtain for the distance detection test in a graph. It is possible to observe that our solution obtained similar accuracy results while improving the maximum detection distance.

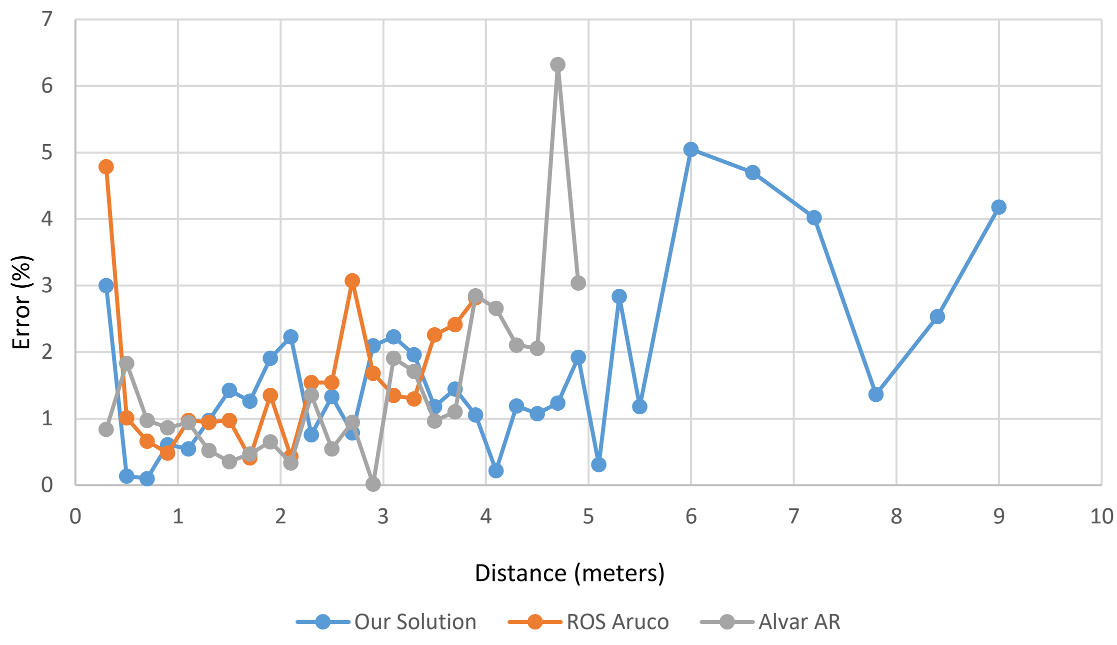


Figure 5.13 - Marker detection distance accuracy graph.

Table 5.3 represents the accuracy of each method considering the medium accuracy of all measures inside of the detection range. It is possible to observe that the solution implemented has similar precision to the other solutions tested.

	Average error (%)		
	Proposed solution	ROS Aruco	Alvar
0-390cm	1,2	1,7	1,1
0-490cm	1,3		1,5
0-900cm	1,8		

Table 5.3 - Accuracy comparison between marker detection algorithms.

Table 5.4 presents the results obtained for marker rotation, our method performed better than the other two algorithms used for comparison, obtaining lower error values.

	Proposed solution		ROS Aruco		Alvar	
Rotation	Dist. (m)	Error (%)	Dist. (m)	Error (%)	Dist. (m)	Error (%)
0	1.981	0.980	2.013	0.631	1.986	0.727
10	1.983	0.872	2.032	1.583	2.017	0.858
20	1.985	0.741	2.031	1.537	2.011	0.564
30	1.989	0.528	2.029	1.415	1.974	1.308
40	2.001	0.040	2.029	1.425	1.971	1.493
50	2.002	0.121	2.03	1.468	2.028	1.357
60	2.004	0.213	2.033	1.639	1.974	1.322
70	1.993	0.341	2.029	1.421		

Table 5.4 - Results obtained for maker perspective distortion.

Table 5.5 represents the results obtain for the rotation test in a graph. Its possible to observe a consistent improvement in accuracy.

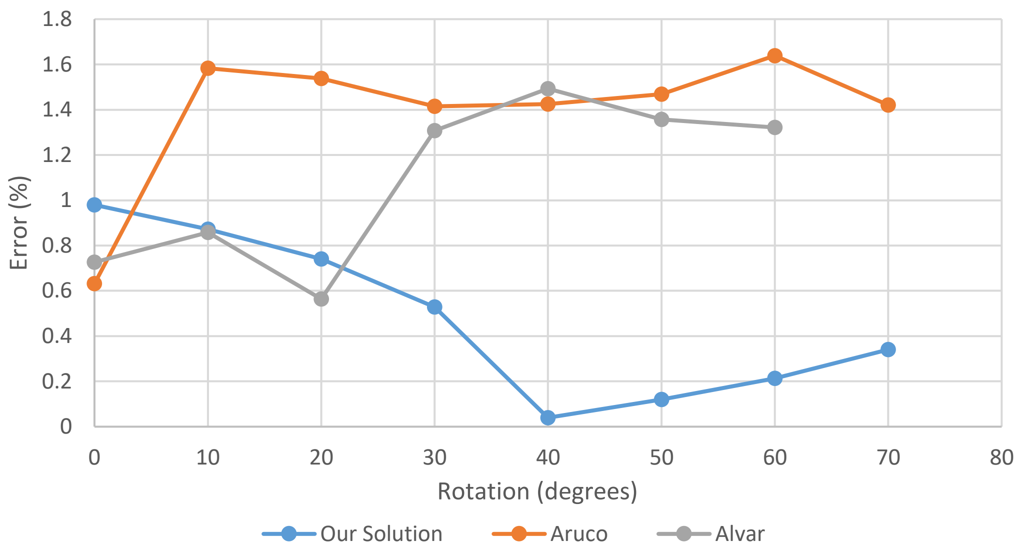


Table 5.5 - Marker detection tolerance to rotation graph

6 Graphical User Interface

In this chapter we will present the development of a new graphical user interface (GUI) and data visualization tools for the WiiGo robot created as a replacement for the old GUI that presented some limitations. The main limitations found were the lack of responsiveness to multiple screen resolutions, no support for multiple languages and use of pre-rendered text.

The old GUI was built with the QT framework using C++. It was integrated as a ROS node into the robot software. It was hardcoded to work with a fixed resolution of 1280x720, and all text was pre-drawn into images used to compose the interface, making it hard to apply corrections or create translations for the GUI.

The GUI is divided into two main section, the main screen used to give feedback and present the current state of the robot to its users and the advanced menu composed of debug and diagnostic utilities. Figure 6.1 presents the old GUI main screen.

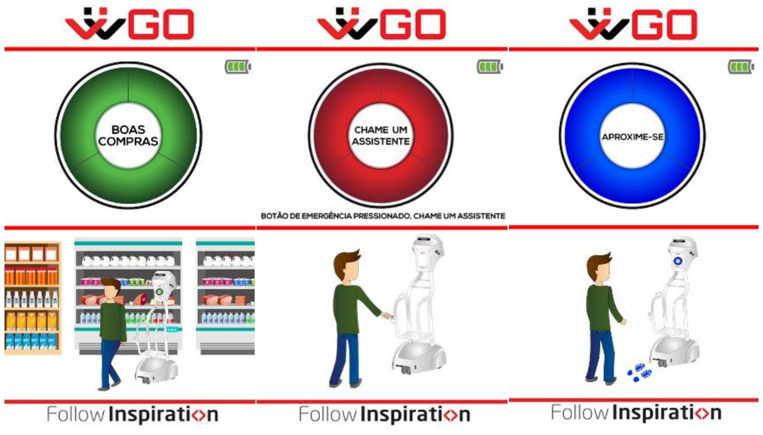


Figure 6.1 - WiiGo robot old GUI main screen.

Figure 6.2 represents the old GUI advanced settings menu that was used for debug purposes allowing the users to visualize information from the multiple robot sensors, and access auto-diagnostic tools. The old visualization tools were only capable of displaying a single type of information at a time and were limited to 2D representations.

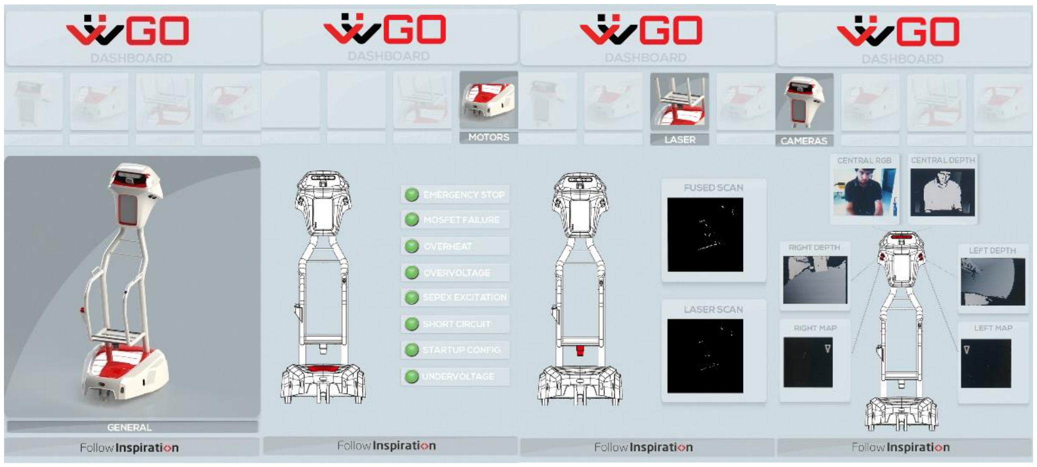


Figure 6.2 - WiiGo robot old GUI advanced menu.

Since the robot it is not equipped with a touchscreen the old GUI required a mouse or keyboard for navigation. This was not a problem since it was only required to access the advanced menu of the robot that contained debug information.

6.1 System architecture

To implement the new user interface and attend to all requirements, we have chosen to use JavaScript as a programming language and NWJS [62] as framework for desktop support. To develop 3D visualization components the three.js [63] library was used.

Figure 6.3 represents the basic blocks used to implement the GUI. Elements are build using DOM elements in JavaScript code. To achieve support for multiple resolutions and screen aspect-ratio the elements are positioned using normalized coordinates relatively to the screen resolution.

Multi-language support was achieved using dynamically loaded JSON files that contain locale related data (e.g. text, images, audio). These files are loaded and managed by the Locale Manager.

The GUI is targeted at multiple robotic platforms, all these platforms use the same base WiiGo technology, but each of them has its own differences (e.g. different sensor configuration, functionality disabled).

To easily adapt the GUI to each different robotic platform a platform managing system was developed. A platform indicates the robot configuration, topics to visualize and which UI components are used. Platforms can implement modified versions of the GUI elements and are managed by the Platform Manager.

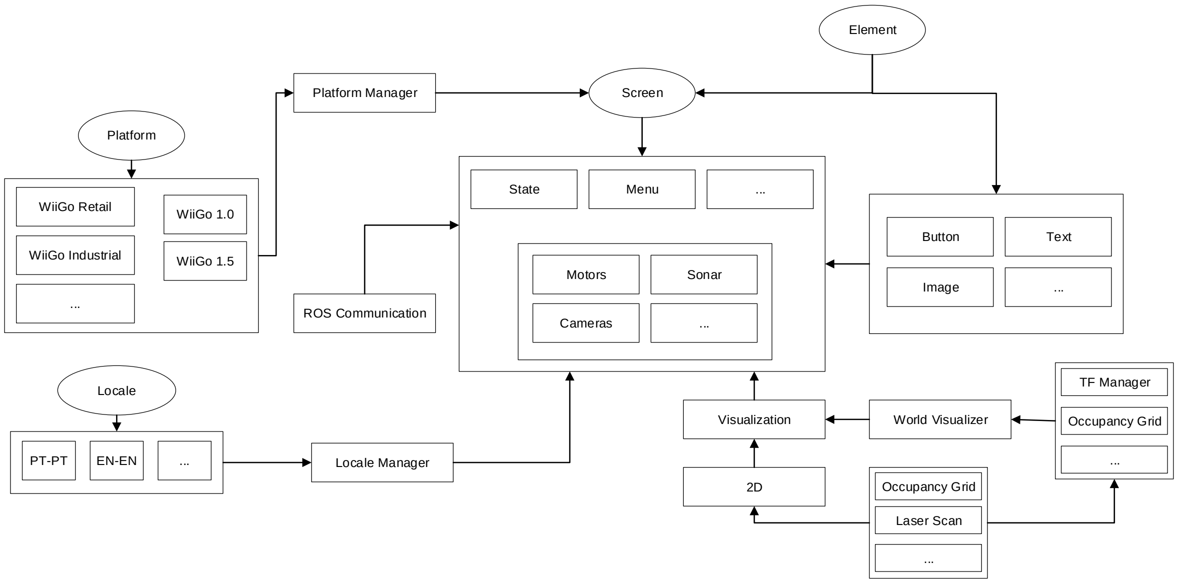


Figure 6.3 - GUI implementation class diagram.

An abstract approach for ROS communication was used presented Figure 6.4. Interfaces for topic and service access where created. ROS communication was implemented by using external libraries. Two different solution were used for communication: ROS NodeJS and ROS LibJS.

ROS NodeJS generates JS definitions for ROS messages and implements the ROS communication model to access and publish data to topics and services. It is used when the software is being run directly in the robot.

ROSLibJS uses ROS bridge, it is a communication middleware that converts the ROS communication model into a web socket based communication model. It is used when accessing the interface from a remote device using a web browser.

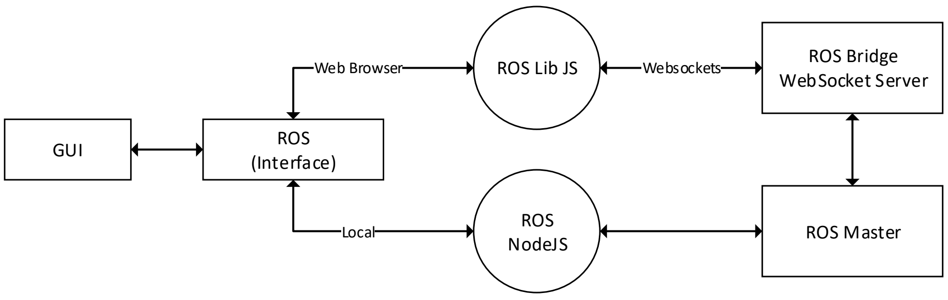


Figure 6.4 - ROS communication model implemented by the GUI.

6.2 GUI Menus

The main screen presented in Figure 6.5 is composed of a central zone containing messages and information for the user (e.g. instructions, warnings) and the zone bellow is used for illustration and interaction.

A new set of simpler illustrations was created by another employee at Follow Inspiration. Vibrant colors are used to represent the robot state. This allows for the user to easily distinguish the robot state. Sound effects are also used, if the robot gets lost or gets blocks a buzzer sound is played to warn the user.

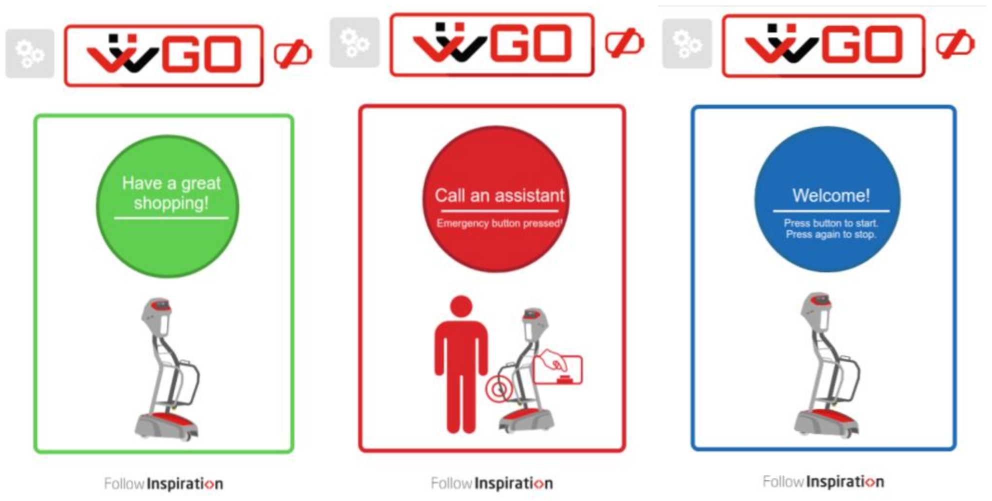


Figure 6.5 - WiiGo GUI main screen.

Figure 6.6 represents the new GUI advanced settings menu. New visualization widgets were developed, some of them were adapted for display some extra information (e.g. the camera viewer can now also display information from the people detection module).

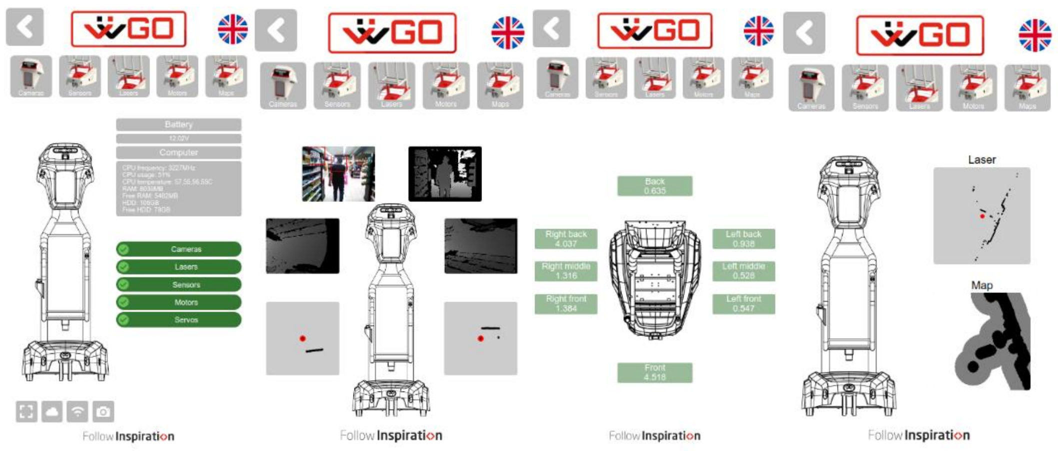


Figure 6.6 – WiiGo robot advanced menu GUI.

Figure 6.7 shows the GUI running remotely on a mobile phone connected to the same network as the robot. All information is updated in real time without any noticeable delay.

It is possible to access the interface remotely using a web browser and the interface is compatible with mobile devices. During tests mobile phones were used to access the interface connected via Wi-Fi, but it is also possible to configure it to external access via internet.

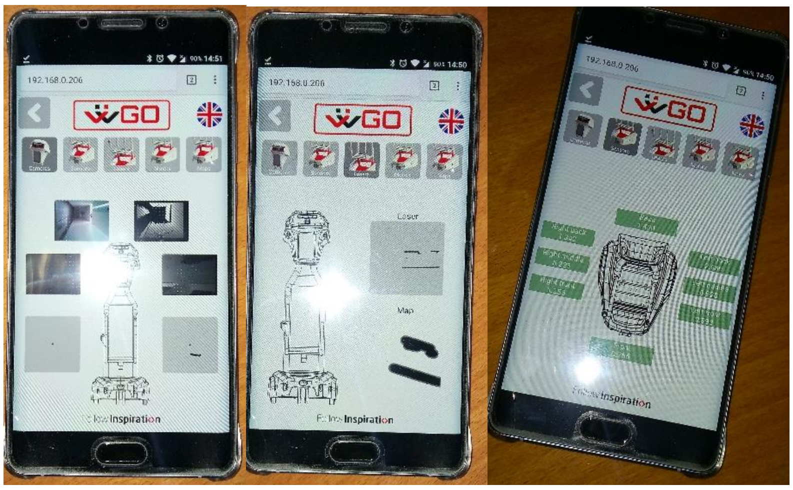


Figure 6.7 - WiiGo GUI running remotely on mobile phone.

6.3 World visualization

To visualize information published by the robot, a set graphical visualization tools were created. The objective for these tools is to visualize robot information in 3D space similarly to RViz. RViz could not be reused for this purpose because it is implemented using different technology.

To draw information published by the robot a ROS tf handling system was developed. A tf message contains a transform, children name and parent name. The transform indicates the position of the children relative to the parent.

Figure 6.8 presents the results obtained. It is possible to observe the robot tf being positioned each one relatively to its parent. To interact with this visualization the use can use a mouse to move the camera around the world. Multi touch gestures were implemented, if the user has a touchscreen or a multitouch enable trackpad its possible use pinch to zoom, or two fingers scroll to move the camera.

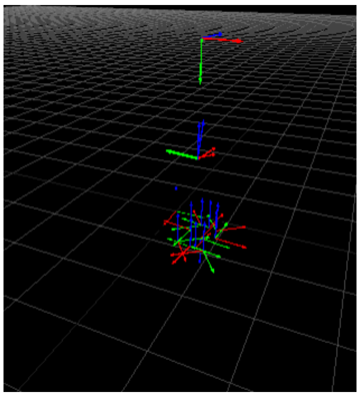


Figure 6.8 - tf visualization in the WiiGo robot GUI.

To visualize occupancy grids a new tool was created. The occupancy grids received are draw into an canvas offscreen. The canvas is attached to a plane geometry which size and origin matches the one indicated in the grid message. Figure 6.9 represents the grid map visualization attached to its tf. It is possible to observe the correct position of the remaining tf relative to the map.

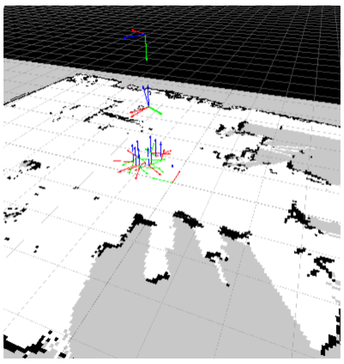


Figure 6.9 - Grid map visualization in the WiiGo robot GUI.

A tool for path creation was developed on top of the world visualization module. It allows the user to create paths using a mouse or touchscreen. Figure 6.10 shows the result obtained. The points in red are the points marker by the user.

Paths created can be used for autonomous navigation or stored into XML files for later user. The WiiGo software uses a A* algorithm to validate the path and applies correction if required. This step ensures that the robot can always reach the defined destination.

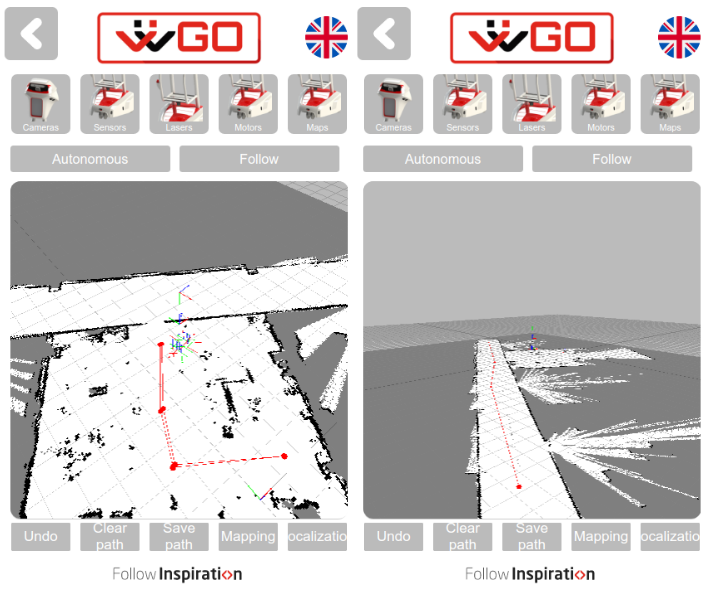


Figure 6.10 - Path creation tool in the WiiGo robot GUI.

7 Conclusion

Multiple objectives were defined for this work. The integration of a SLAM algorithms, visualization tools to interact with this system proposed and a visual marker detection algorithm for landmark detection.

It was possible to successfully achieve all objectives proposed. The results obtained allowed to enhance the WiiGo robot making it able to navigate human populated environments, localize its docking station, and improve user interaction.

In December a test was performed with the WiiGo robot as an internal company event distributing candy to the company employees. Using the map of the Laboratory, a path was created using the tools presented in this document.

The robot roamed independently around the laboratory for 6H until its battery got low. During the test no human intervention was required. The robot was able to navigate the environment and avoid obstacles successfully. Figure 7.1 presents the robot equipped with a tray of candy, made with cardboard, roaming around the laboratory corridors autonomously.

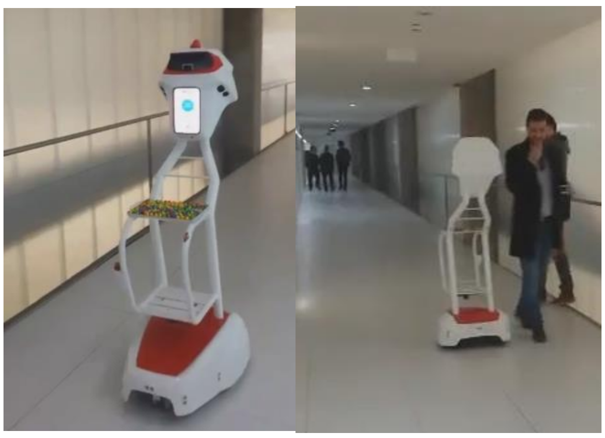


Figure 7.1 - WiiGo robot autonomous navigation experiment as candy distributor.

7.1 SLAM and Localization

Multiple SLAM algorithms were tested and compared. RTABMap was chosen as a SLAM and localization solution to integrate in the WiiGo robot. From all the solutions tested it was the only one capable of good results for all datasets. It obtained good mapping results and was able recover its localization quickly during out tests.

The odometry data present in the dataset used was relatively bad. It could be improved by adding inertial sensors to the robot. (e.g. IMU). Figure 7.2 represents an example where it is possible to observe odometry drift for the WiiGo robot relatively to one if the mapping results.

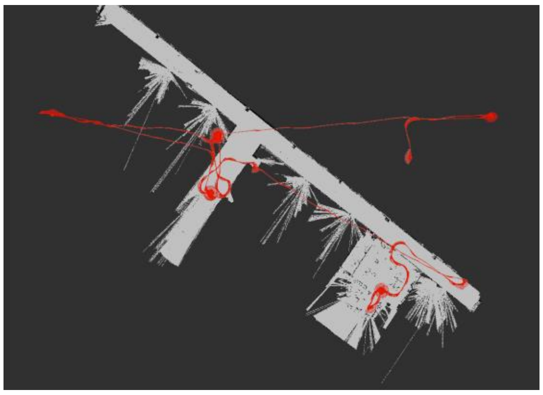


Figure 7.2 - Comparison between map and odometry in Laboratory dataset.

One problem found when testing SLAM algorithms is that due to the use of statistical models that use random data it is hard to obtain consistent results across multiple tests on the same dataset.

The solution proposed presented good results and demonstrates how good SLAM algorithms have become over the years. There is room for improvement specially regarding laser only localization where it was not possible with current algorithms to localize the robot from in a map.

7.2 Visual markers

Experiments conducted showed that the solution developed was able to improve the detection distance when compared to other methods that use similar marker format while keeping similar pose estimation precision

The two algorithms tested and the developed use a similar approach for marker detection. All of them apply adaptive threshold, detect squares, refine corners, apply distortion correction, and decode the marker data. The method proposed uses a different approach for square detection that allow it to detect markers from far away.

We compared our system with two state of art algorithms (ROS Aruco and Alvar) and verified that our algorithm detects the marker up to 9m, when compared with 4m and 5m for the other algorithms it represents a 44% increase in detection distance, with similar precision values. Our method also presents more tolerance to perspective distortion, obtaining better precision results when detecting rotated markers.

To improve the algorithm further a corner refinement method with subpixel accuracy could be added improving the marker corner position estimation.

The implementation of the algorithm described in this document can be found online at `www.github.com/tentone/aruco`.

7.3 Graphical User Interface

The GUI implemented was able to successfully fulfill all requirements proposed and replaced the older implementation completely. It provides a more refined experience and better interactivity to the users.

The solution presented allowed the GUI to be translated into four different languages: Portuguese, English, Romanian and French. Figure 7.3 presents an example on GUI the main screen in multiple languages. More languages can be easily added by creating additional new Locale configuration files. No changes to code are required. The robot GUI language can be selected in the advanced settings menu.

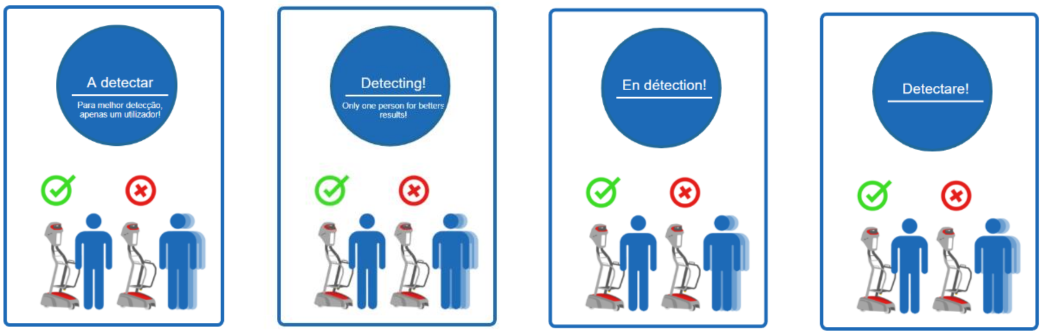

A detectar
Para melhor detecção, apenas um utilizador!
Detecting!
Only one person for better results!
En détection!
Detectare!

Figure 7.3 - GUI main screen in multiple languages (Portuguese, English, French and Romenian).

The new modular system implemented will allow for faster expandability and will be used as a base for future projects. Figure 7.4 presents the new WiiGo GUI running on the robot hardware, were it is possible to observe the main screen and the word visualization menu.

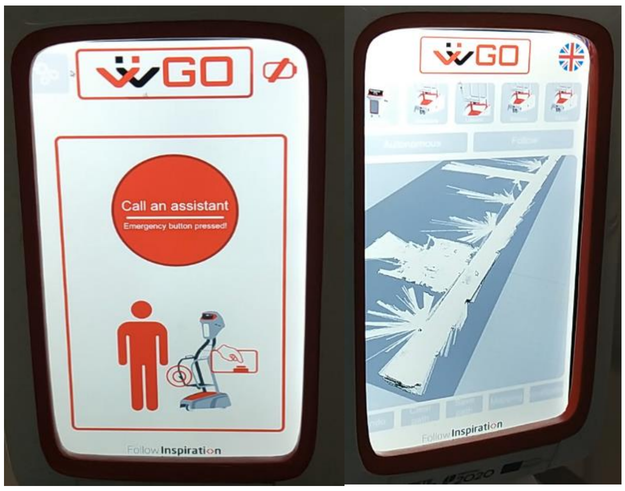


Figure 7.4 - GUI running on the WiiGo robot.

Appendix A - Feature Based Object detection

In this section we will study a feature-based object detection framework. The objective of this experiment was to test and compare the performance of visual feature extraction algorithms and determine the best one to be used with the RTABMap SLAM framework.

These tests will be performed Find-Object [64] framework developed by Mathieu Labbé. It is an object detection solution that uses visual features. It integrates multiple visual feature algorithms (BRIEF, FAST, GFTT, MSER, ORB, STAR, FREAK, SIFT, SURF and BRISK).

Methodology

For these tests three different objects were used, the box of a processor "Intel Core i5 4460", a box of Coffe "DeltaQ Activus 10Uni" and a motherboard box "ASUS H81i-PLUS". All these objects are solid and textured, and theoretically should represent a good test scenario for the feature extraction algorithms tested.

Images of the objects were captured using the Orbbec Astra camera at a resolution of 640x480. A feature database was created with these images. Figure A.0.1 presents the images captured.

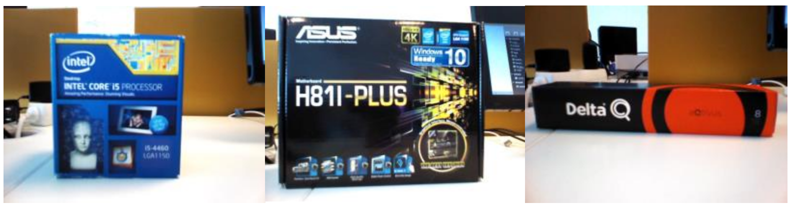


Figure A.0.1 - Testing objects for object detection.

Another image was taken where are three objects are visible. For each algorithm processing time, number of features found, number of objects recognized, false positives where measured. Figure A.0.2 represents the image used for testing.

Find-object implements k-trees based nearest neighbor search for both SURF and SIFT algorithms allowing these two solutions to have better performance in the matching phase. For testing since RTABMap also supports these methods they were kept enabled.

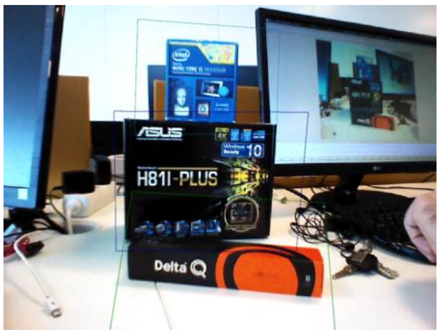


Figure A.0.2 - Test environment setup, with object detection boxes.

Results

Table A.0.1 represents the results obtained for all tests performed, it is possible to observer that the SURF algorithm obtained the best results being the faster algorithm capable of detecting all objects. ORB is considerably faster that all the other algorithms and may be a good solution in low performance scenarios.

Some of the solutions tested could not recognize any object. For these solutions no matches between features stored in the databased and features present in the test image were found. Those results were omitted from the table.

Features	NN Find	Objects	Matches	FP	Delta (ms)	Features
SURF	KDTree	3	3	0	148	711
SIFT	KDTree	3	3	0	303	1051
ORB	BruteForce	2	2	0	35	500
KAZE	BruteForce	2	2	0	956	1314
AKAZE	BruteForce	3	3	0	483	1073
BIRSK	BruteForce	3	3	0	1015	1725

Table A.0.1 - Comparison of visual feature extraction methods for object detection

Conclusion

It was possible to observe that this method for object detection could be easily applied in a real-world case. It was able to detect all test objects. During the tests no false positive were detected, but a false positive was obtained when using on purpose a visually similar product box, represented in Figure A.0.3.

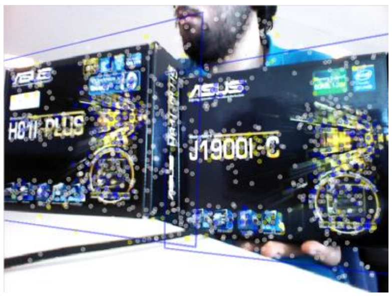


Figure A.0.3 - False positive detection with similar objects.

The use of features for object detection allowed to obtain matches even under occlusion or rotation. Figure A.0.4 represents a case where it was possible to detect the CPU box even under partial occlusion and rotation.

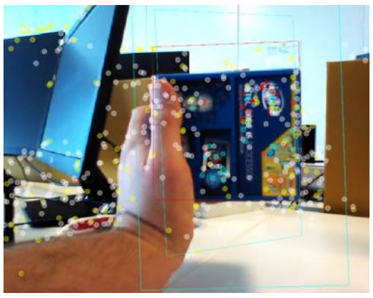


Figure A.0.4 - Object detection under occlusion and rotation.

It was possible to observe during these tests that feature based object detection works well for rigid objects. But it is not reliable for deformable body detection (e.g. bags, fruit). In Figure A.0.5 it is possible to observe a test scenario with SURF features. Multiple images of bananas were used for feature extraction but when presenting two similar bananas to the ones used as reference it was not possible to obtain any matches.

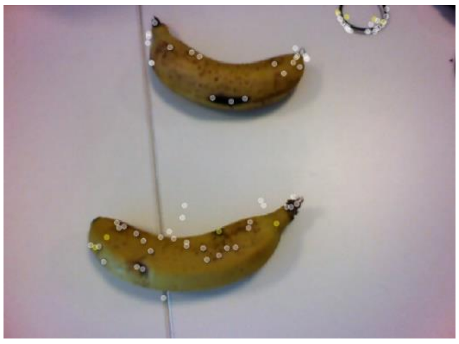


Figure A.0.5 - Fruit detection using visual features.

References

[1] Follow Inspiration, "Follow Inspiration," [Online]. Available: www.followinspiration.pt. [Accessed 16 February 2018].

[2] P. F. Forward, "GITA Robot," [Online]. Available: www.piaggiofastforward.com/gita. [Accessed 16 February 2018].

[3] ClearPath Robotics, "ClearPath Robotics Autonomous Mobile Robots," [Online]. Available: www.clearpathrobotics.com. [Accessed 16 February 2018].

[4] O. Motors, "OTTO Self-driving vehicles for Automated Material Transport," [Online]. Available: www.ottomotors.com. [Accessed 16 February 2018].

[5] OTTO Motors, "Meet OTTO 1OO," [Online]. Available: https://www.ottomotors.com/otto100. [Accessed 25 January 2018].

[6] ClearPath Robotics, "Autonomy Research Kit All-In-One Autonomous Navigation," [Online]. Available: https://www.clearpathrobotics.com/autonomy-research-kit/. [Accessed 24 February 2018].

[7] Stanford University, "Personal Robotics Program," 27 12 2017. [Online]. Available: personalrobotics.stanford.edu.

[8] P. Robotics, "StockBot: Automatic Inventory Solution," [Online]. Available: www.pal-robotics.com/en/products/stockbot/. [Accessed 19 February 2018].

[9] J. Viladomat, "Automating inventory-taking with StockBot," [Online]. Available: blog.pal-robotics.com/blog/automating-inventory-taking-with-stockbot/. [Accessed 19 February 2018].

[10] P. Robotics, "Google Play PAL Robotics REEMote," [Online]. Available: https://play.google.com/store/apps/details?id=com.palrobotics.reemote. [Accessed 24 February 2018].

[11] Xiaomi, "Xiaomi Mi Robot Cleaner 2," [Online]. Available: www.mi.com/roomrobot/. [Accessed 25 February 2018].

[12] C. S. K. A. M. B. Wolfram Burgard, "Introduction to mobile robotics: SLAM," Universität Freiburg, 2012.

[13] P. P. J. S. Niko Sunderhauf, "Robust Optimization for Simultaneous Localization and Mapping," Technische Universität Chemnitz, Chemnitz, 2012.

[14] C. S. Giorgio Grisetti, "Learning Grid Maps with Rao-Blackwellized Particle Filters," University of Freiburg, 2007.

[15] K. Murphy, "Bayesian Map Learning in Dynamic Environments," *Proceedings of the 12th International Conference on Neural Information Processing Systems,* pp. 1015-1021 , 1999.

[16] C. S. W. B. Giorgio Grisetti, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," *Proceedings of the 2005 IEEE*

International Conference on Robotics and Automation, pp. 2432-2437 , 2005.

[17] C. S. W. B. Giorgio Grisetti, "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics,* vol. 23, no. 1, pp. 34-46, 2007.

[18] O. v. S. J. M. U. K. Stefan Kohlbrecher, "A flexible and scalable SLAM system with full 3D motion estimation," *IEEE International Symposium on Safety, Security, and Rescue Robotics,* 2011.

[19] D. K. H. R. D. A. Wolfgang Hess, "Real-Time Loop Closure in 2D LIDAR SLAM," *IEEE International Conference on Robotics and Automation (ICRA),* pp. 1271-1278, 2016.

[20] Google, "Ceres Solver," [Online]. Available: www.ceres-solver.org/. [Accessed 19 February 2018].

[21] Google, "Cartographer project," 2016. [Online]. Available: https://google-cartographer.readthedocs.io/en/latest/. [Accessed 01 February 2018].

[22] F. M. Mathieu Labbe, "Memory Management for Real-Time Appearance-Based Loop Closure Detection," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2011.

[23] M. Labbé, "Simultaneous Localization and Mapping (SLAM) with RTAB-Map," Département de Génie Électrique et Génie Informatique, 2015.

[24] F. M. Mathieu Labbé, "Long-term online multi-session graph-based SPLAM with memory management,1," *Autonomous Robots,* vol. 42, no. 143, pp. 1-18, 2017.

[25] F. M. Mathieu Labbe, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2014.

[26] V. Shah, "ROS Adaptive Monte Carlo localization," [Online]. Available: www.wiki.ros.org/amcl. [Accessed 17 February 2018].

[27] W. B. F. D. S. T. Dieter Fox, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," *Proceedings of the National Conference on Artificial Intelligence,* pp. 343-349, 1999.

[28] Erle Robotics, "Localization in a known map with Erle-Rover," [Online]. Available: docs.erlerobotics.com/erle_robots/erle_rover/examples/localization_in_a_know_map_with_erle_rover. [Accessed 17 February 2018].

[29] C. S. S. G. W. B. Giorgio Grisetti, "A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent," *Robotics: Science and Systems,* 2007.

[30] J. L. S. T. Edwin Olson, "Fast Iterative Alignment of Pose Graphs with Poor Initial Estimates," *International Conference on Robotics and Automation,* pp. 2262-2269, 2006.

[31] M. K. J. J. L. David M. Rosen, "An Incremental Trust-Region Method for Robust Online Sparse Least-Squares Estimation," *International Conference on Robotics and Automation,* 2012.

[32] M. K. Frank Dellaert, "Simultaneous Localization and Mapping via Square Root Information Smoothing," *Journal of Robotics Research.*

[33] R. A. B. Roy E. Welsch, "Robust nonlinear regression using the dogleg algorithm," *National bureau of economic research,* vol. 76, 1975.

[34] G. G. H. S. K. K. W. B. Rainer Kuemmerle, "g2o: A General Framework for Graph Optimization," *IEEE International Conference on Robotics and Automation,* 2011.

[35] P. P. Niko Suenderhauf, "Switchable Constraints for Robust Pose Graph SLAM," *IEEE International Conference on Intelligent Robots and Systems*, 2012.

[36] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision,* vol. 60, no. 2, p. 91–110, 2004.

[37] T. T. L. V. G. Herbert Bay, "SURF: Speeded Up Robust Features," *European Conference on Computer Vision,* 2006.

[38] OpenCV, "Introduction to SURF (Speeded-Up Robust Features)," [Online]. Available: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html. [Accessed 16 February 2018].

[39] T. D. Edward Rosten, "Fusing Points and Lines for High Performance Tracking," *International Conference on Computer Vision,* vol. 2, pp. 1508-1515, 2005.

[40] V. L. C. S. P. F. Michael Calonder, "BRIEF: Binary Robust Independent Elementary Features," *European Conference on Computer Vision,* pp. 778-792, 2010.

[41] V. R. K. K. G. B. Ethan Rublee, "ORB: an efficient alternative to SIFT or SURF," *International Conference on Computer Vision,* pp. 2564-2571, 2011.

[42] A. B. A. J. D. Pablo Fernandez Alcantarilla, "KAZE Features," *European Conference on Computer Vision,* pp. 214-227, 2012.

[43] J. N. A. B. Pablo F. Alcantarilla, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," *British Machine Vision Conference,* 2013.

[44] M. C. R. Y. S. Stefan Leutenegger, "BRISK: Binary Robust invariant scalable keypoints," *International Conference on Computer Vision,* 2011.

[45] Orbbec, "Orbbec Astra, Astra S & Astra Pro," [Online]. Available: www.orbbec3d.com/product-astra/. [Accessed 2018 February 16].

[46] Hokuyo, "Scanning Laser Range Finder Smart-URG mini UST-10LX Specification," 2016. [Online]. Available: http://www.hokuyo-usa.com/application/files/2414/7196/1936/UST-10LX_Specifications.pdf. [Accessed 25 January 2018].

[47] Roboteq, "SDC2130 Brushed DC Motor Controller," [Online]. Available: https://www.roboteq.com/index.php/roboteq-products-and-services/brushed-dc-motor-controllers/279/sdc2130-detail. [Accessed 25 January 2018].

[48] Avago Technologies, "HEDM-55xx/560x & HEDS-55xx/56xx Quick Assembly Two and Three Channel Optical Encoders," [Online]. Available: http://mathcs.holycross.edu/~kwalsh/zebra/1884440.pdf. [Accessed 25 January 2018].

[49] K. C. B. G. J. F. T. F. J. L. E. B. R. W. A. M. M. Quigley, "ROS: an open-source Robot Operating System," *Icra,* vol. 3, no. 5, 2009.

[50] Open Source Robotics Foundation, "ROS Technical overview," [Online]. Available: http://wiki.ros.org/ROS/Technical%20Overview. [Accessed 27 January 2018].

[51] T. Foote, "tf: The Transform Library," *TePRA 2013,* 2013.

[52] A. R. T. Jackie Kay, "Real-time control in ROS and ROS 2.0," [Online]. Available: https://roscon.ros.org/2015/presentations/RealtimeROS2.pdf. [Accessed 25 January 2018].

[53] S. K. T. A. Yuya Maruyama, "Exploring the Performance of ROS2," *EMSOFT '16 Proceedings of the 13th International Conference on Embedded Software,* 2016.

[54] R.-S. F.-C. M.-J. S.Garrido-Jurado, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition,* vol. 47, no. 6, p. 2280–2292, 2014.

[55] S. Siltanen, "Theory and applications of marker-based augmented reality," *VTT Science,* vol. 3.

[56] R.-S. F.-C. R.-C. S.Garrido-Jurado, "Generation of fiducial marker dictionaries using Mixed Integer Linear Programming," *Pattern Recognition,* vol. 51, no. C, pp. 481-491, 2016.

[57] B. S. Mehmet Sezgin, "Survey over image thresholding techniques and quantitative," *Journal of Electronic Imaging,* vol. 13, no. 1, pp. 146-165, 2004.

[58] K. A. Satoshi Suzuki, "Topological structural analysis of digitized binary images by border following," *Computer Vision. Graphics, and Image Processing,* vol. 30, no. 1, pp. 32-46, 1985.

[59] A. C. G. d. S. M. R. G. M. Shin-Ting Wu, "The Douglas-peucker algorithm: sufficiency conditions for non-self-intersections," *J. Braz. Comp. Soc,* vol. 9, no. 3, 2004.

[60] N. Otsu, " A Threshold Selection Method from Gray-Level Histogram," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 9, no. 1, pp. 62-66, 1979.

[61] K. Levenberg, "A method for the solution of certain non-linear problems in least squares,," *Quarterly of Applied Mathematics,* vol. 2, no. 2, pp. 164-168, 1994.

[62] R. Wang, "NW.js," [Online]. Available: www.nwjs.io. [Accessed 22 February 2018].

[63] R. Cabello, "three.js javascript 3D library," [Online]. Available: www.threejs.org. [Accessed 26 February 2018].

[64] M. Labbé, "FindObject project page," [Online]. Available: https://introlab.github.io/find-object/. [Accessed 02 February 2018].