UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTMENTO DE INGENIERIA ELECTRICA

AN EVALUATION AND COMPARISON OF LONG TERM SIMULTANEOUS LOCALIZATION AND MAPPING ALGORITHMS

MEMORIA PARA OPTAR AL TITULO DE
INGENIERO CIVIL ELECTRICO

FABIÁN ALEJANDRO CONTE MARZA

PROFESOR GUÍA:
MARTIN ADAMS

MIEMBROS DE LA COMISIÓN:
FELIPE INOSTROZA FERRARI
MARCOS ORCHARD CONCHA

SANTIAGO DE CHILE
2018

SUMMARY OF THE PROJECT REPORT TO OPT TO
THE TITLE OF INGENIERO CIVIL ELECTRICO
BY: FABIÁN ALEJANDRO CONTE MARZA
DATE: JANUARY 2018
SUPERVISOR: MARTIN ADAMS

AN EVALUATION AND COMPARISON OF LONG TERM SIMULTANEOUS LOCALIZATION AND MAPPING ALGORITHMS

RESUMEN

Este trabajo consiste en la generación de un set de datos con un respectivo ground truth (medición más confiable) y el uso de los algoritmos ORB-SLAM (Orientated FAST and Rotated BRIEF (Binary Robust Independent Elementary Features) Simultaneous Location And Mapping) y LOAM (Lidar Odometry And Mapping) a modo de entender de mejor forma el problema de SLAM (localización y mapeo simultaneo) y comparar los resultados obtenidos con el ground truth.

A modo de entender de mejor forma el set de datos generado, la funcionalidad de los diferentes sensores es explicada. Los sensores utilizados para generar los datos son LIDAR, cámara estéreo y GPS.

Este trabajo posee dos mayores etapas, en primer lugar, el GPS es estudiado para establecer las diferentes formas de extraer los datos desde el dispositivo. Una forma es generar un nodo de ROS que mediante comunicación de Bluetooth otorga un mensaje que puede ser leído. Otra forma es presionar tres veces el botón de encendido del GPS, lo que inicia el almacenamiento de los datos en la tarjeta SD. Mientras el primer método entrega mayor cantidad de información, es menos confiable, existiendo la posibilidad de guardar mensajes vacios o perdida de ciertos datos, afectando la tasa de muestreo. Finalmente una combinación de ambos métodos es implementada.

Un set de datos de prueba es generado cerca de la Universidad De Chile, para probar que los datos están siendo almacenados correctamente. En el test se concluye que a modo de obtener mejor resultado con el GPS es necesario tomar los datos en zonas con baja cantidad de edificios.

Finalmente con los datos y el ground truth el Error Absoluto de la Trayectoria (ATE) es calculado como método de comparación de ambas trayectorias generadas con los algoritmos mencionados. El ATE s la cantidad de energía necesaria para transformar la trayectoria estimada en el ground truth. Dadas ciertas limitaciones en la extracción de los datos estimados, la comparación se realizo entre dos set de datos de prueba, con pequeña cantidad de loops en el camino recorrido. En esta situación los resultados dados por LOAM son mejores que los obtenidos con ORB.SLAM. Pero en un ambiente con mayor cantidad de loops y una trayectoria más larga ORB-SLAM entregaría mejores resultados.

ABSTRACT

This work consists of the generation of a data-set with ground truth and the use of ORB-SLAM (Orientated FAST and Rotated BRIEF (Binary Robust Independent Elementary Features) Simultaneous Location And Mapping) and LOAM (Lidar Odometry And Mapping) algorithms as a way to better understand SLAM and to compare the ground truth and the data-set generated.

To fully understand the data-set generated, the functionality of the different sensors is explained. The sensors used to generate the data-set are LIDAR, Stereo Camera and a GPS.

This work is divided into two stages, in the first place the GPS is studied to establish the different ways to extract the data from it. One way is to generate a ROS node that through Bluetooth communication generates a message which is published. The other way is to press three times the button of the GPS to store the data in the GPS micro SD memory. While the first method is capable of store more data per second, it is less reliable, existing the possibility of store an empty message or simply the loss of data in the process. In the end, a combination of the two methods is implemented, modifying the bag file with the data stored in the micro SD.

A test-data is generated near the University Of Chile, to prove that the bag file (a type of file that can contain any kind of information such as images, video or text, between others) is correctly generated. In these tests, it was concluded that to obtain better performance of the GPS therefore, obtain a better ground truth, it was necessary to generate the data in a zone with a low quantity of high buildings.

Finally with the data-set and the ground truth the Absolute Trajectory Error (ATE) is used as a method to compare the trajectories. The ATE is the amount of energy that would require to transform the estimated trajectory on the ground truth. Since certain limitations of the extraction of the estimated path, the comparison was made between two small data-set which counted with low quantity of closed loops. Therefore the LOAM algorithm shows better results in this trajectory. The ORB-SLAM algorithm shows better results in data-sets with a high quantity of loops in the path.

To those who believed that I could.

Thanks

Thanks to my professor Martin Adams for his kind and helpful advice during this work

To Felipe Inostroza for helping me to understand what is relevant, and so why this could be important in the field; also Im grateful for his help in fixing important details to generate the data-set.

To AMTC (Advanced Mining Technology Center) team for helping in the understanding of the GPS, for facilitate the access to a base place to generate the data-set and for letting me use the hardware to compute the different algorithms.

To my friends for accompany me in this process and keeping me distracted in the hard times.

To my family for convincing me to do this.

Contents

1 Introduction **1**
1.1 General objectives . . . 2
1.2 Specific objectives . . . 2
1.3 Document Structure . . . 3

2 Bibliographic review **4**
2.1 Mobile Platform (Husky A200 Clearpath) . . . 4
2.2 Sensors . . . 5
2.2.1 TOPCON HIPERIÓN GPS [1] . . . 5
2.2.2 Stereo camera (Bumblebee®XB3[2]) . . . 7
2.2.3 LIDAR (VLP-16[3]) . . . 7
2.2.4 RADAR (Acumine 94 Ghz[4]) . . . 8
2.3 State of the art . . . 9
2.3.1 SLAM . . . 9
2.3.2 Graph-SLAM [5] . . . 11
2.3.3 ORB-SLAM [6] . . . 13
2.3.4 LOAM (Lidar Odometry and mapping) [7] . . . 16

3 Methodology **19**
3.1 Sensors connection . . . 19
3.2 Collecting data . . . 20
3.3 Computing The SLAM algorithm . . . 20
3.4 Comparison with ground truth . . . 20
3.5 Working Schedule . . . 21

4 Generating the data set **23**
4.1 GPS configuration . . . 23
4.1.1 Connecting the GPS to the computer . . . 24
4.1.2 Verifying the data . . . 24
4.2 Collecting the data . . . 29
4.2.1 Test data . . . 29
4.2.2 Verifying the final trajectory . . . 31

5 Data Processing **35**
5.1 ORB-SLAM . . . 35
5.2 LOAM . . . 36
5.3 Comparison . . . 37

6 Conclusion **41**

Bibliography **43**

List of Tables

4.1 GPS modes. 26
4.2 Topic used in the data-set. 33

5.1 Comparison between algorithms. 38

List of Figures

2.1 Husky Clearpath. 4
2.2 Triangulation GPS process. 5
2.3 GPS HIPER V. 6
2.4 Base and Rover system. 6
2.5 Stereoscopic camera. 7
2.6 Lidar sensor. 7
2.7 Lidar sensor measurement method. 8
2.8 Radar sensor. 8
2.9 Data taken from a Radar in a rectangular room. Taken from [8]. 9
2.10 Error associated to the movement. 10
2.11 Matlab simulated SLAM. 11
2.12 Representation of the map generation. 12
2.13 Closed loop correction on SLAM. 13
2.14 Points detected in both images 14
2.15 Closed loop correction 15
2.16 Block diagram of the map calculation. 16
2.17 Map generated using LOAM method. 18

3.1 Working schedule first semester. 21
3.2 Working schedule second semester. 22

4.1 Topic of the data received from the GPS. 25
4.2 Data received from the GPS. 25
4.3 Data received from the GPS displayed on Google maps. 27
4.4 Data recorded on the GPS memory card displayed on Google maps without post processing. 28
4.5 Data recorded on the GPS memory card displayed on Google maps with post processing. 28
4.6 Husky mobile platform used torecord the data-set. 29
4.7 Big arc with the GPS mounted. 30
4.8 Small arc with the GPS mounted. 30
4.9 Coordinate system. 31
4.10 Results from the test data GPS. 31
4.11 Image used in ORB-SLAM. 32
4.12 Path generated by ORB-SLAM. 32
4.13 Path generated by GPS (The ground truth). 33

5.1 Output of the ORB-SLAM algorithm. 36
5.2 Output of the LOAM algorithm. 37
5.3 Output of the LOAM algorithm. 37
5.4 Output map of the LOAM algorithm when running same bag file twice. . . . 38
5.5 Initial ORB-SLAM posses. 39
5.6 Initial LOAM posses. 40

Chapter 1

Introduction

Human beings, have had been able to use our understanding of the world, to solve problems. When humans started to evolve, the main troubles to solve was how to survive. For this reason most of the tools created were designed to supply something that humans lack, or to improve human's abilities. When human was more able to survive, man started to design more sophisticated tools to understand the world, in that moment, the idea of generating automatic systems started to grow.

First, autonomous systems were made in a way that they don't process information, only react to different impulses. A good and simple illustration is a water mill, which moves by the flow of water making possible to grind the wheat. Obviously more, and more complex autonomous systems where created. These systems not only could process information sensing their surroundings, they could learn from past experiences (data) and inform their outcomes. Some of them were possible by the development of computational systems.

Nowadays autonomous robotics systems are developed to do things that are beyond human capabilities. For example: explore places unreachable before; deep down our planet, beyond the solar system, inside of tiny things. Also, they can be used as assistance for humans that have lost some body functionality or to execute activities in the industry which are too dangerous to be developed by a person.

In this context, it is necessary to develop a way to transfer knowledge acquired by generations of humans, to a robotic processor based system. This information can be described as human interaction with the environment, and if it is possible for the robot to reproduce these interactions in some way, it should be possible to develop a complex autonomous system. Some of this knowledge could be for example recognition of objects, detection of distance by sound waves and being able to move in a terrain avoiding obstacles or hazards.

This work is going to focus on the movement of a robot. For a robot to be able to move as a person could, it needs three things: its location, the planned position and the best way to reach it. To get the first one it is necessary to know, with a high certainty, where is the robot located. And to know the environment in which it is enveloped, as a person would require. It is possible to solve one problem at a time, or even both at the same time, if external information is given to the subject. This external information could be a pose, to know where the robot is located, or a map, to fully understand the environment in which it is localized. But, it would be better to make both processes simultaneous, letting the robot sense the environment, generate a map, and locate itself in the map at the same time. This process is called SLAM (Simultaneous Location And Mapping).

The quantity of algorithms developed in this area is extensive. Some of these algorithms focus on creating the map and localizing the robot offline, which means, to take data with the robot and then compute the map and the trajectory. One of the applications of SLAM is to automate the movement of the robot. This makes it necessary for the algorithm to work online (in a time that allows the results obtained to represent the actual pose) as the estimated trajectory will help the robot to make real time decisions.

In this moment not only the possibility of determining the position and map take importance, the computational load of the algorithm also takes priority. More precise algorithms can be very slow, but it is still possible to reduce the load and obtain accurate results. It is in this context that it is necessary to be able to compare two different SLAM algorithms, which might generate similar results, but can differ in execution time and methodology. In this way the methodology used to compare the algorithms will be detailed to future references.

1.1 General objectives

The objective of this work is to generate a large data set with an associated ground truth. As a way to test the ground truth an established method to compare results given by two different SLAM algorithms is going to be used. To define the best result it is necessary to compare its trajectory with the true path (ground truth), as well as the computational load for each algorithm. The final decision on which algorithm is better, is relative to the application in which the algorithm is being used, since a more precise algorithm could take more processing time, but if the time is not a problem to the application, it can be chosen as a better algorithm. But if instead, the time of computation is important, the faster, less accurate algorithm should be chosen as a solution.

1.2 Specific objectives

To ensure the correct comparison between the algorithms it will be necessary to:

- Use the rtk GPS (real time kinematic Global Positioning System) to generate the ground truth.
- Connect every sensor used to collect the data.
- Use ROS (Robotic Operating System) software to handle sensors and collect the data.
- Execute the SLAM algorithms to obtain the calculated path.
- Compare the results obtained using the ground truth.

1.3 Document Structure

This project is focused on testing two SLAM algorithms and compare them with a GPS ground truth mapping. In Chapter 2 the sensors used to obtain the data are described, explaining how the algorithms work. In section 3 of Chapter 2, the SLAM algorithm problem is explained completely, also specific algorithms were explained for better understanding. These algorithms are not necessarily the ones used in this work, but they will be explained as a way to understand how they work and their outcomes. Chapter 3 includes the designed methodology to get results and do a fair comparison, followed by project timetable used. Chapter 4 includes outcomes of data-set and ground truth. Results are presented in Chapter 5, and, finally, Chapter 6 includes the conclusion, limitations, and implications of the project, and ideas for further research.

Chapter 2

Bibliographic review

To obtain information from the environment that surrounds the robot (detect objects around the robot and use them as landmarks), and to be able to define a map and a trajectory to compare two SLAM algorithms. It is going to be necessary to use different kinds of sensors. These sensors will be mounted in a skid-steer robot. This robot is going to move a predefined route, saving the data generated by the sensors to the computers memory.

2.1 Mobile Platform (Husky A200 Clearpath)

The Husky A200 robotic platform consists of a chassis with four wheels and a support for batteries (Figure 2.1). On this platform any sensor may be mounted, these sensors must be connected, to an external computer, to control and extract information from the sensors. The robot can be controlled using a remote control, by connecting the husky through a serial port to the external computer (to handle the instructions given by the control). Since the robot has a set of batteries it is possible to keep the sensors energized at the same time as the route defined is traveled.

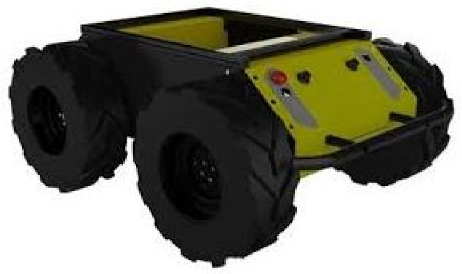


Figure 2.1: Husky Clearpath.

The platform includes [4]:

- Encoders

- Regulated voltage sources (5V, 12V, 24V)
- Custom aluminum structure to mount the sensors on.

2.2 Sensors

In the next section the sensors used are going to be described:

2.2.1 TOPCON HIPERIÓN GPS [1]

In the mobile robotics field, it is difficult to fully know the actual position of the robot in study. This is because the movement of the robot adds uncertainty to the pose, this uncertainty is controlled by the studied algorithms, keeping it from growing indefinitely. It is for this reason that it is necessary to have an external way to guarantee a correct position. This correct position is, in this work, given by a GPS (Global Positioning System).

The GPS is connected to a satellite network which is orbiting orderly around the globe. The GPS needs at least 4 satellites to correctly estimate the actual pose (See Figure 2.2). To compute the position GPS uses two identical pseudo aleatory signals, one generated by the GPS itself, and other sent by satellite. With those signals, it is possible to compute the distance between the GPS and the satellite. Finally with at least three signals received (if the clock is synchronized with the clock of the satellite ([9], p.3) it is possible to triangulate the position of the GPS in the globe with low accuracy (approximately 30 meters), but useful enough in certain applications.

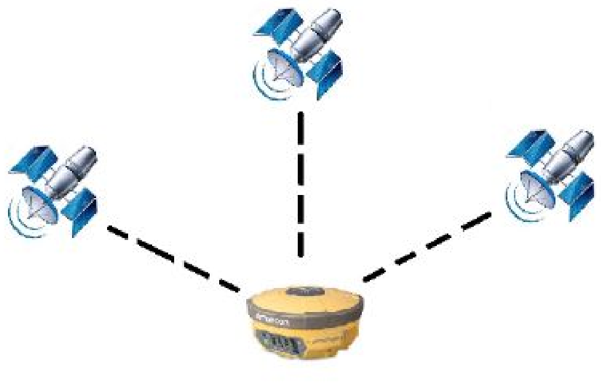


Figure 2.2: Triangulation GPS process.

Another type of GPS in existence is denominated "differential GPS", this one consists in the use of a base device which is capable of obtaining its position with a high level of certainty, using another method which is different to the one described (for example: obtaining the actual position by another sensor with higher accuracy). With this information and the information obtained from the satellite signal comparison, it is possible for this device to calculate the error associated with the measurement. Finally, with the hypothesis that every

GPS in a near radius has the same error or similar, this device sends the error information to the other devices. These devices then subtract the error from the measurement obtained and obtain a new position. With this method, it is possible to reduce the uncertainty to the order of half a meter approximately.

The GPS used in this work is denominated “rtk GPS” from the words “real time kinematic GPS” (2.3).

Figure 2.3: GPS HIPER V.

This GPS consists of two devices of similar characteristics in which one is used as a base station and the other is used as a rover, mounted on the object that is desired to track, which in this case corresponds to the Husky robot. It is possible to connect various rover devices to a single base station. The main advantage of these devices is that it is possible to identify the relative position of the rover from the base station with a low level of uncertainty. To make this possible the base station is configured to compute the phase of the received signal and send this information to all the rover stations connected to it, the rover stations compute the phase of the signal received by them (same delayed signal) and then obtaining the difference on the phase of both signals (that should be zero if the devices are in the same place) it is possible to obtain the relative position with a good level of certainty (in the range of cm.) (Figure 2.4).

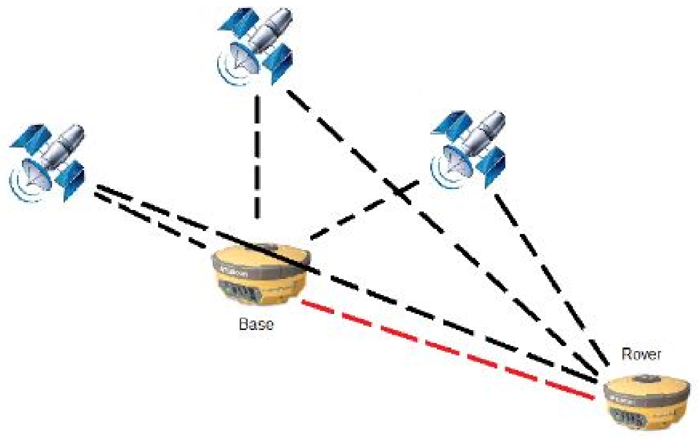


Figure 2.4: Base and Rover system.

2.2.2 Stereo camera (Bumblebee®XB3[2])

This camera helps to obtain information from the front part of the mobile platform in form of an image. This camera is a stereoscopic camera (Figure 2.5), which also gives information about the depth of the objects perceived. The depth is given by the calculation of the difference of the same real object between the two images obtained. Another special characteristic of this camera is that all the pixels of the image are obtained at the same time. With algorithms like ORB-SLAM[6] it is possible to track the trajectory that the robot follows as it moves.

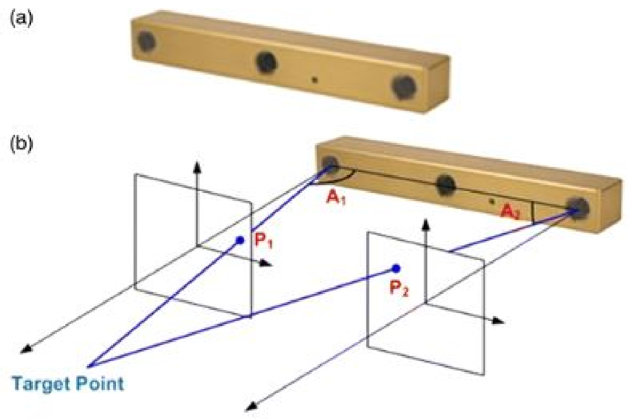


Figure 2.5: Stereoscopic camera.

2.2.3 LIDAR (VLP-16[3])

This sensor corresponds to an active sensor which uses a laser beam of light to detect nearby objects (Figure 2.6). The sensor emits the laser then it bounces on the detected element to finally be detected by the sensor when it comes back. To calculate the distance to the object, the sensor computes the time of flight, obtaining the time that took the laser to go off the sensor and come back. Given the time and the speed of light, it is simple to obtain the distance between the sensor and the object (Figure 2.6).

Figure 2.6: Lidar sensor.

This sensor has a laser transmitter which is mounted on an actuator. This laser has an opening angle, which allows the laser to obtain information from upper and lower elements, not only the elements which are in front of it directly. The actuator has the function to make the emitter spin in 360^o which gives measurements from all the surrounding.

This sensor stores the reflectivity information in a 256-bit data value. This value stores the reflective of the object using values from 0-100 to indicate the percentage of reflection. The values 101-255 indicate the retro-reflector property, in which 255 is an ideal retro-reflector, and lower values, represent a partial retro-reflector(Figure 2.7).

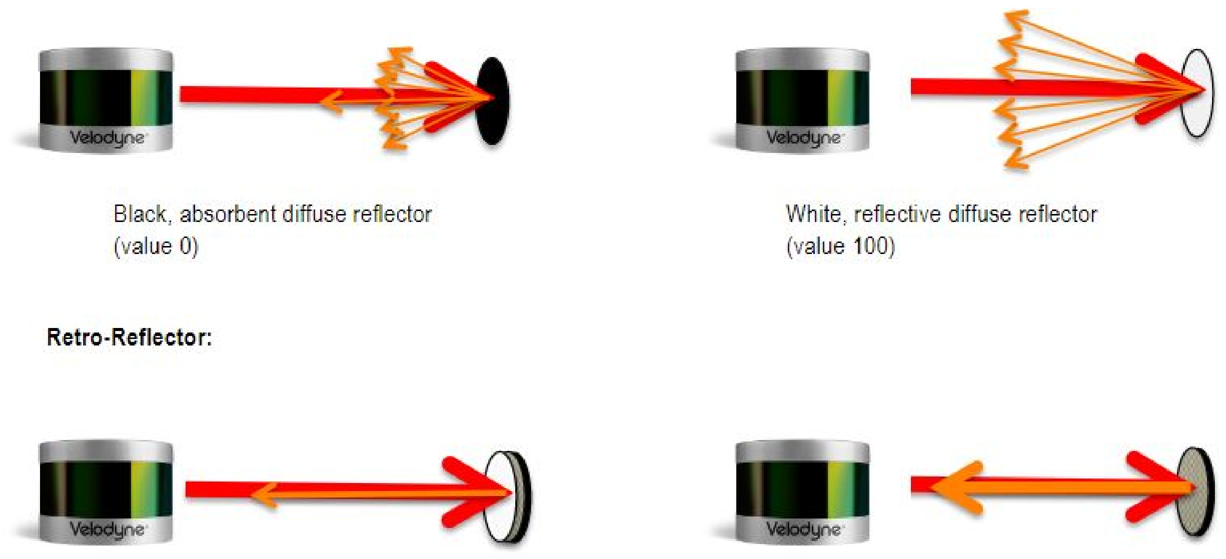

Velodyne
Black, absorbent diffuse reflector
(value 0)
White, reflective diffuse reflector
(value 100)
Retro-Reflector:

Figure 2.7: Lidar sensor measurement method.

2.2.4 RADAR (Acumine 94 Ghz[4])

This is an active sensor which emits a radio signal, the radio signal travels through the environment and collides with one or more objects (2.8). The signal received gives information about the location of the different objects.

Figure 2.8: Radar sensor.

The detection is given by the "echo" of the signal sent. This means that to detect an object it is necessary for the radio wave to bounce on the object. The wavelength of the signal affects the minimum size of a detectable object, this means that the wavelength of the radio signal needs to be smaller than the object detected. To obtain a distance measurement with this sensor, the FMCW (Frequency Modulated Continuous Wave) technique is used. This technique consists of the use of a continuous radio signal that is formed by a sinusoidal modulating signal and a sawtooth carrier signal. Then the distance is given by $t_\mathrm{i} = \frac{cT_\mathrm{d}}{2} f_{b_\mathrm{i}}$

where t_{i} is the distance between the radar and the object i, is the frequency amplitude of the radar sweep, c is the velocity of light, T_{d} is the period of the frequency curve and $f_{b_{\text{i}}}$ is the variance between the frequency received by the reflection of the signal in an object and the carrier signal.

An advantage of the RADAR is the fact that the power of the bounced radio signal is not reflected completely with the encounter of an object. This allows identifying a series of objects located one behind each other. The possibility to detect objects behind obstacles is given because when a radio signal encounters an object part of the energy that is transmitted is reflected and part of it energy continues its path (2.9).

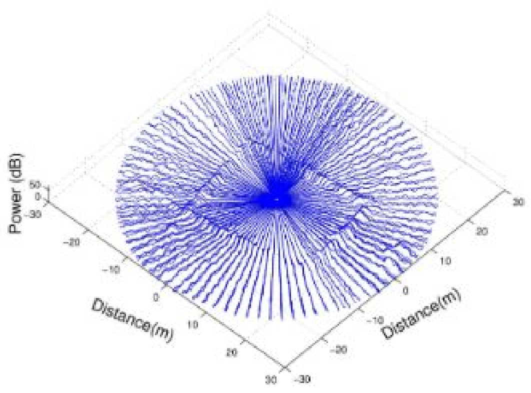


Figure 2.9: Data taken from a Radar in a rectangular room. Taken from [8].

This sensor is greatly affected by materials that do not produce an echo. This means, that objects that absorb radio signals affect negatively the detection capability of this sensor.

2.3 State of the art

In this section, the concept of SLAM algorithm is explained detail. Also, then the algorithm that are used in this work are described.

2.3.1 SLAM

The SLAM problem consists on estimating the localization of the robot in a continuously estimated map given a sensors data that the robot handles. The processes of generating the map and locating itself in the map are realized at the same time.

It is possible to define the initial position of the robot in as the origin $(x, y, z) = (0, 0, 0)$ of the robots coordinate system. Then it is possible to estimate the pose of the robot given the movements that it follows. The estimation of the robot motion is called odometry. At first impression, the use of the rotation of the wheels is a good solution considering the computation load and the quantities of sensors required to do so, but in reality, it is intractable to perform this procedure. This is because the movement of the robot adds uncertainty to the robot pose, in other words, without any external data, the robot's pose estimate would be highly inaccurate.

The procedure of measuring the movement of the robot with the odometry is not reliable, given the uncertainty that this implies. This is given by the fact that the sensed data does not give information of the reality as it is. For example, if it is necessary to know the distance that a wheel traveled when the robot moves it is possible to estimate the angular velocity. Then with the known wheel radius and the time that the wheel spins calculate the distance desired. The problem with this measurement is that the wheels can slide on rough terrain, making the distance calculated bigger than the one actually traveled by the robot. It could also occur that the mentioned problem affects only one wheel making the robot spin changing the heading direction.

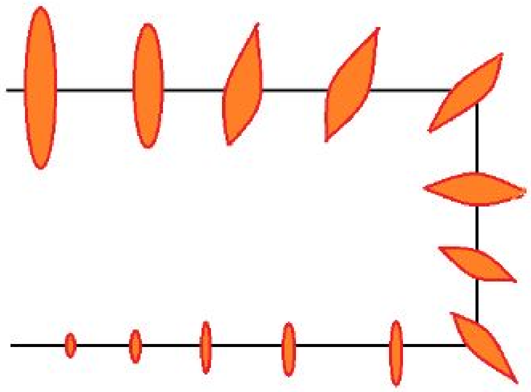


Figure 2.10: Error associated to the movement.

In Figure 2.10 the red ellipses represent the delimited zone where the robot can be found, the orange space is the possible location of the robot in that zone. Note that with every movement the error expands and errors in rotations are often larger than in translation.

It is noticed then that using the odometry by itself, there exist a lot of errors associated with the movement (Figure 2.10), which make the estimated position of the robot uncertain. Is because of this reason that a position given after a certain movement is associated with a certain amount of uncertainty, which means that the position is given by a possibility zone by every movement that the robot makes. It is necessary to notice that not only there exist a probability of correctly estimating the pose, also this probability decreases with every movement because errors are cumulative.

To solve this problem and reduce the zone of the possible pose of the robot in a manageable distribution, various sensors and algorithms are used. These sensors give information about the environment, more precisely, the position of the objects surrounding the robot. With the objects detected it is possible to generate a map of the environment. At the same time, the robot is capable of estimating its own position in the estimated map. Is in this situation that the SLAM problem arises, because it is necessary to simultaneously generate the map and locate itself on the map. The solely mapping and then localizing or localize and then the map would lead to an incorrect result since errors associated with mapping and localization are correlated[10].

It is necessary to mention that the sensors by themselves do not reduce the uncertainty itself. To clearly reduce the robot uncertainty it is necessary to compute algorithms that make it possible to estimate the robot position in a good way as it maps the environment. This is given because of the sensors as it is mentioned on the Bibliographic review, also are susceptible to errors, and as it was stated before, the errors on the localization are cumulative.

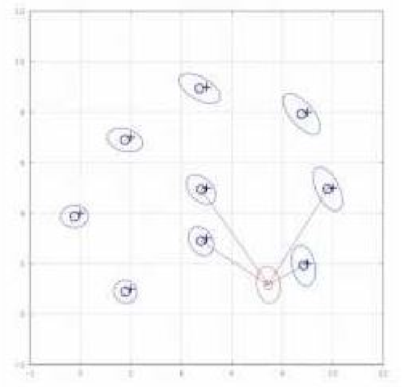


Figure 2.11: Matlab simulated SLAM.
The robot on the down right side of the image detect the 4 marked landmarks, estimating its position. Each landmark and the robot are associated with an uncertainty region, which is represented by the ellipse around each one.

SLAM algorithms differ from each other in the way that the data is delivered as an output (the map and trajectory obtained from computing the algorithm) and the way that the pose is estimated from one state to another, among other things. These algorithms are based on the same principles. The robot recognizes the environment in which it is, estimating a map, and simultaneously estimate its position on that map. The environment is not only sensed, it is also labeled, generating "landmarks" (landmarks are not used by all SLAM algorithms). Each landmark must be differentiated from others landmark or the estimated position could be miscalculated. This miscalculation can occur if the algorithm computes that: it is possible to be in two different positions at the same time because the landmarks detected generate this possibility (2.11). The consecutive detection of the landmark reduces the uncertainty of the robot's pose and the landmark's position in the map. Some of the SLAM algorithms that have been developed will be described in more detail below.

2.3.2 Graph-SLAM [5]

The SLAM problem consists on estimating the pose of the robot and the map in which it is located. Is for this reason that the SLAM problem is referred to the calculation of probabilities or means. Then the path followed by the robot is given by a succession of random variables defined as $X = [x_1, x_2, \ldots, x_n]$ which represent the pose on the map in a given moment, an odometry vector is given too, which has the information of the movements performed by the robot in the trajectory in a given moment, this vector is defined as $u = [u_1, u_2, \ldots, u_m]$. Also, the sensed information of the environment is given, which is defined as the set $Z = z_1, z_2, \ldots, z_m$. Finally the SLAM problem is able to calculate the posterior probability p (equation 1) the estimated position ($x_1\colon x_T$) and the map (m) given the previous position (x_0), the odometry ($u_1\colon u_T$) and the observations ($z_1\colon z_T$) at the given time T.

$$p(x_1\colon x_T, m|z_1\colon z_T, u_1\colon u_T, x_0) \tag{1}$$

This algorithm stores the received information obtained from the environment, with the use of the sensors, and the position of the robot itself in a graph form. Graphs are a structure formed by nodes and their interconnections. The nodes represent the observations, and the interconnections represent the distance between the robot and the given observation, or by the observations themselves. Each node can be distinguished by the information that it holds or by the interconnections that it has with other nodes (Figure 2.12).

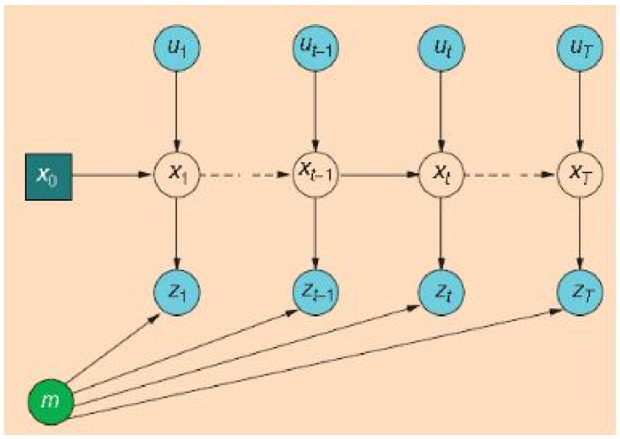


Figure 2.12: Representation of the map generation.
X_0 is the initial state. X_t is the given state, U_t is the input, which means, the action executed by the robot, and Z_t is the observation that corrects the estimated state given the input.

Meanwhile, the robot moves along the path, and it acquires observations, the generated graph begins to expand with each observation set. While the movement of the robot is adequate along the frequency of the data acquisition (the movement allows to correlate sensed data with the stored data), given by the sensors in use, it is possible to add information to the graph by correlating sensed information with the nodes on the graph. With this process it is possible to expand the graph with the new observations, making a graph with more information on the environment. The likelihood equation is given by $I_{\mathrm{i}j}$ (equation 2) where $\Omega_{\mathrm{i}j}$ is the information matrix between node i and node j, $\hat{Z}_{\mathrm{i}j}(X_{\mathrm{i}}, X_j)$ is the prediction of a measurement and $Z_{\mathrm{i}j}$ is, therefore, the measurement.

$$I_{\mathrm{i}j} \ \alpha \ [Z_{\mathrm{i}j} \ - \ \hat{Z}_{\mathrm{i}j} \ (X_{\mathrm{i}}, X_j) \ ]^T \ \Omega_{\mathrm{i}j} \ [Z_{\mathrm{i}j} \ - \ \hat{Z}_{\mathrm{i}j} \ (X_{\mathrm{i}}, X_j) \ ] \tag{2}$$

Given the error on the data that is related to the measurement of the sensor, it is necessary to use a way to minimize the error. To obtain this reduction on the error it is necessary to use the graph already generated and the new observations. To reduce the error on the nodes in the graph, they are compared to the possible new nodes, and it identifies which nodes are already in the graph and which ones are not. To correct the position of the nodes in the graph, the difference on the distances between the nodes that belong to the graph and the new observation are calculated. And to define where to add the new points the position that maximizes the likelihood (equation 3) is computed, this means, the new points are added in a way that the points that belong to the new observation and the graph are best superposed.

$$X^* = argmin_x \ F(x), \ F(x) = \sum_{(\mathrm{i},j)\in C} \mathrm{e}_{\mathrm{i}j}^T \Omega_{\mathrm{i}j} \mathrm{e}_{\mathrm{i}j} \tag{3}$$

Where $\mathrm{e}_{\mathrm{i}j}$ is the error between the estimated measurement and the real measurement, $\Omega_{\mathrm{i}j}$ is the information matrix between a node i and a node j, and X^* is the maximum likelihood.

As the track that the robot follows grows, the graph generated from the observations also grows. A good recommendation to make the data collection is to make sure that the robot makes a closed loop trajectory from time to time. This way, it is possible to correct the distance of the nodes on the graph, this is because when the same node is observed more than one time it is possible to correct accumulated errors in the trajectory, as can be seen in the figure 2.13. In this way, the map of the observations and the trajectory followed by the robot are estimated in the best possible way.

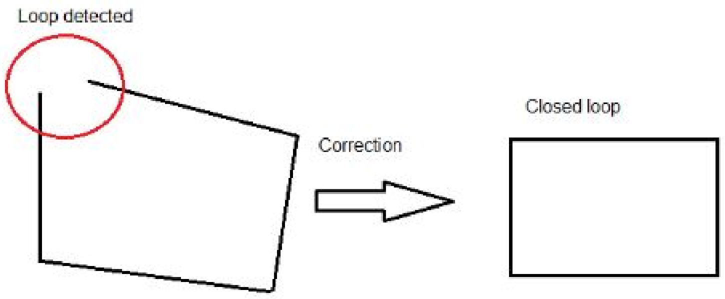


Figure 2.13: Closed loop correction on SLAM.

2.3.3 ORB-SLAM [6]

ORB-SLAM (Orientated FAST and Rotated BRIEF(Binary Robust Independent Elementary Features)-SLAM) is based on feature extraction from images, which means, that the principal sensor used in this algorithm is the stereoscopic camera. This algorithm separates the computations realized, in three different threads (a computational process that can be executed in parallel in a multi-core processor). The first thread is responsible for extracting the features from the images (such as corners) to track the position of the camera, and add new keyframes (frames with information of the movement of the camera and the landmarks detected) to the map, the second generates the local map using the keyframes and landmarks, and the third computes the error reduction when a loop is detected (this means that this thread is used only in a certain areas of the path). With this computational measures, the algorithm makes possible to distribute in a good way the computational load, making it possible to run the algorithm online (in real time).

In first place, it is necessary to establish an initial stage where the robot tries to identify its actual pose. To make this possible, the first image taken is considered as the reference point. To obtain this state, the image is first taken as a planar image (all the feature points belong to the same plane) and as a nonlinear plane (some features belong to a certain plane and others to another parallel plane). A homographic matrix H_{cr} is defined to the planar case and an elemental matrix F_{cr} is defined to the nonplanar case. X_r contains the ORB information of the reference frame, while X_c contains the ORB information of the current frame, the matrix values are computed as it stated in (4). In order to this two matrix, the first landmarks are extracted to compute the SLAM algorithm.

$$X_c = H_{cr} X_r, \quad X_c^T F_{cr} X_r = 0 \tag{4}$$

The name of the algorithm is given by the feature extraction algorithm used to obtain the landmarks, which consist of an algorithm named ORB. This algorithm explores the image and extracts the corners of the objects encountered in the image, identifying each of them by a oriented BRIEF (Binary Robust Independent Elementary Features) descriptor. The BRIEF descriptor consists of a binary string. To generate the string, a set of pixels is selected and compared in a defined order, depending on the intensity of the pixels and the order defined, a 1 (pixel(i)<pixel(j)) or a 0 (pixel(i)>pixel(j)) is assigned. Following this method in another frame is possible to detect the same point even if it is rotated [11]. With this characteristic is relatively simple to find the same corner again in another image, even if the image is rotated with respect to the other. When the same point is detected, and a set of points along with it, it is possible to calculate the translation and rotation between the images, which implies the rotation and translation of the robot itself (2.14).

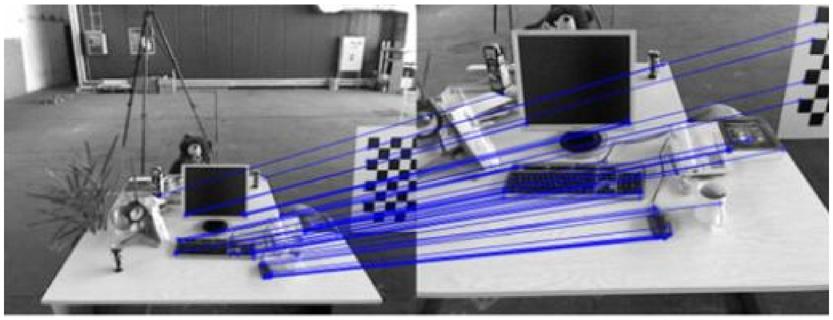


Figure 2.14: Points detected in both images
These points can define the movement that the robot made.

To establish the trajectory of the robot, the algorithm uses a local map and keyframes. The local map contains the information of the environment detected by the image and the features extracted using the ORB feature extraction algorithm, this includes all the keyframes generated and the landmarks detected while the robot moves. A keyframe is generated when the change between frames is significant but makes possible to obtain enough landmarks to estimate the movement of the camera, and therefore, the movement of the robot. The keyframe contains the landmarks detected in the most recent frame which enables the algorithm to compare it with the local map and determine the relative pose between the previous and current keyframe. The points from the local map that are going to be compared with the landmarks from the current keyframe are estimated by the movement of the camera, this allows to estimate the movement of the characteristics in the image. This estimation reduces computational cost because it gives possible places in which the characteristics are going to be found in the next image. Avoiding the computational load that would require exploring all the image searching for a desired characteristic.

At the moment of obtaining new characteristics using the ORB algorithm, the number of characteristics needs to surpass a certain threshold to be considered as a valid data set. The area in which the correspondences are searched can be limited in relation to the movement of the robot. If the quantity of points that fit with each other is below from pre-defined criteria, the searching area increases. To add new sectors, generally produced by the turn of the robot in a curve, it must exist at least twenty new measurement[6] from the last time that a sector was added, detect more than fifty points[6] in the image or being unable to correlate more than ninety percent[6] of the points to points in the reference map.

To successfully add a new map point to the global map a defined quantity of conditions must be satisfied. At first, when a new point is detected it must be present and be consistent in at least three consecutive frames[6]. At second, the point must be observed in at least twenty five percent of the times[6] that the node was estimated to be detected by the sensor. Some points are removed from the global map if they are considered not important, in this way the amount of information handled is optimal.

Finally, to detect a loop, it is necessary to detect it in three consecutive frames[6] the possibility to close a loop, which means, that a point in the global map has been detected again given another trajectory. One time the criterion is fulfilled, the error of the position of the point is calculated to make them fit in the global map. The correction then affects all the nodes in the neighborhood directly connected with the loop, modifying in that way all the points in the loop. Since the nodes are stored in the form of a graph, specialized optimization methods are used to execute these calculations (Figure 2.15).

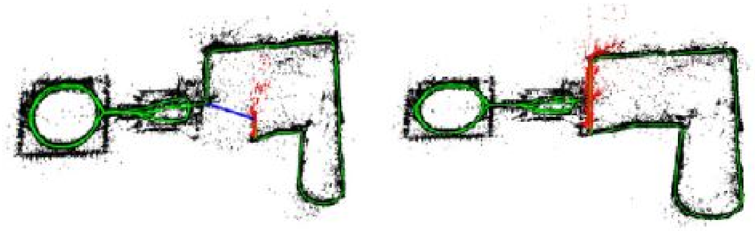


Figure 2.15: Closed loop correction

2.3.4 LOAM (Lidar Odometry and mapping) [7]

This SLAM algorithm is focused principally on the Lidar sensor. This algorithm selects the points to be used, this is necessary because the sensed signals that make possible to track the movement of the robot, can be received at undesired times, given that the received signal could have rebound on a surface. This means, at a given signal there are always two options for the possible location of the object. The first ones are that the perceived object is in fact in the direction in which the signal was received and the other is that the signal bounced in another surface and then was received by the Lidar. The points then are selected to avoid these errors, selecting principally corners and places where the signal is perpendicular to the surface of the object detected.

This algorithm consists of two simultaneously executed parts. On one side there is in execution an algorithm that determines the odometry of the Lidar, which means that it detects and correct the detected points of the sensed data to make a good mapping. On the other side, there is an algorithm in execution that makes the actual mapping and estimates the position of the robot. To handle the information, the graph structure is shown in Figure 2.16.

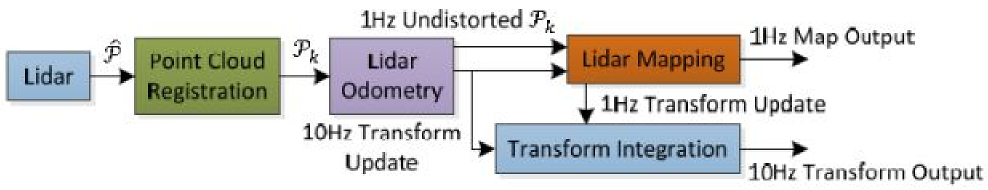


Figure 2.16: Block diagram of the map calculation.

To obtain the odometry, the Lidar data is going to be used as a cloud of measurements, which consist of all the points detected by the sensor. In first place, the points that came from bouncing on planar surfaces and corners, because these are easier to detect. The points near to already selected points are discarded since not much information is provided by them, also planar surfaces that are not perpendicular with the laser are avoided, as the points detected in an occluded section (a section behind an object) are considered not viable. To evaluate the smoothness of a surface the equation (5) is used

$$c = \frac{1}{|S| * \|X^L_{(k,\mathrm{i})}\|} \sum_{j \in S, j \neq \mathrm{i}} \|(X^L_{(k,\mathrm{i})} - X^L_{(k,j)})\| \tag{5}$$

Where S is each consecutive points returned in the same scan, and $X^L_{(k,j)}$ the i point that belongs to the P_k set of features obtained from previous scans. Then the c value is used to determine the edges or the corners detected. When c take a high value(a defined threshold) it corresponds to be an edge, when c has a low value it corresponds to be a planar point, which means a point in a plane perpendicular to the laser.

When the selected points are well defined, it is possible to work with more than one cloud of points, to be able of estimate the Lidar odometry. At the beginning, points that belong to both clouds sets are detected. If the movement of the robot is slow enough relative to the frequency of the data received by the sensor, a large number of points should be located in the studied clouds. When the data related to both clouds is determined, it is possible to estimate the translation and rotation of the Lidar in the map, which is the same as the location of the robot in the created map. The distance associated to the edges d_ε is represented by equation (6) while the distance between planes d_H are given by equation (7) where $\hat{X}_{(k+1,\mathrm{i})}, \bar{X}_{(k,j)}, \bar{X}_{(k,l)}$ and $\bar{X}_{(k,m)}$ are the coordinates of the points i, j, l and m in the L set. Minimizing this distances will lead to a well-generated map.

$$\mathrm{d}_\varepsilon = \frac{|(\hat{X}^L_{(k+1,\mathrm{i})} - (\bar{X}^L_{(k,j)}) \times (\hat{X}^L_{(k+1,\mathrm{i})} - (\bar{X}^L_{(k,l)})|}{|(\bar{X}^L_{(k,j)} - (\bar{X}^L_{(k,l)})|} \tag{6}$$

$$\mathrm{d}_H = \frac{|(\hat{X}^L_{(k+1,\mathrm{i})} - \bar{X}^L_{(k,j)}) * ((\hat{X}^L_{(k,j)} - \bar{X}^L_{(k,j)}) \times (\hat{X}^L_{(k,j)} - \bar{X}^L_{(k,m)}))|}{|(\hat{X}^L_{(k,j)} - \bar{X}^L_{(k,j)}) \times (\hat{X}^L_{(k,j)} - \bar{X}^L_{(k,m)})|} \tag{7}$$

Finally, the mapping algorithm (Figure 2.17), is executed with a slower frequency than the previous algorithm. This algorithm is the responsible for generating the global map. To generate the global map consecutively data clouds are received after they are “cleaned” selecting the desired points in the way stated previously. Using the superposition of the points (in the data clouds received), the disposal of redundant points data, and the previously estimated position of the robot, the desired map is generated. At the same time, the actual estimated position of the robot in the map is calculated. The position transform can be estimated as a linear transform, as equation (8) propose.

$$T^L_{(k+1,\mathrm{i})} = \frac{t_\mathrm{i} - t_{k+1}}{t - t_{k+1}} T^L_{k+1} \tag{8}$$

Where $T^L_{(k+1,\mathrm{i})}$ correspond to the pose transform between $[t_{k+1}, t]$ of a point i, given the actual time stamp t, the time stamp of the point i, t_i, the started time of the Lidar swap t_{k+1}, and the rigid motion T^L_{k+1} of the Lidar.

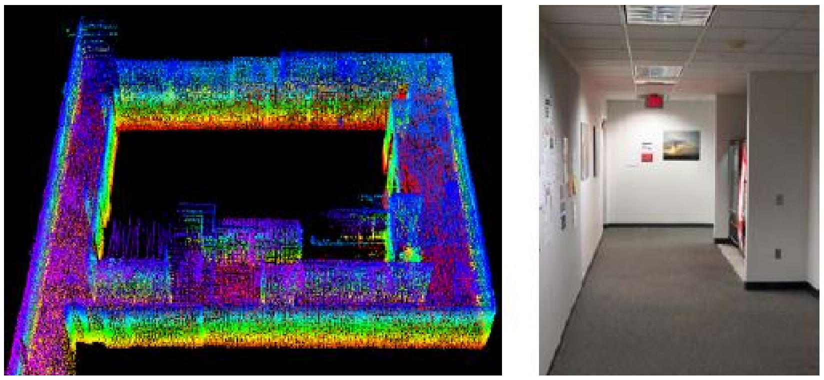


Figure 2.17: Map generated using LOAM method.
The picture on the left is the generated map, the different colors refer to different heights.
the picture on the right is the real corner (lower right) of the generated map.

Chapter 3

Methodology

In this work, a comparison between two different SLAM (Simultaneous Location And Mapping) algorithms is going to take place, using a ground truth. Therefore this work will be split into three different stages. The first one consists of sensor connections (these sensors correspond to the ones explained in the Bibliographic review), in this part, it will be secured that the sensors are transmitting the information that is wanted. In the second part, a remote control robot will be taken through a pre-defined path, and the sensors will collect data from the surroundings. In the last phase, the data sensed from step two will be used to generate a trajectory and map with the SLAM algorithms, to finally compare with the GPS data and see which one improves better.

3.1 Sensors connection

The majority of the sensors used in this work have been previously configured by other students or the robotics team. The only sensor which needed to be re-studied was the rtk GPS Hyper V. This sensor would be used as the ground truth of the robot by the reason mentioned in the Bibliographic review. To make the sensor work there will be a specific time defined to work on it.

In this work, the ROS (Robot Operative System) software is going to be used. This software helps to control the robot by giving a defined way on how to exchange information between sensors and process. To achieve this, each control or calculation process is executed in a "node", this allows to differentiate all the process in execution. The execution of these nodes is controlled by the ROS environment. To exchange information between nodes, a publisher and a subscriber are defined. The subscriber creates a message with a certain structure (for example a set of values with their respective names) and then publish it as a topic (an identifier to the message published), then the subscriber can see that topic and extract the given information from it. With this structure, it is possible to handle all the complex processes that a robot executes in a more reliable way.

The sensors used in this work are the ones mentioned in the Bibliographic review. All the sensors will be connected to a notebook, which, with the help of the ROS software, is going to be the dataset collector. This notebook not only works as dataset collector, also is the central part of the whole system, making it possible to communicate between the different parts of the robot in movement. The ROS program controls the different sensors and takes the data from them generating a re-running simulation, which means, that once the data is recorded it is possible to run it as if the robot was moving along the path defined, without really moving the robot.

3.2 Collecting data

When all the sensors are correctly connected to the notebook and configured, so the data is taken in a good manner, it will be time to collect data from the surrounding. To obtain the data a trajectory in which the GPS sensor will have the best performance possible, will be defined. In this trajectory, the different sensors would be running, and the data will be stored in the notebook for later use of the SLAM algorithms.

3.3 Computing The SLAM algorithm

When the dataset is defined and recorded, it will be the time to execute the SLAM algorithm and compare the estimated computed trajectory with the ground truth. In the ROS community there exists different packages designed by students or professors to freely use with the corresponding recognition. The two algorithms used in this work are going to be extracted from these sources.

It is important to configure the different topics in order to compute the algorithm. Even when, the use of an already created algorithm makes things a lot easier, the process of making an algorithm work can take some time too. The time of configuration is undefined, because it depends on many factors, as the compatibility with ROS version or different drivers. With the correct implementation of the algorithms, a path and a map that the robot should have followed according to the data sensed will be obtained.

3.4 Comparison with ground truth

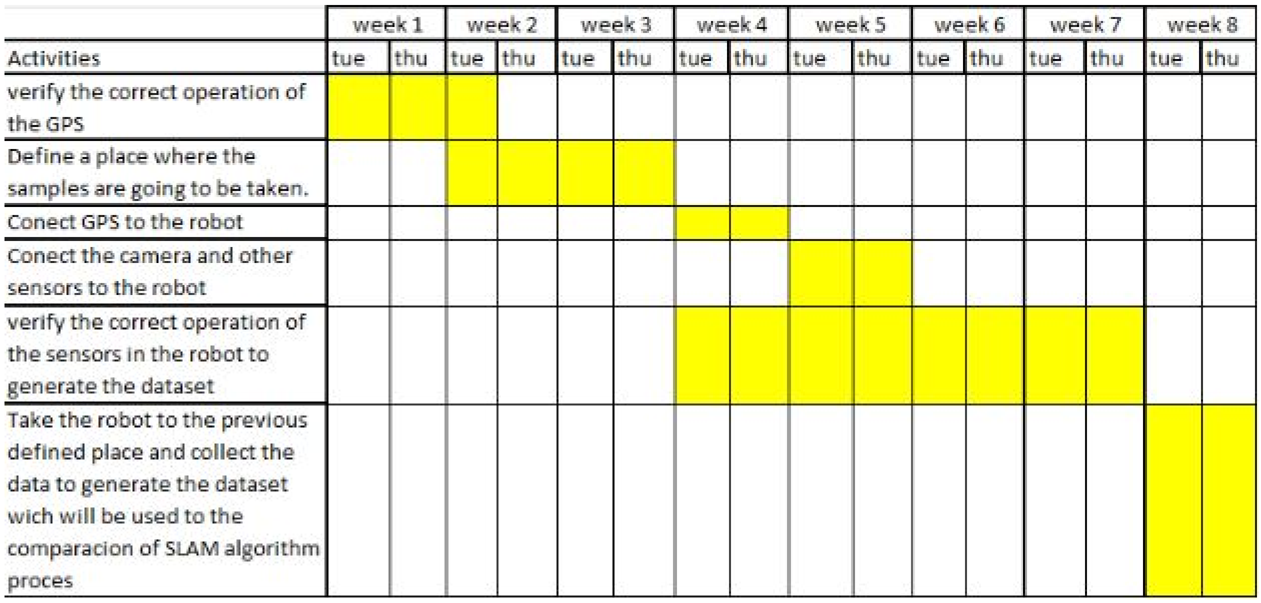
With the GPS trajectory, a ground truth of the robot will be defined. This one should be the most trustworthy of all the possible trajectories obtainable by SLAM, this is the reason it is called ground truth. The two trajectories obtained by the two SLAM algorithms will be compared with this trajectory. To be sure that the trajectory described by the GPS is trustworthy it will be superposed in the map (using Google Maps for example) and see if it really fits the real walkthrough.

To compare the different trajectories is necessary in first place to be able to superpose the two trajectories in the best way possible. This considers the rotation of the track and also the scale. This is necessary because some SLAM algorithms generate a result in which the distance between elements is not well defined. For example, the distance is given without units, this means that positions are relative in scale, i.e. the distances in meters or centimeters are not defined until it is specified with extra data.

When the trajectories are well related it is possible to compute the difference obtained in the trajectory by Euclidean distance or other distance method selected. The one with less distance should be the one with better performance. In other instances the computational time used is also important. This, is given that the ideal for these algorithms is to be used online, so the robot can know its pose in real time. The algorithms used should be designed to operate in real time, therefore it is necessary to take this into consideration when the discussion about performance takes place.

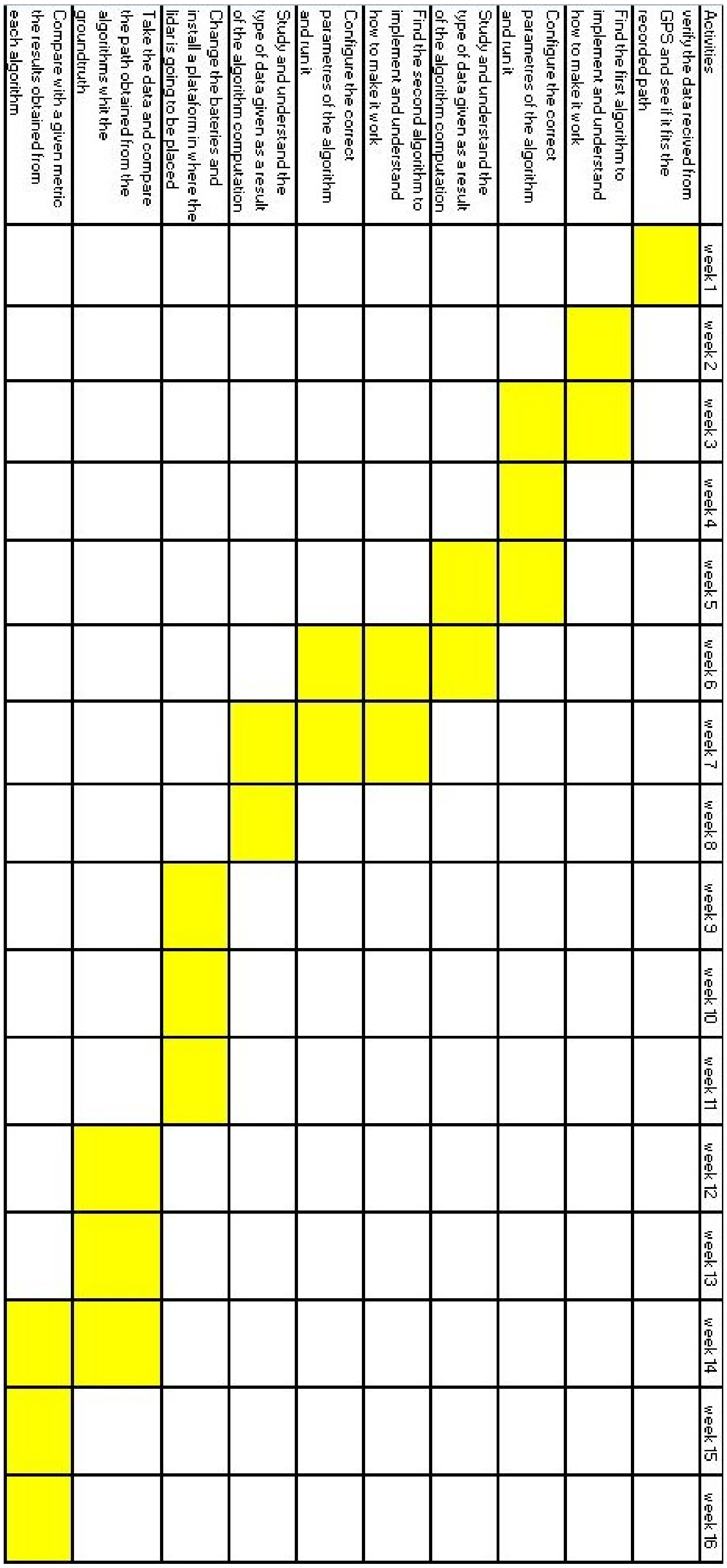
With the information obtained from the comparison, it would be possible to tell which algorithm gives better performance. This work should help then to compare new algorithms that could be implemented in the future, explaining clearly the necessary work to do and how to do it.

3.5 Working Schedule



	week 1		week 2		week 3		week 4		week 5		week 6		week 7		week 8	
Activities	tue	thu	tue	thu	tue	thu	tue	thu	tue	thu	tue	thu	tue	thu	tue	thu
verify the correct operation of the GPS	🟨	🟨	🟨													
Define a place where the samples are going to be taken.			🟨	🟨	🟨	🟨										
Conect GPS to the robot							🟨	🟨								
Conect the camera and other sensors to the robot									🟨	🟨						
verify the correct operation of the sensors in the robot to generate the dataset							🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨		
Take the robot to the previous defined place and collect the data to generate the dataset wich will be used to the comparacion of SLAM algorithm proces															🟨	🟨

Figure 3.1: Working schedule first semester.



Activities	week 1	week 2	week 3	week 4	week 5	week 6	week 7	week 8	week 9	week 10	week 11	week 12	week 13	week 14	week 15	week 16
verify the data recived from GPS and see if it fits the recorded path	🟨															
Find the first algorithm to implement and understand how to make it work		🟨	🟨													
Configure the correct parametres of the algorithm and run it			🟨	🟨	🟨											
Study and understand the type of data given as a result of the algorithm computation					🟨	🟨										
Find the second algorithm to implement and understand how to make it work						🟨	🟨									
Configure the correct parametres of the algorithm and run it						🟨	🟨									
Study and understand the type of data given as a result of the algorithm computation							🟨	🟨								
Change the bateries and install a plataform in where the lidar is going to be placed									🟨	🟨	🟨					
Take the data and compare the path obtained from the algorithms whit the groundtruth												🟨	🟨	🟨		
Compare with a given metric the results obtained from each algorithm														🟨	🟨	🟨

Figure 3.2: Working schedule second semester.

Chapter 4

Generating the data set

To be able to execute a SLAM algorithm and obtain an estimated trajectory from it, it is necessary to process data taken from different sensors. Different SLAM algorithms work processing data from a different kind of sensors, for this reason, the data set generated will contain information of different kinds of sensors. In this way, the possibility to compute different algorithm with the same data set is given. The sensors used in this work correspond to a camera, a Radar, and a Lidar, to sense the environment, and a GPS to track the movement of the Husky robot.

To handle the movement of the robot, control the sensors and record the data, the ROS software [12] will be used. This software is specially designed to work with robot environments, making it easier to control the flow of information between the different processes executed. The ROS software manages the computational processes in form of nodes which make computational calculations and, publish or subscribe to, a topic. A topic is a defined structure which stores data. A publisher modifies this data and a subscriber reads it, in this way it is easier to manipulate the data flow. ROS allows the user to work with the programming languages "python" and "c++". While python is easier to use to generate a program, it is sometimes necessary to use c++, to obtain better computational performance. In this work, python will be used in most cases, since the programs created aren't complex enough to require the use of c++ for performance.

4.1 GPS configuration

To verify the solution given by a SLAM algorithm it is necessary to compare the result obtained with a truthful data. This truthful data is denominated ground truth. The ground truth can be generated by different methods, in this work, the ground truth will be given by an rtk GPS 2.3. This type of GPS as it was stated in the bibliography, reduces the uncertainty of the location of a rover GPS by correcting the error from it. To accomplish this correction data is given from a base station GPS. The base station is used as a reference point, therefore it is recommended to keep it static while the rover moves around. The rover is the GPS used to track the position of the desired object, in this case, the Husky robot.

To record the data from the GPS it is fundamental to establish a way to extract the data from the GPS and store it in some way. As it was mentioned before this process will be controlled by the ROS software. A package created with the c++ program will be used to extract the information from the GPS. To do this, a serial port associated with a Bluetooth communication between the GPS and the computer used is used.

4.1.1 Connecting the GPS to the computer

To establish communication between the computer and the GPS Bluetooth communication will be used. For this reason, it is necessary that the computer used has a Bluetooth module or an external Bluetooth connector. To configure the Bluetooth connection the software Bluetooth Manager from the Ubuntu operating system will be used, and it will be configured to connect through a serial port. When the computer connects successfully to the GPS, the GPS reproduce a sound message saying "Bluetooth connected".

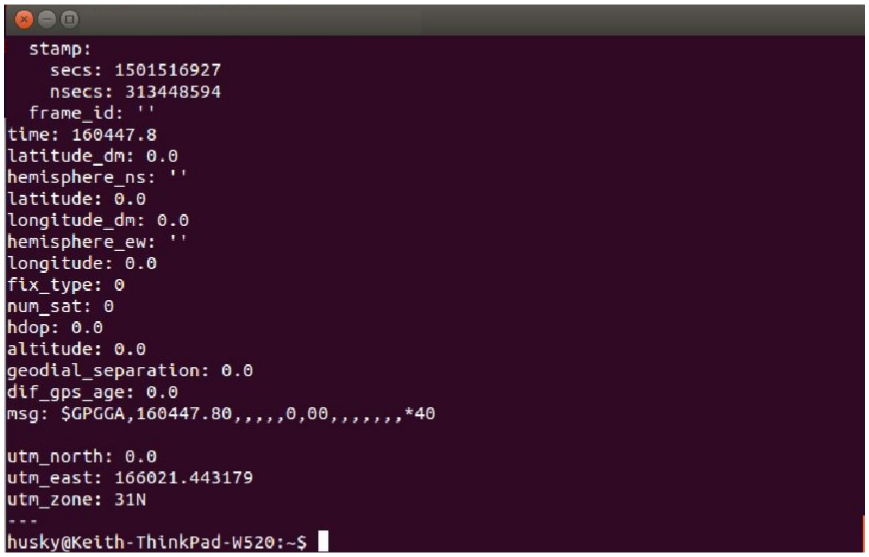
A package developed by the AMTC laboratory staff is used to correctly extract the data from the GPS and publish it in a ROS environment. The package, among other things, include the message type created to handle the information of the GPS, and the program that reads the data from the serial port, define variables of the information available, and publish it as a topic named "/rtk/GGA_message". to install a package, [13] can be followed step by step.

To extract information from the GPS, the serial port used, identified as "/dev/rfcomm0". these ports have administrative privileges, so it is mandatory to grant permission to the actual user to being able to extract the desired data. To grant the permissions it is necessary to execute in the console the command: "sudo chmod 777 /dev/rfcomm0".

To verify that the data is being published correctly in the ROS environment, the program "rtk.launch" of the package installed will be run with the console. To execute the program the command used in the console is: "roslaunch sensor_topcon_hiperv rtk.launch" where "sensor_topcon_hiperv" corresponds to the name of the package used. After the correct execution of the program, the topic mentioned should be published, to view the data received the command in console "rostopic echo /rtk/GGA_message" is used and the figure 4.1 should be displayed.

4.1.2 Verifying the data

To check that the data is received according to what is expected, a small route is defined and followed, the data is recorded and then displayed in Google maps. To achieve this it is required to configure the base and rover GPS station. To configure this options the GPS counts with programs that work with Windows OS, but the toolbox in which the GPS was bought also includes a portable device with this software. To establish the rtk function a job will be created with the software "Topcon Magnet Field", in which the rover and the base are defined, if the configuration is correct, the rover GPS should reproduce a sound effect



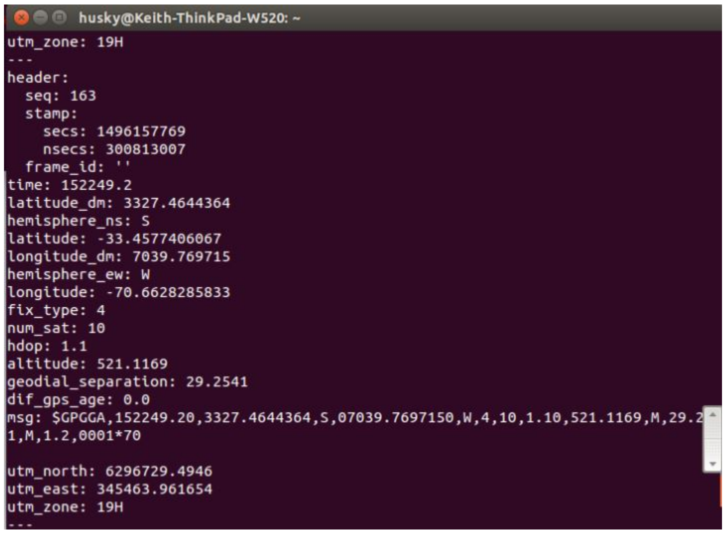
```
  stamp:
    secs: 1501516927
    nsecs: 313448594
  frame_id: ''
time: 160447.8
latitude_dm: 0.0
hemisphere_ns: ''
latitude: 0.0
longitude_dm: 0.0
hemisphere_ew: ''
longitude: 0.0
fix_type: 0
num_sat: 0
hdop: 0.0
altitude: 0.0
geodial_separation: 0.0
dif_gps_age: 0.0
msg: $GPGGA,160447.80,,,,,0,00,,,,,,,*40

utm_north: 0.0
utm_east: 166021.443179
utm_zone: 31N
---
husky@Keith-ThinkPad-W520:~$
```

Figure 4.1: Topic of the data received from the GPS.

that says "rtk fixed" when the rtk is available and "rtk lost" when rtk is no longer available.

To follow the path and record data correctly it is required to locate the base station in a good place since it is used to correct the errors of the rover. Therefore it is mandatory to locate the base in a place that is assured a satellite connection. This is necessary for the GPS to determine it‘s own position, and to have a good radio communication with the rover GPS to share the data correction. To fulfill these requirements a high place without a roof is recommended as a place to locate the base.

If the configuration is realized in a good way, the message received from the GPS should look like figure 4.2. This message gives, among other information, the latitude and longitude position of the GPS, the number of satellites detected in each measurement and the GPS mode (Table 4.1).



```
husky@Keith-ThinkPad-W520: ~
utm_zone: 19H
---
header:
  seq: 163
  stamp:
    secs: 1496157769
    nsecs: 300813007
  frame_id: ''
time: 152249.2
latitude_dm: 3327.4644364
hemisphere_ns: S
latitude: -33.4577406067
longitude_dm: 7039.769715
hemisphere_ew: W
longitude: -70.6628285833
fix_type: 4
num_sat: 10
hdop: 1.1
altitude: 521.1169
geodial_separation: 29.2541
dif_gps_age: 0.0
msg: $GPGGA,152249.20,3327.4644364,S,07039.7697150,W,4,10,1.10,521.1169,M,29.2
1,M,1.2,0001*70

utm_north: 6296729.4946
utm_east: 345463.961654
utm_zone: 19H
---
```

Figure 4.2: Data received from the GPS.

GPS mode identifier	GPS mode
0	Fix not available
1	GPS fix
2	Diferential GPS fix
3	PPS fix
4	Real Time Kinematic
5	Float RTK
6	Estimated (dead reckoning)
7	Manual input mode
8	Simulation mode
9	Added from SD card

Table 4.1: GPS modes.

With the data recorded successfully, it is required to confirm in some way that the data saved correspond to the actual path traveled.

Verifying the test trajectory

To check the data saved from the GPS a simple subscriber is created to generate an excel file (.xls) to handle the data in a better way. This excel can be used to create a Google map path, with the latitude and longitude information extracted from the GPS. The mode of the GPS and the quantities of satellites can also be known and added to the excel. As it is seen in figure 4.3, the path recorded belongs in most of the trajectory to the actual path, the errors are given by the absence of rtk in certain areas.

Wwith this procedure, it is assured that the data is correctly recorded and that it corresponds to the information desired. On the other hand, the data recorded shows that the data has errors associated. Since this data will be used as a ground truth it is mandatory to establish a way to reduce these errors to the minimum, the most desired situation is the absence of errors. To accomplish this objective a route where rtk is always available should be defined if the data is handled solely by this method.

As an alternative method, it is also possible to record the data directly in the GPS (base and rover) memory card, extract it and post process it if the rtk is not available. To save the data in each device, it is required to press three times consecutively the power on/off button, if it is recording the device should emit a beep noise time to time. To extract the data from the GPS, the Topcon tools can be used. Another way to extract the data from the GPS is to simply remove the memory card and insert it into a computer with the corresponding adapter. The data file should be called "logXXYYZZZ.tps" where XX is the month that the data was recorded, YY is the day that the data was recorded and ZZZ is a unique identifier to differentiate each recording of the day.

The data recorded will be processed with the Topcon software "Topcon tools". In this software, a job is created to being able to import the data recorded from the GPS. This data is imported as a static value, which means that all the points stored are used to define

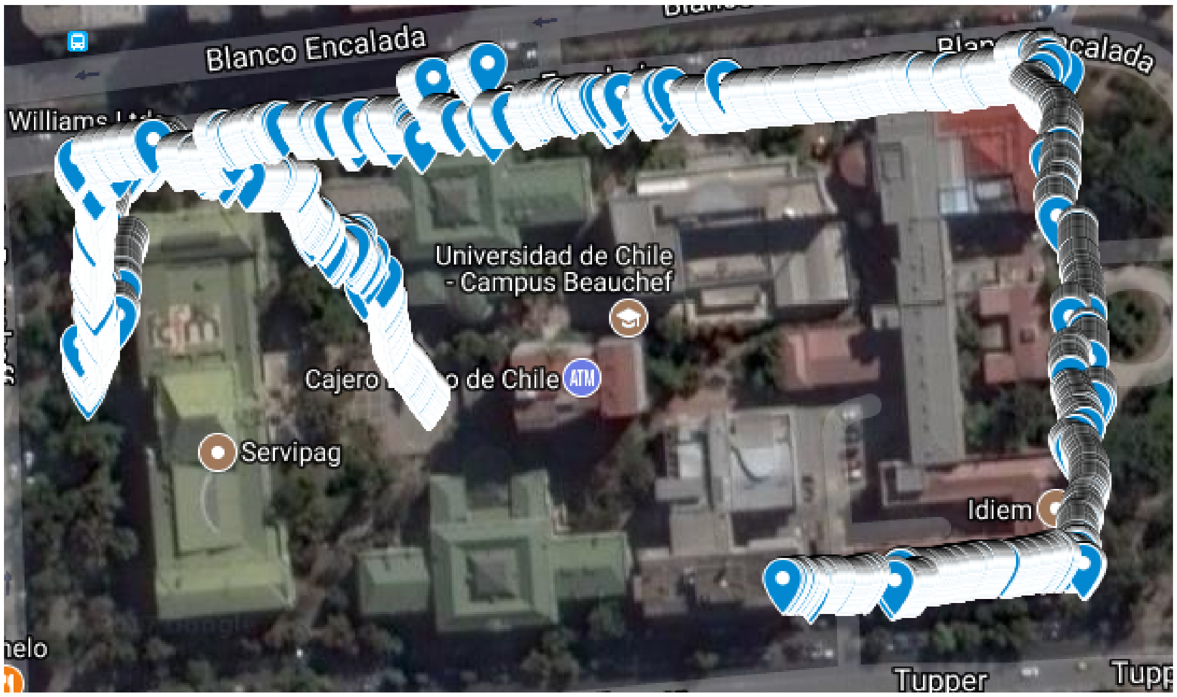


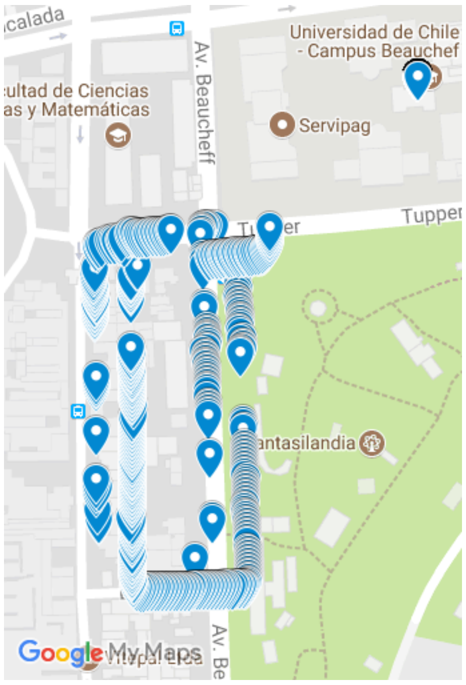
Figure 4.3: Data received from the GPS displayed on Google maps.

a single point (useful to define the base), but it also can be changed to "kinematic" which display each point of the trajectory followed (useful to check the rover movements). These datasets contain extra data as the time windows that the GPS were recording the data and the satellites that were available to track at each time.

When the data is correctly loaded, it can be exported to a ".csv" file, which can be used to plot the information in Google maps. In this way, it is possible to compare between this method and the method described before that takes the data from the GPS using ROS. The figure 4.4 shows the data before the post-processing and figure 4.5 shows the processed data. It can be noted that most of the points where correctly fixed, but some errors remain.

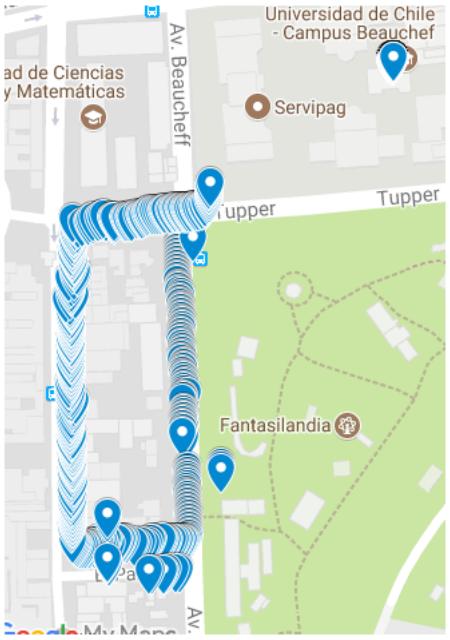
Defining a final trajectory

It is important to take into consideration some characteristic to be able to take the GPS in a way that it can be used as a ground truth. It is important that the base is located in a place that is possible for it to obtain enough satellites on sight to determine its position. Also is important that the base position makes possible to transmit that information to the rover. To guarantee satellite coverage it is essential to locate the base station in an open place, preferably a high place to not be blocked by the surrounding buildings. To establish correct radio communication between devices it is necessary that the rover is near enough to the base station, and that the surrounding buildings of the path traveled do not block the radio communication. In the same way, the buildings could interfere with the satellite communication with the rover, therefore it is necessary to define a rural path that allows to reduce these problems.



calada
Universidad de Chile - Campus Beauchef
ultad de Ciencias as y Matemáticas
Av. Beaucheff
Servipag
Tupper
antasilandia
Av. Be
Google My Maps

Figure 4.4: Data recorded on the GPS memory card displayed on Google maps without post processing.



Universidad de Chile - Campus Beauchef
ad de Ciencias y Matemáticas
Av. Beaucheff
Servipag
Tupper
Fantasilandia
Google My Maps

Figure 4.5: Data recorded on the GPS memory card displayed on Google maps with post processing.

With all the information in consideration, the better area to record the data would be a zone that allows to locate the base in a high place and also have small buildings.

4.2 Collecting the data

To collect the data , in first place it is required, to mount and connect all the sensors on the husky robot (see Figure 4.6). To transport the GPS on the Husky an arc made from aluminum profiles is mounted on the Husky. When the sensors are correctly connected and it is verified that the data is recorded correctly, the robot is taken to the defined place to follow the path and record the data.

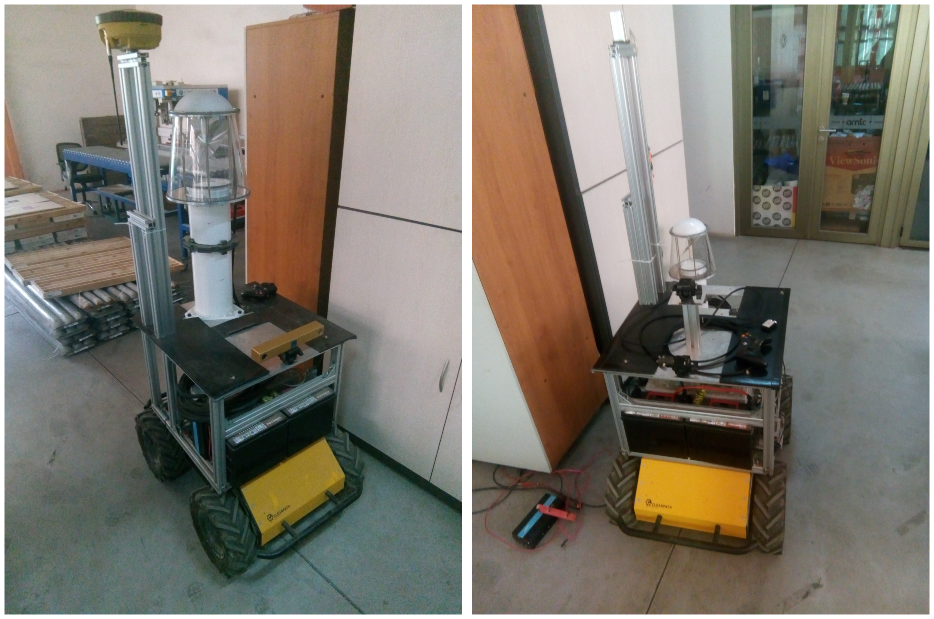


Figure 4.6: Husky mobile platform used torecord the data-set.

It is possible to notice in Figure 4.6 that the aluminum structure was implemented to mount the GPS on the husky. To make a firm structure, the use of pins was necessary, the aluminum profiles were designed for that purpose, but to fit the pins it was necessary to modify the profiles with mechanical tools. The modification consisted on to generate the pin thread to join the different profiles and make the final structure. These aluminum structures consisted of 2 different arcs, one small and one big. These two sizes where implemented considering the possibility to freely mount the radar without interfering with the GPS signal. In this way, if the Radar is mounted, the big arc (Figure 4.7) is used, and if the Radar is not mounted, the small arc (Figure 4.8) is used. Also, a transform of coordinates is implemented and published as a topic in ROS (Figure 4.9), in this way the relative position of the sensors can be known in every moment.

4.2.1 Test data

Before taking the husky to the place where the final data-set is going to be recorded, a test data set is taken near the faculty of Engineering of the University of Chile. Active sensors in this test where the GPS, the stereo camera, and the radar. The data was stored in memory using the command "rosbag record -a", which takes all the topics published and

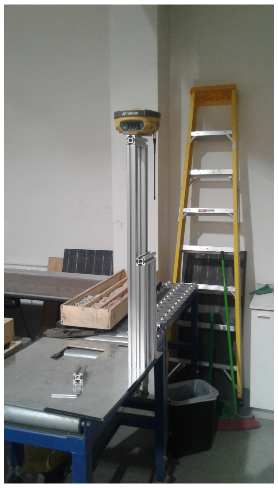


Figure 4.7: Big arc with the GPS mounted.

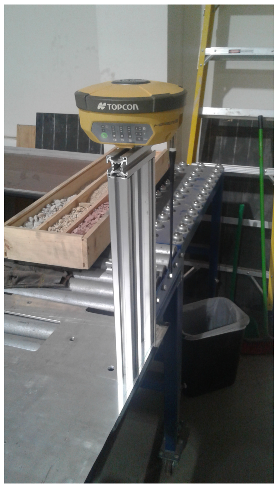
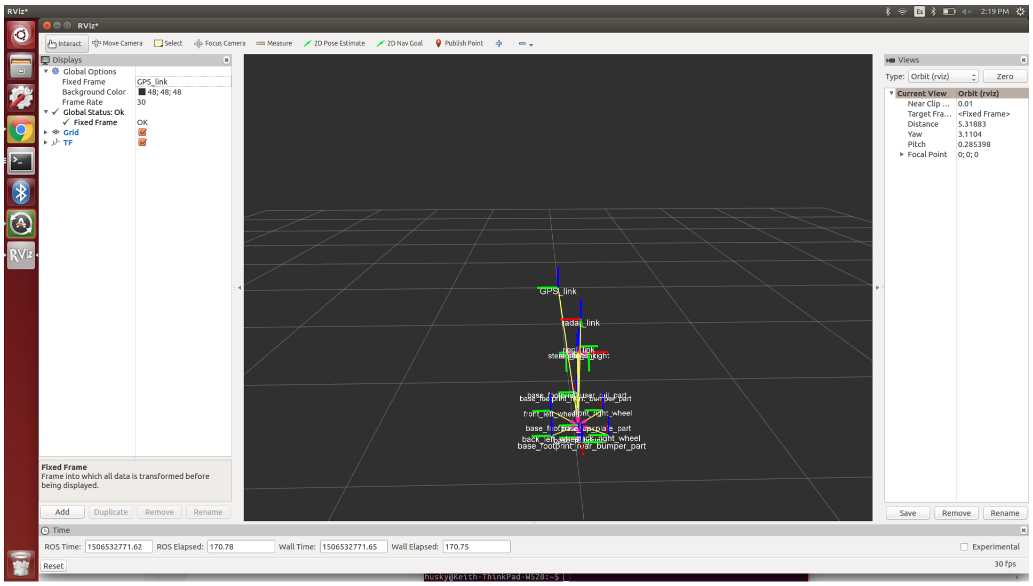


Figure 4.8: Small arc with the GPS mounted.

save them in a bag file. A bag file is a structure that can contain all kind of messages with its respective timestamp, this allows to reproduce the data as if it was taken in real time The bag file contains the images of the stereo camera, and the messages of the GPS, between other information. The bag file can be used later, to play the information from the topics as if the sensors were taking the data. The stereo camera images were used to run the ORB-SLAM2[14] algorithm, and the results are displayed in Figure 4.12. The GPS data was plotted and the results from the Figure 4.10 were obtained.

The ORB-SLAM2 algorithm is a ROS package which uses images to estimate the robot odometry [6]. Given the stereo camera, it is possible to estimate a real distance to locate each landmark and the camera pose. The landmarks used are corners(Figure 4.11), these are obtained using the ORB detection and describer algorithm, and each landmark is characterized by the BRIEF descriptor. Each corner can be detected in different scales of the image. Identifying the same points between different frames allows it to estimate the odometry of the camera, therefore the odometry of the robot is estimated too. The corners detected are used as landmarks and they are used to generate the map, the location of the keyframes form



Global Options	
Fixed Frame	GPS_link
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
TF	✓

Fixed Frame
Frame into which all data is transformed before being displayed.

Type: Orbit (rviz)

Current View	Orbit (rviz)
Near Clip ...	0.01
Target Fra...	<Fixed Frame>
Distance	5.31883
Yaw	3.1104
Pitch	0.285398
Focal Point	0; 0; 0

Time
ROS Time: 1506532771.62 ROS Elapsed: 170.78 Wall Time: 1506532771.65 Wall Elapsed: 170.75 Experimental
30 fps

```
husky@Keith-ThinkPad-W520:~$
```

Figure 4.9: Coordinate system.

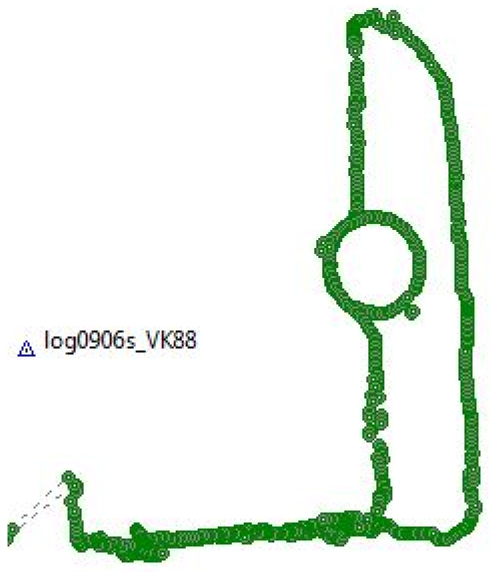


Figure 4.10: Results from the test data GPS.

the path that the robot followed(Figure 4.12).

4.2.2 Verifying the final trajectory

With the test data, it was possible to conclude that not all the topics generated are useful to the data-set, instead, they tend to use great hard disc space and process speed, hence only certain topics were selected to be recorded in the final trajectory. It is important to notice this because the computer used to record the data-set has limited disc space. Also,

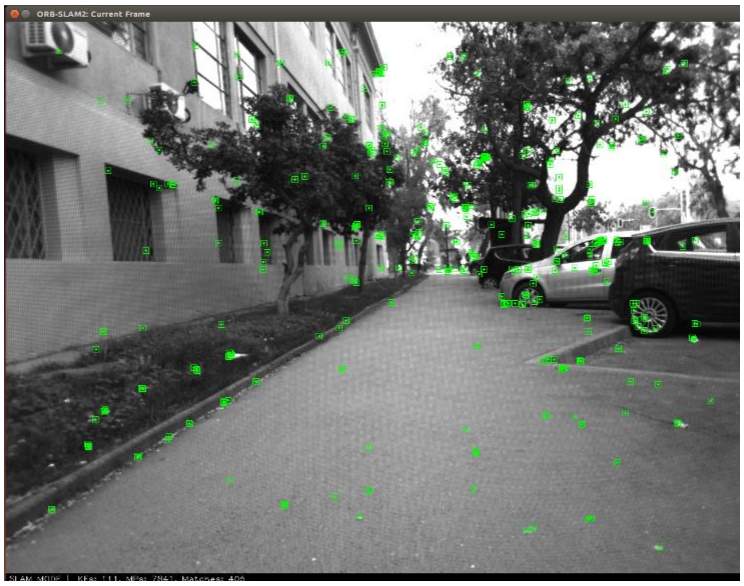


Figure 4.11: Image used in ORB-SLAM.
The green squares represent the points detected with the ORB algorithm.

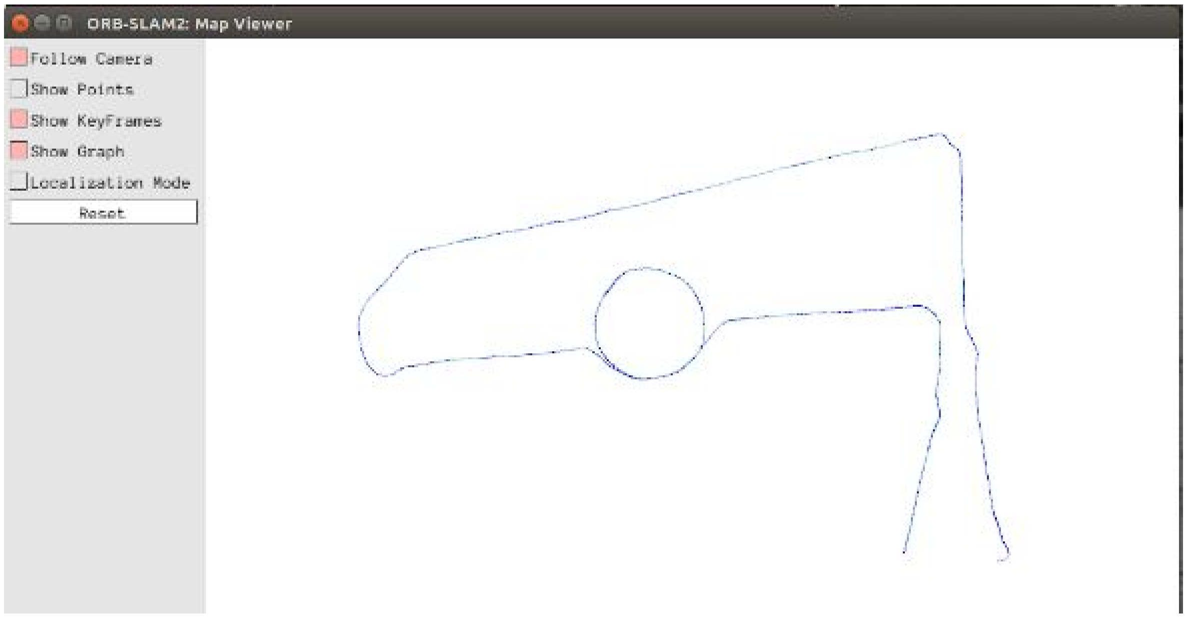


Figure 4.12: Path generated by ORB-SLAM.

the fact of recording a lot of topics generate errors in the sampling frequency of the recorded data of the sensors. Finally, the Radar was discarded from the data-set since the high power consumption of it affected the data-sampling, producing the power off of the camera.

The topics used to generate the bag file were the ones are seen in the Table 4.2

Finally, the husky was taken to the selected place to record the data. This place was selected for the low quantity of high buildings in the surroundings which led to better GPS performance. The GPS data is seen in Figure 4.13 which corresponds to the path taken. In

Sensor	Topic
Stereo Camera	$/stereo/left/image_raw$
	$/stereo/right/image_raw$
Transformed Locations	/tf
GPS	$/rtk/GGA_message$
Husky Odometry	/husky/data/encoders

Table 4.2: Topic used in the data-set.

this way, it is proved that the GPS can be used as a good ground truth.

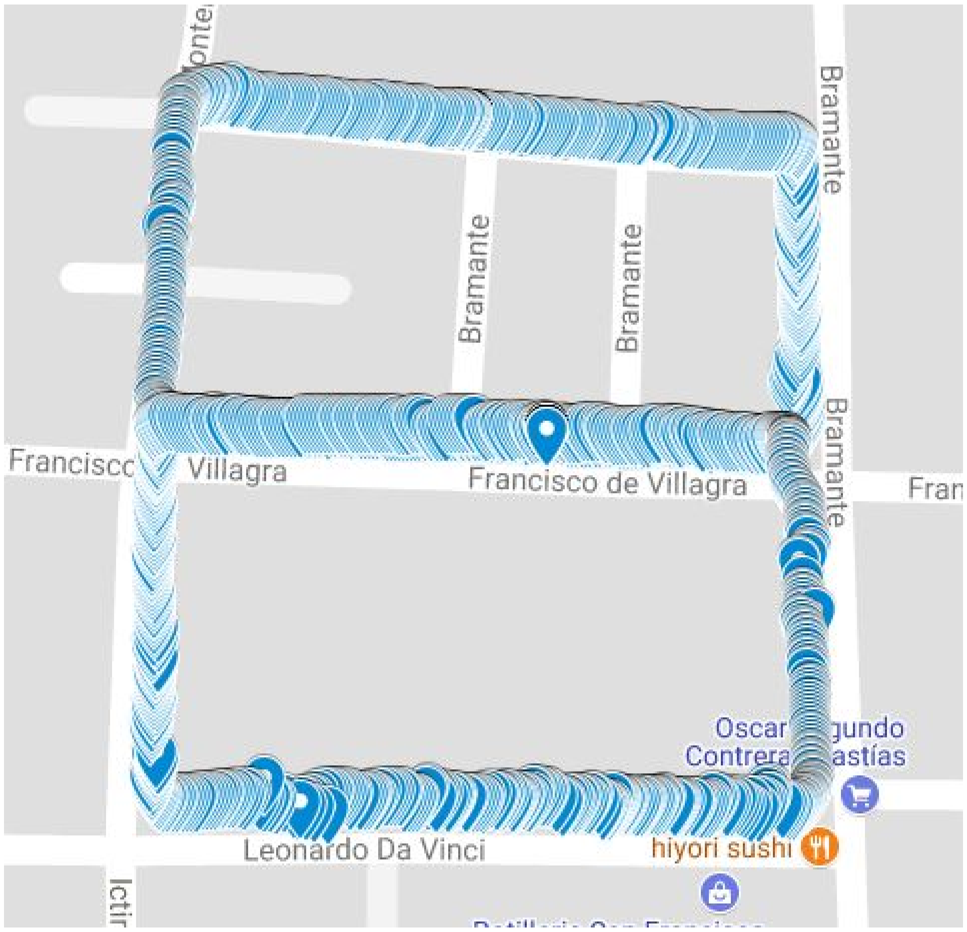


Figure 4.13: Path generated by GPS (The ground truth).

The ROS topic that contains the GPS data was recorded incorrectly, with errors in the message, some messages where empty messages, some messages were simply not recorded. These errors could not be fixed in the implementation, but a different and better way to record the data was used simultaneously. The data was recorded directly in the GPS memory and it is going to be taken and added to the bag file, modifying the topic recorded. With this modification, the topic generated will be filled with the correct data.

To add the data to the bag file a python script was created to generate an excel file (.xls) in which the data is corrected manually by adding the data from the SD card and filling the empty messages. When the excel file is completely corrected, another python script is used to copy the previous bag file while adding the data from the excel file. The added messages are correctly labeled to distinguish them from the messages obtained directly from the GPS (Table 4.1).

Chapter 5

Data Processing

Given the data-set recorded, and to use the ground truth as a way to validate it, two SLAM algorithms are going to be computed. These algorithms will use the data-set generated as an input and will give a map and a trajectory as an output. The trajectory will be compared to the ground truth as a way to compare the two algorithms used.

5.1 ORB-SLAM

The ORB-SLAM[14] algorithm uses the stereo camera information to compute the odometry of the camera, therefore the odometry of the robot. This algorithm uses the ORB detection algorithm to generate landmarks and characterizes each landmark with a BRIEF descriptor. In this way, it is possible to identify a landmark every time it is seen again (while the perspective of the landmark corresponds). Thanks to this it is possible to relocate the robot in the given trajectory and map if the robot gets lost in the tracking. This algorithm also counts with a loop detection phase and a correction phase when the loop is detected. In this way when the robot detects the same points and detects a loop, the accumulated error is corrected.

In general, the use of this algorithm does not require too many computational resources, but it can lose track very easily in certain conditions. As the algorithm creates the map, it tends to add more and more points, which implies the use of more RAM, therefore in a large map, a large amount of RAM is recommended. The lack of RAM can lead to a change in the sampling frequency, which can lead to a lost track. The loop closing part of the algorithm is the phase that uses the most computational resources, since it requires to re-compute all the points and the keyframes in the map. When a loop is detected it is recommended to reduce the computational load or stop the bag while the correction is calculated. The result of the ORB-SLAM algorithm can be seen in the Figure 5.1.

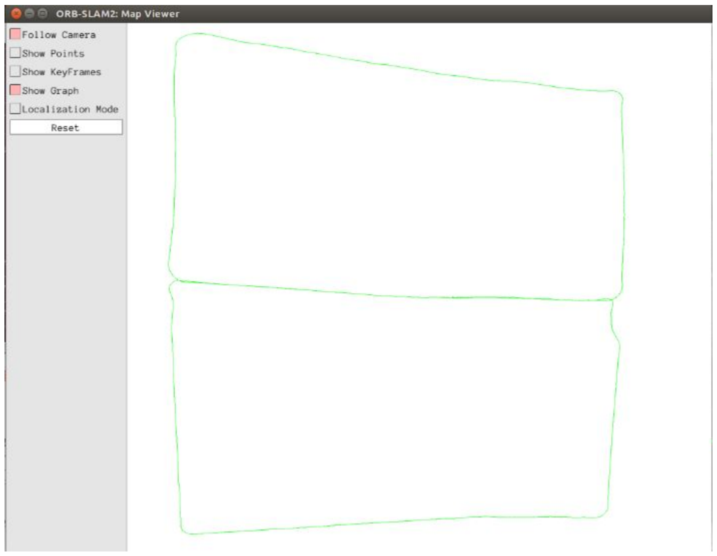


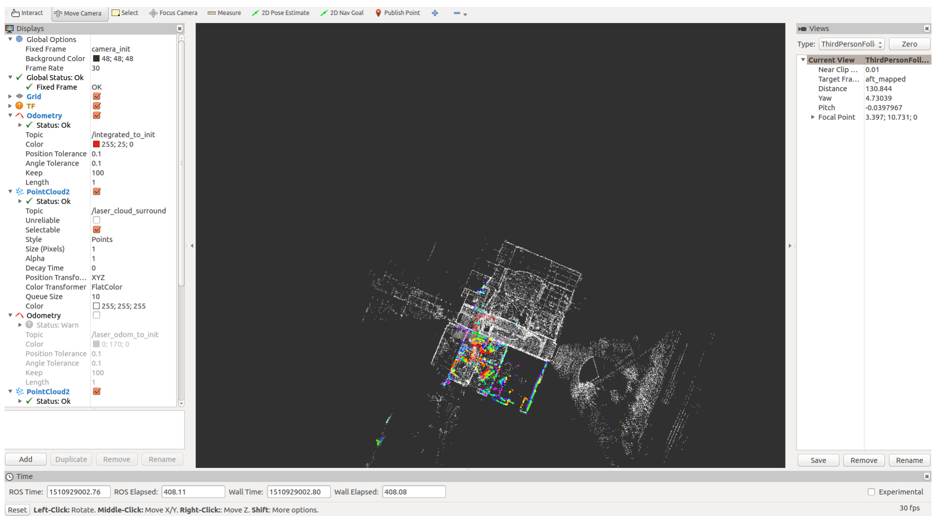
Figure 5.1: Output of the ORB-SLAM algorithm.

5.2 LOAM

The LOAM[15] algorithm uses LIDAR data to determine the odometry of the LIDAR, therefore the odometry of the robot. This algorithm, unlike the ORB-SLAM, lacks on a descriptor for the points used, therefore, this algorithm does not use landmarks that can be recognized by the algorithm. This algorithm is based on the data correction of the sensor, to better estimate the trajectory and the map in which the robot moves. Since the points of the map are not identified by a tag or a characteristic, if the track is lost there is no way to relocate the robot. Thanks to this, LOAM lacks a loop closing process. In Figure 5.2 the output of the algorithm is shown, and in Figure 5.3 it is possible to notice that rerunning the bag, generates that the points superpose generating the same map rotated. This is given since the algorithm does not detect that the new points correspond to previously stored points.

The LOAM algorithm uses three maps, a current map, which is the current measure of the LIDAR, a local map, which is a small part of the map that contains various measures, but not all of them, and a global map, which is the map that contains all the measurements. The local map is used to compute the likelihood of the actual data to add this new information in the global map and compute the odometry. Given these characteristics this algorithm is easier to compute, but, similar to the ORB-SLAM, while the data-set grows, the map will contain more points, therefore more RAM is needed.

Figure 5.2 and Figure 5.3 correspond to the results of a test data to verify that the path is estimated correctly and that a map it generates. Finally, the LIDAR is used in the same path that the ORB-SLAM was computed obtaining the results observed in the Figure 5.4



Global Options	
Fixed Frame	camera_init
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
TF	✓
Odometry	✓
Status: Ok	
Topic	/integrated_to_init
Color	🟥 255; 25; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1
PointCloud2	✓
Status: Ok	
Topic	/laser_cloud_surround
Unreliable	☐
Selectable	✓
Style	Points
Size (Pixels)	1
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	FlatColor
Queue Size	10
Color	☐ 255; 255; 255
Odometry	☐
Status: Warn	
Topic	/laser_odom_to_init
Color	0; 170; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1
PointCloud2	✓
Status: Ok	

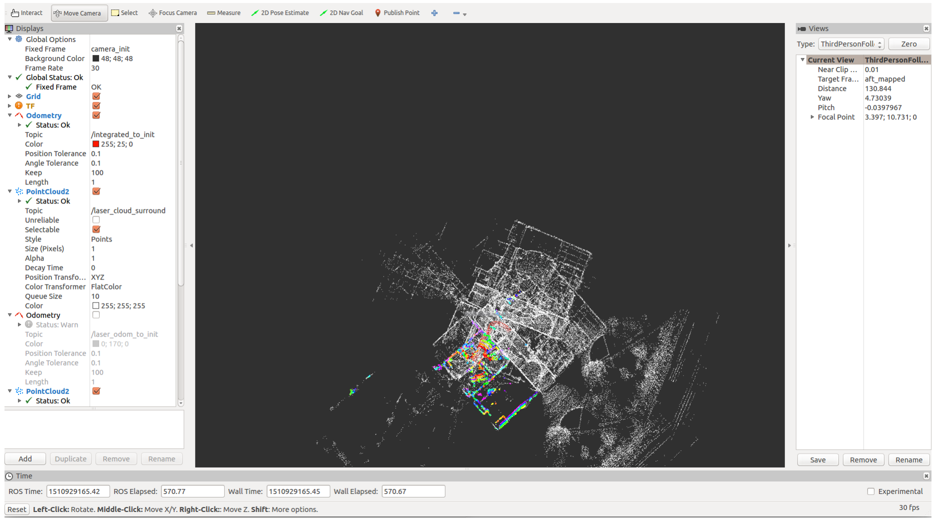
Type: ThirdPersonFoll

Current View	ThirdPersonFoll...
Near Clip ...	0.01
Target Fra...	aft_mapped
Distance	130.844
Yaw	4.73039
Pitch	-0.0397967
Focal Point	3.397; 10.731; 0

ROS Time: 1510929002.76 ROS Elapsed: 408.11 Wall Time: 1510929002.80 Wall Elapsed: 408.08 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options. 30 fps

Figure 5.2: Output of the LOAM algorithm.



Global Options	
Fixed Frame	camera_init
Background Color	48; 48; 48
Frame Rate	30
Global Status: Ok	
Fixed Frame	OK
Grid	✓
TF	✓
Odometry	✓
Status: Ok	
Topic	/integrated_to_init
Color	🟥 255; 25; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1
PointCloud2	✓
Status: Ok	
Topic	/laser_cloud_surround
Unreliable	☐
Selectable	✓
Style	Points
Size (Pixels)	1
Alpha	1
Decay Time	0
Position Transfo...	XYZ
Color Transformer	FlatColor
Queue Size	10
Color	☐ 255; 255; 255
Odometry	☐
Status: Warn	
Topic	/laser_odom_to_init
Color	0; 170; 0
Position Tolerance	0.1
Angle Tolerance	0.1
Keep	100
Length	1
PointCloud2	✓
Status: Ok	

Type: ThirdPersonFoll

Current View	ThirdPersonFoll...
Near Clip ...	0.01
Target Fra...	aft_mapped
Distance	130.844
Yaw	4.73039
Pitch	-0.0397967
Focal Point	3.397; 10.731; 0

ROS Time: 1510929165.42 ROS Elapsed: 570.77 Wall Time: 1510929165.45 Wall Elapsed: 570.67 ☐ Experimental

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Move Z. **Shift**: More options. 30 fps

Figure 5.3: Output of the LOAM algorithm.

5.3 Comparison

In Table 5.1 the differences and similarities of the algorithms are studied.

It was intended to compare the final data-set trajectories in the first place, but the code that extracts the points of the ORB-SLAM algorithm could not end successfully in that iteration (presumably for a deadlock), for that reason the test data is used instead to compute the comparison. Also the data-set generated with the LIDAR is the same generated in the test data.

To compare the results of the algorithm the path given as an output are going to be compared with the ground truth with a certain metric. The most basic metric in these cases should be to use Euclidean distance, but since the quantities of points in each case vary, it is hard to correlate the points to compute this type of error. To compare the trajectory

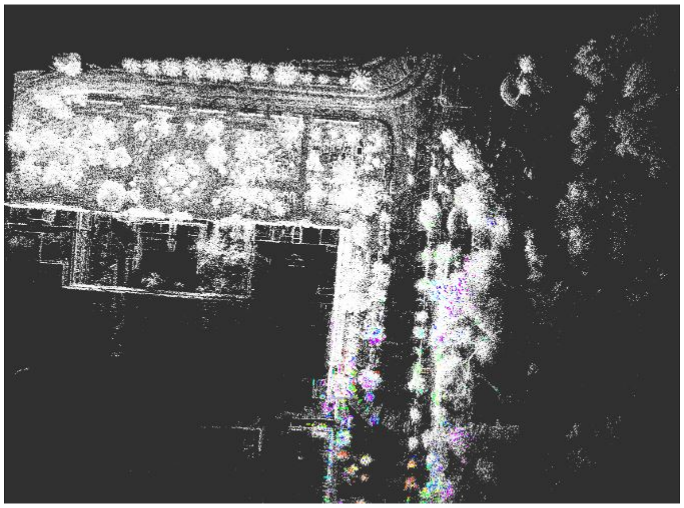


Figure 5.4: Output map of the LOAM algorithm when running same bag file twice.

Description	ORB-SLAM	LOAM
landmarks identification and recognition	Yes	No
Computation of loop closing	Yes	No
Optimization of the code using threads	Yes	Yes
Real time execution algorithm	Yes, but relying on computer specifications	Yes, but relying on RAM based on the quantity of points needed
3D-map	quantity of points low to being able of distinguish the surrounding	quantity of points adequate to identify the surrounding

Table 5.1: Comparison between algorithms.

the metric used corresponds to the energy required to transform the estimated path into the ground truth.

To obtain the result, the Absolute Trajectory Error (ATE) [16] is computed. This error is the energy that takes to transform the estimated data to the ground truth. To compute the ATE is necessary for the first place to obtain the difference between the estimated trajectory and the same trajectory, rotated, translated and scaled, that better fits the ground truth. This means that it might not be possible to transform the estimated data in the ground truth, but there is still a transformation that makes them the best match possible.

To obtain the transformed data the Horn's Method [17] is used. This algorithm receives the two trajectories, in a 2D or 3D path and gives as an output the transformation, rotation, and scale necessary to fit the two trajectories in the best possible way. As an input the points given can be seen in the Figure 5.5 and 5.6 for the ORB-SLAM and LOAM respectively. The ground truth corresponds to the red trajectory, while the estimated one corresponds to the blue curve. The estimated trajectory was scaled to better fit the ground truth.

Finally, with the two trajectories at hand, it is possible to calculate the ATE which

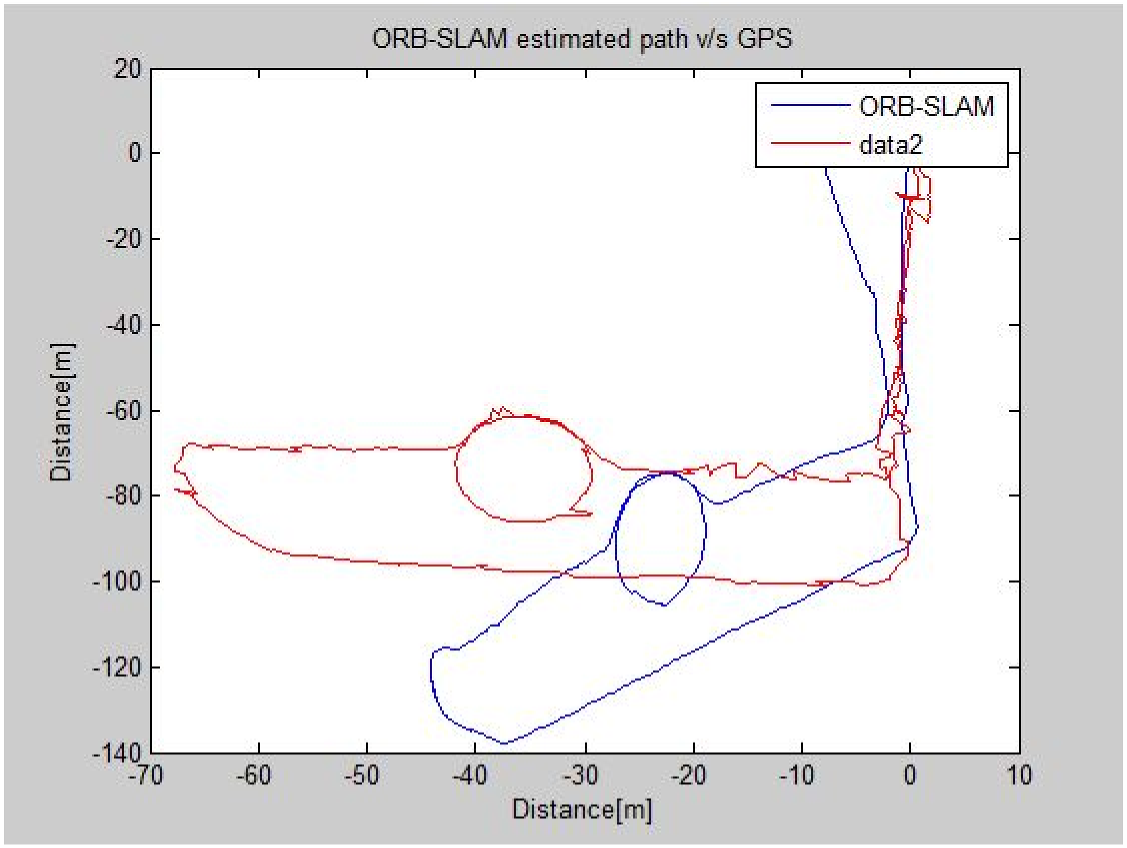


Figure 5.5: Initial ORB-SLAM posses.

correspond to $\sqrt[2]{\sum error * error'}$ which in the case of the ORB-SLAM estimated path gives a result of 605.08 and in the case of the LOAM it gives: 524.78 . Which indicates that in the path given, the LOAM estimated was better.

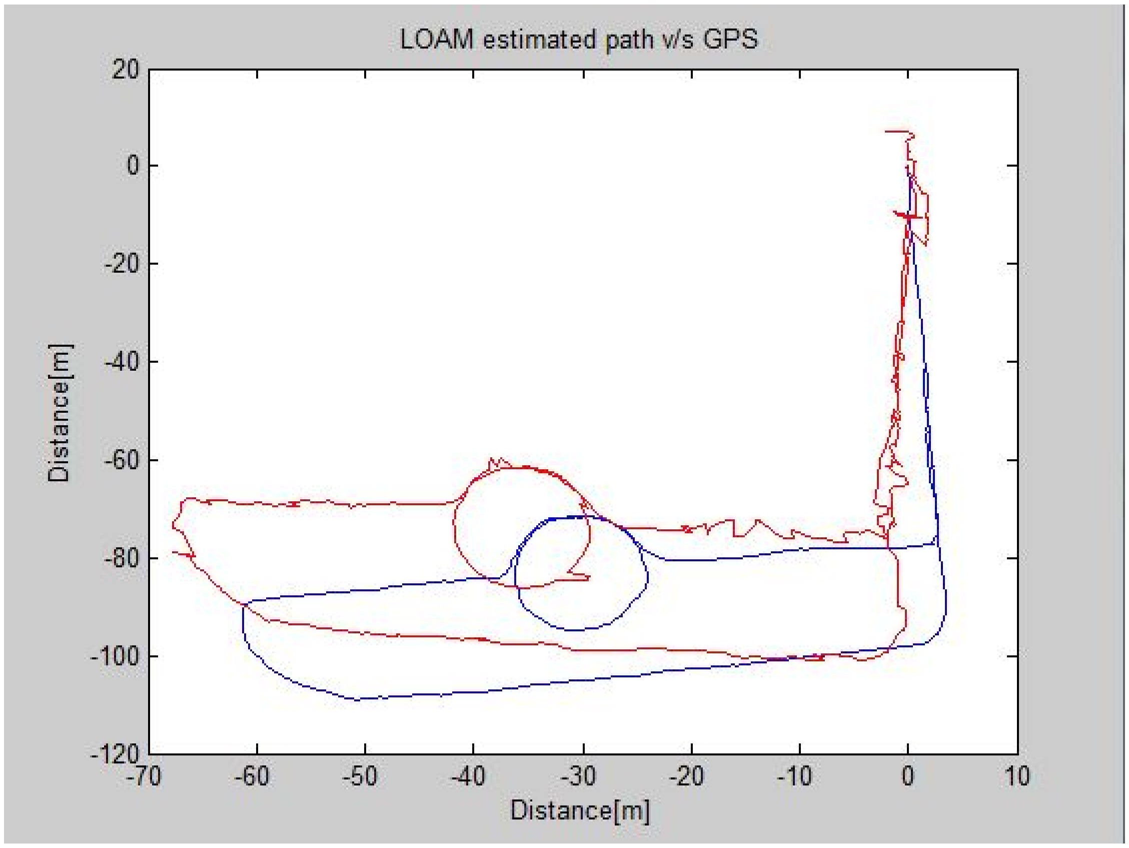


Figure 5.6: Initial LOAM posses.

Chapter 6

Conclusion

To use the GPS as ground truth it was necessary to test the data obtained from it. The rtk GPS is a good ground truth if the data is post-processed or if the fix mode is active. To rely on the fix mode it is necessary that the base GPS and the rover GPS are in the line of sight of enough satellites, therefore it is required a clear vision of the sky. It is also necessary to communicate the base and the rover all the time if it is possible to guarantee the fix data. the communication is given by radio signal, which allows a distance of 0.5-1 Km if there is a low quantity of objects that can interfere the signal. The most reliable way to store the data is to store it in the micro SD memory card in the GPS.

The ORB-SLAM landmarks are well identified and characterized so it is easy to re-track the path if the movement is lost by simply re-running the bag file in the time-stamp close to the lost track. Sometimes could be necessary to lower the frequency of the bag file, since the change between frames is too fast, generating the loss of the track, and therefore it is possible to fix the track if the frequency is lowered.It is possible to correct errors when a loop is detected, which is useful since the path without loop closure can vary enormously in relation to the loop closed one. The loop closure process is the part of the algorithm that require the most processing resources.

The LOAM algorithm does not characterize the points used. The LOAM algorithm consist of pre-processing the points of the LIDAR to estimate in a better way a trajectory. It is not computationally heavy, but it would need a large amount of RAM if the data-set is big. Since there is no characterization of the points it is not possible to close loops or to relocate the robot if the bag is replayed.

The performance of one algorithm in relation to the other can be described in relation to existent loops and the size of the data-set. In a small data-set without loop closing paths the LOAM algorithm could obtain better results, while the ORB-SLAM should be most useful in a big data-set with a lot of loops.

Bibliography

[1] *Operation manual of HiPer® V.* Topcon positioning systems, 2012.

[2] *Getting Started Bumblebee®XB3 IEEE-1394b Stereo Vision Digital Camera System.* Point Grey, 2010.

[3] *Users Manual And Programing Guide VLP-16.* Velodyne, 2010.

[4] C. G. Ignacio, *Implementaciòn De Una Librería De Control De Un Radar Con Interfaz A La Plataforma ROS.* Memoria, U. de Chile, 2014.

[5] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, *A Tutorial on Graph-Based SLAM.* Browse Journals & Magazines, IEEE Intelligent Transportation system magazine, Volume: 2 Issue: 4, 2011.

[6] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, *ORB-SLAM: A Versatile and Accurate Monocular SLAM System.* IEEE Transactions on Robotics, Volume: 31, Issue: 5, Oct. 2015, 2011.

[7] J. Zhang and S. Singh, *LOAM: Lidar Odometry and Mapping in Real-time.* Robotics: Science and Systems, 2014.

[8] M. Adams, *Apunte de curso: Robotics, Sensing and Autononomous Systems.* U. de Chile, 2016.

[9] E. D. Kaplan and C. J. Hegarty, *Understanding GPS: Principles and Applications.* Artech House, INC., 2006.

[10] J. Mullane and M. Adams, *Random Finite Sets for Robot Mapping & SLAM.* Springer Tracts in Advanced Robotics 72, Springer-Verlag Berlin Heidelberg, 2011.

[11] O. Library, "BRIEF." `https://docs.opencv.org/trunk/dc/d7d/tutorial_py_brief.html`, 2016. [Online; accessed 17-November-2017].

[12] O. S. R. Foundation, "ROS software." `http://www.ros.org/`, 2008. [Online; accessed 01-August-2017].

[13] O. S. R. Foundation, "How to install a package in ROS." `http://wiki.ros.org/ROS/Tutorials/BuildingPackages`, 2008. [Online; accessed 01-August-2017].

[14] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras," *arXiv preprint arXiv:1610.06475*, 2016.

[15] J. Zhang and S. Singh., "LOAM: Lidar odometry and mapping in real-time.," *Robotics: Science and Systems Conference (RSS)*, 2014.

[16] raulmur, "evaluate ate scale." `https://github.com/raulmur/evaluate_ate_scale`, 2016. [Online; accessed 17-November-2017].

[17] M. J, "Absolute Orientation - Horn's method." `https://www.mathworks.com/matlabcentral/fileexchange/26186-absolute-orientation-horn-s-method`, 2015. [Online; accessed 17-November-2017].