PoLIDAR: a six wheel-legged mapping robot

Esteban Gissinger, Matthias Vermot-Desroches

***Abstract*—In order to remove the human need in mapping, we realized an autonomous mapping robot after making a pseudo-prototype. We used a LIDAR sensor to scan the area and get a series of point to, later, make the robot autonomous by choosing where he should go according to the series to scan the area. We also used ROS to generate and save a map with the LIDAR's series of point.**

***Index*—LIDAR, autonomous, mapping, ROS, six-wheeled**

I. Introduction

TO map an area is the process of creating a map using coordinates of the relief in points by scanning an area.

Mapping an area can be a complicate task for a human. Indeed, the relief can have rocks, mud, hills, humps, cliffs (mostly for coasts) or just for a post-disaster area 's exploration [1]. Moreover, mapping is a complicate task for a human in general [2].

To ease this task, many mapping robots were made depending on the type of area: on the earth [3], in the air [4], under water [5], in Mars [6].

Our goal was to make a mapping robot that does not require human presence to work properly (with a LIDAR sensor). We chose to make a mapping robot that works on the earth. There are two types of earth mapping robot, those with tracks and those with wheels. We chose to use six wheel-legs for their efficiency [7].

We wanted the robot to be autonomous in its movement, so we used the LIDAR to allow the robot to map and locate itself without human assistance [8].

In this article, we will first speak about the robot's structure, from the frame to the wheel ensemble, from its sketches to the frame used, passing by the wooden model made as a prototype. In the second part, we will speak about how we manage to scan the area thanks to the LIDAR and ROS. Finally, we will conclude on what we have done.

II. The Robot's structure

A. The frame

The frame is an important piece of our robot, it will support the microprocessors and the wheel-legs. After looking at different existing robots similar to ours [9] [10], we first decided to go with a T-shaped frame: the top would support everything, and the tip would be the fixing point of all the wheel-legs.

However, after counting everything the top would have to support, we switched for a sort of cubic-shaped frame as it is more resistant than a T-shaped one. The wheel ensemble would be fixated on a side of the cube.

B. The wheel ensemble

The wheel ensemble is the centerpiece of our robot, it must be able to support relief-changes as well as rotating the wheels when needed. We decided to make an ensemble consisting of a wheel, a 5V servomotor, a 12V motor, a suspension and supports pieces. The servomotor will allow the ensemble to rotate, the suspension will make the ensemble able to sustain the relief-changes and the supports pieces will make the link between it and the frame. You can see on figure 1 a sketch of the ensemble.

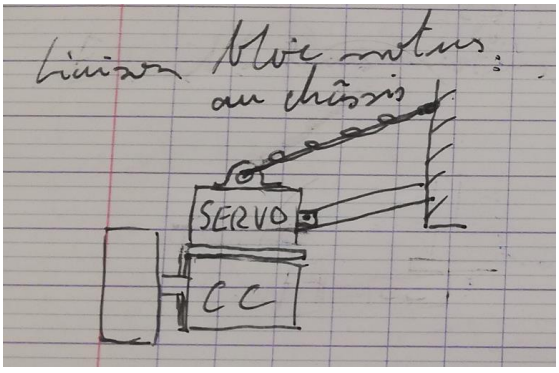

Fig. 1. The wheel ensemble

C. The wooden model

In order to have a prototype of our robot, we decided to make a wooden model, at least one for the frame and for the wheel ensemble. Firstly, we realized CAD models of the supporting pieces (named joints) and plate for the frame. Then, we used the cutting machine to make them with wood. This step created a lot of problem for us, as we didn't know how wide we wanted our model to be (we had a large choice of width, between 3mm and 8mm for example) and ended up making one of 3mm and one of 5mm. The other problem was with the joints, at first, we wanted them to be the same as the supporting pieces of the wheel ensemble, but in the end, we decided to make them different. That slowed the process a lot since we also made the first batch using the wrong width.

E. Gissinger is in Université Côte d'Azur, Polytech Nice Sophia, in the robotic department, 930 route des Colles, 06410 Biot, France (e-mail: esteban.gissinger@etu.univ-cotedazur.fr).

M. Vermot-Desroches is in Université Côte d'Azur, Polytech Nice Sophia, in the robotic department, 930 route des Colles, 60410 Biot, France (e-mail: matthias.vermot-desroches@etu.univ-cotedazur.fr).

In the end, we managed to make the frame and the wheel ensemble (we used two small aluminum bars with the wood). You can see a picture of the wheel ensemble on figure 2.

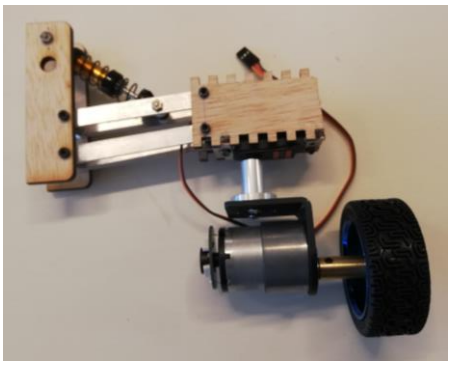

Fig. 2. The first wheel ensemble

D. *The final frame*

Unfortunately, after finishing the first wheel ensemble, we realized that we would lack time if we repeated the creation process. So, we ordered a premade frame to speed our process and start the programming part earlier. Unfortunately, the frame we ordered used motors instead of servomotors, so all our previous studies were put aside for the time being, even if we hoped to use them later when we will have the time to make our own frame.

After receiving it, we assembled every piece and had to fix every sensor and microprocessor on it. We had to fit two HC-SR02 ultrasound sensors, one 2D LIDAR that we hoped to replace later with a 3D one, the motors drivers, an Arduino Uno card, and an Nvidia Jetson Nano card on it. We fixated the ultrasound sensors using support pieces made for them that we had bought. For the rest of the sensors and microprocessors, we used screws, nuts and bolts to fix them. When we needed to place the Nvidia card and the LIDAR, we lacked space. So, we cut a new floor made of PMMA before fixing everything on the frame.

You can see a picture of the fully assembled frame on figure 3.

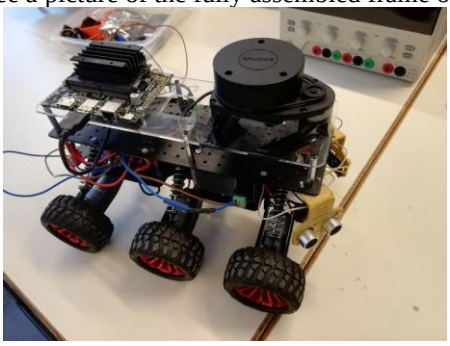

Fig 3. The fully assembled frame

E. *The motor program*

We wanted the robot to turn when an obstacle is encountered, so we needed to program the motors. In order to do this, we used ultrasound sensor to measure the length between the robot and an obstacle to then increase or decrease the motors' speed accordingly. For example, if an obstacle was detected on the right, the right sided motors' speed would increase and the left sided motors' speed would decrease in order to make the robot rotate to the left.

III. THE MAPPING

A. *The LIDAR*

To be able to effectively scan the surrounding environment, we decided to add a LIDAR based module on top of the robot. We chose to use a LIDAR-Lite v3HP as our LIDAR sensor because it is fast (>1000 Hz) and accurate. We also decided to attach the LIDAR sensor and redirect the LASER with a mirror to achieve 3D vision instead of moving the LIDAR to get a more robust and faster robot [11]. We achieve this with a structure that enables the mirror to rotate on the vertical and horizontal axes. To achieve this, we use four pieces that we designed using Fusion 360 and then printed using a Prusa: a rotor, a ring, a synchronizer, and the mirror's support. The ring slides around the rotor, while the synchronizer lies on top of the ring and rotate with the rotor while being also attached to the mirror's support as seen on figure 4. With this basic setup, we can control the mirror's pitch angle by sliding the ring and rotating the rotor. We use two 9g servomotors on each side of the ring to slide the ring because it is cheap, small, precise, and easy to control. Moreover, the two servomotors move together to avoid any friction with the rotor. On the other side, we use a nema 17 stepper motor and its A4988 driver to rotate the rotor because of its high precision. Moreover, they are coupled with a belt instead of gears in an attempt to reduce the propagation of vibration and thus achieve greater accuracy. Finally, we cover one face of the support's mirror with a "mirror effect" film and we attached everything to a board that was also printed as well as a ball bearing to ease the rotation.

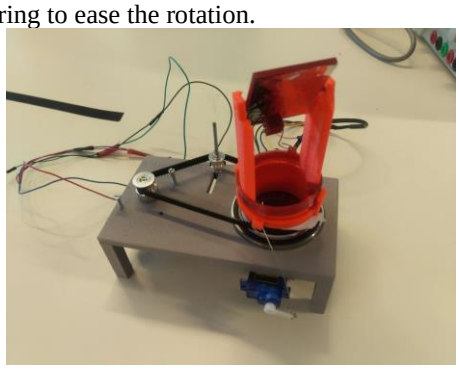

Fig 4. The stacked ring and synchronizer on the rotor.

Everything is controlled using an Arduino UNO on which runs a straightforward program. First, we rotate the stepper motor by a step, then we get the distance using the LIDAR sensor. Once received, we convert those coordinates into cartesian coordinates using basic trigonometry and send them to the computer. We repeat this process until the rotor makes one turn and then we slightly increase the servo-motor pitch before repeating the same process.

Unfortunately, this first prototype suffered greatly from various issue. First of all, the rotor something gets out of its place because of the torque applied by the belt on it. An easy

solution is to replace the current bearing with a bigger one so that we can increase the diameter of the rotor and fit the lidar in it, which then allow us to put the bearing in the middle of the rotor, reducing the torque. We can also add another bearing at the bottom of the rotor for extra-stability. There is also an issue with the rings that are to tights and prevent the rotor from turning properly because the belt is sliding on it instead of making it turn. Once again, an easy solution is possible by adding gear teeth on the rotor and reprinting the rings with a slightly bigger diameter. However, the biggest issue is the connection between the servomotors and the ring which doesn't work as intended. Solving this issue required to redesign the model. This time, we used another stepper motor to replace the servo motors, in combination with a screw on which two plate are mounted. Moreover, between those plates comes the synchronizer. The system works as following: the stepper makes the screw turn, which make the plates go higher, which move the synchronizer up, which make the mirror tilt as we can see in figure 5.

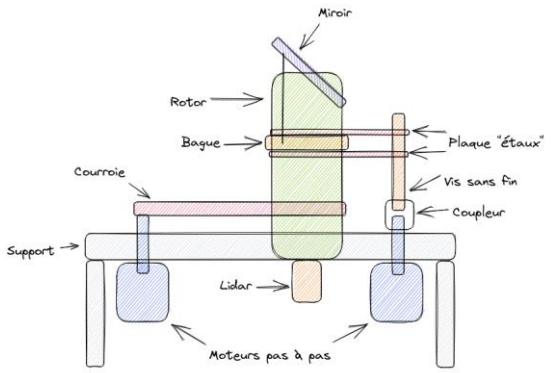

Fig 5. The redesigned LIDAR module

B. ROS

In order to generate and save a map when our robot is in function, we decided to install the ROS operating system on our Nvidia card. It is an operating system that allowed us to separate scripts or programs in nodes, optimizing the memory, the data processing, and the microprocessors management [12].

After finishing the download (it took a really long time thanks to a lot of errors or modules unable to be downloaded), we followed the beginners' tutorials [13] to understand how the OS worked. Then, we searched ROS modules or libraries that could map an area and save it. After finding a tutorial on how to create a map using ROS [14] and a 2D LIDAR, we began creating scripts. We decided to use a 2D LIDAR at first to ease our learning and programming process and then update the scripts and programs to have a 3D LIDAR compatibility. However, following this tutorial made a lot of errors occurred and we had to spend a lot of time searching for a solution on each of them, while solving some errors created new ones. After solving all the issues, we managed to create a first 2D map. However, this one was just an ensemble of points, it was not very representative of the area, and it kept being update without saving the previous position. You can see a picture of this map in figure 6.

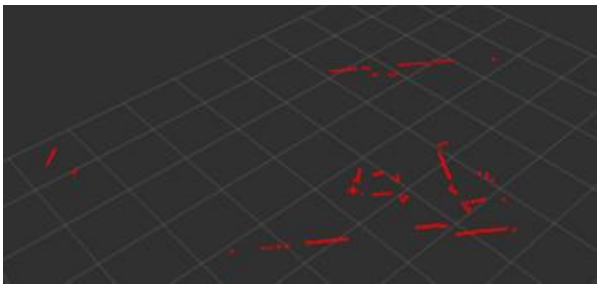

Fig 6. The first generated map

We decided to improve the generated map with a ROS library named hector_slam. This library used a bit of a SLAM algorithm, Simultaneous Localization And Mapping, an algorithm that allowed the robot to map an area while localizing itself in the process.

After adding this module, the map looked much more representative of the scanned area. The "walls" were represented by black points and if the robot moved, the map updated itself while keeping the previous points. If the robot height changed, new black points would appear, meaning that in theory, if we used a 3D LIDAR, we could create a 3D map.

You can see an example of a map generate with SLAM in figure 7.

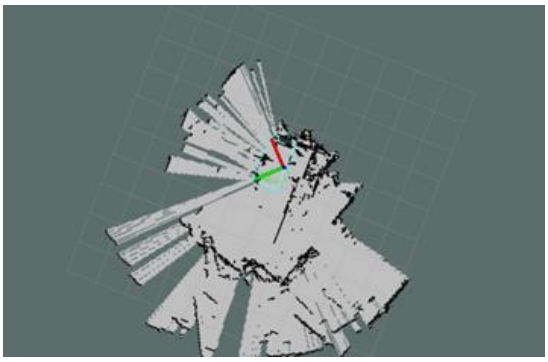

Fig 7. An example of a map generated with SLAM

Then, we wrote a script and a service that allowed the mapping process to be activate on the robot's launch. After that, we created another script, service and a timer that would make the robot save the generated map in a png file every ten minutes. Unfortunately, we didn't have the time to update our scripts to have a 3D LIDAR compatibility.

IV. CONCLUSION

In conclusion, we realized a frame for our robot that has every sensor and microprocessor we needed on it. We also made a program allowing the robot to turn when an obstacle is detected by ultrasound sensors. We made the robot able to map an area and save the generated map thanks to ROS and a bit of SLAM.

Also, the LIDAR module offers 3D scan capability with automatic speed adjustment to match the desired precision in

most circumstances. The latest model gives significant improvement, although fine tuning is still needed.

REFERENCES

[1] https://journals.openedition.org/vertigo/15743
[2] https://www.persee.fr/doc/quate_0004-5500_1981_num_18_1_1397
[3] https://www.irobot.fr/fr_FR/roomba.html
[4] https://escadrone.com/gammes/drones-aeriens/cartographie/
[5] https://veillecarto2-0.fr/2022/01/14/des-robots-electriques-cartographient-le-fond-marin-la-derniere-frontiere-de-la-terre/
[6] https://fr.wikipedia.org/wiki/Mars_Exploration_Rover
[7] https://spectrum.ieee.org/wheels-are-better-than-feet-for-legged-robots
[8] https://pastel.archives-ouvertes.fr/pastel-00935600/file/2012ENMP0103.pdf
[9] https://www.instructables.com/YAAR-Yet-Another-Autonomous-Robot/
[10] https://community.robotshop.com/robots/show/parovoz-6wd-all-terrain-robot-platform
[11] https://charleslabs.fr/en/project-3D+Lidar+Scanner+MK2
[12] https://www.generationrobots.com/blog/fr/ros-robot-operating-system-3/#:~:text=Qu'est%20ce%20que%20ROS,pour%20la%20robotique%20de%20service.
[13] http://wiki.ros.org/fr/ROS/Tutorials
[14] https://automaticaddison.com/how-to-build-an-indoor-map-using-ros-and-lidar-based-slam/

E. Gissinger has done an intensive two-year study course preparing for the competitive entrance to the French “Grandes écoles” in Camille Pissarro Highschool.
M. Vermot-Desroches has a two-year undergraduate degree in mechanical engineering in Université Claude Bernard Lyon1.