MASTERS THESIS

SLAM algorithms applied to collaborative robotic systems

Localization and Mapping into a lunar analogue and unknown environment

Author:
Maximilien DREIER

Supervisors:
Jasmine RIMANI
Stephanie LIZY-DESTREZ
Damien VIVET

Department of Mathematics
SORBONNE UNIVERSITES (CAMPUS PIERRE ET MARIE CURIE)

NOVEMBER 1^{st} 2020 - APRIL 30^{th} 2021

Jasmine RIMANI
IGLUNA 2021 CORODRO Leader
Visiting PHD Student at
ISAE-SUPAERO

Stephanie LIZY-DESTREZ
Professor Associate
Head of SACLAB
DCAS department at
ISAE-SUPAERO

Abstract

The CORODRO (COllaborative ROver and DROne) team from ISAE-SUPAERO is a student team dedicated to the study of autonomous navigation and operations for space robotics systems. It participates in the Igluna 2021 event which consists of university students applying their knowledge to solve a technical challenge, to sustain life in an extreme environment, increasing in parallel the maturity of technologies relevant to the space domain. As space missions turn bolder and push toward deep space, the concept of autonomy becomes an essential asset to space exploration. The study of my team aims to mature the algorithms and the software for autonomous navigation and operations of robotics systems. More in detail, we want to study and advance technologies linked to the exploration of the lunar lava tubes. Lava tubes are volcanic architectures that unravel under the Moon surface. They may become a radiation and micro-meteoroids shielded human outpost. To assess the safety of these caves, a series of robotic missions are envisioned. They should autonomously map, navigate and operate inside the tubes as well as relay sensitive data toward Earth. That is why our team is divided into 3 main development sections: Task-Planning, Path-Planning, and Simultaneous Localization And Mapping (SLAM). My principal part inside of the project was to be in charge of the SLAM section, which consisted of developing the algorithms that would enable our robotic systems to map and localize in an unknown environment, as well as relocalize themselves into the mapped environment. This paper presents worked perform by the SLAM section, the environment of work needed for the testing of the developed algorithms as well as the mathematical theories lying behind them. Results of used algorithms are showed and discussed relatively to the project requirements. At the end, a discussion session occurs about different aspects of the project on which I have worked on and which step out of the SLAM framework.

Goal of the Project

During the lava tube exploration, the rover will deliver a thruster-propelled bot toward the lava tube skylight, Annex subsec: Lunar Mission Analysis. The thruster-propelled bot, embodying a high mobility platform, will map the environment and detect several points of interest for the rover to analyze. The rover will then receive the map from the bot and use it to navigate between the obstacles toward the points of interest. The lunar lava-tube environment imposes several severe constraints. Firstly, any technological achievements deployed on Earth such as the GPS is non-existent on the Moon. Secondly, all current and past space missions are constantly monitored from Earth. Navigation relies on human decisions and observations of the surrounding environment of the robots from Earth telescopes. However, in a cavernous environment, there is no way for human based on Earth to track, monitor, and analyze the environment of the mission target. That is why our robotic systems shall be fully autonomous. Then, the limitation of payload implies a limited amount of computing power embedded in the robotic systems sent to space.

Therefore, softwares must be very optimized to allow our robotic systems a real-time navigation system, which consist of processing data faster than the rate at which it is generated. Finally, the low communication data flow rate between Earth and Moon imposes to optimize and condense the mission results for them to be shareable with Earth operators. To simulate a similar set-up and effectively study the operational layer of our collaborative systems, a test bed with a rover and a drone will occur in July 2021 on top of Mount Pilatus in Switzerland (jeopardized by the Covid). The test area will be at an altitude of 1820m, and the control centre will be in Lucern. A communication network between the two areas will be artificially set to Moon-Earth analogue conditions, with a flow rate of 0.45Mb/s and a delay of 2,56s.

Acknowledgments

I would like to thank my remarkable tutor Jasmine RIMANI and supervisors Stéphanie LIZY-DESTREZ & Damien VIVET for their mentorship and for creating a congenial working environment during the 6 months of this internship.
Special thanks also goes to engineers of the DISC Corentin CHAUFFAUT and Louis TRETON, as well as Pascal CHAUVIN for their precious help and participation in our student project. Sharing their expertise with us was essential for the project success.
I am thanking also the SacLab team composed of PhD students, postdocs, and of the professor Stéphanie Lizy-Destrez for their warm welcome in the department and for the very interesting weekly meeting where I learned a lot about diverse topics related to the Space field.

Dedication

I would never be able to thank enough my classmate Hamed JOORATI for not gaving up on me and for supporting me through the years of this master. Without him, I genuinely would not have hold this master until its completion. I am dedicating this master to you Hamed, thank you so much.

Table of Contents

List of Figures

CHAPTER 1

INTRODUCTION

The SLAM Team

The SLAM section was composed of me and Maanasa SACHIDANAND, a Master student in Aerospace engineering who joined me later in February. I worked in the Département Conception et Conduite des Véhicules Aéronatuiques et Spatiaux (DCAS) of ISAE-Supaero, and more particularly as an intern of the Space Advanced Concepts Laboratory (SacLab) section of that Department. My work was supervised by Stéphanie LIZY-DESTREZ, the head of the

Figure 1.1: SLAM team : from left to right Maximilien Dreier (me), Maanasa Sachidanand

SacLab, and Jasmine RIMANI, the project Leader of the CoRoDro project and also in charge of the Task-Planning section in which she directly applies her work of her ongoing PhD. I was however enjoying a real Independence and autonomy of work as I was in charge of choosing and tailoring the algorithms to the wanted objectives. As the field of robotics and space were completely new to me, I was lucky to benefit from the experience and the knowledge of Damien VIVET in autonomous mapping and localization, of Corentin CHAUFFAUT and Louis TRETON in embedded systems. Initially, during the first weeks of the project (November), two others persons were in the SLAM section. They contributed to work on the 1. Maanasa worked on the detection of AR-tags that would simulate the points of interest we aim to detect and visit. Those detected points in the environment would also be used as reference points for a robotic system to relocalize itself in a given map. Her work and our collaboration is discussed in section 4.1.1.

Project Relevance

Many space agencies, including NASA and ESA, foresee that mankind will be back on the Moon in the 2020s [16]. However, strong engineering actions are required to sustain the realization of a permanent human outpost. In particular, one of the most critical aspects of both robotic and human space exploration is related to operations. As space exploration missions grow in complexity, there is a need to balance the mission return, the autonomy level and the workload of the control center operators. That aim can be reached thanks to the technological advancement of the last decade related to artificial intelligence [13]. In this context, space agencies, companies, and universities are engaged in the definition of a broad spectrum of technological maturation studies toward autonomous operations and navigation [14] [13] [12] [10]. For any innovation of the space sector, there is a need to engage in extensive testing to prove the suitability of the algorithm for collaborative robotics, autonomous navigation and autonomous operations. The project starts in this context of the technological maturation of autonomous navigation and operations for robotics systems. The topic of interest can easily double in successful terrestrial applications. It is easier and safer for robotics systems, such as rovers and drones to access the heart of a volcano or a radioactive area than for a human [22] [19]. Collaboration algorithms can be deployed in the agricultural sectors [23] [28]. Multiple collaborative and autonomous drones can be used for delivery purposes, especially in the current situation where human contacts shall be minimized. [7]. Our tested algorithms of autonomous navigation and operation will permit our systems to explore challenging targets on the Moon, like the lava tubes. Lava tubes are volcanic structures that unravel under the lunar surface [17]. During their time inside the tunnels, the robotic exploration systems, may they be rovers or hopping bots, should rely on themselves both to navigate in the unknown environment and to operate. During the field campaign, we will test a rover collaborating with an "Earth" drone. The first objective of this collaborative drone is to simulate the operations of the lunar bot. During the lunar mission, the small propelled bot will fly to map and explore the environment before the rover. The same bot will use propelled flight to

slow down its fall in the lava tubes while mapping the skylights. In the lava tube, the bot may decide to perform short flights to better map the environment. The second focus of the test at the field campaign is to test the collaboration between diverse robotic platforms and how this difference can help to shape the success of an exploration mission.

Project Design

Concept of Operations

On a given terrain, the aim of the drone is to autonomously generate a map of the environment. The drone must detect what could be potential points of interest for the rover to visit, as well as the elements that constitute an obstacle for the rover. Once the map is generated, it is sent to the rover via a local network. The rover must be able to localize itself in the map generated by the drone and use it to plan the optimal path given the available information. In the case of contingency where the drone's map is incomplete or non-existent, and to be more robust, the rover is also equipped with an autonomous navigation system and map generating software similar to the drone's one. As the rover progresses toward its objectives, it scans its environment to detect previously unseen obstacles by the drone, and if some are detected, the rover updates the map it has with the new information and adjusts locally its path to avoid the said obstacles.

SLAM Introduction

The Simultaneous Localization and Mapping algorithms answer a basic need in robotics. Indeed, before engaging in any task that requires a movement, a robot must be aware of both itself and its environment. As we design robots to perform tasks for us the humans, it is not a surprise that localization and mapping is also a basic need for humans. When we think about it, mapping is the first process by which mankind brings civilization to a place. A navigation system includes a topographic scan of the ground coupled with a positioning system, resulting in a well-structured map. Intelligible for both humans and its machines, that map represents sophisticated information that enables the capability for an entity to simultaneously localize itself and be precisely aware of its environment. Therefore, the area is no longer raw and unknown, the map has been integrated into it. In other terms, the foundations of civilization have been settled on it. SLAM algorithms have emerged in the 80', and among the most popular ones that have been developed ever since, a general structure of the SLAM algorithm can be extracted:

- Step 1: Estimation and scan:

  1.a) The robot estimates its position thanks to an odometry sensor (e.g. IMU or VIO).

  1.b) In the meantime, it scans its environment with a depth sensor (e.g. Lidar or Depth Camera).

- Step 2: Optimization: To avoid heavy computational software routines, it is essential to filter the depth scan provided by the sensor.

- Step 3: Landmark extraction: As humans use unique elements that can be easily remembered in order to locate themselves (e.g. the monuments of Paris as intersection points of straight Haussmannian boulevards), our system emulates a similar behaviour. Landmarks won't be monuments, but something that still can be catchy for the eyes of the robot. That could mean points with a high-gradient variation relative to a physical quantity (e.g. corner points). Those landmarks should also be frequently observable, as there is no utility in a landmark that will be seen only once by the robot.

- Step 4: Association and Localization: The algorithm tries to recognize current landmarks among the past stored landmarks. Thus, given the position of the recognized landmark, and its current distance and angle between the robot and the landmark, the algorithm is able to retrieve the position of the robot. Knowing its position and its distance with all the new landmarks, it can calculate their positions and store them in the landmarks database. Various methods are used. Among them, one can mention Extended Kalman Filter methods, particle filters.

- Step 5: DEM map generation: Position of the robot and depth data are merged to form a local DEM map. That local DEM map is added to the global DEM map containing the previous local DEM map generated. The robot returns to step 1 until the global DEM map is complete. Every cell of the environment is identified by its position and contains depth information if the robot has already scanned this area, or a default value if the robot has not come yet. The map uses the Cartesian (x,y) system, where (0,0) is the initial position of the robot, and the y axis the straight forward line when the robot is initializing.

The Step 5 mentions that the generated map is a 2 dimensional map. It could be perceived as strange as the drone evolves in a 3D environment. Actually, we supposed that the drone is operating in an area without obstacles for it. Depending on the chosen sensors for the drone (see 1), we computed a flight altitude of 3m. The obstacles on our test field do not exceed 1,5m. Moreover, the map is aimed to be used by the rover which evolves in 2D dimension. We suppose there is not a bridge configuration that would allow the rover to be at two different positions with the same (x, y) but a different z. The final map that is generated is tailored to the rover's characteristics, especially its capability to cross obstacles. Empirical tests showed that the rover cannot cross anything that is higher than 6cm with a slope of 90° or something that is higher than 15cm with a slope of 60°. That information is used to autonomously analyze the 3D map generated and compute the areas reachable or unreachable by the rover. The result is then projected onto a 2D grid map on which a cell is white if the rover can theoretically be, black if not, and grey if data was lacking.

Mechanical Design

The rover is composed of a lower-part commercial platform [1] and an upper-part designed and assembled by the DISC. The drone has been fully designed and assembled by the DISC of ISAE-Supaero as well. The choice of the sensors equipped on the drone is explained in Section 1. The sensors for the rover were already mounted on the platform to perform experiments similar to ours, so we kept the same winning set-up.

Drone components	Subsystem	Reference	Specific characteristics
Cameras	Lidar	HOKUYO UST-20LX	Scan angle: 270¬∞
	Depth Camera	Intel D435i	Maximum frame rate: 90 fps
	Tracking Camera	Intel T265	Maximum frame rate: 120¬∞
Communication	Radio Receptor	Futaba R6208 SB	Frequency: 2.4 GHz
	WiFi System	Odroid WiFi Module	Frequency: 2.4 GHz
Computer	Computer	ODROID XU4	
Electronic Avionics	Flight Controller	3DR Pixhawk 3	
	Electronic Speed Controllers	BL-C trl V 3.0	
Motors	4 Motors		
Power System	Battery	Lipo 4s	Capacity: 3700 mAh

Table 1.1: List of components: Drone

Rover components	Subsystem	Reference	Specific characteristics
Cameras	Leo Rover Camera	HOKUYO UST-20LX	Lens: 170 deg Resolution: 5 Mpx
	Depth Sense Camera Lidar	Intel D435i HOKUYO UTM-30 LX-EW	
Communication	WiFi Internal PPI Antenna Leo Rover		Frequency: 2.4 Ghz
	WiFi Access Point Antenna Leo Rover		Frequency: 2.4 GHz
	WiFi System Upper Part Rover		
-2*Computer	Computer Leo Rover	Raspberry Pi 3B+	Frequency: 2.4 GHz
	Computer Upper Part Rover	ODROID XU4	
Controller	Controller	Husarion CORE2-ROS	
Mobility System	4 Wheels 4 Tires 4 Wheel Actuator 4 Wheel Encoder	Buehler DC Motors Pololu Romi 12 CPR Magnetic Encoders	Diameter: 130 mm Material: rubber with foam insert (non pneumatic)
Inertial Measurement Unit		SBG IG 500A-G5A2P1-P	
-2*Power System	Battery Leo Rover	Battery Li-Ion with internal PCM	Capacity: 5000 mAh
	Battery Upper Part Rover	KUNZER multi pocket booster MPB 150	Capacity: 15000 mAh

Table 1.2: List of components: Rover

Trade-Off

The first work to be done was to determine which depth sensor(s) to use on the drone to perform the SLAM task. After the first selection of sensors, we were mitigated between 3 final sensors.

The first trade-off aims to decide which camera to use. From Table 1.3, we can see that the D435i scores better on our benchmarks. However, we have decided to mount both cameras on the drone. Indeed, the scores are pretty close, and since there is the possibility to put both cameras on the drone, we preferred this solution. This would provide a native stereo algorithm and localization algorithm at the same time, and help the software development.

The second trade-off aims to decide whether to use a LiDAR instead of the D435i as a depth sensor. We can see on Table 1.4 that the camera scores much better than the LiDAR, according to our benchmarks and weights. Also, we believe that embedding the LiDAR on the drone may generate many problems related to power consumption and overall drone autonomy. However, even in this case, we do not exclude the use of the LiDAR. In a cavernous environment, LiDAR is not disturbed by the natural IR wavelength light from the sun, which is the wavelength at which the LiDAR operates. Moreover, it is easier to scan a relatively flat ground with a LiDAR than

with a depth camera. That is why we decided to mount both the LiDAR and the D435i (as well as the T265). Our strategy was to implement and develop two configurations, one which works with the T265 + D435i, and one with the T265 + LiDAR. That way, we would regularly compare the two configurations and we would keep at the end the best one.

Features	Weight	Intel RealSense T265	Intel RealSense D435i
Range	1	2	2
Accuracy	2	0	1
Field of View	1	2	2
Weight	2	2	2
Integrated Stereo	3	1	2
Integrated V$_SLAM$	2	2	1
Power Consumption	1	2	2
Weighted Average		2.43	2.86

Table 1.3: Trade-off to decide which camera was the most suitable payload to enable localization with the drone

Features	Weight	RPLidar A1	Intel RealSense D435i
Range	1	2	2
Accuracy	2	2	1
Field of View	1	2	2
Weight	2	0	2
Integrated Stereo	3	0	2
Integrated SLAM	2	0	1
Power Consumption	1	2	2
Weighted Average		1.43	2.86

Table 1.4: Trade-off to decide whether to use a LiDAR instead of the D435i

CHAPTER 2

SLAM ALGORITHMS

2.1 Extended Kalman Filter

A robotic system presents two types of sources of errors. The first is related to the inner system; given an input order, the system output never matches perfectly the desired input. The second is related to the sensors' measures; if we want to observe the state of the robotic system through a sensor, there will always be an uncertainty on the retrieved value as the sensor is not perfect. That is why when a robotic system uses a deterministic model with its own sensors' observations as input to execute the next order, an accumulation of errors coming from the two previously said sources will quickly occur, and a drift between the final desired output and the final exact state of the robotic system will be non-negligible. The Extended Kalman filter [20] comes to remedy that problem. It is an optimal recursive data processing algorithm. It can integrate any information we have on the uncertainties of the sensors and the errors of the dynamics of the system. As a result of a stochastic control algorithm, it gives the optimal estimation of the state of the robotic system. Let's dive into detail.

Applied to our objective, we want to retrieve at each time k the discrete position process $X_{rob}(k)$ of our system. The first assumption of the Kalman Filter is that the measurement uncertainties are modelled by a Gaussian noise V_{sys}. In practice, the covariance uncertainty of our IMU and Cameras can be retrieved directly on the constructor documentation, or with a calibration tool. As a result of the integration of the uncertainties as a Gaussian distribution, the position $X_{rob}(k)$ will also be a probabilistic process.

$$X_{rob}(k+1) = M_{rob}(k)X_{rob}(k) + U(k+1) + V_{sys}(k+1) \tag{2.1}$$

where M_{rob} is the state transition matrix, $U(k+1)$ the control wanted between time k and $k+1$, and V_{sys} a time independant Gaussian noise with mean 0 and covariance Q_{sys}. The second as-

sumption of the Extended Kalman Filter is that X_{rob} is a Markov process. Indeed, it is legitimate to consider that the control $U(k+1)$ moving the system from the point $X_{rob}(k)$ to the point B is a function of $X_{rob}(k)$ only.

Let $Z(k)$ be the observation of the landmarks. As said in the SLAM introduction, landmarks are points extracted from the point cloud obtained by the depth sensor. Various elements can be extracted among them we can cite the corner points (Harris points[18]) which is a commonly used point detection method. The observation $Z_i(k)$ of the i_{th} landmark is described by the following equation :

$$Z_i(k) = HL_i - H_{rob}X_{rob}(k) + W_i(k) \tag{2.2}$$

where $L := (L_1, ..., L_N)$ is the matrix of positions of all landmarks (containing 0 for any unobserved landmark). L is commonly taken as the first observed values of the said landmark:

$$L_i = H^{-1}(H_{rob}X_{rob}(0) + Z_i(0)) = H^{-1}Z_i(0) \tag{2.3}$$

because $X_{rob}(0) = 0$. H_{rob} and H are the observation matrixes that contain the functions to compute the position of a landmark given the position of the robot and the distance and bearing between the robot and the landmark (Distance and bearing are indeed the only information we have on a landmark thanks to a depth sensor).$M_i(k)$ is the measurement related to the i_{th} landmark at time k (distance and bearing from the robot). $W_i(k)$ is again a time-independent Gaussian noise relative to the depth sensor uncertainties with mean 0 and covariance $R_i(k)$.

The whole purpose of the Extended Kalman Filter is to produce an estimate [11] [26] of the robotic state thanks to the density probability of $X_{rob}(k|Z(0), Z(1), .., Z(k-1))$. One of the main advantage of the Extended Kalman Filter is that it also produces estimations of the landmarks' states thanks to the density probability of $Z(k|Z(k-1), ..., Z(0))$.

2.1.1 Propagation of Conditioned probability

Let's consider one estimate Z_1 and one observation Z_2 of a process $x(k)$. Z_1 and Z_2 are considered Gaussian with respective mean $\hat{x_1}(k)$ (the estimate) and x_2 (the measured value), and their respective uncertainties covariance Q_1 Q_2. The conditional density of the process $x(k)$ given Z_1 and Z_2 is still a Gaussian with mean m and covariance $Q(k)$ as follows :

$$1/Q(k)^2 = (1/Q_1^2) + (1/Q_2^2) \tag{2.4}$$

and mean :

$$m = [Q_2^2/(Q_2^2 + Q_1^2)]m_1 + [Q_1^2/Q_2^2 + Q_1^2)]m_2 \tag{2.5}$$

Considering a probability density function conditioned by observations , the optimal estimate can be chosen among three possible choices:

- the mean which is the center of probability mass
- the median which is the value that separate in half the probability weight
- the mode which is the value with highest probability.

In our case, with a linear system and Gaussian noise, those three choices coincide. The Extended Kalman Filter proposes then an estimate of the process $x(k)$ given the known information Z_1, Z_2 to be $\hat{x}(k|Z_1, Z_2) = m$. We can rewrite the equation 2.5as follow :

$$\begin{aligned} &\hat{x}(k|Z_1, Z_2) = m_1 + K_2[m_2 - m_1] \\ &K_2 = [Q_1^2/(Q_1^2 + Q_2^2)] \end{aligned} \tag{2.6}$$

and we can rewrite the equation 2.4 :

$$Q(k)^2 = Q_1^2 - K_2 Q_1^2 \tag{2.7}$$

We can understand these equations as follows. Looking at the equation 2.6, the optimal estimate of the process $x(k)$ is the previous estimate added with a weighted adjusting term that takes into consideration the difference between the estimate and the observed value. Looking moreover at the equation 2.7, we understand that the new estimate $\hat{x}(k|Z_1, Z_2)$ and Q_2 contains all the information of the process $x(k)$ conditioned to the estimate and observation Z_1 Z_2.

Let's propagate that information in time. At the time $k+1$ our process x, which follows the equation 2.1, is moved with a control U and with a Gaussian noise term V related to the system source error with mean 0 and covariance Q_{sys}. Between time k and $k+1$, the density of the process $(x|Z_1, Z_2)$ has moved thanks to the control U and spread around its mean because of the uncertainty V. However, the density of the process x conditioned to Z_1, Z_2 at time $k+1$ is still a Gaussian with mean and covariance given by:

$$\hat{x}(k+1|Z_1, Z_2) = M_{rob}(k)\hat{x}(k|k) + U(k+1)[(k+1) - k] \tag{2.8}$$

and

$$Q(k+1)^2 = Q(k)^2 + Q_{sys}^2[(k+1) - k] \tag{2.9}$$

We understand now that without observation, $\hat{x}(k+1|Z_1, Z_2)$ is the best estimate of the process $x(k+1)$ given the information at time k. If an observation $Z(k+1)$ is made then we can return to the first step in order to estimate $\hat{x}(k+1|k+1)$ given the estimation $Z_1 := \hat{x}(k+1)$ and observation $Z_2 := Z(k+1)$.

We will apply what precedes to the extended process $x(k) = (X_{rob}(k), L)$, the state of the robotic system and of the landmarks. We compute the estimate of the process $x(k+1|Z(0), .., Z(k))$ under the form $\hat{x}(k+1|Z(0), ..., Z(k)) = \mathbb{E}[x(k+1|Z(0), ..., Z(k)]$. Given the equation 2.1 and the fact that

$\hat{Z}(k+1|k) = \hat{Z}(k|k)$ because the landmarks are supposed static, the extended process x follows the model :

$$x(k+1) = M(k)x(k) + u(k+1) + v(k+1) \tag{2.10}$$

. with $M(k) := \begin{bmatrix} M_{rob} & 0 \\ 0 & I \end{bmatrix}, u(k+1) = \begin{bmatrix} U(k+1) \\ 0 \end{bmatrix}, v(k+1) = \begin{bmatrix} V_{sys}(k+1) \\ 0 \end{bmatrix}$ which has mean 0 and covariance matrix $Q'(k) = \begin{bmatrix} Q_{sys}(k) & 0 \\ 0 & 0 \end{bmatrix}$

The Extended Kalman Filter (EKF) computes also the estimate of the covariance $P(k+1|k)$ corresponding to the uncertainty of the error $\tilde{x}(k+1|k) = \hat{x}(k+1|k) - x(k+1)$

$$\begin{aligned} P(k+1|k) &= \mathbb{E}[\tilde{x}(k+1|k)\tilde{x}(k+1|k)^T | Z(k)] \\ &= M(k)P(k|k)M^T(k) + Q'(k) \end{aligned} \tag{2.11}$$

Then, the EKF computes estimates of the future measurements conditioned to the previous observations.

$$\begin{aligned} \hat{Z}_i(k+1|k) &= HL_i - H_{rob}(k)\hat{X}(k+1|k) \\ &= H_i(k)[\hat{x}(k+1|k)] \end{aligned} \tag{2.12}$$

where $H_i(k) := \begin{bmatrix} -H_{rob} & 0 & \dots & H & \dots & 0 \end{bmatrix}$. Once the observation at time $k+1$ is obtained, the innovation is computed as follows :

$$J_i(k+1) = Z_i(k+1) - \hat{Z}_i(k+1|k) \tag{2.13}$$

with its covariance matrix

$$S_i(k+1) = H_i(k)P(k+1|k)H_i^T(k) + R_i(k+1) \tag{2.14}$$

The optimal estimate of the states at time k+1 conditioned to the past estimate and the new observation is now computed as :

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W_i(k+1)J_i(k+1) \tag{2.15}$$

with the Gain matrix

$$W_i(k+1) = P(k+1|k)H_i^T(k)S_i^{-1}(k+1) \tag{2.16}$$

and the estimated error covariance as :

$$P(k+1|k+1) = P(k+|k) - W_i(k+1)S(k+1)W_i^T(k+1) \tag{2.17}$$

The refinement at each iteration of the state of the landmarks and of the robotic state is now directly readable on P: $P(k+1|k) = \begin{bmatrix} P_{rob,rob}(k+1|k) & P_{rob,lan}(k+1|k) \\ P^T_{rob,land}(k+1|k) & P_{land,land}(k+1|k) \end{bmatrix}$ with $P_{rob,rob}(k+1|k)$ the matrix of covariance of the error estimates on the robotic system position, $P_{land,land}(k+1|k)$ the covariance of the error estimates on the position of the landmarks L, and $P_{rob,land}(k+1|k)$ the correlation between the robotic system position and the landmarks. At this stage, we have at each time k the optimal estimate of the robotic state $\hat{x}(k|k)$ as well as its uncertainty $P_{rob,rob}(k|k)$ which will act as a trust weight on the position $\hat{x}(k|k)$ of our system. We can show that at each iteration, the covariance on each landmark decreases. Indeed, because the initialization $P(0|0)$, Q, and R_i are positive semi-definite matrixes, then P, S, WSW^T are also positive semi-definite. Thus we can show that

$$\begin{aligned} \det P(k+1|k+1) &= \det[P(k+1|k) - W_i(k+1)S_i(k+1)W^T(k+1)] \\ &\leq \det P(k+1|k) \end{aligned} \tag{2.18}$$

This is still true for any square submatrix because a square submatrix of a positive semi-definite matrix is also positive semi-definite. Then

$$\det P_{land,land}(k+1|k+1) \leq \det P_{land,land}(k+1|k) \tag{2.19}$$

As we said, there is no stochastic influence of the observation of landmark at time k on the estimation of landmarks at time $k+1$ because landmarks are supposed static. So naturally $\det P_{land,land}(k+1|k) = \det P_{land,land}(k|k)$. Then,

$$\det P_{land,land}(k+1|k+1) \leq \det P_{land,land}(k|k) \tag{2.20}$$

This inequality is also true for any submatrix of $P_{land,land}$ [15]. So for all landmark i,

$$\sigma^2_{ii}(k+1|k+1) \leq \sigma^2_{ii}(k|k) \tag{2.21}$$

and the uncertainty on the landmarks around their value L_i decreases over time. We have now a successful way to improve the absolute position of the landmarks. Indeed, as we get several observations on a particular landmark i, we can trust more and more our latest observation $Z_i(k)$ that occured at time k. We can compute a new position of the i_{th} landmark with :

$$L_i(k) = H^{-1}(H_{rob}\hat{X}_{rob}(k|k) + \hat{Z}_i(k|k)) \tag{2.22}$$

Because x is a Markovian process, we can pretend that we reset the experiment at the time k and launch the process again with first observation $L_i = L_i(k)$ and associated covariance $R_i(0) = \sigma^2_{ii}(k|k)$ for all i. Thus, not only the uncertainties on the state of the landmarks decrease over time, but also the accuracy of the absolute position of landmarks increases.

2.2 Iterative Closest Point

Knowing the position of the robotic system was half of the problem. We must now use our position and our depth scan to generate the wanted map of the environment. The scan of the environment performed by the depth sensor is a point cloud. At the initialization, the empty map is integrated with the first point cloud, without any particular transformation. The map is then updated incrementally by adding every new obtained point cloud to the set of previous point clouds which constitute the map. The goal is to find the best transformation $T : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$ between two sets of points $\mathscr{M} = \{m_i\}_{i\in N}$ and $\mathscr{S} = \{s_i\}_{i\in N}$ which are the landmarks $\mathscr{M}$ of the current map that have been recognized and associated to the points $\mathscr{S}$ in the new point cloud. This ensures the two sets have the same cardinal and that we want to match the common portions between the current map and the new information. First, as the landmarks have been recognized and associated, we assume that the two sets can be assembled as a set of pairs of points $(m_i, s_i)_{i\in N}$ where for all i m_i and s_i correspond to the same landmark. We made the legitimate assumption that for each point m_i, the point s_i will be the closest point in $\mathscr{S}$ to m_i. The searched transformation T aims to minimize the Root Mean Square Distance (RMSD) between two sets which is defined as follows:

$$RMSD(\mathscr{S},\mathscr{M}) = \frac{\sum_{i\in N} |s_i - m_i|^2}{N} \tag{2.23}$$

The rigid transformation T is composed of a rotation matrix $\mathscr{R}$ and a translation vector t. A good initial guess for $\mathscr{R}$ and t can be computed from the robotic state. Indeed, the orientation θ of the robot in the map (that is retrieved by EKF) is directly the inverse rotation we aim to perform onto the set $\mathscr{S}$. The translation vector is equivalent to the translation between the origin 0 of the map (origin of the set $\mathscr{M}$) and the robot current position. We are lucky to benefit from a very good initial guess. To refine our transformation, the algorithm proceeds with the Singular Value Decomposition (SVD) [21] which consists in computing R and t given two sets of points:

$$\begin{aligned}
\text{Computation of centroids } C_{\mathscr{M},\mathscr{S}} &= \frac{1}{N}\sum_{i=1}^{N} m_i, s_i \\
\text{Aligned sets } \mathscr{M}' &= \{m'_i = m_i - C_{\mathscr{M}}\}_{i\in N} \\
\mathscr{S}' &= \{s'_i = s_i - C_{\mathscr{S}}\}_{i\in N} \\
\text{Covariance Matrix } H &= \mathscr{M}'\mathscr{S}'^T \\
\text{Singular Value Decomposition [21]} H &= U \wedge V^T \\
\text{Rotation matrix given by } R &= VU^T \\
\text{Translation vector } t &= C_{\mathscr{S}} - RC_{\mathscr{M}}
\end{aligned} \tag{2.24}$$

The transformation is then applied to the set $\mathscr{S}$ as $T(\mathscr{S}) = \{Rs_i + t\}_{i\in N}$. The full algorithm consists of starting from the said initial guess T_0 computed from the current robotic state. At each iteration, the algorithm computes a new transformation T_{k+1} given the sets $\mathscr{M}$ and $T_k(\mathscr{S})$

thanks to the SVD. It also computes the error $RMSD(T_k(\mathscr{S},\mathscr{M})$. If the $RMSD$ error is below a chosen threshold value, the algorithm stops and returns the wanted transformation T as the product $T_n \circ \cdots \circ T_0$ of all transformations computed. That rigid transformation will then be applied to the entire new point cloud before adding it to the set of points clouds that constitutes the 3D map. This algorithm constitutes the basic approach to solve the problem. We treated the continuous world as a finite set of points, and we treated the points as absolute positions of the landmarks. However, the first section on Extended Kalman Filter showed us that from the robot perspective and due to the uncertainty R of its sensor, a landmark point m_i is an observation of a Gaussian with covariance R and mean the absolute position of the landmark. Following this idea, the sets of points can be translated into continuous densities functions of what we call Gaussian Mixture Models. What was a discrete optimization problem in our basic approach is now a continuous optimization problem and can benefit from the powerful mathematical theory about it. We will stick to the basic approach [25] [8], but more details about Gaussian Mixture Models can be found in [9].

2.3 Algorithm Architecture

The overall architecture of the implemented software is shown in Figures 2.12.2.

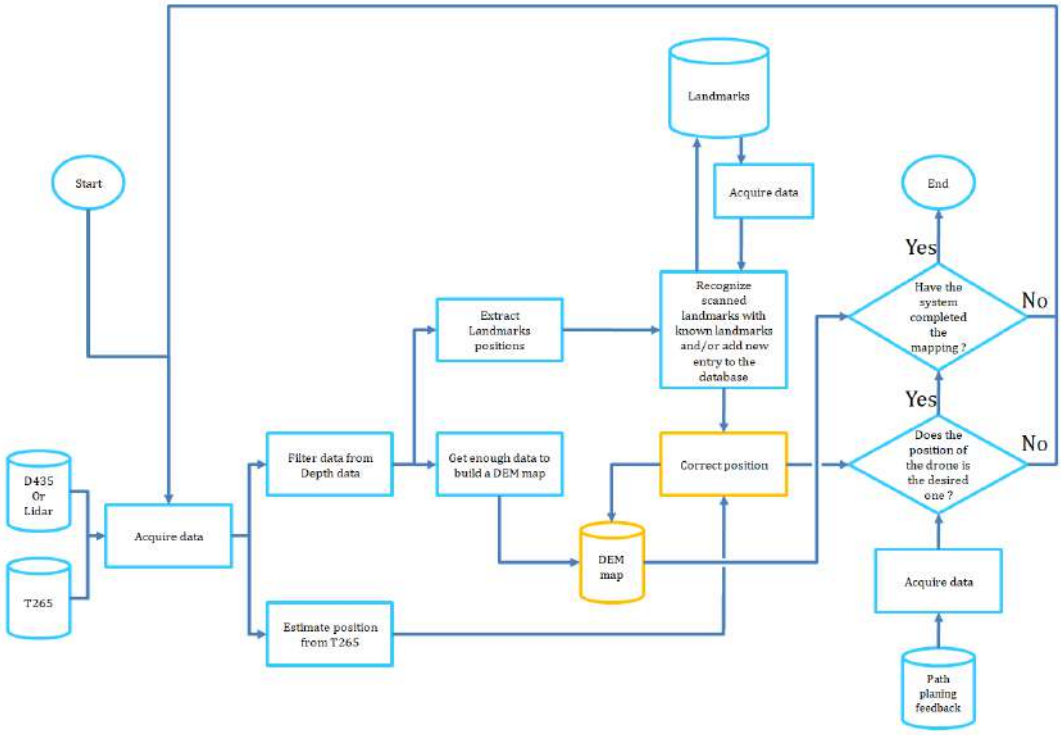

Figure 2.1: Process Flow Diagram Drone SLAM algorithm

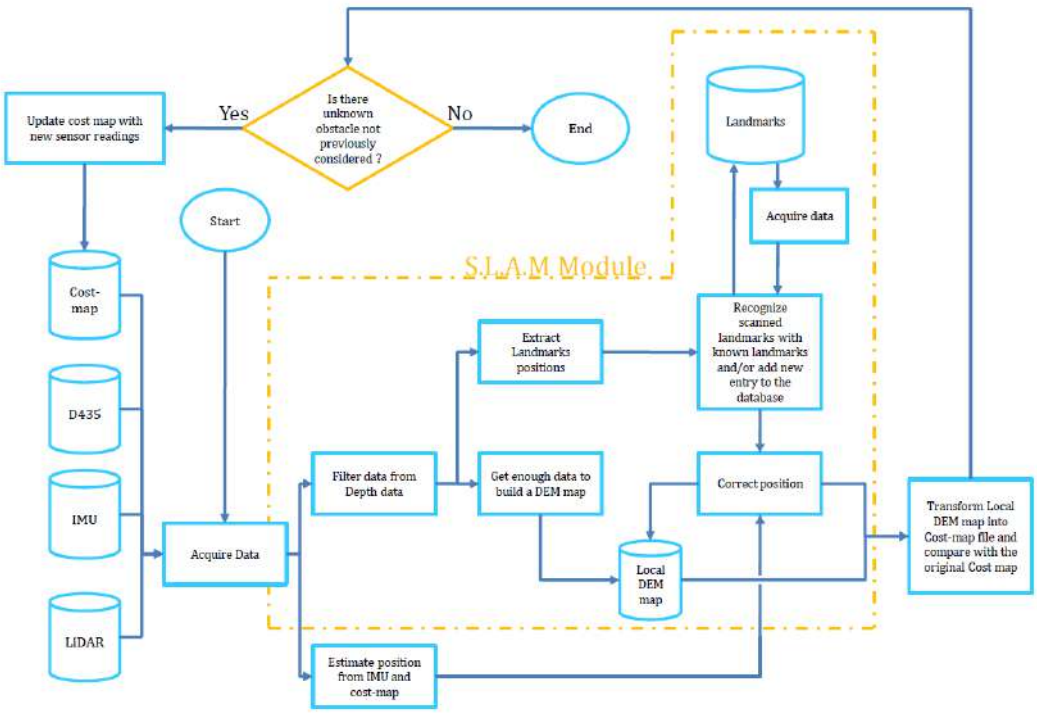

Figure 2.2: Process Flow Diagram Rover SLAM algorithm

2.4 Conversion of 3D map toward 2D cost-map

Task Planning and Path Planning systems of the rover require a 2D map called "cost-map", therefore a conversion module is needed to translate the obtained 3D DEM map to the format needed by the other teams.

The first step consists of extracting one layer of the 3D map that represents the topography of the ground. Indeed, the 3D map obtained by the SLAM algorithm is in general very dense as it is the stacking of a large number of noisy point clouds that when are stacked thicken the elements seen by the camera. The output of that step is a 3D dem map where a column (x,y) contains a maximum of one point. In other words, in that step, we remove all the points that are not strictly needed to identify obstacles, reachable or unreachable areas for the rover. An example of this step is shown in Figure 2.3.

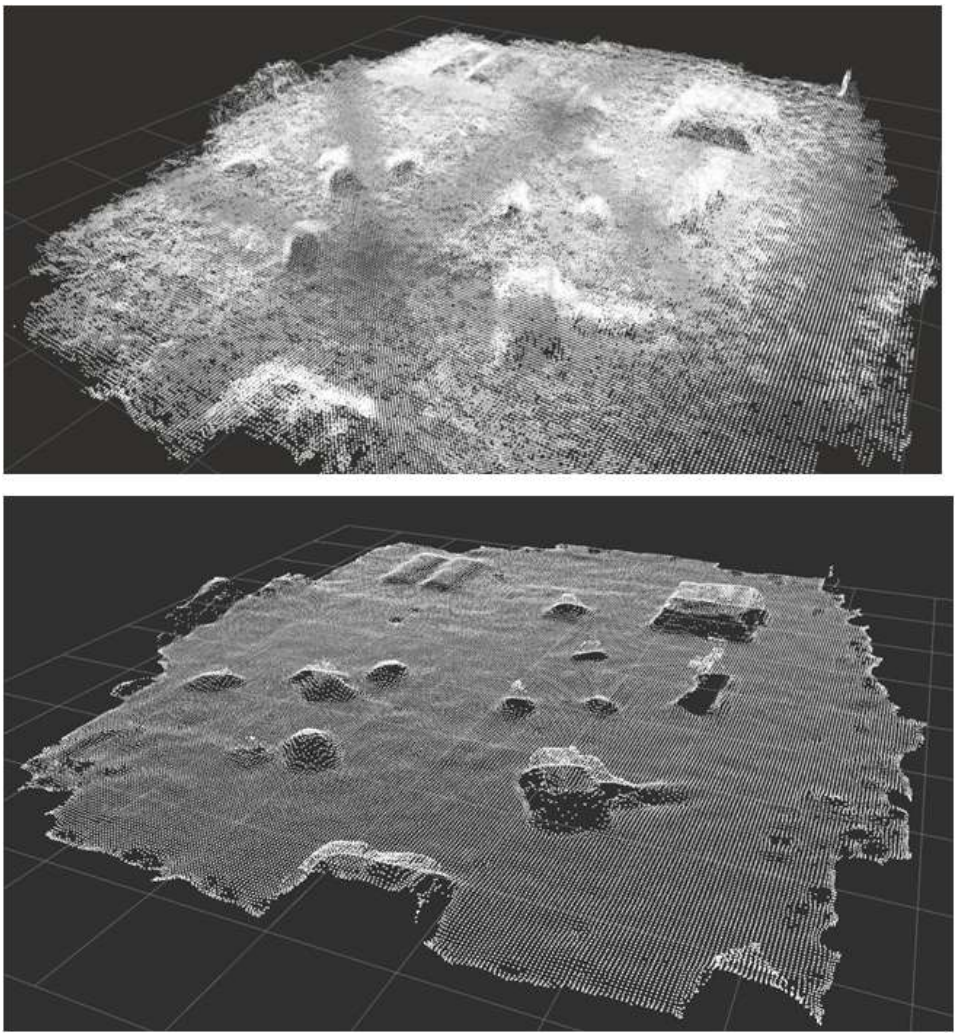

Figure 2.3: 3D dem map before and after being extracted

The second step consists of the detection of unreachable areas for the rover. An iterative process on each point analyses its closest neighbours in order to deduce if that point is reachable or not by the rover. More in detail, thanks to the slope characteristics that have been empirically obtained with tests conducted on the rover in section 1, the algorithm computes the steepness of the ground at each point and deduces if it is below or above the maximum steepness crossable by the rover. The algorithm then removes every point that is considered reachable for the rover.

The third step consists of the projection of the remaining points onto a horizontal plane which is the 2D occupancy grid. The occupancy grid contains only three values: 0 (white) for unoccupied, 100 (black) for occupied, -1 (grey) for unknown. The final output of the conversion is the 2D map written as an occupancy grid file format [2]. It contains information such as the coordinates (x,y) of the origin, the resolution, the height and width of the map, and a list that contains the values (-1, 0 or 100) of all the cells. Graphical results will be presented in the next chapter.

CHAPTER 3

EXPERIMENTATIONS

3.1 Robotic Work Environment

Testing our implemented algorithms on a real robot required to work with the Robotic Operating System (ROS). Indeed, as our algorithm takes sensors' data as input, ROS is the middleware that interfaces the physical components of a robot with the embedded computing programs that run the robot's functionalities. Ros allows any part of the robotic system to be connected to another part of the robot. In our case, we configured our algorithms to listen to the camera and odometry sensors' outputs, and synchronize them before using them as inputs of our algorithm. The algorithms then publish their outputs, which are a map and the position of the robotic system, as messages that are available at their turn to any other part of the system that will listen to them. Moreover, our mathematical models and algorithms act as if the robotic system was reduced to a point. However, in reality, the coordinate frame at which the odometry is measured and the coordinate frame at which depth scans are generated are fundamentally different because they correspond to different sensors. To tackle that issue, ROS allows us to create a Transformation Tree (TF) which links every component of the robotic system between each other with transformation functions (static or time-dependant). That way, we can bring all data to a unique 3D coordinate frame belonging to the TF. Its origin will be the point at which our algorithm will compute the position and generate the map. With the TF again, we can then compute the position in the map of any part of the robot. A Transformation Tree of our drone is shown in Figure 3.1.

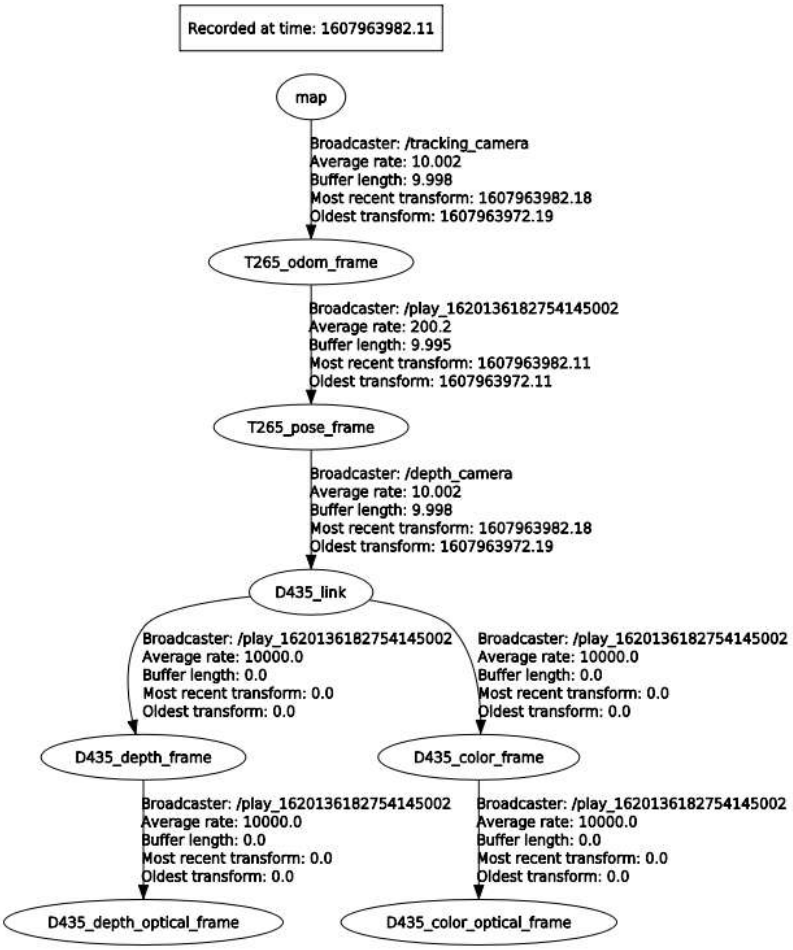

Figure 3.1: Example of drone TF

Learning and working with ROS was the principal activity of my internship. Applying an idea to reality is a challenging task. Implementing the algorithms can be considered an easy task. Testing them however expect you to solve a lot of issues that occur all the time with real a physical system. Among them I can exhaustively cite solving communication issues between different part of the robot, communication issues between the remote computer with which you control the robot and the robot itself, finding the right way to run a lot of programs in parallel, monitoring the CPU utilization so that every program can run smoothly, making sure that all used programs are compatible with each other and with the version of the operative system we are working with. I also spent time interfacing my code with ROS so that my algorithms become easy manageable by the State Machine (SM) (implemented by my supervisor Jasmine the Task-Planner) which is the "head" of the robot that dictates autonomously the sequence of actions to be executed by the robot. In a nutshell, I learned a lot about the inescapable challenges to apply even a simple idea to reality. I am very grateful because that was an experience that

taught me about the whole chain between the idea and the final product, and I am now more confident to be able to design a project and brings it to completion. Further discussion in section 4.

3.2 Post-Processing Real Data

Designed algorithms are great, but in the conception part, we tend to forget that reality is not as smooth as we depicted it in our models. Once the robot starts moving, there are vibrations, tiny shocks, fast movements, and a lot more effects that disturb if not disrupt, the sensors and algorithms. Moreover, we want our algorithm to process data faster than the rate at which the data is generated because we want the whole process to be running in real-time. To achieve that, we need to optimize the sensors' data by reducing their size and reduce the impact of disturbing effects. This section presents a list of tiny algorithms that were implemented for which the underlying mathematical theory is not very complex. Their use on depth scans will improve the data's quality and will improve the map quality as well as the computational process of the SLAM algorithm.

3.2.1 Voxel Grid Filter

The first thing we can do the reduce the size of a point cloud is to apply a voxel grid filter[6]. A Voxel[5] is the 3D equivalent of a 2D pixel, so it is a cube that can be used to partition the 3D space. We can set the size of the cube and subdivide the 3D space into voxels. If a voxel is occupied with one or more points of the point cloud, we then reduce all those points into a single point which is the centre of the voxel. That way, we reduce drastically the number of points of the original point cloud, keeping however reliable information of the topography of the environment.

3.2.2 Outliers Removal

Very often, a depth scan of the environment contains aberrant data, which are points of the point cloud that have no physical sense. For example, if a point is isolated, very far from others, it is a sign that this point is not corresponding to a physical object that has been scanned. Indeed, if it was the case, then the sensor would have detected a small portion of the obstacle and thus the point would have been surrounded by a set of other points. Following this idea, we can implement a Radius Outlier Removal[3] that removes a point of a point cloud if its number of neighbours within a chosen radius is below a chosen threshold.

A similar idea to detect aberrant point is to take into account the Gaussian noise injected in the point cloud due to the system uncertainty (vibration, sensor precision,...). We suppose that the points of a detected element follow a Gaussian distribution with a specific mean and standard deviation Following this idea, we then analyze a point as follows. We compute the mean

distance between it and its N closest neighbours. If that mean is outside the standard deviation, we conclude that that point was aberrant and we remove it from the point cloud [4].

An example of Point cloud where the outliers were removed according to these filters is shown is Figure 3.2.

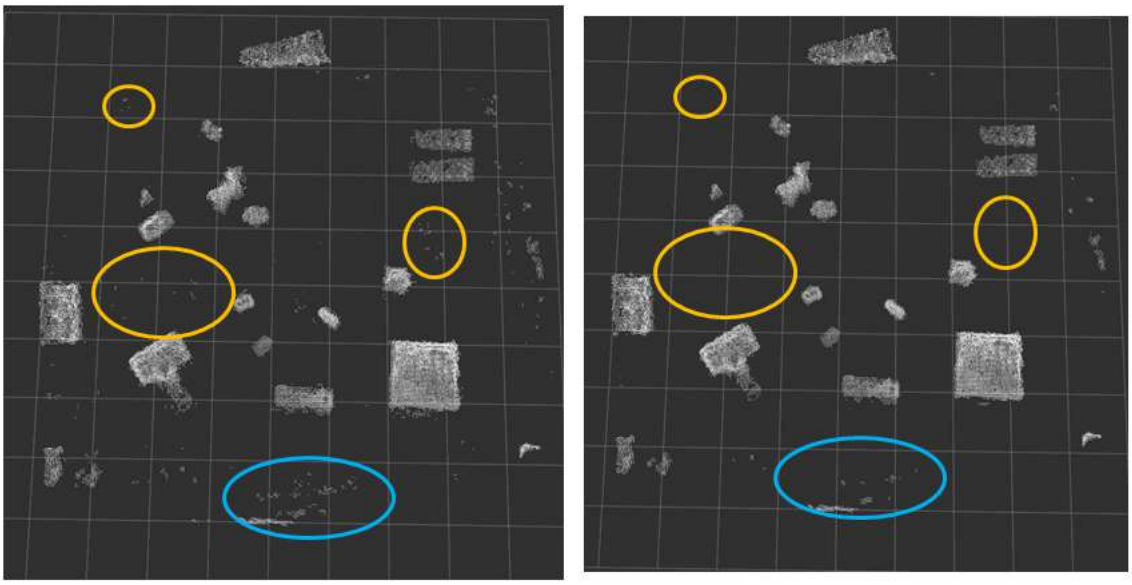

Figure 3.2: 3D DEM map obtained from the D435i Configuration before and after being filtered with the Outliers Removal algorithms

Note that this process can be stacked several times until we reach the desired outliers purge level. Of course, it has a limit but we empirically showed that stacking these filters 2 times on the depth sensor output gives even better results.

3.2.3 Statistical Denoising

To reduce the noise on a point cloud, we followed this strategy: For each point p, we search for all of its neighbours within a certain range. Then, we compute the mean point m and the covariance matrix A of the found neighbours. When then replace the point p in the point cloud by the point q where :

$$\begin{aligned} q &= (A + \varepsilon I)^{-1} p + b \\ b &= m - Am \\ 0 &< \varepsilon < 1 \end{aligned} \tag{3.1}$$

ε is a parameter that can be empirically chosen (value 0.01 in our case). The idea behind this algorithm is to move a point where its probability to be according to the local distribution of its neighbours is the highest. The literature on denoising algorithm is wide [27] but this basic

approach was efficient enough regarding our requirements so we decided not to go further even if it is a very interesting computer vision subject. An example of a denoised outside bench that was scanned by the drone can be seen in Figure 3.3.

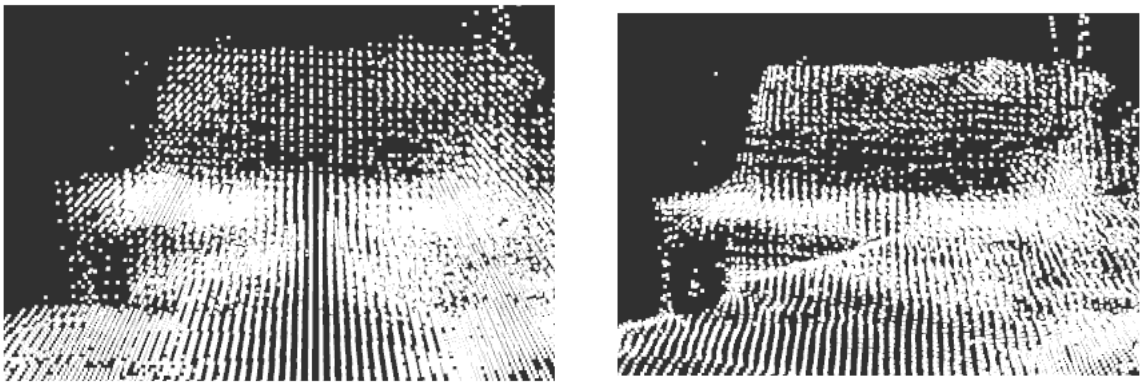

Figure 3.3: An outside scanned bench with the mapping process (D435i Configuration) before and after the denoising process

3.3 Results of the Mapping process

The mapping process is constituted by the chain as follows : The raw data coming from the depth sensor is firstly optimized thanks to the 1.Voxel Grid Filter, 2. Outliers Removal, 3. Statistical Denoising. Then, the SLAM algorithm is performed on the odometry data and the post-processed depth data. At this point, the drone's position is used by the other part of the system (Task-Planning and Path-Planning), while the map is incrementally generated and saved in the system. At the end of the mapping phase, the 3D map is post-processed with the same tool as used on the depth scans. Indeed, it is not because each scan is post-processed that the final map with the stacked and transformed point cloud with ICP will be smooth as well. So again, on the map we perform a 1. Voxel Grid Filter, 2.Outlies Removal, 3.Statistical Denoising. Then, the conversion algorithm described in the previous chapter is used to generate the 2D occupancy grid that is our wanted final product. As mentioned in the first Chapter, we were working with two configurations: one with the D435i Depth camera as a depth sensor, and one with the LiDAR as depth sensors.

3.3.1 Indoor Tests

Indoor tests occured in the "Volière" of the ISAE-Supaero DISC (Département d'Ingénierie des Systèmes Complexes).

Results Test SM-1.D and Test SM-1.L The SM-1 test goal was to test the sensor's capabilities in the easiest situation. We placed the sensors on a rotary bench to get a very stable movement.

Tests SM-1.D refers to the "depth camera configuration of the of test SM-1" and SM-1.L refers to the "LiDAR configuration of test SM-1". At that time of the project, the Conversion Module that translates the 3D map into 2D was not complete, so a simplistic version of it has been used.

The test results SM-1.D are shown in Figure 3.4 and Figure 3.5.

Videos :
D435i:
https://drive.google.com/file/d/12nqdqYho1jYAUjkdk27BWWxLcf8UhDAY/view?usp=sharing
https://drive.google.com/file/d/1kCQ78E4ByOLaxlw51W9RyJ-CV12ojdLl/view?usp=sharing
LiDAR:
https://drive.google.com/file/d/12GdnFwZLJIVtIQtD5eAvc-J5_Zy0xs1v/view?usp=sharing
https://drive.google.com/file/d/1OYr9FEUMdMZtDFHH8_gSrT8SBgv-sDB1/view?usp=sharing
https://drive.google.com/file/d/1viXe_AJQNi9RchnESL4PLZdaSWPxZbb_/view?usp=sharing

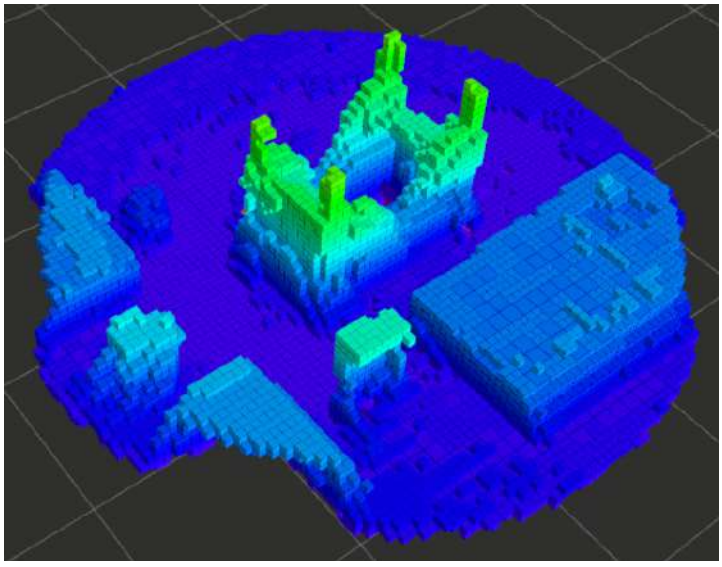

Figure 3.4: SLAM DEM Map Test SM-1.D Resolution 0.03m.

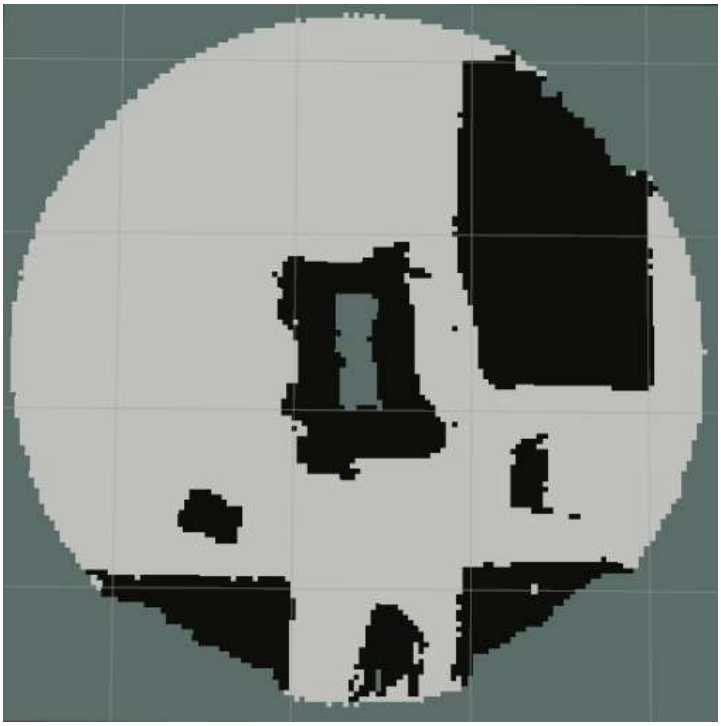

Figure 3.5: SLAM Occupancy Grid Test SM-1.D Resolution 0.03m.

The test results SM-1.L are shown in Figure 3.6 and Figure 3.7.

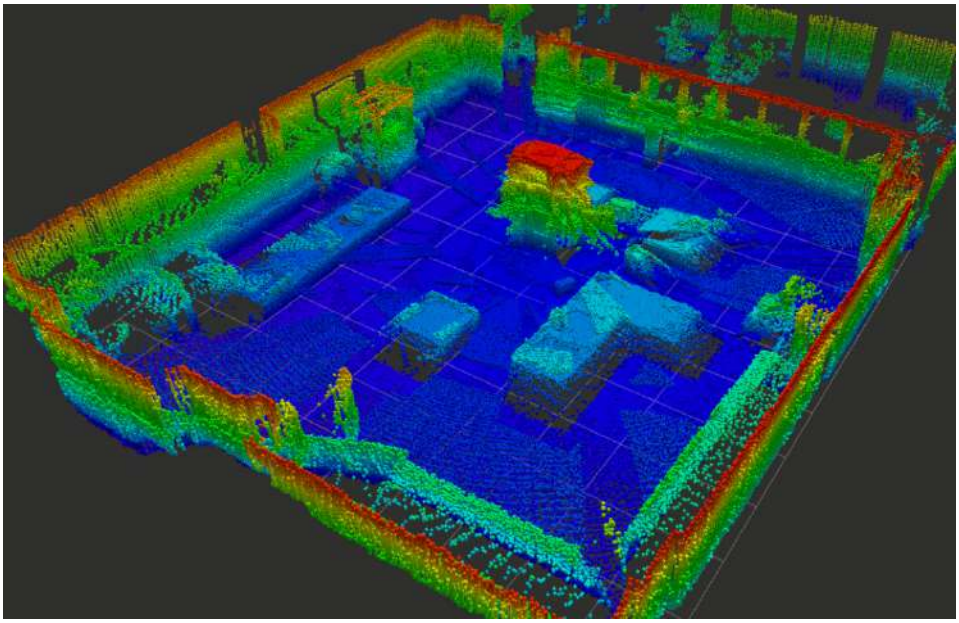

Figure 3.6: SLAM DEM Map Test SM-1.L Resolution 0.03m.

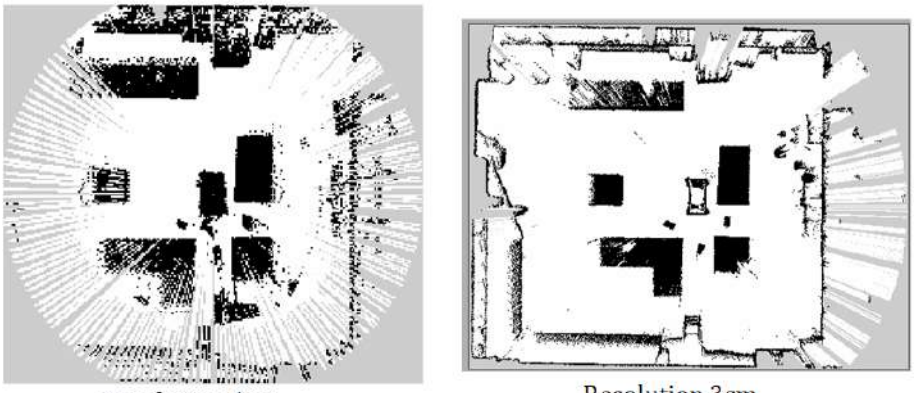

Figure 3.7: SLAM Occupancy Grid Test SM-1.L before and after Depth Filtering Module implementation

Results Test SM-2.D and Test SM-2.L The drone was flying at a height of 2m +/- 20cm. The flight time was 3 minutes and 39 seconds. It was manually piloted and performed a zig-zag trajectory over an area of 8x8m. The Trajectory was not the optimal one as the drone repeated some parts of its trajectory. The simplistic version of the Conversion Module started showing its limitations. The Conversion Module work started after that test. The results of this test are shown from Figure 3.8 to Figure 3.14.Once the conversion module was finished, it was applied to the same data set. So, an example of the final occupancy grid of the SM-2.D test is shown in Figure 3.12.

Videos:
https://drive.google.com/file/d/1WHupc7u8lN_LTh31nvroBs170tEmyLe2/view?usp=sharing
D435i:
https://drive.google.com/file/d/10GhugTR0lmueM6Kfo1j9O7ZlspbZC_CF/view?usp=sharing
https://drive.google.com/file/d/1B94ydef2N-n60dMnLzaiyhNpEd-uXCwU/view?usp=sharing
LiDAR:
https://drive.google.com/file/d/1Q8otgCMhTQW7WP3C72zD0M3BsFVJqvfg/view?usp=sharing
https://drive.google.com/file/d/1cAdpl9bAwJDTWo7a4HWmCGmLwFCrYMpf/view?usp=sharing

Figure 3.8: Test SM-2 Set-Up

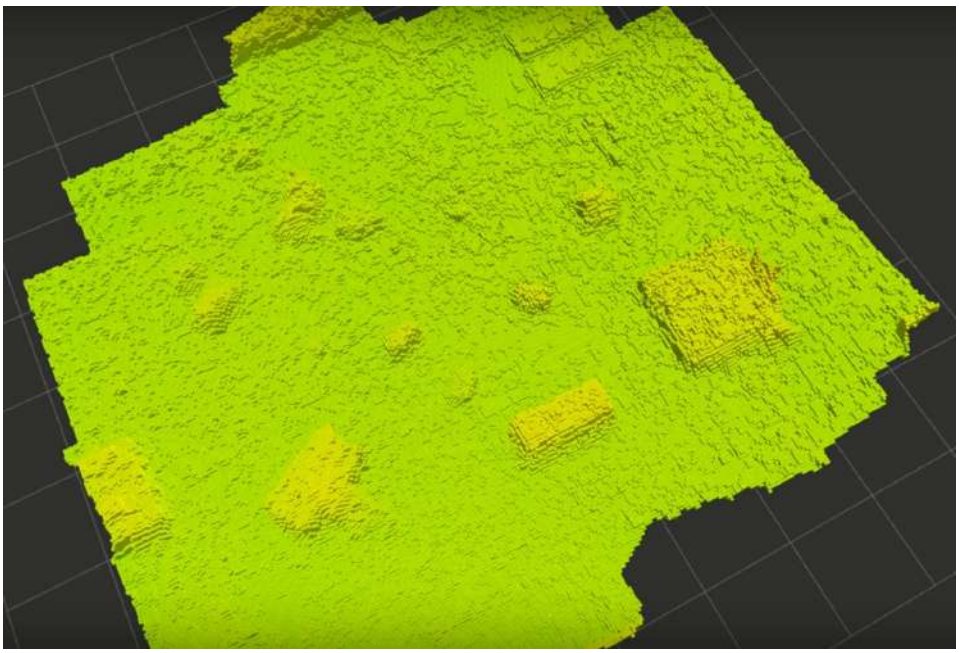

Figure 3.9: SLAM DEM Map Test SM-2.D Resolution 0.03cm

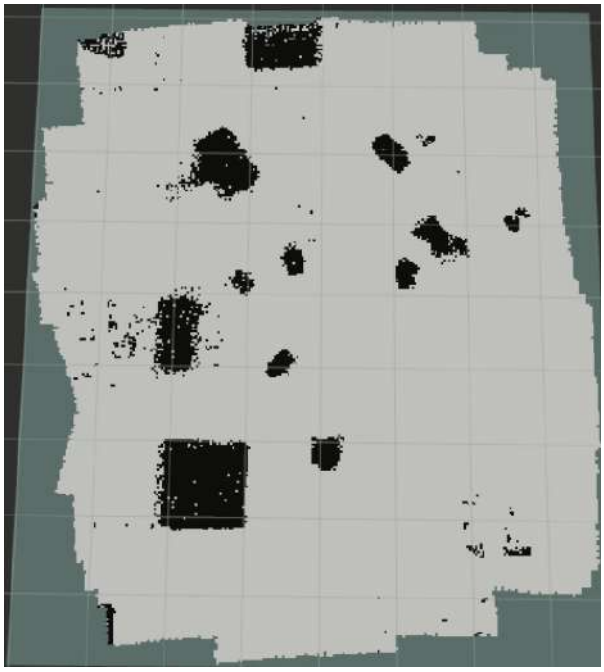

Figure 3.10: SLAM Occupancy grid Test SM-2.D Simplistic version of Conversion Module

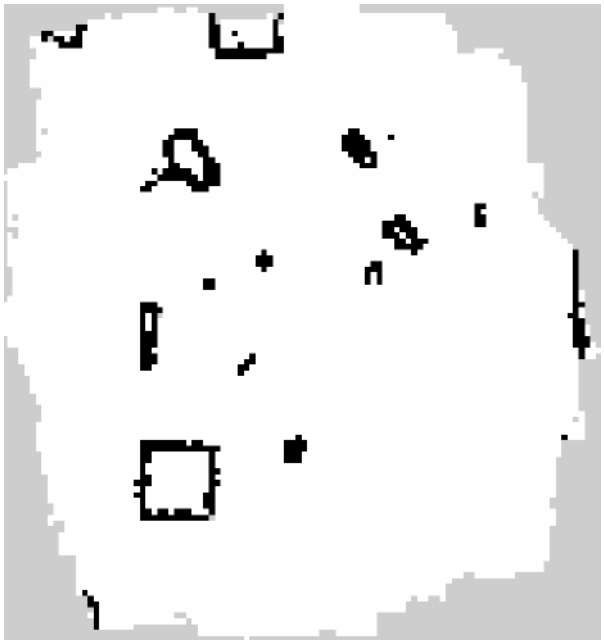

Figure 3.11: SLAM Occupancy grid Test SM-2.D Final version of Conversion Module Resolution 8cm

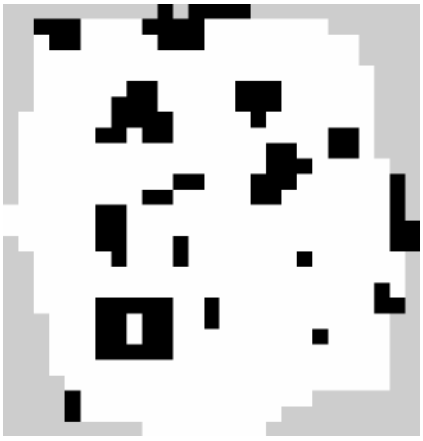

Figure 3.12: SLAM Occupancy grid Test SM-2.D Final version of Conversion Module Resolution 32cm

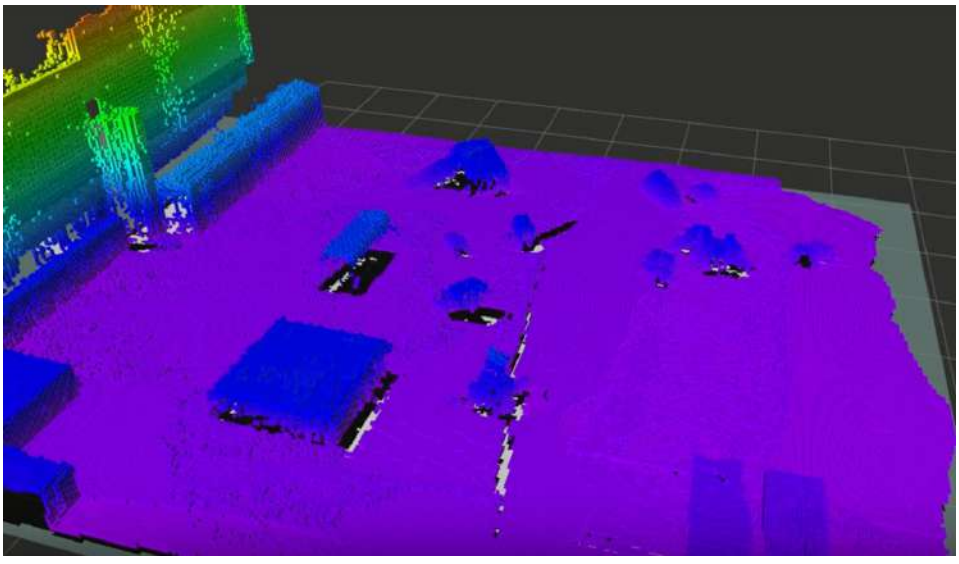

Figure 3.13: SLAM DEM Map Test SM-2.L Resolution 0.03cm

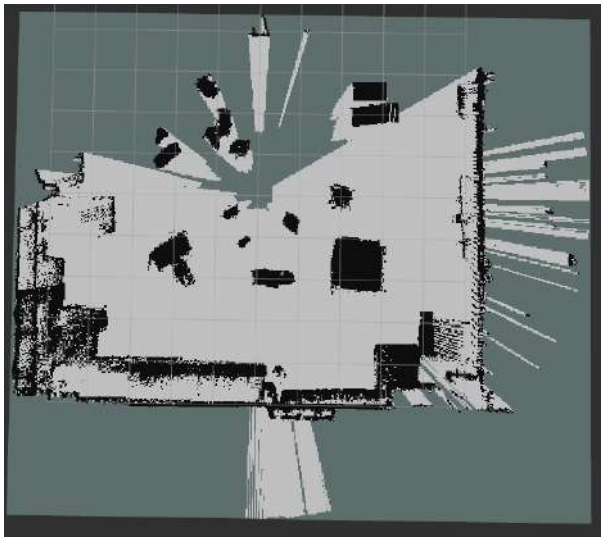

Figure 3.14: SLAM Occupancy grid Test SM-2.L Simplistic version of Conversion Module

Figure 3.15: SLAM Occupancy grid Test SM-2.L Final version of Conversion Module Resolution 0.06m

3.3.2 Outdoor Tests

In the outdoor environment set-up, the main obstacle was a bench. It was a tough obstacle to be rightly detected by the sensors as it was composed of an alternation of metallic strips and void

(allowing the light to get through). The area near the bench was covered with no or little grass about 5cm max. The area beyond the bench was all covered with grass, from 5cm to more than 10cm (with dead leaves). As we need legal authorization and a drone pilot(which we have for the final field campaign) to make a drone fly, we conducted the tests by holding the drone by hands and simulating a flight pattern.

Figure 3.16: Outdoor test set-up

Figure 3.17: outdoor test set-up Grass detail

Figure 3.18: SLAM Test SM-3.D Bench Picture by the drone D435i camera

Figure 3.19: SLAM Test SM-3.D Grass Picture by the drone D435i camera

Results Test SM-3.D The results of the test are shown from Figure 3.20 to 3.22. Videos:
`https://drive.google.com/file/d/1DAItEeQEMOC3aJH_yCfp6e3kAsdFAY4o/view?usp=sharing`
`https://drive.google.com/file/d/1yJ9LtHGM8_jtaAOS3jDp8Fd-9Pamg2nJ/view?usp=sharing`

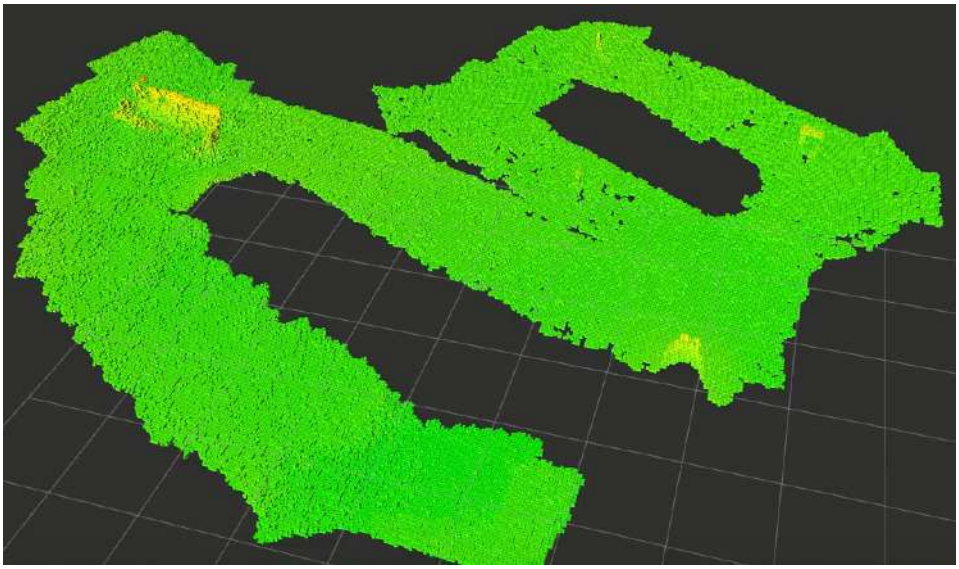

Figure 3.20: SLAM DEM Map Test SM-3.D Resolution 0.03cm

Figure 3.21: SLAM Test SM-3.D Occupancy grid Resolution : 8cm Size 150x172 pixels

Figure 3.22: SLAM Test SM-3.D Occupancy grid Resolution : 32cm Size 38x43

The D435i Configuration gave outstanding results as we obtained a map that registers all real obstacles for the rover which were on the testing field, including the bench, the tree, and a Lego Brick (that is not visible in the photo) that we put arbitrarily on the ground. However, a starting effect has been identified where we can see in the bottoms a set of black cells which are supposed to be white, and it is due to the quick drone lift-off. Let's recall that the EKF algorithm is more and more robust with time so the position of the drone is the weakest at the beginning of the experiment. Adding up to that the fact that at the beginning, the drone lifts off from 0 to 2/3m in less than 5seconds, and it is enough for the map to contain some mistakes in that area. A solution has been proposed in collaboration with the Task-Planning team to ask the drone to start the mapping process after the drone performed its lift-off.

Results Test SM-3.L Concerning the LiDAR configuration, the results of the test are shown from Figure 3.23 to 3.25.

Video:
`https://drive.google.com/file/d/1iq1hqqbj6m84CBInZ6BcLb0IPVGlM7tu/view?usp=sharing`

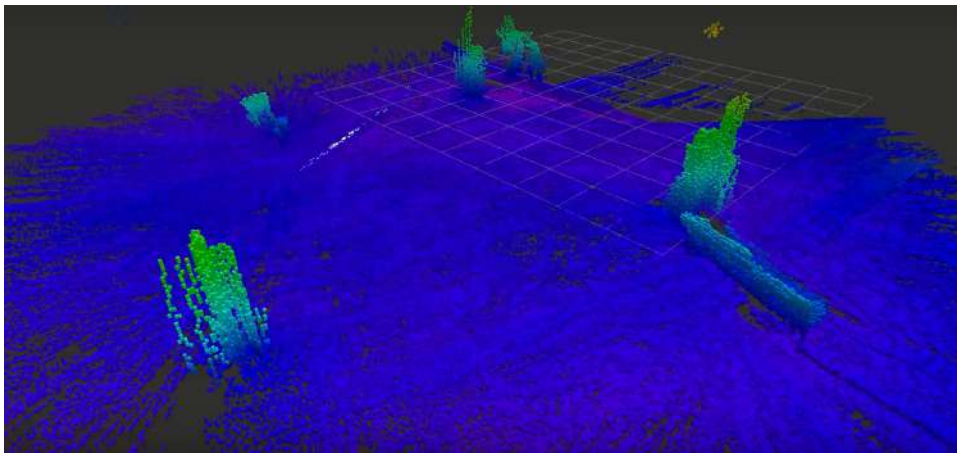

Figure 3.23: SLAM DEM Map Test SM-3.L Resolution 0.03cm

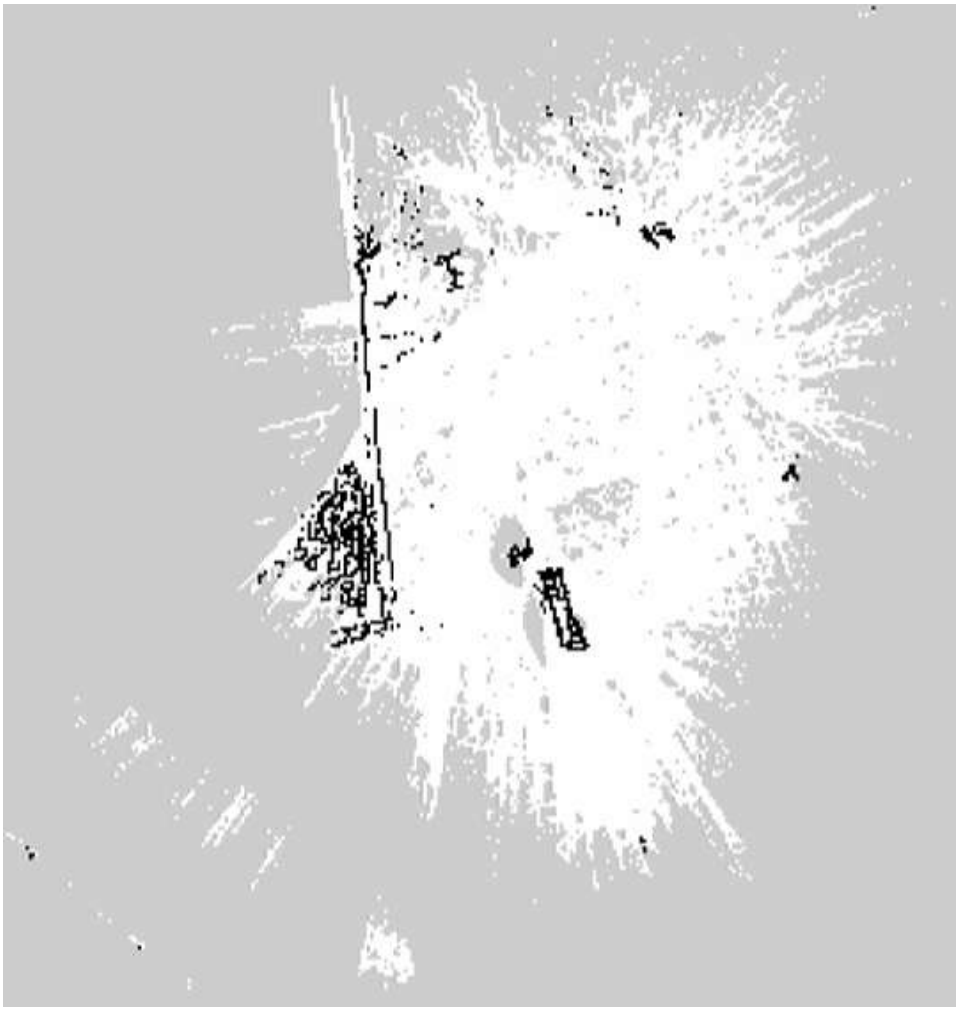

Figure 3.24: SLAM Test SM-3.D Occupancy grid Resolution : 0.08m

Figure 3.25: Occupancy Grid LiDAR Outside Resolution : 0.3m Size

The LiDAR did not react well to the outside conditions. It was predictable in the sense that the Infrared sun rays blind the sensor, and the complex texture of the grass created a very chaotic 3D ground in the resulting map. The advantage of the D435i was its weakness. Because the D435i is less accurate, it thickens the grass observed and the resulting 3D ground is noisy, but smoother than what the LiDAR has generated. In conclusion, the Depth camera generated a 3D map that reacted better with the Conversion Module which consists of the computation of slope gradient.

CHAPTER 4

DISCUSSION

4.1 Meeting the project requirements

To assess the accuracy of the position of the drone indoor, we used the opti-track system of the Volière. On average, we measured the accuracy of position about 15cm +/- 5cm. From time to time, especially at the beginning, the system can quickly lose track of itself to about 0.5/1m but will in general easily find back its position after few seconds. Concerning the map quality, we reached the Mission-Analysis requirement of 80% of the elements that represent an obstacle for the rover were registered on the map. To be more precise, tall and noticeable obstacles were fully detected. The obstacles that might not be registered are the ones with a scale that is close to the sensor accuracy limits. Indeed, the maximal obtained resolution of the 3D map was 3cm. An element can constitute an obstacle from 6cm height. That means that those tiny obstacles are translated on the 3D map by only two stacked cubes. The D435i camera uncertainty creates noise that has more or less the same scale as those tiny obstacles. With the denoising process, we were successful in identifying tiny 6cm obstacles, but some of them were flooded in the ambient noise. Concerning the big obstacles, like the bench we saw in the previous chapter, it happens that we do not detect the entire obstacle. However, we still detect the main shape of them and it is enough for the Path-planning to set a trajectory that will avoid them. We saw that when we downsize the resolution of the obtained 2D map (like 30cm resolution), the area of the whole bench is covered by black cells. In any scenario, the rover will progress on its designed path by analyzing its environment and adjusting its path if it detects unregistered obstacles.

4.1.1 Localization in the generated map

When the rover receives the map, there is no way a priori to determine its position in it without engaging a localization process. Indeed, we would need the transformation between the 3D coordinate frame that was used to generate the map and the rover's coordinate frame. One way would be to localize the rover with the drone while the drone is moving. Another way, which is safer because it does not involve the drone flying over the rover to detect it, is to use the knowledge the drone has gathered about the environment. Indeed, while the drone was mapping the environment, it also detected several points of interest (which are cubes with AR-tags on their faces). Those elements, which represent something noticeable, were registered along with their coordinates in the map in the memory of the drone. That data is also shared with the rover as the rover must visit them for the mission. The idea is the following: when the rover starts its mission, it begins by searching in its local environment such elements. When it detects 3 of them, it can use their coordinates in its frame and their coordinates in the frame of the map to triangulate itself in the map. That way, it will compute the transformation between its coordinate frame and the coordinate frame of the map, and it will know exactly its position in the map. When the drone is shut down and when it is started again (for example when we change the battery, or in an emergency case where it has to stop its functions), the drone will lose its coordinate frame. Each time it will be powered on it will generate a new coordinate frame. Thus, exactly as the rover, we will use that presented method to relocalize it in the map.That task is not fully done yet and will be a collaboration work between Maanasa who implemented the AR-tag detection algorithm and me. At the moment, starting from the ar-tag detection program implemented by Maanasa, I have written a program that enables the robotic system to automatically store in a database the information about all the points of interests(AR-tags) it has detected. I'm working on giving an automatic order to the robotic system to take a photo each time it sees a new point of interest (for the mission goal and for human analysis), as well as the triangulation program. Concerning the rover, we have an even better method in some cases. Because the rover is operating in a 2D relatively plane field, it needs to detect only one or two points of interest to localize itself. Indeed, we only need an initial guess of its position in the map to perform a localization algorithm, AMCL[24], which compares the scans generated by the rover with the known map, and try to match the topography of the scan with the map in order to find the localization of the rover.

4.1.2 Updating the map with the rover obstacle avoidance module

Currently, the rover can operate autonomously, with or without a map provided by the drone in case of a contingency case. What is left to be done is to implement the Updating Module that uses the detected obstacles by the rover to compare them with the map and updating it if some of them are not registered. This task is in theory simple but in reality challenging. The map accuracy depends on the drone's position accuracy, the drone's sensor accuracy. Then, the rover's position must be very reliable as well as its sensors in order to be sure that a seen obstacle is really an

unregistered obstacle and not just an obstacle that exists in the map but few centimeters away from its registered position. We use inflation radius on the registered obstacles in order to add safety to our operations and be certain that the rover will use a safe path. But with the updating module, all the uncertainties from the drone's and the rover's are stacking up and that's why it is challenging.

4.2 Outside the SLAM subject

The CoRoDro project is a very complete project proposed for the IGLUNA 2021 event. The requirements from the Space Innovation who holds the event are complex and diverse. Beyond the algorithm developing part (which requires 3 sections as mentioned : Path-Planing Task-Planing and SLAM), a whole team of five students is in charge of the Mission Analysis. Their role includes setting up all the requirements of the project such as Performance, Safety, Communication, Functional, constraints requirements and making sure they are met. They are also in charge of the Risk-Analysis which represents a huge task where they list every possible risk and compute their associated likelihood of occurrence, and they design as well their associated contingency plan. Of course, the developing teams know how the system behaves so we all collaborate to talk about the weaknesses of our robotic systems and to design the contingency plans.
Outside my expertise, we have a Communication team that is in charge of the Outreach and communication of the project on the media and was in charge to find grants to fund the project. I can mention and thank Valentine Bourgeois who found two grants of 5 000euros each that would pay the travel costs for students related to the Field Campain (FC) that would happen at Mount Pilatus (but who might be cancelled due to Covid).
More about my work, I had to inform the Space Innovation of the characteristics of our robotic system so I learned precisely about the components architecture of the drone and rover along with their power consumption. To stay in the competition we had indeed to prove to the Space Innovation jury that our system was well-designed and matching the space conditions.
Moreover, we had to demonstrate the relevance of the project and its economic viability. In other words, we had to pretend to be a new company that would effectively start an economic activity with the systems we are developing. I collaborated with a member of the Project Management team to write a Business Plan for our CoRoDro company. It includes a market analysis of several sectors that could make use of our technology. Among the Earth applications, we can cite the application of our project in the agricultural sector where our drone and rover would autonomously scan the state of the farmer's crops and deliver customized treatment only to the plants in need instead of treating currently the whole field. We identified that our principal added value is the ability of our drone and rover to collaborate, which is a capability that is currently very rare in the robotic market.
From my implication in several key parts of the project, especially the developing part and the

engineering part of the system design, I acquired expertise and a global vision of the project. For that and because other parts of the project still need to be implemented, I was hired as an Engineering Assistant at the DCAS department until the end of the project which will occur right after the demonstration of our systems at Mount Pilatus in July. If the field campaign can't happen in Switzerland due to the Covid, we will perform it at the CNES on their simulated Martian soil. Also, we have submitted our project to the IAC, and we have been selected to compete in the Student Competition Category. That opportunity allows us to present our work at the 2021 IAC that will be held in Dubaï.

4.3 Conclusion

The initial problem of localizing and mapping in a GPS-free environment on which my internship was focused has been solved with respect to the mission requirements. Regarding the performance of the Lidar vs the Depth Camera, we have decided to choose the D435i + T265 as final configuration for the drone. The interface has been made with the Path-Planing and Task-Planing algorithms which were able to generate a path that successfully avoided all the registered obstacles. However, because this project is completely new, I focused my work on securing the whole blockchain of processes from the sensors' data to the final position and 2D map. That is why I identified several interesting parts that I have implemented in simple ways and that can be improved. Among them, I can cite the extraction of 3D dem map (the process of the Figure 2.3), where a huge advantage could be obtained from more advanced analyzes on shape recognition and identification of aberrant data. The denoising program can be as well upgraded to a version more adapted to the D435i sensor. The chain of filters to improve the quality and reduce the complexity of the sensor data can be enlarged. The project CoRoDro 2021 might continue next year with another team of students from Isae-Supaero to participate in the Igluna 2022 and with the objective of always present a better version of the robotic systems. So in conclusion I would say that I have paved the way for future students to dive into diverse and very interesting topics in the field and vision computer. As I am continuing the project until its final delivery in July, I am going to push the robotic systems as far as possible in complexity and capabilities. For now, the objective is to interface the work of every developing team and to perform the final tests to secure a working version of our robotic systems. If time is left, I will begin to improve the interesting parts mentioned just above.

BIBLIOGRAPHY

[1] *Leo rover*.
https://www.leorover.tech/.
Accessed: 2020-10-21.

[2] *Occupancy grid format*.
http://docs.ros.org/en/diamondback/api/nav_msgs/html/msg/OccupancyGrid.html.
Accessed: 2021-01-30.

[3] *Radius outlier removal filter*.
https://pointclouds.org/documentation/tutorials/remove_outliers.html#remove-outliers.
Accessed: 2021-01-30.

[4] *Statistical outlier removal filter*.
https://pointclouds.org/documentation/tutorials/statistical_outlier.html#statistical-outlier-removal.
Accessed: 2021-01-30.

[5] *Voxel definition*.
https://en.wikipedia.org/wiki/Voxel.
Accessed: 2021-01-30.

[6] *Voxel grid filter*.
https://pcl.readthedocs.io/projects/tutorials/en/latest/voxel_grid.html.
Accessed: 2021-01-30.

[7] AMAZON PRIME, *Amazon prime air*.

[8] P. . J. BESL AND N. D. MCKAY, *A method for registration of 3-dshapes*, IEEE Trans Pattern Anal. Mach. Intell., vol. 14, no. 2, pp.239,Äì256„ (1992).

[9] B. V. BING JIAN, *Robust point set registration using gaussian mixture models,*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2011).

[10] S. CHIEN, R. KNIGHT, R. STECHERT, A.AND SHERWOOD, AND G. RABIDEAU, *Integrated planning and execution for autonomous spacecraft*, IEEE Aerospace Conference, (1999).

[11] J. J. DEYST, *Estimation and control of stochastic processes*, M.I.T.Dept. of Aeronautics and Astronautics, Cambridge, Massachusetts, (June 1972).

[12] ESA ADVANCED CONCEPTS TEAM, *Ai in space workshop*, 2013.

[13] C. FROST, *Challenges and opportunities for autonomous systems in space*, 2011.

[14] HORIZON-2020, *European robotic goal-oriented autonomous controller (ergo)*.

[15] C. R. HORN, ROGER A.; JOHNSON, *Matrix analysis (2nd ed.) isbn 978-0-521-54823-6.*, Cambridge University Press, (2013).

[16] ISECG, *The global exploration roadmap*, The International Space Exploration Coordination Group, (2019).

[17] T. KAKU AND ET AL, *Detection of intact lava tubes at marius hills on the moon by selene (kaguya) lunar radar sounder*, Geophysical Research Letter, (2017).

[18] L. J.-J. LI YI-BO, *Harris corner detection algorithm based on improved contourlet transform*, Procedia Engineering 15:2239-2243, (December 2011).

[19] P. MARTIN, S. KWONG, N. T. SMITH, Y. YAMASHIKI, O. PAYTON, F. RUSSELL-PAVIER, J. FARDOULIS, D. RICHARDS, AND T. SCOTT, *3d unmanned aerial vehicle radiation mapping for assessing contaminant distribution and mobility*, Int. J. Appl. Earth Obs. Geoinformation, 52 (2016), pp. 12–19.

[20] P. S. MAYBECK, *The kalman filter,Äîan introduction for potential users*, TM-72-3, Air ForceFlight Dynamics Laboratory, Wright-Patterson AFB, Ohio, (June 1972).

[21] B. P.-A. A. S. MICHAEL W. BERRY, DANI MEZHER, *Parallel algorithms for the singular value decomposition*, Handbook on Parallel Computing and Statistics, (2005).

[22] NASA, *Nasa robot plunges into volcano to explore fissure*.

[23] P. I. RADOGLOU-GRAMMATIKIS, P. SARIGIANNIDIS, T. LAGKAS, AND I. MOSCHOLIOS, *A compilation of uav applications for precision agriculture*, Comput. Networks, 172 (2020), p. 107148.

[24] ROS, *Amcl algorithm*.

[25] ——, *Octomap server algorithm*.

[26] STR$\sqrt{\partial m.K.J.,Introductiontostochasticcontroltheory,AcademicPress,NewYork,}$(1970).

[27] M.-J. W. W. J. XIAN-FENG HAN, JESSE SHENG JIN, *A review of algorithms for filtering the 3d point cloud*, SIGNAL PROCESSING IMAGE COMMUNICATION 57, (2017).

[28] C. YINKA-BANJO AND O. AJAYI, *Sky-farmers: Applications of unmanned aerial vehicles (uav) in agriculture*, 2019.