SANCHEZ
Joris
Groupe : C

10/04/2019 to 19/06/2019

INTERNSHIP REPORT

Multi-robot simulations for evaluating mission performance

Tutors :
CHRISTOPHE François
FAGERHOLM Fabian
MIKKOMEN Tommi

UNIVERSITY OF HELSINKI

Acknowledgment :

Thanks Matteo Valenza director of the UIT (University Institut of Technology) of Montpellier and Pierre Gillet the director of the department Electrical Engineering and Computing Science (EECS) for allowing me to do this internship in your UIT.

Thanks the international relationship office with Marie-Claude Artaud-Gillet and Eva Legrais to propose different internship abroad and thanks for this opportunity.

Thanks the University of Helsinki to welcome us in your offices and to propose this intenship with the University of Montpellier.

Thanks Christophe François, Fabian Fagerholm and Tommi Mikkomen to welcome us and guide us during this internship and thanks to trust us.

Introduction :

For the Electrical Engineering and Computing Engineering (EECE) diploma, we have to do an internship in a company or a research laboratory between 10 and 12 week for validate our diploma, we don’t have to do this intership abroad but this is an opportunity to use the UIT relationship with the international relationship office to do an internship abroad and this is a plus on the resume.

My internship is in Finland, more precisely in the University of Helsinki in Helsinki, in the department of computer science.

The subject of my internship is Multi-robot simulations for evaluating mission performance. I’m supervised by Tommy Mikkomen with Christophe François and Fabian Fagerholm.

Through this report, I will introduce you my 10 week as an intern with the presentation of the University of Helsinki, the places where we work, our task, our work and our problems.

Content :

Acknowledgement …… 2

Introduction …….. 3

1. Presentation of the University of Helsinki …… 6

a. Presentation of the department of computing science …….. 6

b. Presentation of the work place ……. 7

2. Internship work …… 10

a. Gazebo …….. 10

1 - What is Gazebo ? ……. 10

2 - The insterface ……. 11

3 - The model editor User Interface …….. 12

4 – Modelization of Pulurobot …….. 14

b. ROS …….. 15

1 - What is ROS ? …….. 15

2 – The installation …… 15

3 – ROS learning …… 15

4 – Re-modelization of the pulurobot ……. 16

5 – Sensors …….. 17

a. The LIDAR ……. 17

b. The sonar …… 18

c. The 3D camera / depth camera …….. 18

6 – The visualization on RVIZ …… 20

a. The LIDAR in Rviz ……. 21

b. The sonar in Rviz …… 21

c. The 3D camera in Rviz …… 22

7 – The movement for the pulurobot …….. 23

8 – Mapping …… 24

9 – Collision avoidance …….. 26

a. Collision detection representation …….. 26

b. Navigation from A to B …….. 27

c. 3D navigation ……. 27

10 – Area covering ……. 28

11 – Test the difference between the real pulurobot and the simulation …… 29

12 – Create my own robot ……. 30

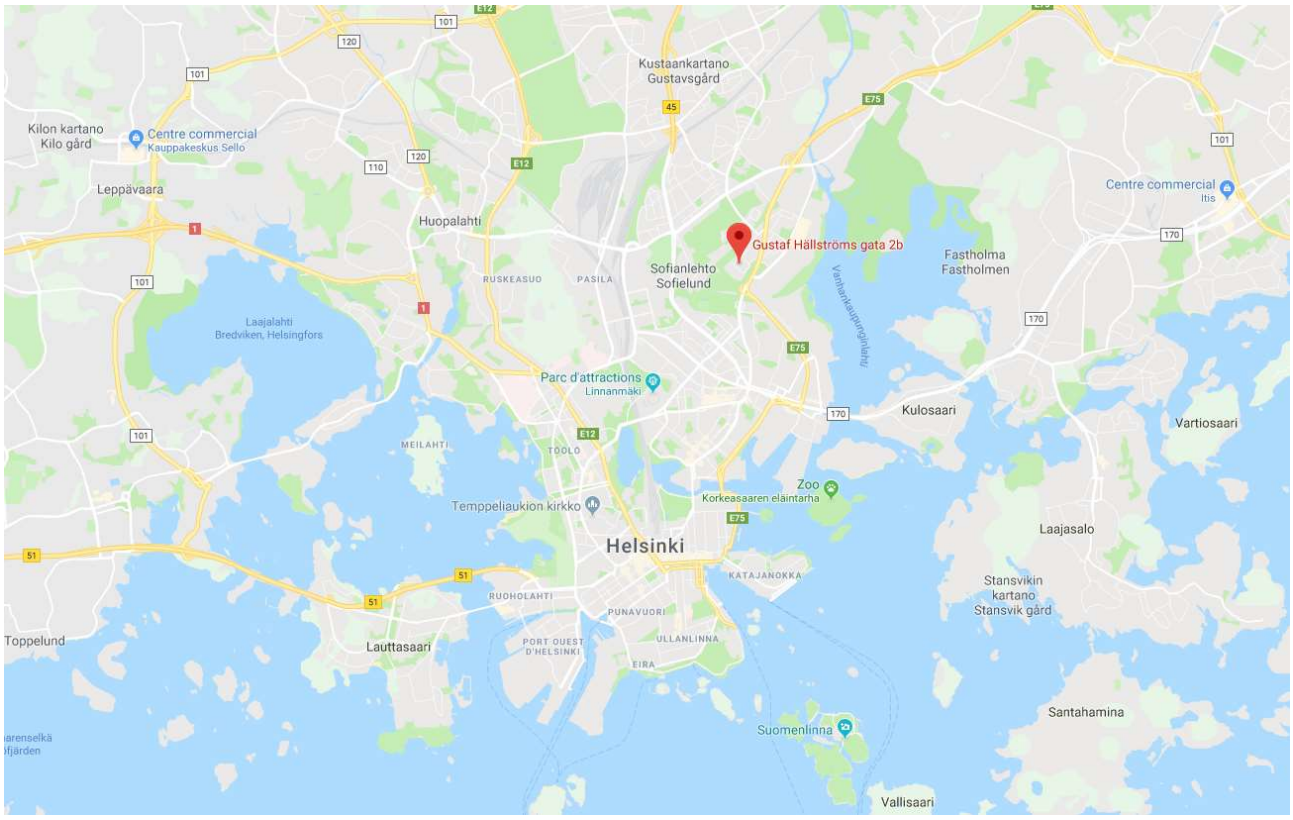
13 – The architecture of my own robot …….. 31

Conclusion …… 32

Documentation ……. 33

1. Presentation of the University of Helsinki :

The internship is proposed by the University of Helsinki, the oldest university and the main university of the Finland, with him approximaly 20 000 students and 4 500 teacher and researcher she is the biggest university of Finland. She have 11 faculty within her. For the internship we are in the department of computing science, the different department isn't side by side, they are scattered around Helsinki like the different department of litteratrue is more on the center of Helsinki and all science department is more on the north of Helsinki as shown in the picture below :



Map with a marker on the place where are the science department of the University of Helsinki.

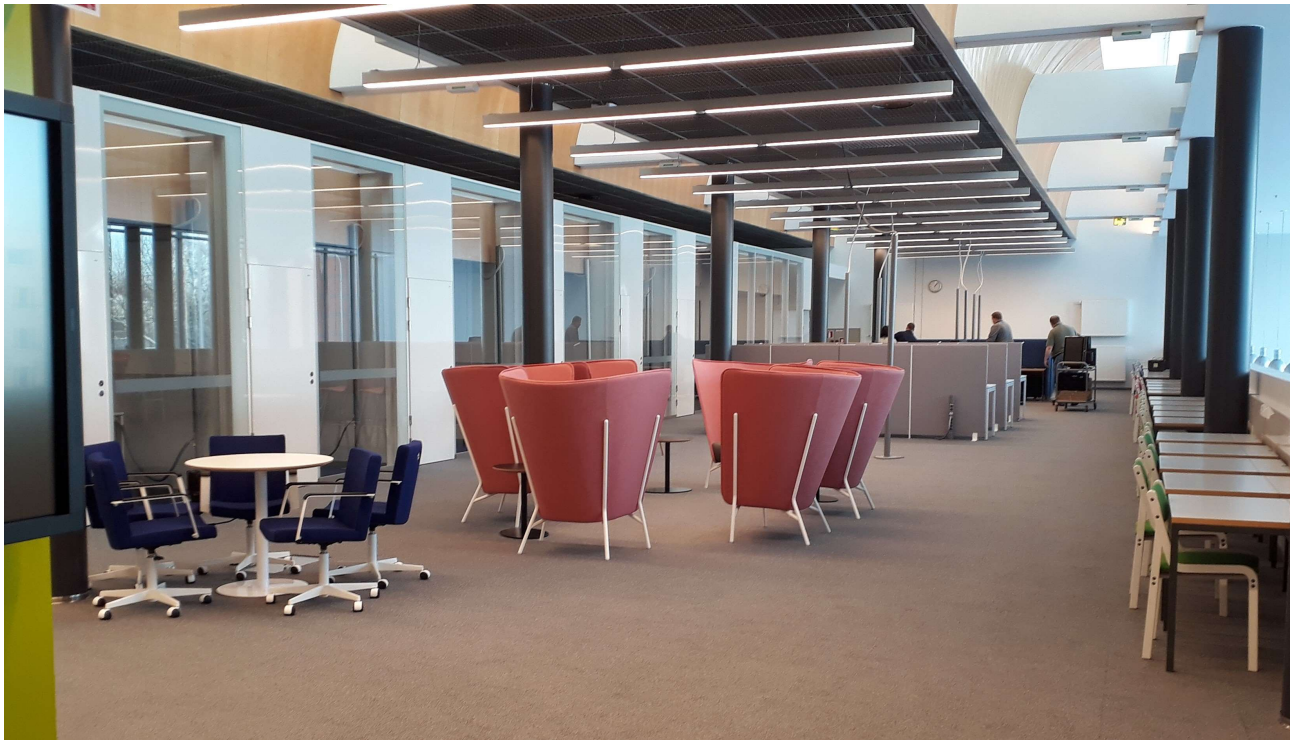
a. Presentation of the department of computing science :

The place is composed with different building with their own name. Their name is Dynamicum, Exactum, Physicum and Chemicum. This different name corresponds to their subject teach, for example the building Exactum our building for information, this building teach the science exact like computing, electronic and mathematic, and this is valid for all building.

Exactum building.

b. Presentation of the work place :

In the Exactum building, there are different room for the students like different auditorium or just different class room but there are the researcher office too, the tutors office and others researcher.
For us, we have two room available for work the first room is the « ubikampus » the place where spend the most of our time. In this room we have different desk with screen, keyboard and mouth for work, all desk is available for everyone, it’s the first-come first served, this is a room staff only with the second room.

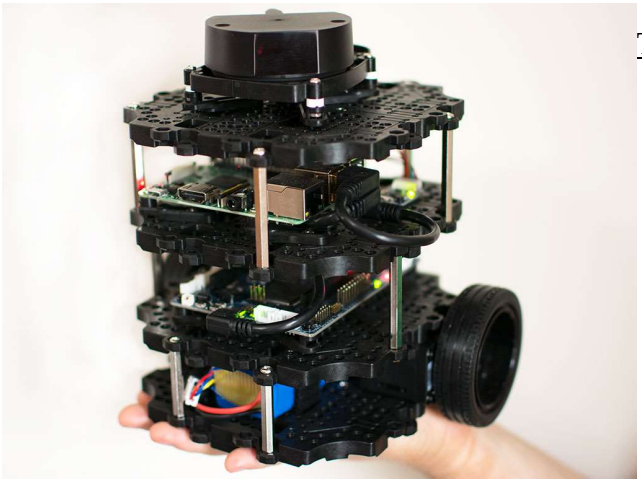

Ubikampus in construction almost finished.

The second room is the software factory, in this there are computer for work on different programs during our intern we don’t need to use these computer because we don’t work only on programs and they don’t the software use by us.

Software factory.

In this room there are the different robot that we work on and we go in this room for test these robot and take some measure. The three robot are the MBot, the Turltebot and the Pulurobot.

Mbot.



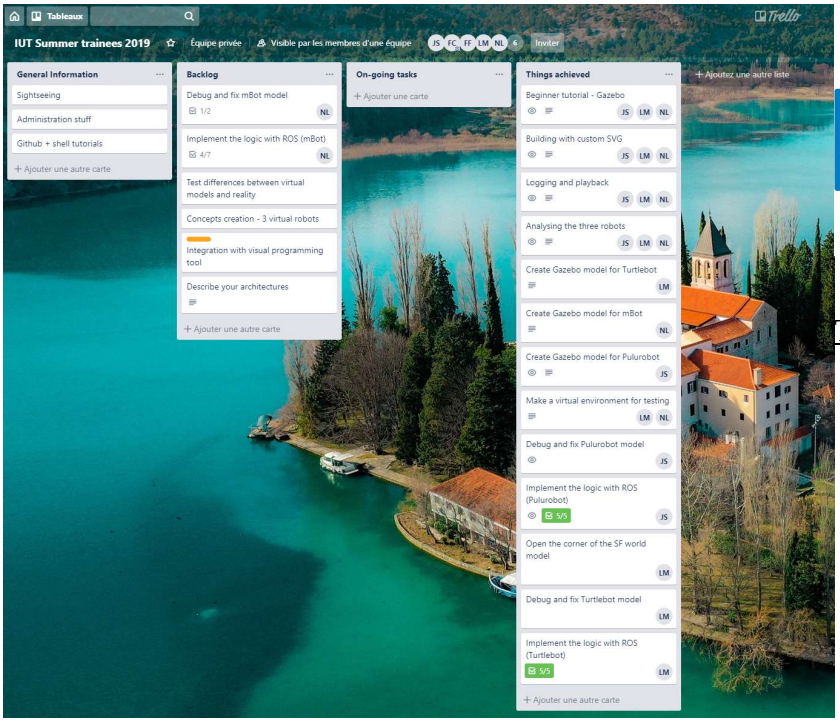
Turtlebot.

Pulurobot.

* These robots are available for us only in the software factory.

2. Internship work :

For know where we were in our work and know what we have to do, we work with the platform trello.



Trello’s interface.

Trello is very easy to use and allow tutors to see what we do and see if we advance, it’s a system with different post-it with a task and drag it into the corresponding column. You can see on the trello’s interface 4 column (General information, backlog, on-going tasks and things achieved), at the beginning all the task was in the backlog, when we begin to work on a task we drag her in the on-going tasks column and when we finish this task we drag her in the things achieved column.
The General information column is for different information as the administration stuff or for different tutorials for help us in our work.

a. Gazebo :

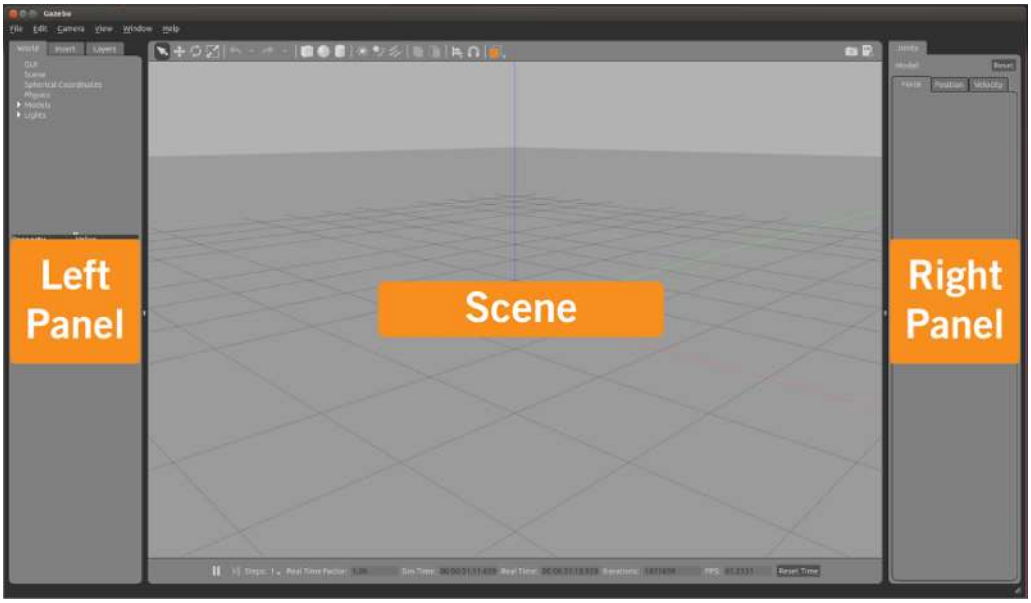
1 - What is Gazebo ?

Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. While similar to game engines, Gazebo offers physics simulation at a much higher degree of fidelity, a suite of sensors, and interfaces for both users and programs. Typical uses of Gazebo include : testing robotics algorithms,designing robots,performing regression testing with realistic scenarios. Gazebo include : multiple physics engines, a rich library of robot models and environments, a wide variety of sensors, convenient programmatic and graphical interfaces.

Definition by Gazebo.

Our first work is to learn how to use Gazebo, firstly we have to do the installation on our computer and we see a problem, the problem is the Gazebo's compilator doesn't work on windows and our computer is on windows, we can't use the university's computers because our login doesn't work and the computer aren't powerful enough to support Gazebo, so to remedy that we use a virtual machine on the computer and we install linux ubuntu version 18.10 and the problem is solved. We install Gazebo and we begin the different tutorials for learn how it work.

2 - The insterface :



Gazebo's interface.

Gazebo have in total 3 windows :

- **The Scene :** Is the main window, this is where the simulated objects are animated and you interact with the environment.

- **The left panel :** The left panel appears by default when you launch Gazebo. There are three tabs in the panel.

- WORLD : The World tab displays the models that are currently in the scene, and allows you to view and modify model parameters, like their pose.

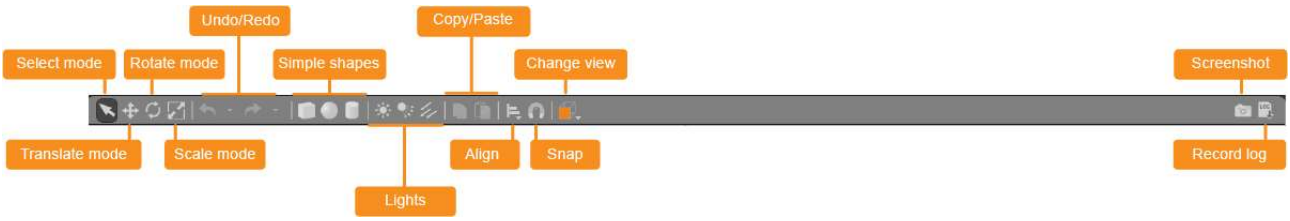
- INSERT : The Insert tab is where you add new objects (models) to the simulation.

- LAYERS : The Layers tab organizes and displays the different visualization groups that are available in the simulation, if any.

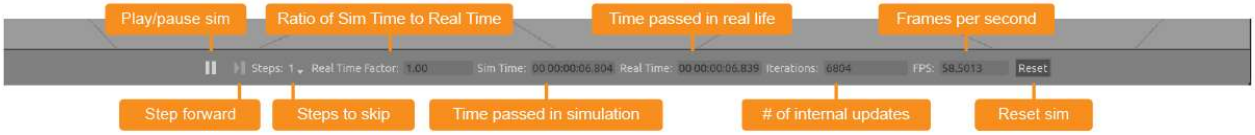
- **The right panel :** The right panel is hidden by default. The right panel can be used to interact with the mobile parts of a selected model (the joints).

With this 3 windows added 2 toolbars :

The upper toolbar :

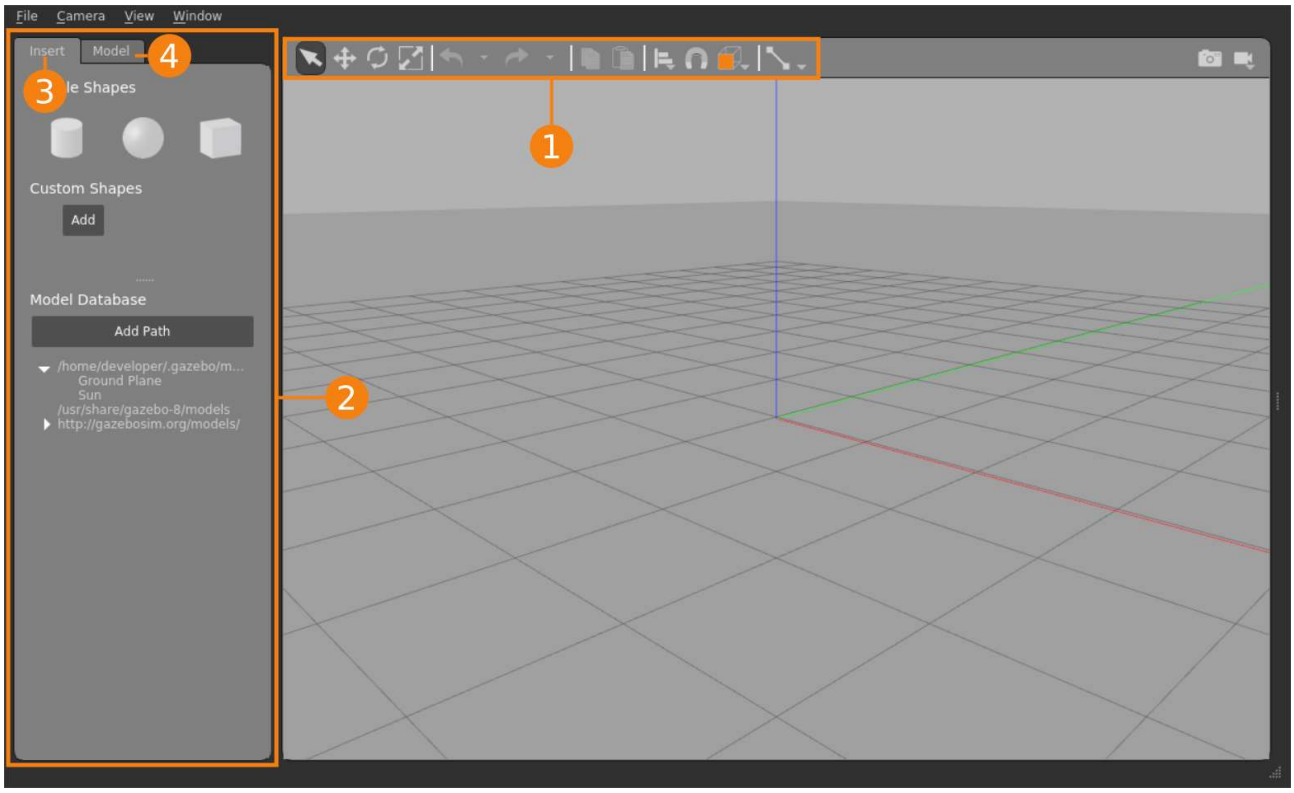


The bottom toolbar :



With this tools and windows we can do any simulation in Gazebo with a condition, have a robot model. For that Gazebo have a model editor to modelize everything.

3 - The model editor User Interface :



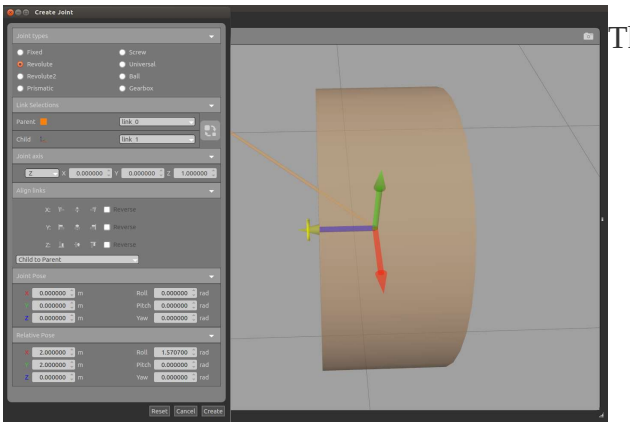
The model editor user interface.

1. Toolbar : Contains tools for editting the model
2. Palette : Also known as Left Panel. Has two tabs for editing the model.
3. Insert tab : Tools for adding links and nested models
4. Model tab : Allows editing model properties and contents

Now, you can modelize every robot in the world.

To validate our learning of Gazebo, we modelize a simple robot with a cube as chassis, two cylinder for the wheels and a sphere for the caster wheel.

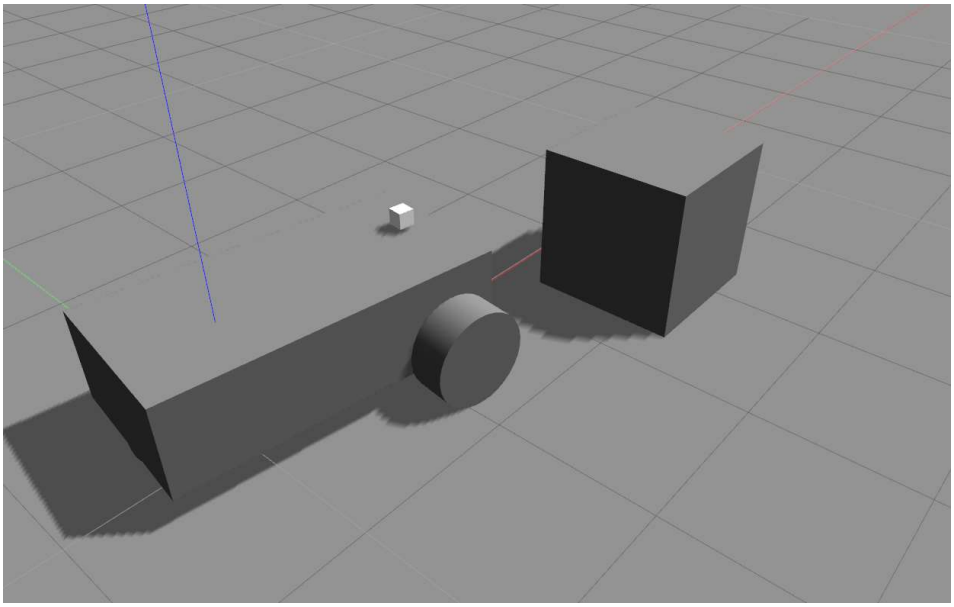
With this different simple shapes we need to joint all of them together, for that Gazebo have a tool with all joint we need (fixed, revolute, revolute2, prismatic, …) for the wheels we use a revolute joint for fix the position but not the rotation and we choose the good axis for the rotation.



The axis of rotation is indicated by the yellow arrow.

And for the caster wheel we choose the ball joint because the sphere rotate on all axis and the position don't change.

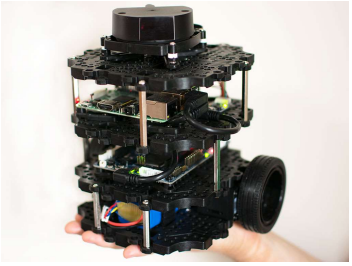
Meanwhile we added a sensor (the white cube) and it gives the result below :



Now that we know the software better, our tutor have given to us a robot to modelize with these three robot on Gazebo :

Mbot

Turtlebot

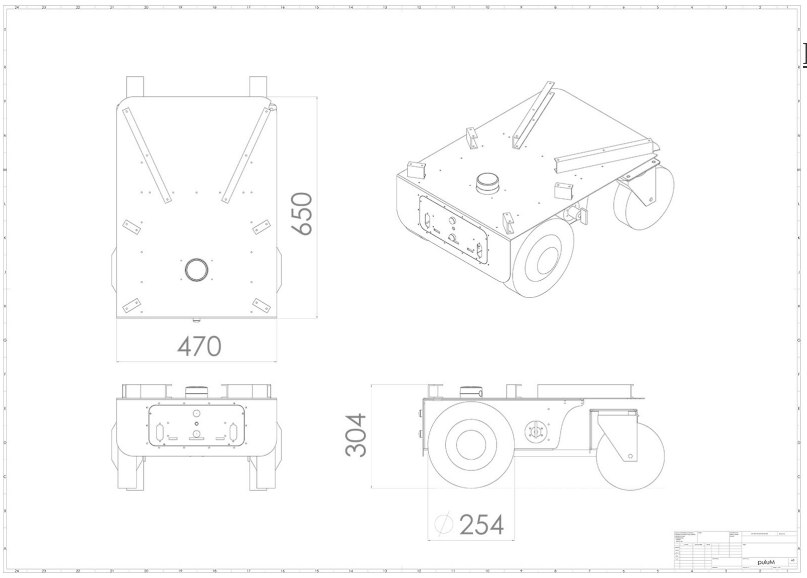


Pulurobot

And the robot that I was given is the Pulurobot, and my is modelize this robot with the real proportion in Gazebo.

4 – Modelization of Pulurobot :

Before begin the modelization We have to do some measure on the robot for know the dimension of the robot and where are the different sensors. For the Pulurobot we have help from internet because we found the plan of the chassis but with that we don't have the all dimension of the robot, so we take some measure for more precision.

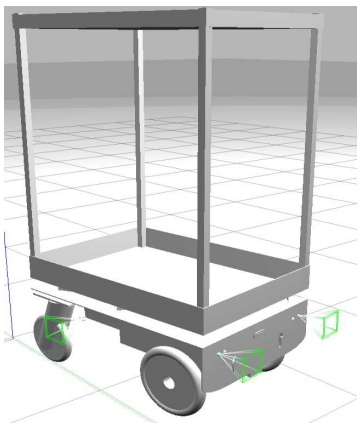
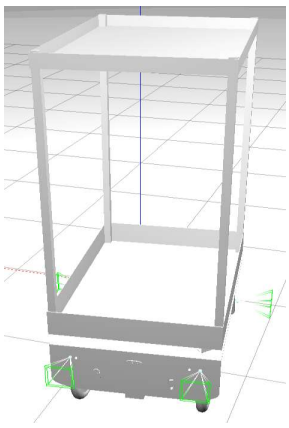
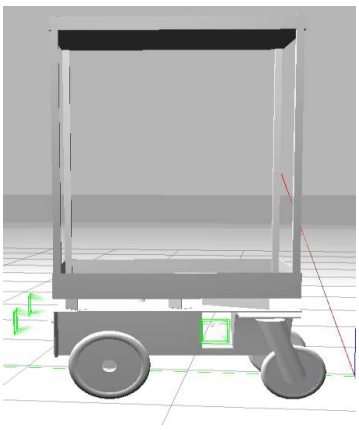
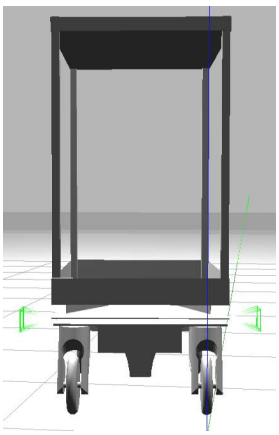


Plan of the Pulu's chassis.

For more precision a decided to do the different pieces on a 3D software and importe them in gazebo after the modelization.

The 3D is Blender a free software.

After modeling all the parts of the Pulurobot, I assembled them on Gazebo and the result is :

b. ROS :

1 – What is ROS ?

ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers.

Definition by ROS.

2 – The installation :

ROS have different version and depending the version of ubuntu different version of ROS are available or not. For my version of ubuntu, I can download the last version of ROS who named ROS melodic.

3 – ROS learning :

As Gazebo, ROS have a lot of tutorials.

For the beginning, we learn how to create a workspace for ROS, how to navigate between the different folder, how to use the different packages and how create your own packages.

We learn how to use the different command of ROS and functionnality. For example, the nodes in ROS for communicate with your packages, or rosrun for launch a compatible program with ros, the different topics for send or read a message send by your package.

The node is /teleop_turtle, the program is /turtlesim and the topic is /turtle1/command_velocity. The node communic automaticaly with the program because he is a program and he use the topic /turtle1/command_velocity for the communication.

That’s all we need for modelize and command our robot.

4 – Re-modelization of the pulurobot :

Why re-modelization ? Because the model editor on Gazebo use the sdf format when you save your project and ROS use the urdf format for work, as we don't have access to the Gazebo model editor we have to do this by the hand.

How work the urdf modelisation ?

Part of the pulurobot's urdf file.

```
699     <link name="link_right_link_wheel">
700     <inertial>
701       <mass value="1"/>
702       <origin rpy="0 0 0" xyz="0 0 0"/>
703     </inertial>
704     <collision name="link_right_link_wheel_collision">
705       <origin rpy="0 0 0" xyz="0 0 0"/>
706       <geometry>
707         <cylinder length="0.16" radius="0.02"/>
708       </geometry>
709     </collision>
710     <visual name="link_right_link_wheel_visual">
711       <origin rpy="0 0 0" xyz="0 0 0"/>
712       <geometry>
713         <cylinder length="0.13" radius="0.02"/>
714       </geometry>
715     </visual>
716   </link>
717
718   <joint name="joint_right_link_wheel" type="continuous">
719     <origin rpy="0 0 0" xyz="-0.225 -0.17 -0.085"/>
720     <child link="link_right_link_wheel"/>
721     <parent link="link_chassis"/>
722     <axis rpy="0 0 0" xyz="0 0 1"/>
723     <joint_properties damping="0.0" friction="0.0"/>
724   </joint>
```

Urdf writing present itself with links and joints like Gazebo.

A link have always an inertie, a collision and a visual. In the link's inertie we have the mass and the origin of the gravity point, in the link's collision and the visual we have the geometry of them and the origin because in a link we can have several visual and collision and with that we can assemble different pieces without joint.

A joint have his type of joint, is origin between the parent link and the child link it permit us to place the different pieces as we want, with that we have the axis of rotation and translation (0 is blocked and 1 is not blocked) and we have the joint properties with the damping and the friction.

Different type of joint in urdf :

Revolute : a hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits.

Continuous : a continuous hinge joint that rotates around the axis and has no upper and lower limits.

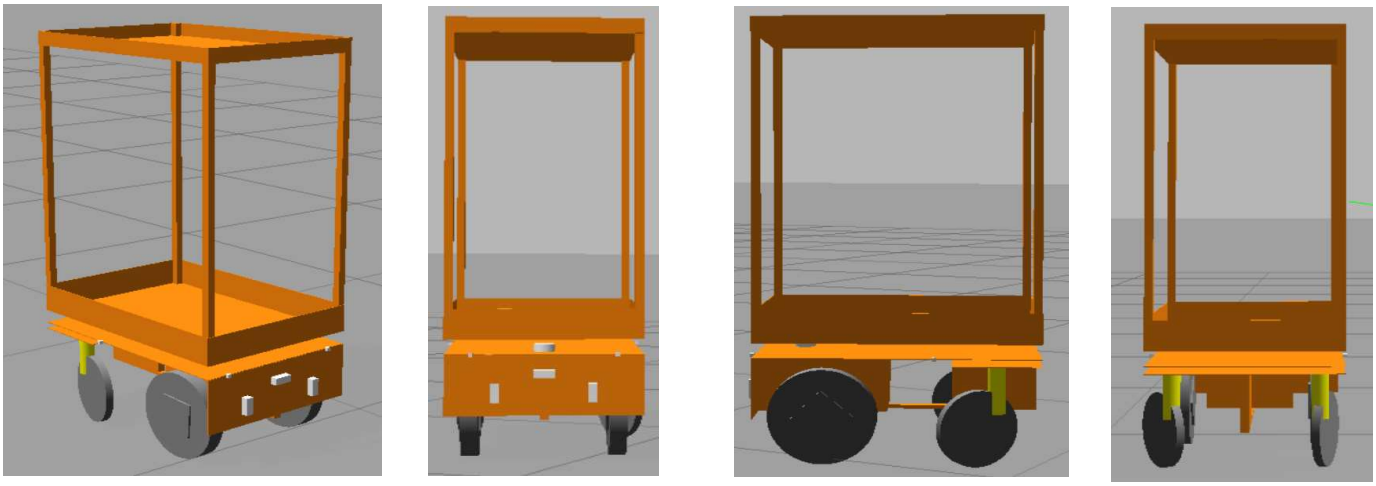
Prismatic : a sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits.

Fixed : This is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint does not require the axis, calibration, dynamics, limits or safety_controller.

Floating : This joint allows motion for all 6 degrees of freedom.

Planar : This joint allows motion in a plane perpendicular to the axis.

The result of the modelization :



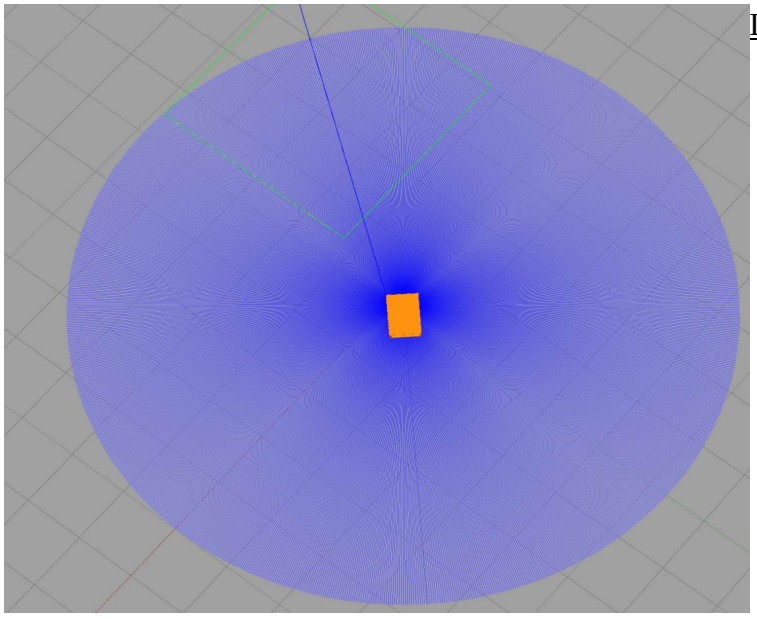
We can on the pulurobot different color, the orange is for the carcass she don't move and she is the origin of the robot, yellow for the links wheels, grey for the wheels and white for the sensor we will detail later.

5 – Sensors :

On the pulurobot we have different sensors (3 sonar, 4 3D camera and 1 LIDAR) and it's needed on the model for do the simulation in a virtual environnement, with Gazebo we access to different plugins for simulate this different sensor.

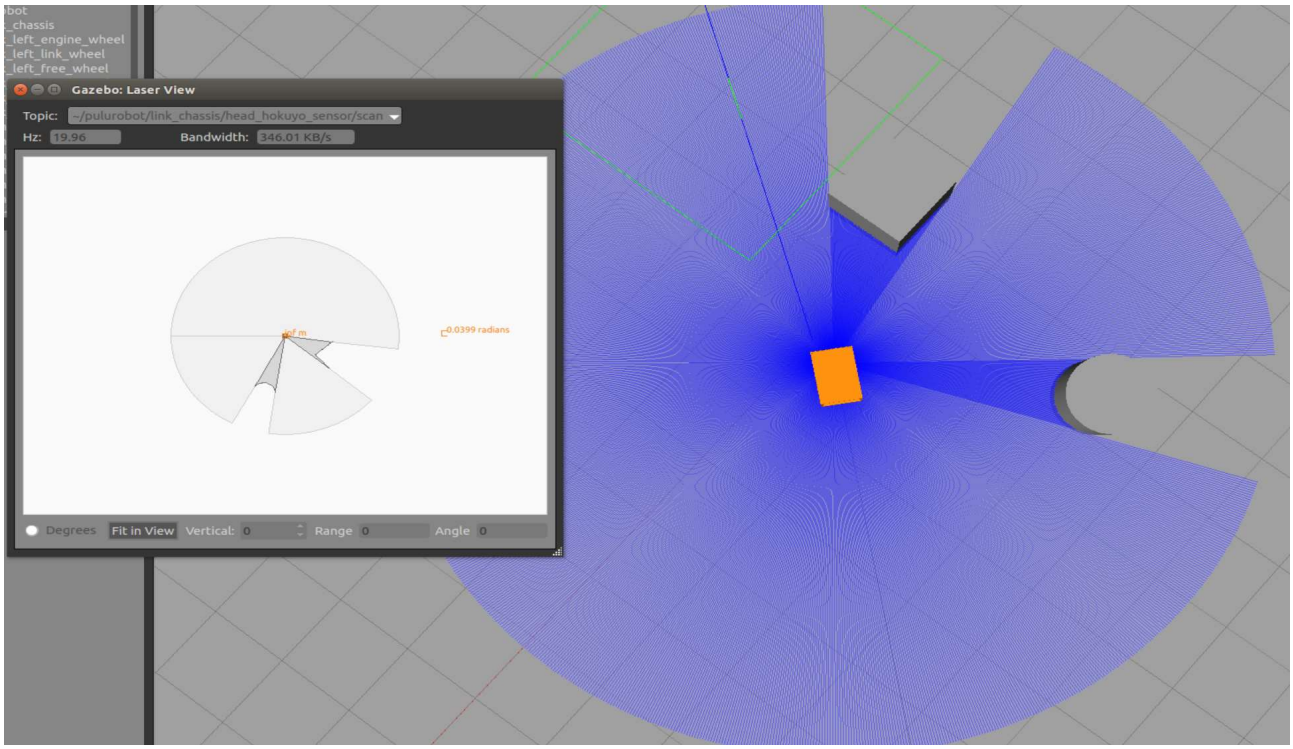
a. The LIDAR :

In Gazebo, the name of the LIDAR's plugin is « gazebo_ros_head_hokuyo_controller ». For the representation on Gazebo is a circle with a lot of ray from the center to the outside and we can choose the angle of the perception, the resolution and choose the number of ray :



LIDAR in Gazebo on the pulurobot.

Now we have the circle, the goal of the LIDAR is to make a map with different point detected with the different ray and the plugin do that alone too :

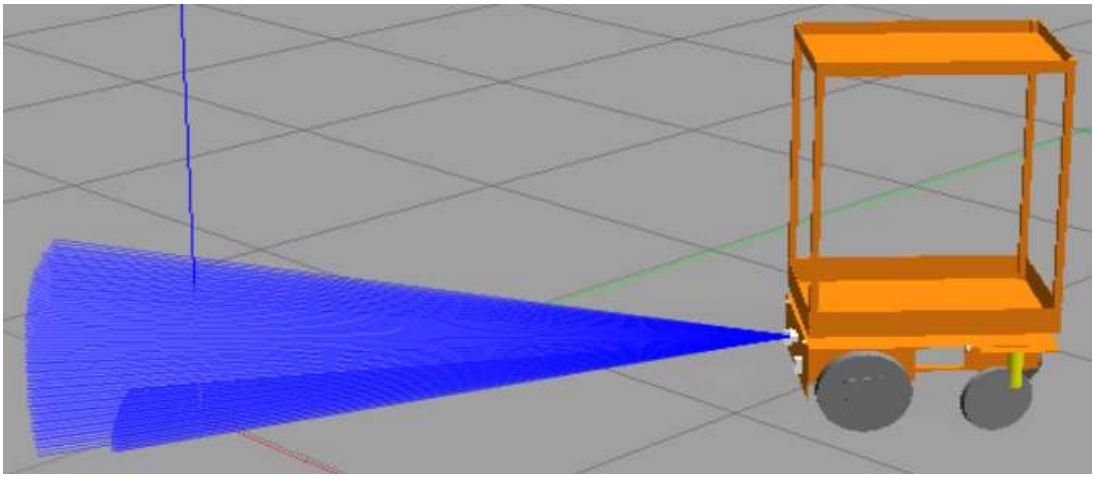


LIDAR with two obstacles.

We can see on the picture above two obstacles a cube and a cylinder. In the 3D view the ray stop on the different obstacles. In the other window, we can see the représentation of the 3D view in 2D with the position of the obstacles and this is the message send by the topic use by the LIDAR for use this information later.

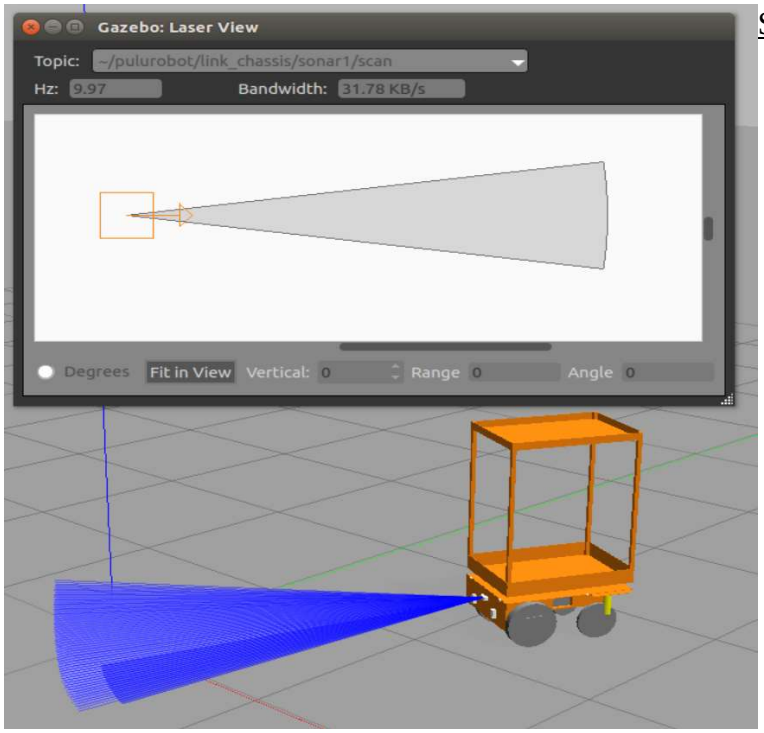
b. The sonar :

The pulurobot have 3 sonar, and the name of the sonar's plugin is « gazebo_ros_range ». For the representation we use a cone. In this cone we choose the angle, the resolution and the number of ray like the LIDAR, he use the same technique for the perception :



Sonar in Gazebo on the pulurobot.

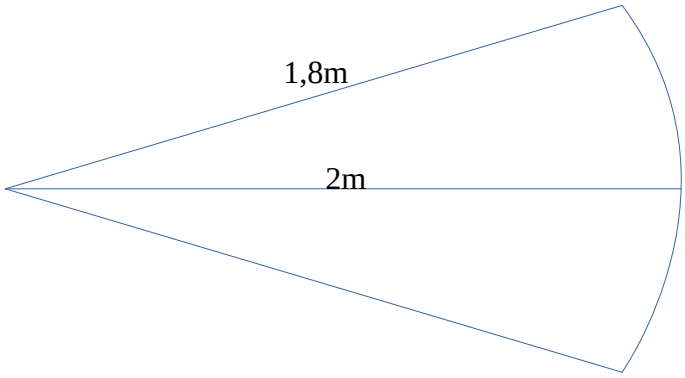
As the LIDAR we search to send the distance between any obstacle :



Sonar with the 2D representation.

On the 2D representation the message send is only the nearest distance.

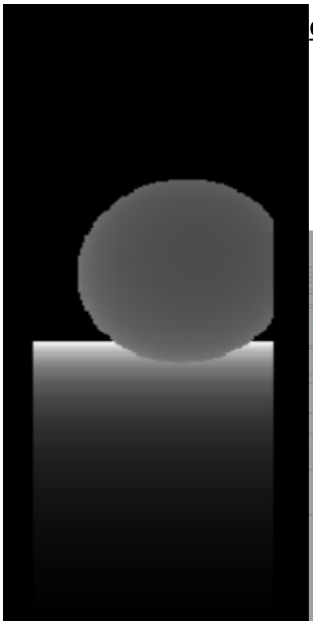
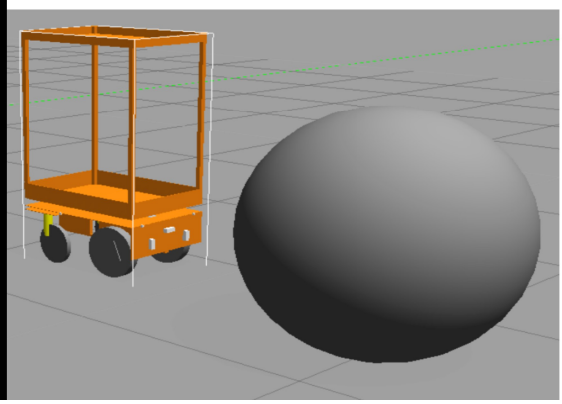
Example :



Finally the data send is 1,8m by the topic.

c. The 3D camera / depth camera :

the pulurobot possess 4 depth camera, and Gazebo have a plugin named « kinect_camera_controller » for that. The camera do an picture in monochrome, the information of black and white is the information of depth and these information are transmit by the topic :

camera depth view with the pulurobot in front of a sphere.

We can see the different shades on the picture and we can notice more is white more is far and the completly black is not detected.

6 – The visualization on RVIZ :

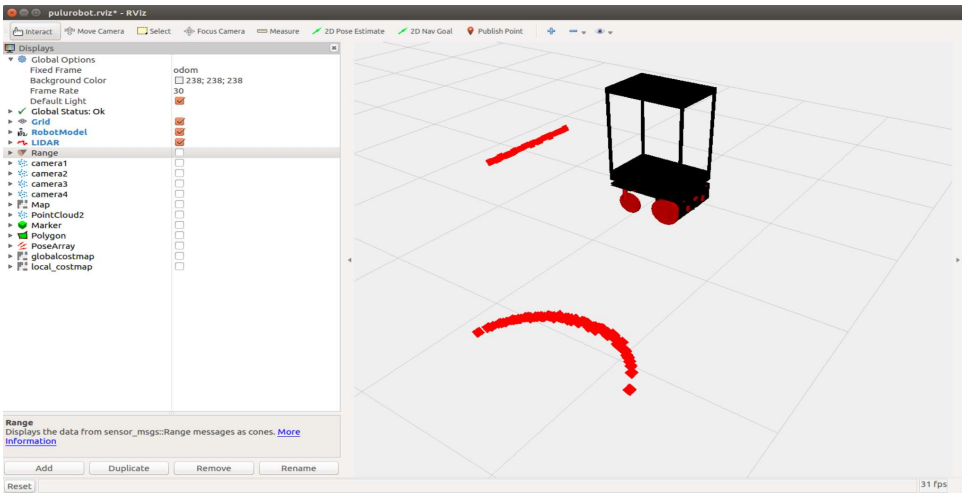
What is RVIZ ?

Rviz (ros visualization) is a 3D visualizer for displaying sensor data and state information from ROS. Using rviz. You can also display live representations of sensor values coming over ROS Topics including camera data, infrared distance measurements, sonar data, and more.

Why RVIZ ?

Because with Rviz you can visualize all sensor in same time in contrary with Gazebo you can't visualize all sensor without a lot of windows open. Rviz is easier than Gazebo for see the data sensor.

a. The LIDAR in Rviz :

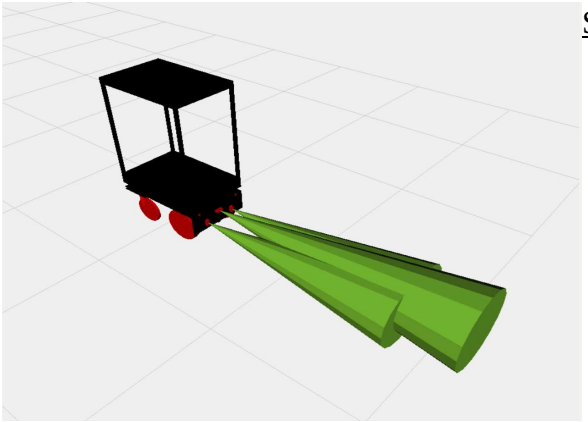


RVIZ interface with LIDAR data representation.

This is the user interface of Rviz a left panel and the scene like Gazebo with the difference the left panel show all sensor use for the visualization and the scene show all information by the sensor of the left panel.

For the LIDAR, we take the same situation before with a cube and a cylinder and we show the information send by the topic. We can see two shapes this is the cube and the cylinder, they are modeled by points.

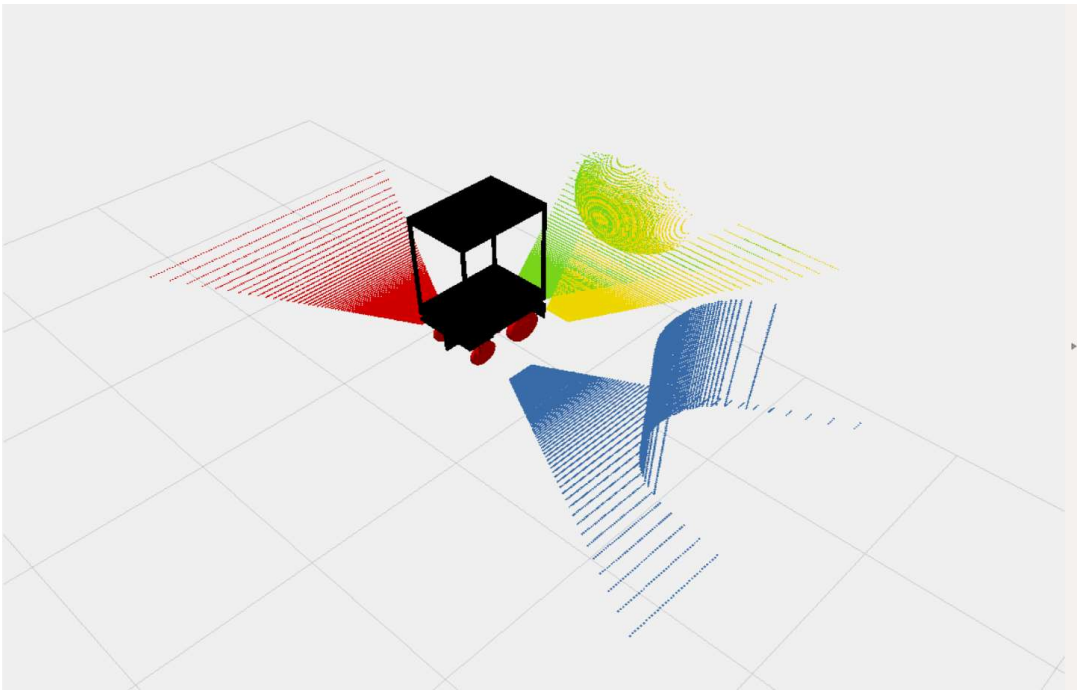
b. The sonar in Rviz :



Sonar representation.

Each sonar are represented by a cone who correspond at the distance between the sensor and the obstacle.

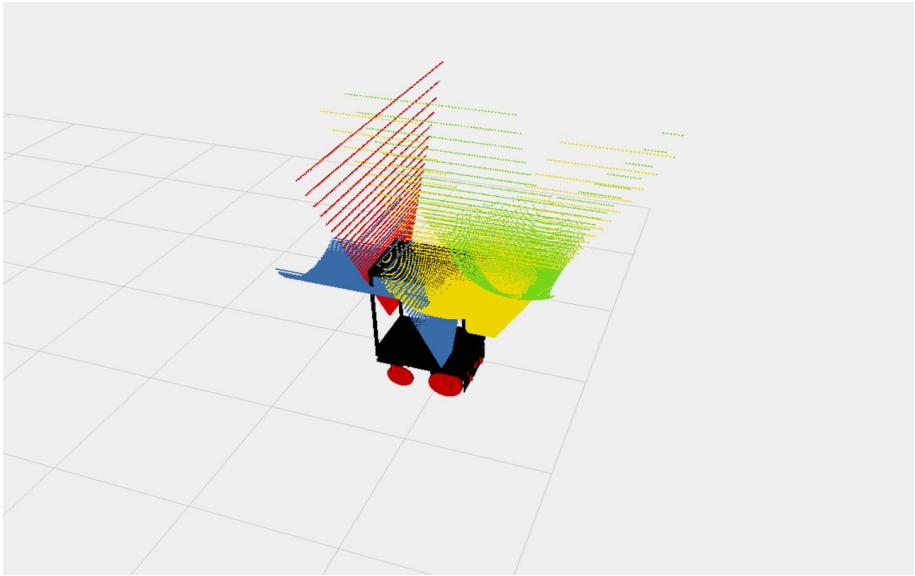
c. The 3D camera in Rviz :



Camera representation.

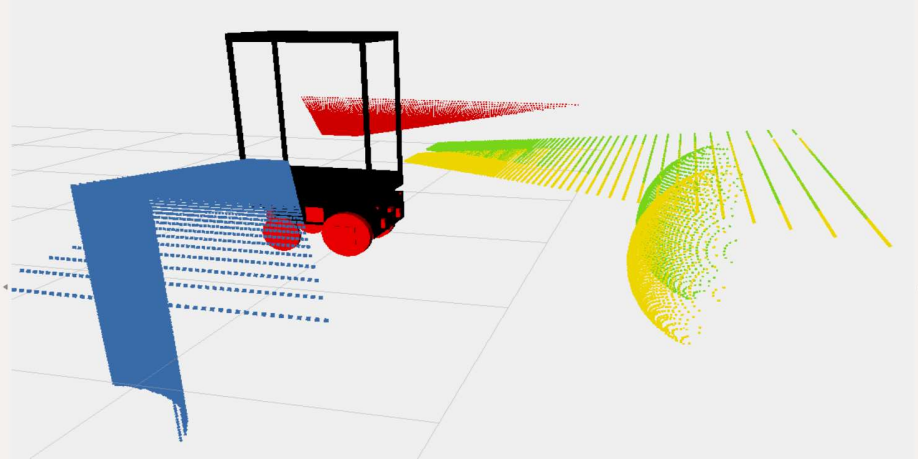
We can see all the data represented by points and we can determinate the shapes in the point cloud.

But I met a problem with the plugin for the camera because before this result the camera sent wrong information to the topic. The different axis were reversed (ZXY instead of XYZ).



Result on Rviz with the plugin unchanged.

And when the first problem solved the axis XY were in negative.



2Nd result with the plugin altmost corrected.

When I see this I correct this for have the result like the first pictures (Camera representation.) above.

7 – The movement for the pulurobot :

For control the pulurobot, Gazebo have a plugin (« differential_drive_controller ») :

```
<!-- control pulurobot -->
<gazebo>
  <plugin filename="libgazebo_ros_diff_drive.so" name="differential_drive_controller">
    <alwaysOn>true</alwaysOn>
    <updateRate>20</updateRate>
    <leftJoint>joint_left_engine_wheel</leftJoint>
    <rightJoint>joint_right_engine_wheel</rightJoint>
    <wheelSeparation>0.34</wheelSeparation>
    <wheelDiameter>0.125</wheelDiameter>
    <torque>0.1</torque>
    <commandTopic>cmd_vel</commandTopic>
    <odometryTopic>odom</odometryTopic>
    <odometryFrame>odom</odometryFrame>
    <robotBaseFrame>link_chassis</robotBaseFrame>
  </plugin>
</gazebo>
```

the control plugin in urdf.

To make work this plugin, we have to enter the joint of the left and right wheels, the separation between the two wheels and her diameter. The topic who received the command message is cmd_vel. If you want control him with your keyboard you just run the program teleop :

rosrun teleop_twist_keyboard teleop_twist_keyboard.py

This program send automaticaly the data to the topic cmd_vel when you press the good bouton :

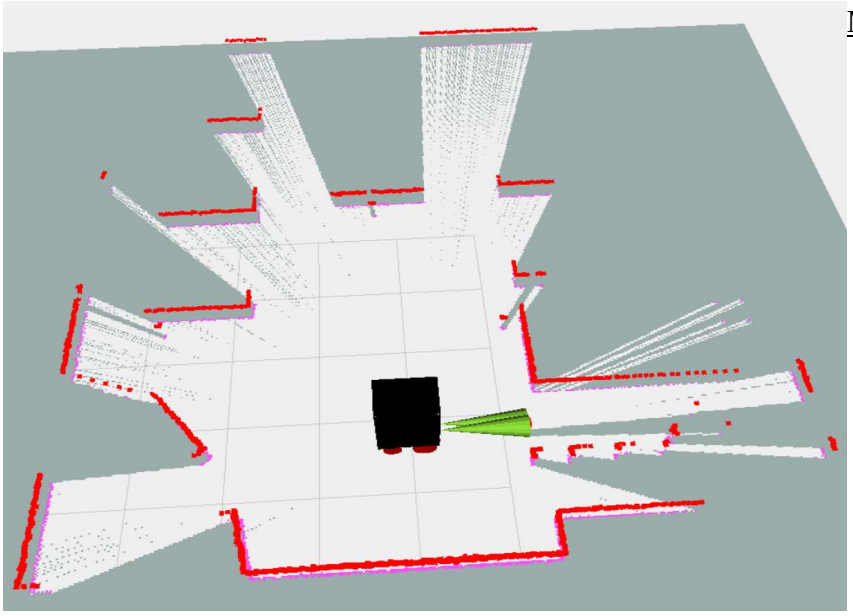
```
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit
```

The control command show when the program run.

8 – Mapping :

Now that our robot can go everywhere, we have to do a 2D map of the robot's explored places. For that ROS have a packages named « gmapping » and with this package we can scan the place and save into a 2D map the different data sensor.

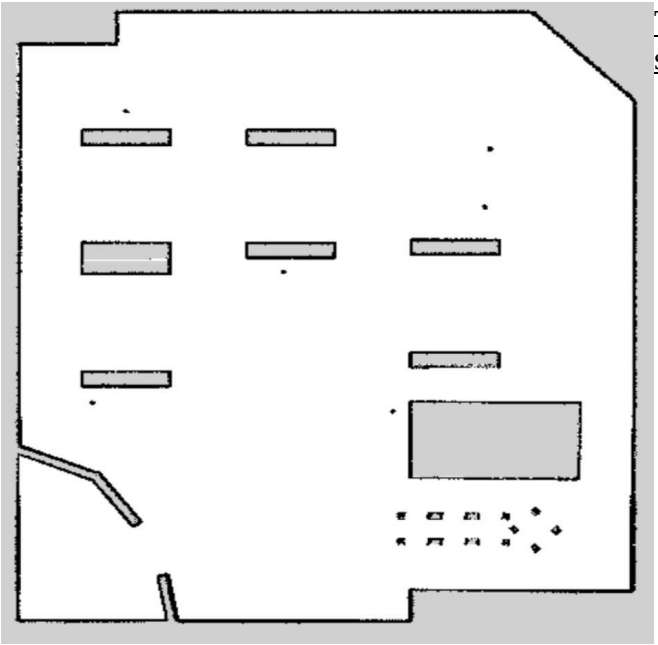
To implement this package to our project we just enter the topic of the LIDAR and run the package :



Map work in progress.

Here we use the virtual map create by my colleagues this is the software factory of the university of Helsinki.

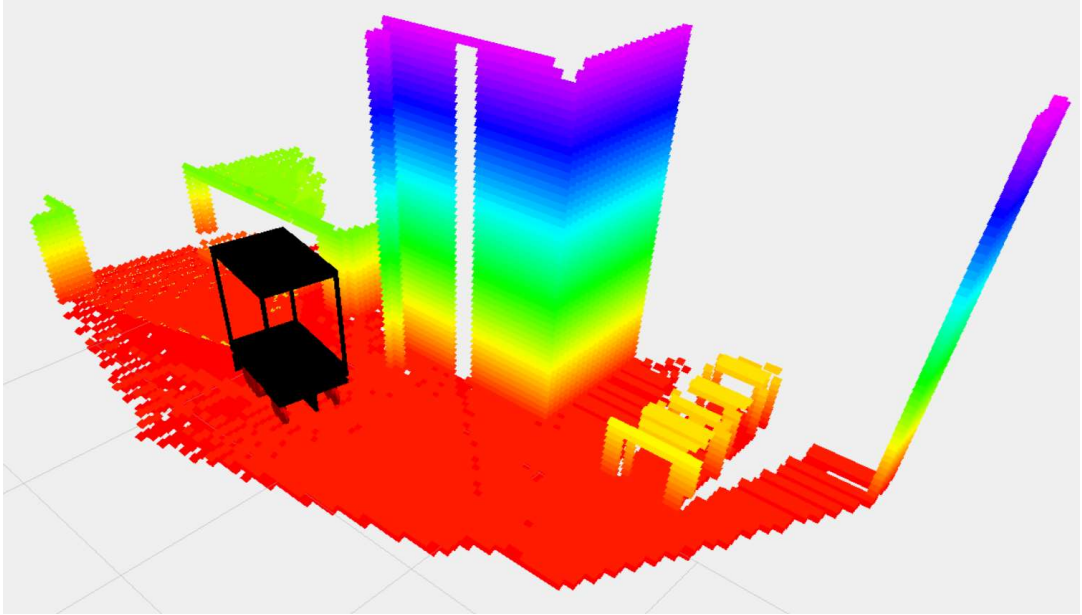
We can see the LIDAR's point and the map in white and grey, the white part is the discovered map and the grey part is the undiscored part, the wall of the map is represented by color points (here the points is in pink) and with the control before we can explore the entire map.



The software factory map complete (this is a simple image in bitmap).

With the Pulurobot we can make a 3D map too because he have 4 3D camera but we can't the same plugin because the 3D doesn't work on the gmapping's plugin, for the 3D mapping we use the plugin named « octomap ».

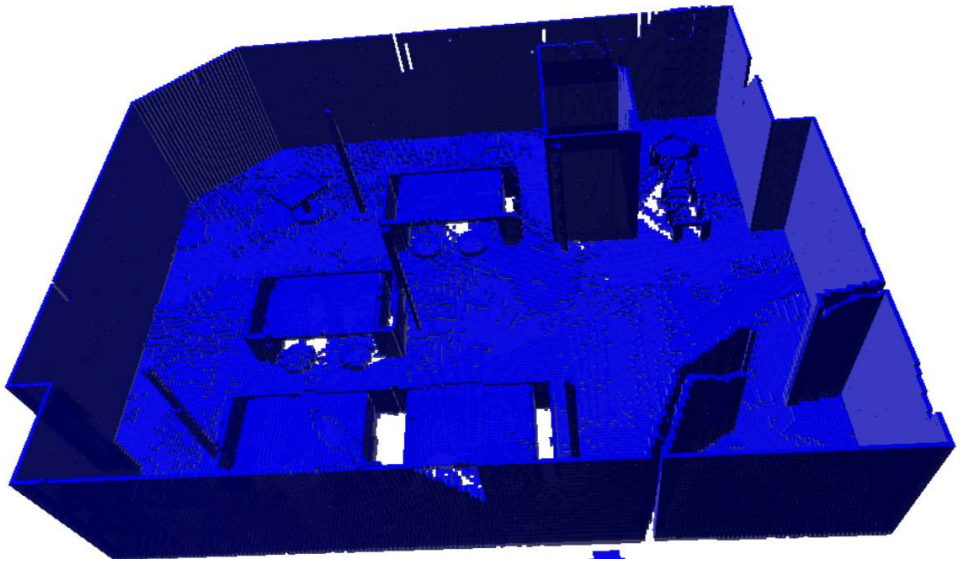
To implement octomap to our project like the gmapping we just enter the topic of the different camera and run the package :



Octomap work in progress.

Here like the gmapping we use the softaware factory of the university for do the map.

We can see the different points taken by the camera, we have the height with the color red for the bottom and purple for the top.



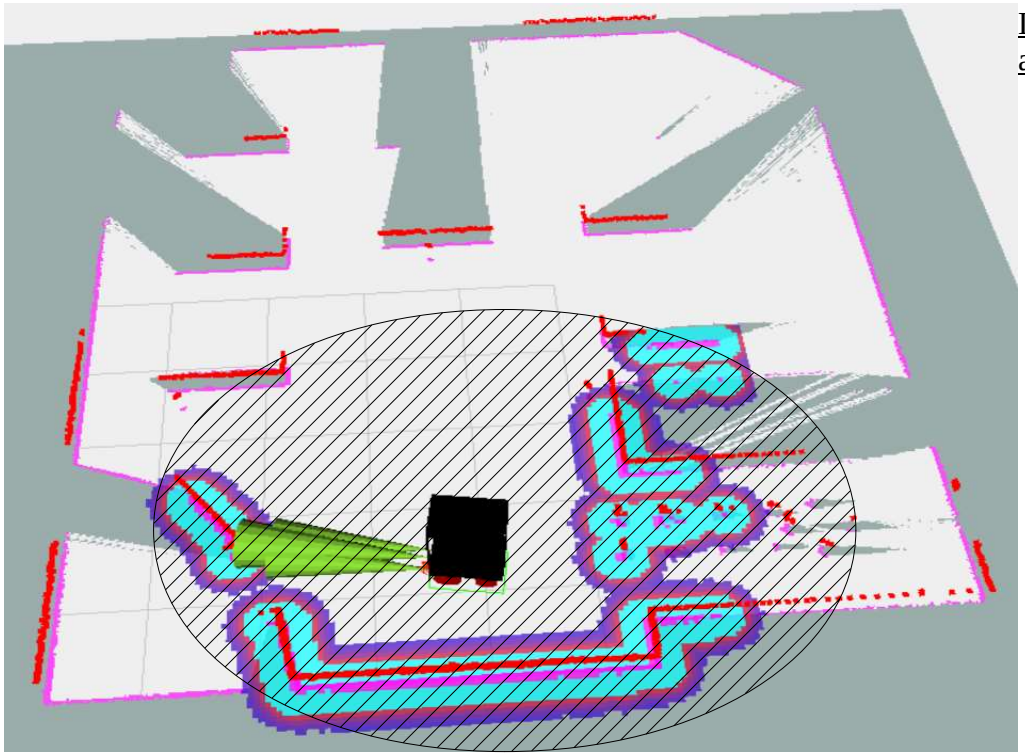
The software factory octomap complete (this is a map format .ot open only with octovis).

9 – Collision avoidance :

For the collision avoidance we use a package named « teb_local_planner » and for implement to our project we just run the package.

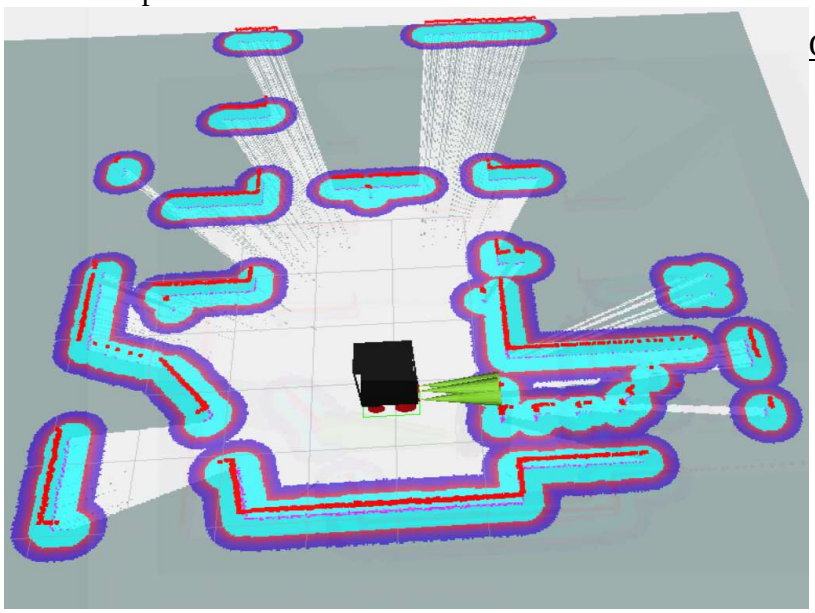
a. Collision detection representation :

For the collision detection the package use the map, two map more precisely, a local map and a global map. The local map is an area around the robot who detect the wall and show the areas where the robot are not allowed to go and the global map show all the prohibited areas of the map.



Local map with his area.

We can see the different prohibited areas for the robot, in blue for the most prohibited and in purple for the less prohibited.

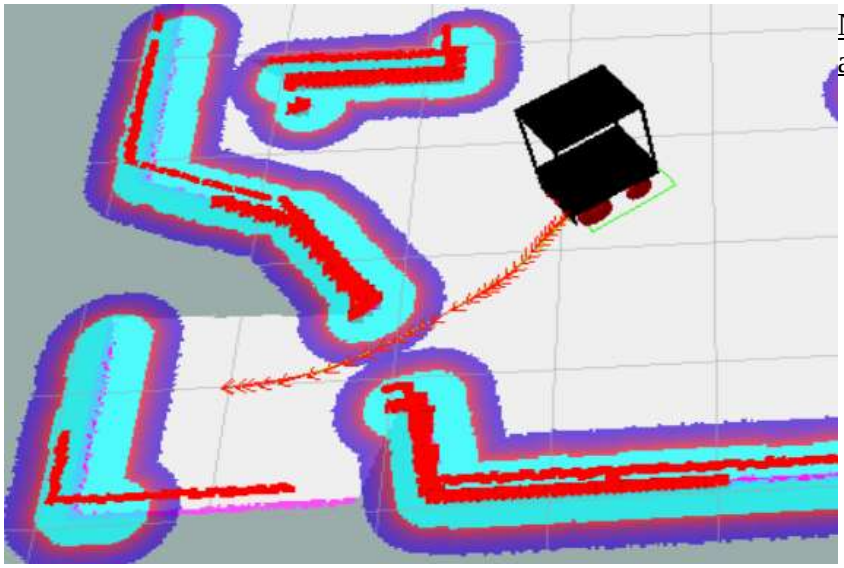


Global map.

We can see on the picture the global map show the prohibited areas on the entire map when the robot spawn.

b. Navigation from A to B :

The plugin can calculate a way for the robot with collision detection and permit to the robot to go every where without our support for control him we just tell to him where he have to go.



Navigation with collision avoidance.

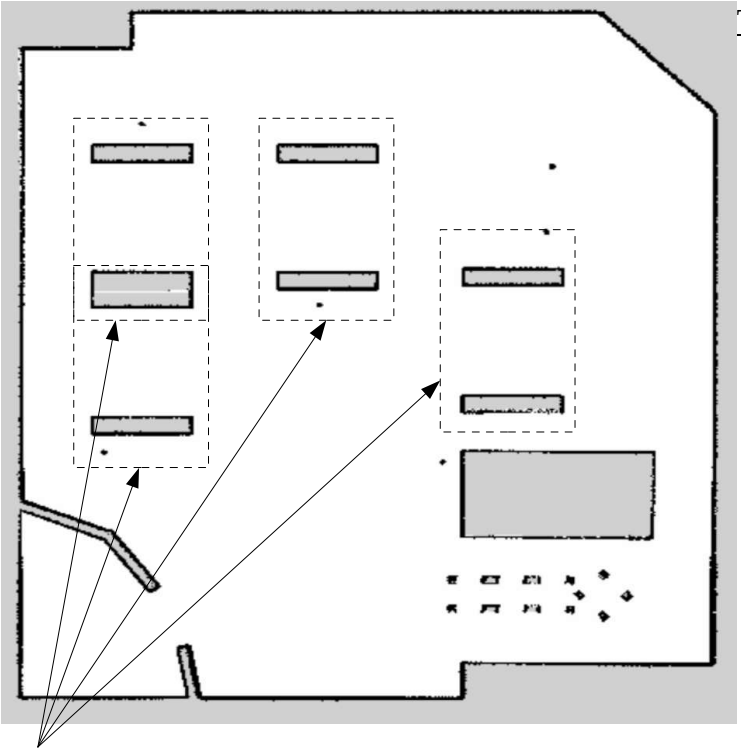
We can see on the picture the way take by the robot he is calculate by the package and apply for the simulation, and he avoids alone all obstacles.

c. 3D navigation :

We have a problem with the 3D navigation because he is unavailable, the ros package is in development, we have a message on the ROS website :

The message on the ros website.

The idea is the same than the 2D navigation but with more functionality like the detection of table because with the software factory room we have a lot of table and the pulurobot don't detect this table and the navigation see the table like a way and the pulurobot bangs against the table.



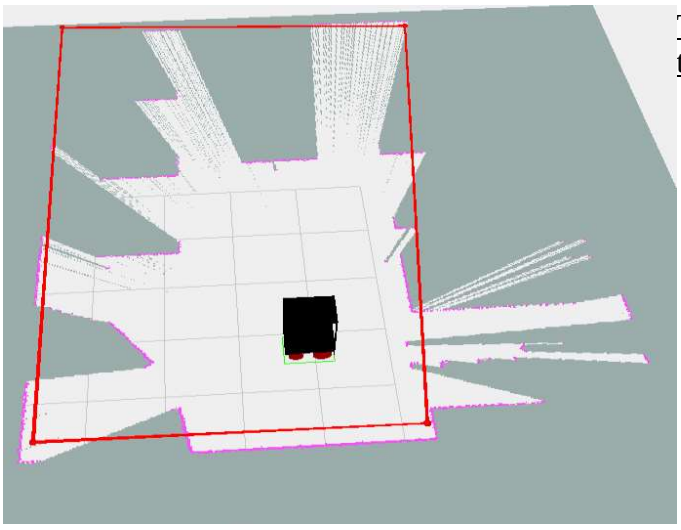
The software factory map.

These are table for us but different possibly way for the pulurobot.

10 – Area covering :

Now we have the navigation and the collision avoidance, we can't automatised the robot for permit him to do an exploration of the room automaticaly.

For that we have a package named « frontier exploration ». This package need the local map and the chassis link to work, with these two things we just run the package and give to the robot an area to explore, an area in the map because if not the robot would like go outside and the package is planting.

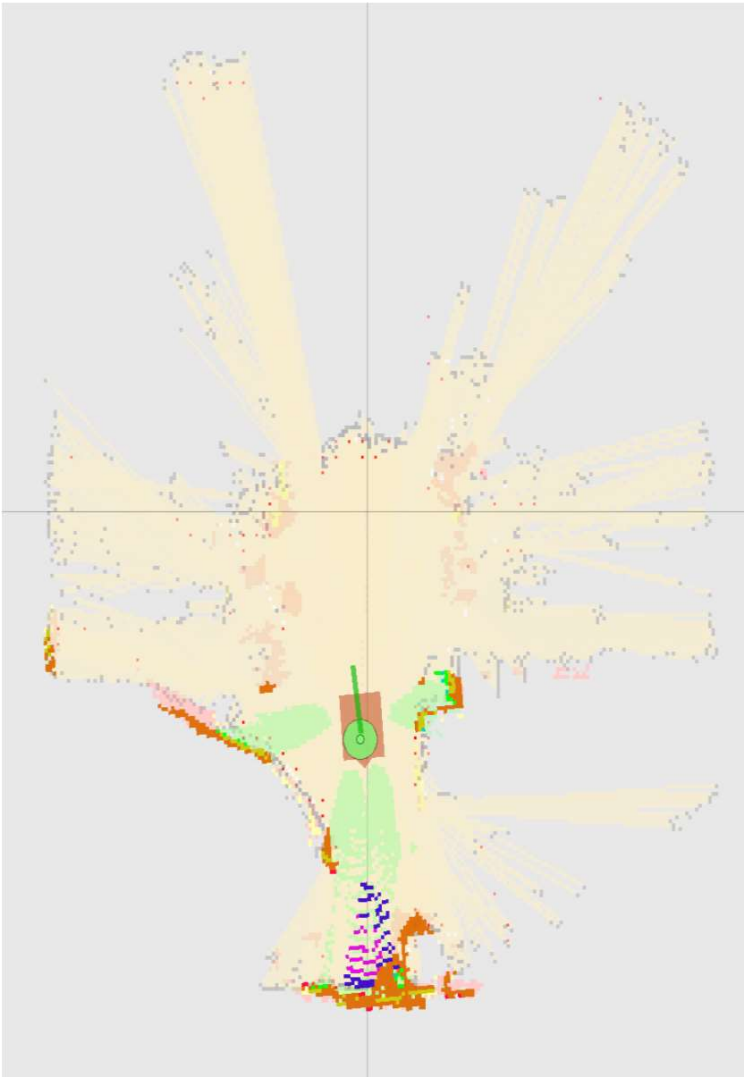


The software factory map work in progress with the area to explore by the robot in RVIZ.

The area in red is the area selected by us for the exploration and when we valid this area the robot work alone for discover the unknown parts of the map. When the robot finish his mission he stops and wait an other mission.

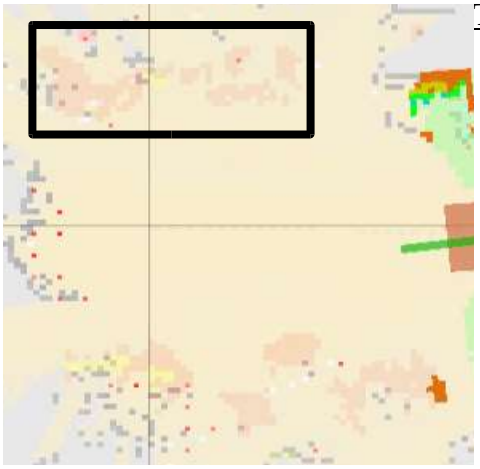
11 – Test the difference between the real pulurobot and the simulation :

For the test between the real robot and the simulation we go in the real softaware factory.



The real software factory with the real pulurobot.

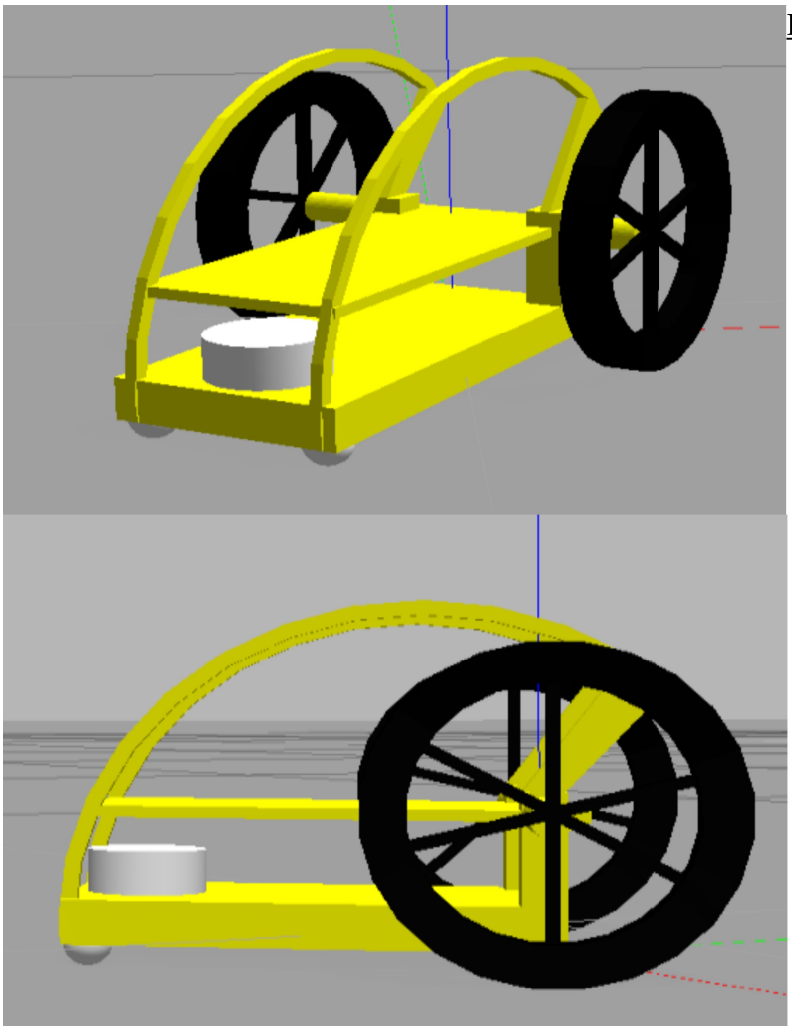
We can see a lot of similarity with the real and the false pulurobot like the mapping, the 3D sensor, the LIDAR and the navigation. The real difference between the two is the 3D mapping, the real pulurobot don’t do the 3D mapping but he detect the obstacles in 3D (Ont the picture these are the different area in light red) unlike the simulation the fake pulurobot do the 3D mapping but he doesn’t detect the obstacles in 3D.



The areas of 3D obstacles.

12 – Create my own robot :

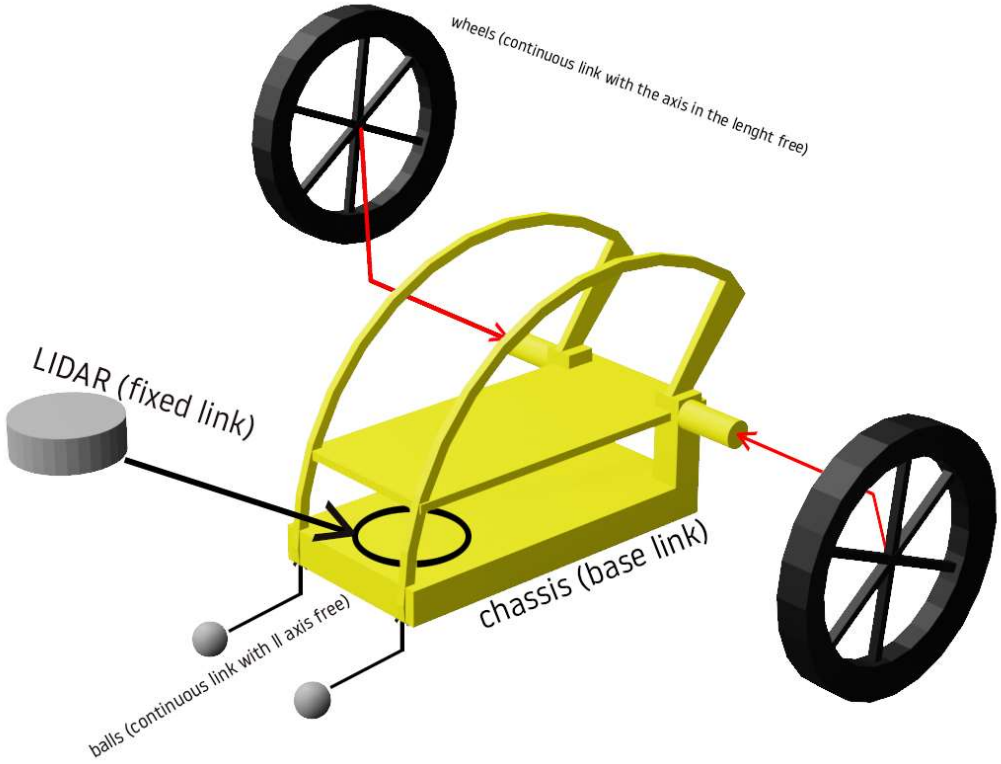
We have to do a robot with no rule, no mission. I decided to do a robot for mapping faster a room.



Design of the robot.

He possess two big wheel for the movement, two ball for slide on the floor and a LIDAR for do his mission. I put an arc on the top to maintain the printed cardboard and protect the cardboard and the LIDAR because the robot is light and faster, he can do a flip this is for his security.

13 – The architecture of my own robot :



Architecture of the robot.

This is the complete architecture of the robot with his type of joint.

Conclusion :

This intership was informative with the different skills acquired during the intership.

The mission entrust by our tutors was interesting because she mix the research and team work. The different missions given by the tutors was interested because when we do an action we see the result in live with the simulation.

The university left us free for our work and the place for work is very pleasant with the ubikampus and the software factory and others places to work in the calm.

And I take pleasure for see a different way of working it’s change to France, I see a different culture of mine and I’m really proud of that.

Documentation :

Tutorials on the Gazebo's website : http://gazebosim.org/tutorials

Tutorials on the ROS's website : http://wiki.ros.org/ROS/Tutorials

There are the two main documentation we use to do our missions with internet.