Project: Where am I ?

Vincent FORTINEAU, R&D Engineer, Paris, France

Project due : 12nd September 2018

Goals and steps of the project

- Create your own ROS package, develop a mobile robot model for Gazebo, and integrate the AMCL and Navigation ROS packages for localizing the robot in the provided map.
- Using the Parameter Tuning section as the basis, add and tune parameters for your ROS packages to improve your localization results in the map provided.
- After achieving the desired results for the robot model introduced in the lesson, implement your own robot model with significant changes to the robot's base and possibly sensor locations.
- Check your previous parameter settings with your new robot model, and improve upon your localization results as required for this new robot.

Evaluation criteria

Abstract

This report presents 2 simple robots created in a simulated environment. An ***Adaptative Monte Carlo Localization*** *(AMCL) algorithm was implemented in those robot using* ***ROS*** *packages to help them navigate in a maze like map. The parameters and configurations of the AMCL package are studied here.*

Table of Contents

Mobile Robot

1. Introduction

As said in this class, robots are defined by three actions :

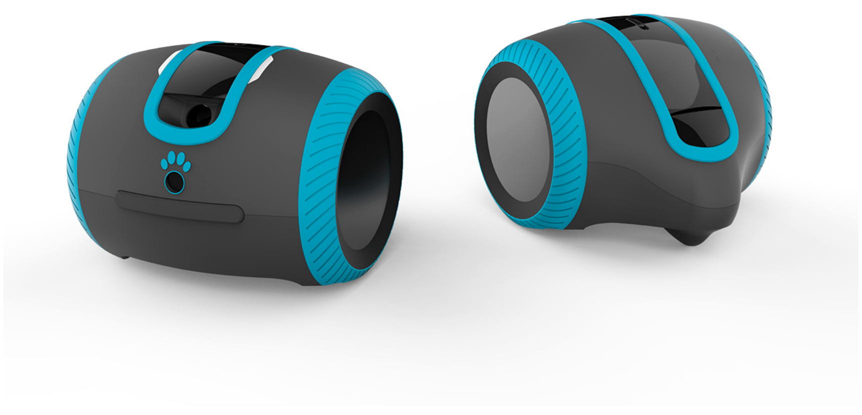
- **Sensing** the environnment (*sensors*)
- **Computing** and making decision (*processors*)
- **Acting** and interracting with the environnment (*actuators*)

For a mobile robot, the goal is often to go from a place to an other location. To reach this purpose, the robot need to know its **pose** (position and orientation), before beeing able to move in the right direction.

Sensors helps the robot to determine its location, we can talk about GPS sensors, that give a position on earth (thanks to satellites), inertial measurment unit that give information about the orientation and angular speed, LIDAR that helps getting distance from objects (landmarks...), camera and more.

Most of this sensors cannot give directly the position of the robot in its environnement. GPS has a precision within a meter, and is not available inside buildings, so it does not provide sufficient information.

Some sensors are very useful but require a **known environmment**, to be used for localization.


(C) Laïka, mobile robot for dogs entertainment, developped by Camtoy

2. Background

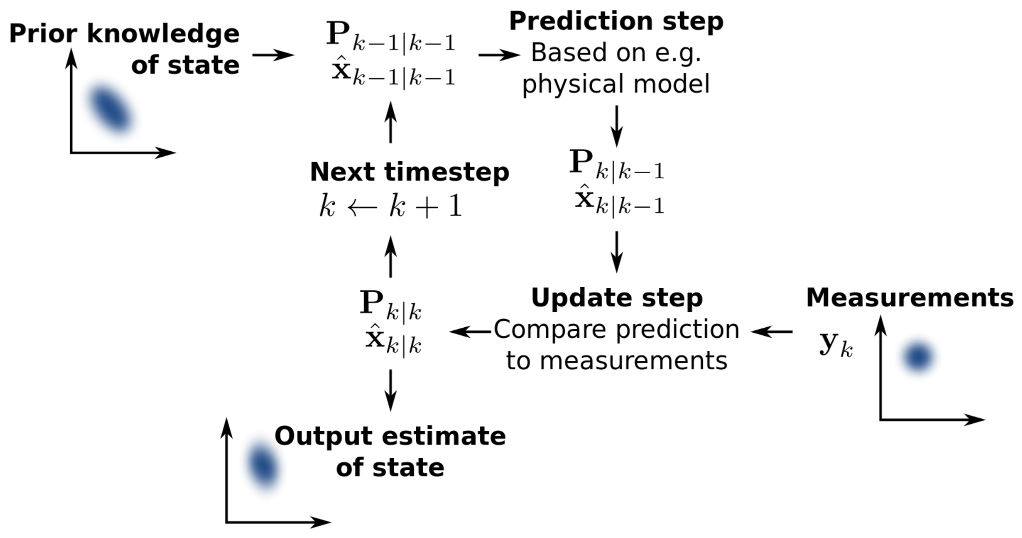
The localization problem might seem quite simple, at first sight. However, knowing that sensors are inaccurate and noisy and that actuators can be faulty and imprecise it becomes more of a challenge to reach good estimation of a robot pose.

In this report, we will use an method based on the **Monte Carlo** algorithm also know as **particle filter**. We will briefly presents the **Kalman filter** method that is an other know solution to the localization problem.

Even though we will not dicuss them, it is relevant to cite Markov and Grid Localization methods.

a. Kalman Filter

This method estimates the state of a dynamic system with noisy and incomplete data. It works with 2 phasis, the **prediction** and the **update**. The prediction phase uses the previous estimated state to produce a first guess of the current position, then the update phase correct this prediction using observation of the current state.


Prior knowledge of state
$\mathbf{P}_{k-1|k-1}$ $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
$\mathbf{P}_{k|k-1}$ $\hat{\mathbf{x}}_{k|k-1}$
Next timestep
$k \leftarrow k+1$
Update step
Compare prediction to measurements
Measurements
$\mathbf{y}_k$
$\mathbf{P}_{k|k}$ $\hat{\mathbf{x}}_{k|k}$
Output estimate of state

Basic concept of Kalman filtering by Petteri Aimonen, 2011 (CC)

It is relevant to notice that the Kalman filter makes the assumption that motion is linear and that the state of the system can be represented with a unimodal Gaussian distribution.

To overcome those limitation, the Extended Kalman Filter (EKF) works with differential fonctions, a Jacobian matrix to linearize the function near the current estimation. This solution uses more complex mathematics and requires more computational time and CPU.

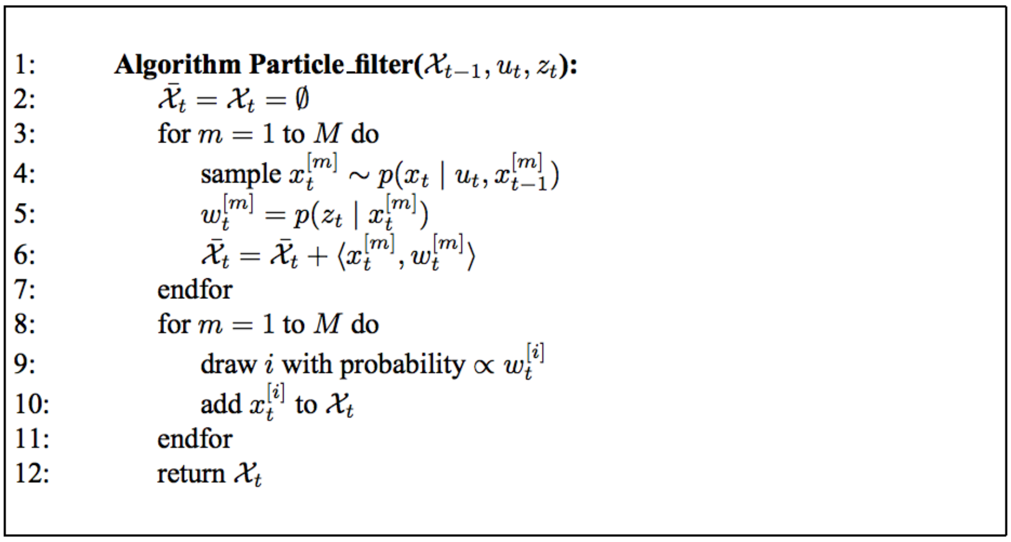
b. Particle Filter

Also know as the Monte Carlo (MCL) method in reference to the casino, this algorithm distributes weighted particles and adjuts the weights according to the probability of each particle. The least likely particules are remplaced by more probable ones.

The probability density, called belief, is estimated using recursive Bayes filtering.

The algorithm has five steps :

- Previous Belief
- Motion Update
- Measurement Update
- Resampling
- New Belief



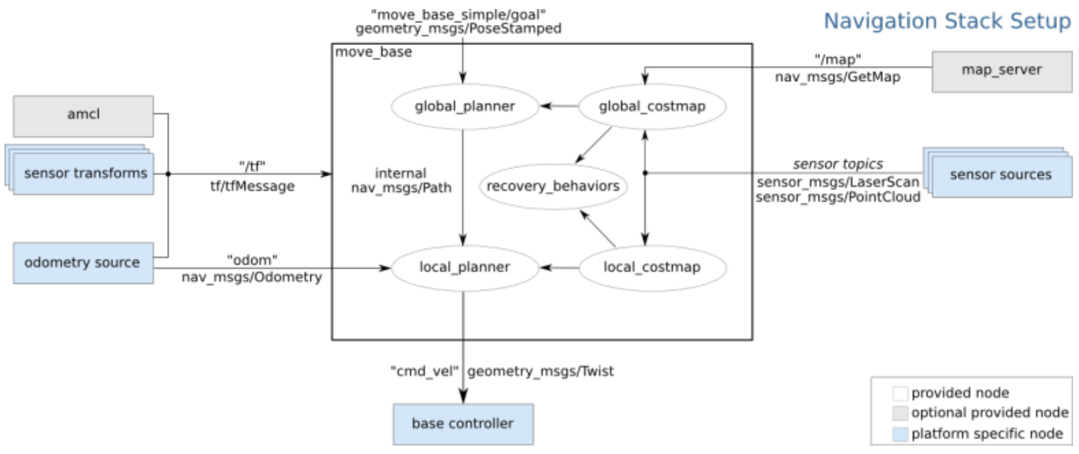
```
1:      **Algorithm Particle_filter**($\mathcal{X}_{t-1}, u_t, z_t$):
2:          $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3:          for $m = 1$ to $M$ do
4:              sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
5:              $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
6:              $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7:          endfor
8:          for $m = 1$ to $M$ do
9:              draw $i$ with probability $\propto w_t^{[i]}$
10:             add $x_t^{[i]}$ to $\mathcal{X}_t$
11:         endfor
12:         return $\mathcal{X}_t$
```

Particle filters provide a good alternative to Kalman filters, with a sufficient number of samples, this approch reach the optimal Bayesian estimate with more robustness.

	MCL	**EKF**
Measurements	Raw	Landmarks
Measurements noise	Any	Gaussian
Posterior	Particle s	Gaussian
Memory Efficiency	+	++
Time efficiency	+	++
Implementation	++	+
Resolution	+	++
Robustness	++	/
Mem.&res. ctrl	yes	no
Global localization	yes	no
State space	multimodal discrete	unimodal continuous

An adaptative particle localization method was used with ROS package. AMCL ROS package provides a dynamic algorithm that adpats the number of particles over time. This model combines **laser** data and **odometery** data.

3. Model Configuration



Tf overview, Robot Setup ROS (CC)

As described in the ROS documentation for AMCL:

"*AMCL is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to* *track the pose of a robot against a known map.*"

For a proper behaviour of the amcl package, several parameters can/must be tuned about the overall filter, the laser model and the odometery model.

The computational limitation of the CPU forces some parameters down, for example the maximum number of particles is by default set to 5000, and was reduced to 250, with a minimum number of particles set to 10.

It was also advised to modify some odometery parameters, called alphas, according to the model used. In our case, the model considered is the **corrected differential model**, that is defined as a "new model" in the documentation. It is said that "*the default settings of the **odom_alpha** parameters only fit the old models, for the new model these values probably need to be a lot smaller*".

The values suggested for the 4 alpha parameters are provided in the following table according to this discussion "*Tuning AMCL's diff-corrected and omni-corrected odom models*":

parameter	value
odom_alpha1	0.005
odom_alpha2	0.005
odom_alpha3	0.010
odom_alpha4	0.005

Other configurations need to be chosen for the base local planner, documentation can be found on the TrajectoryPlannerROS. The robot configuration, goal tolerance, forward simulation, trajectory scoring, oscillation prevention, and global plan can be tuned in this section.

```
TrajectoryPlannerROS:

  holonomic_robot: false
  max_vel_x: 0.45
  min_vel_x: 0.1
  max_vel_theta: 1.0
  min_in_place_vel_theta: 0.4

  acc_lim_theta: 3.2
  acc_lim_x: 2.5
  acc_lim_y: 2.5

  meter_scoring: yes

  pdist_scale: 0.5

  #sim_time: 5.0 # this value depends on the CPU

  controller_frequency: 10
```

The controller frequency had to be limited to 10Hz because of CPU limitation.

pdist_scale give the weighting for how much the controller should stay close to the path it was given, the default value was reduced a little.

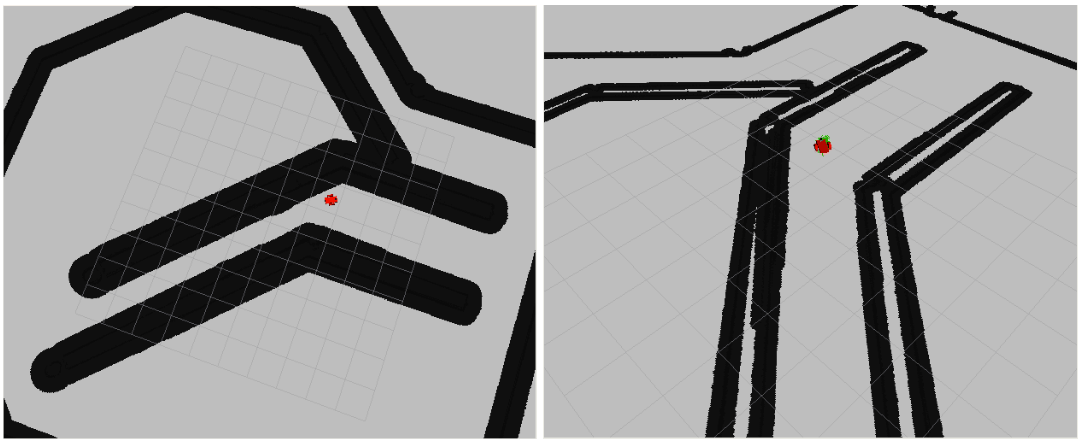
To avoid the following warning, the meter scoring was set to yes to use the metric system instead of cells (*relevant for pdist_scale and gdist_scale parameters*) :

```
[ WARN] [1536595508.057011340, 1527.939000000]: Trajectory Rollout planner initial
ized with param meter_scoring not set.
Set it to true to make your settins robust against changes of costmap resolution.
```

Finally the last parameters that need to be dealt with are about the Costmap configuration. Most of the configuration here were defined as suggested in *Common Configuration (local_costmap) & (global_costmap)*.

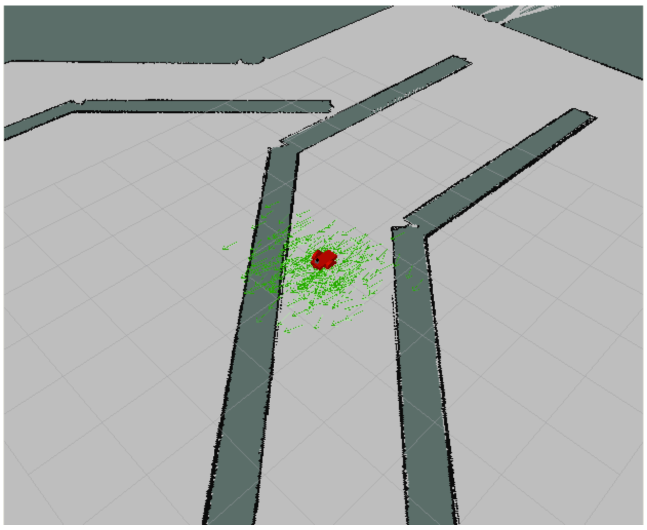
param	value
obstacle_range	2.5
raytrace_range	3.0
transform_tolerance	0.2
inflation_radius	0.40 - 0.15

The only parameter that had to be modified is the **inflation radius**, the robot was having unexpected behaviour with a value around 0.55, and above he was even getting stuck. It was experimentaly determined that with value between 0.40 and 0.15 inflation radius was not blocking the mobile robot (*a value too small could result in blocking scenario*). The inflation radius is highly dependent on the map.


On the left, an inflation radius of 0.4, on the right an inflation radius of 0.15, Vincent F. 2018 (C)

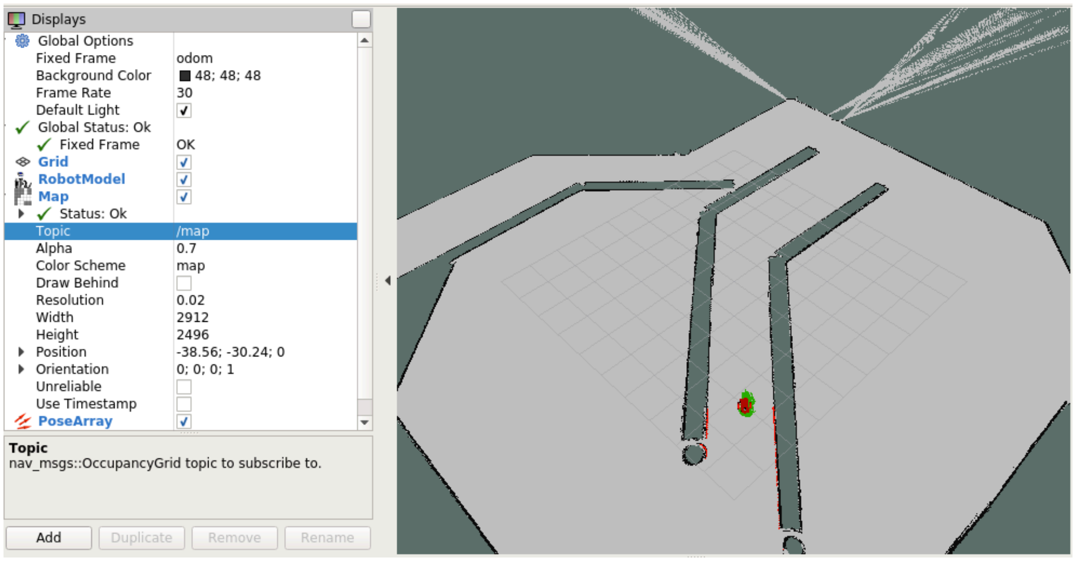
4. Results

The robot is created using URDF file, and animated in an simulated environment thanks to ROS packages. The simulated environment is provided by Gazebo, while we observe relevant data thanks to RViz.


Udacity bot at its start location, Vincent F. 2018 (C)

At first, as shown on the image above, the particle cloud is quite spread showing an important uncertainty.

Given all the previous configuration, the amcl algorithm was giving very encouraging results. Within a few seconds the robot is quickly located, and even if the robot is manually moved (*kidnapping problem*) it is able to found its new location. Althought it was not expected, the AMCL algorithm in Gazebo environment dealt really well with the kidnapping problem.



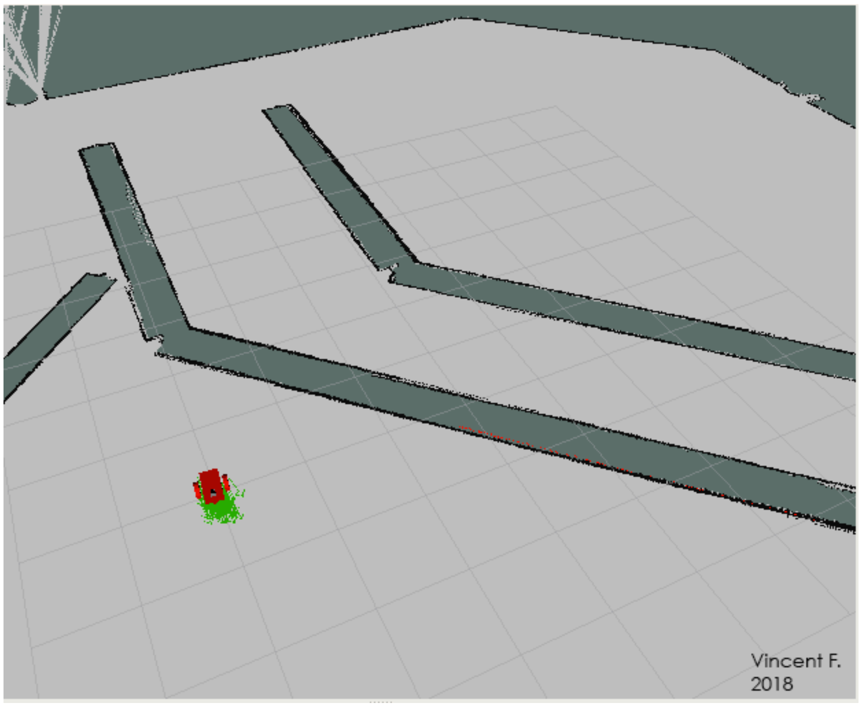
Displays

Global Options	
Fixed Frame	odom
Background Color	■ 48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
✓ Fixed Frame	OK
Grid	✓
RobotModel	✓
Map	✓
✓ Status: Ok	
Topic	/map
Alpha	0.7
Color Scheme	map
Draw Behind	☐
Resolution	0.02
Width	2912
Height	2496
Position	-38.56; -30.24; 0
Orientation	0; 0; 0; 1
Unreliable	☐
Use Timestamp	☐
PoseArray	✓

Topic
nav_msgs::OccupancyGrid topic to subscribe to.

The robot is represented in red, the amcl particles are represented with green arrows, and the laser scan are the red dots that feats the walls, Vincent F. 2018 (C)

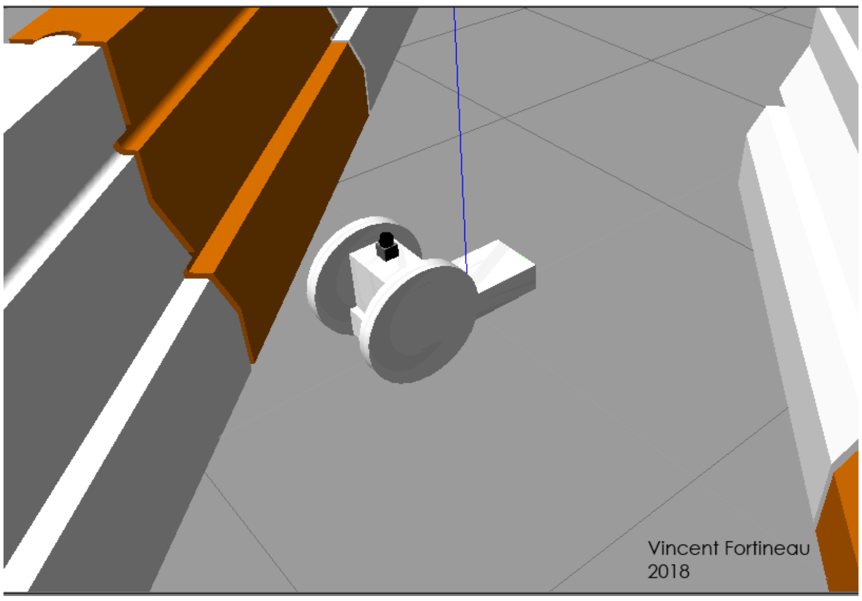
Even if it took few minutes to reach the goal with a non optimized trajectory, the robot was able to reach its destination with precision.



Goal reached with sufficient certainty about the robot pose, Vincent F. 2018 (C)

5. Discussion

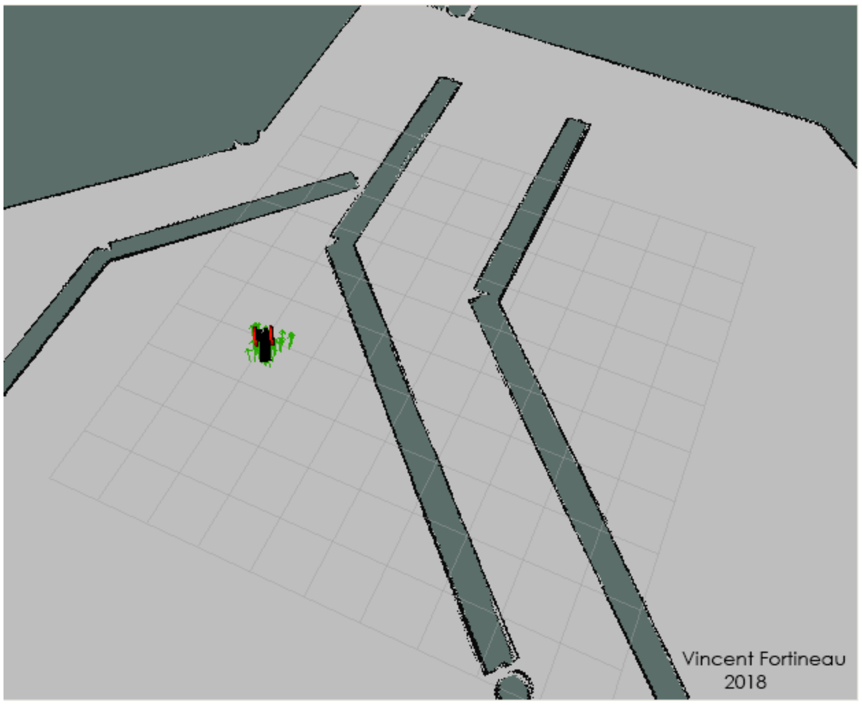
The customized robot had a notable different shape, had only one symetry axis. Le laser scanner was mounted on a more elevated position with no significant difference. However, the size of the front wheel had to be taken into consideration to avoid undesired obstruction of the sensor. The robot was completely defined with a URDF file, a great tutorial is available in the wiki ROS.



Customized robot shape in Gazebo, Vincent F. 2018 (C)

Considering the configuration, only few things had to be modified, the inflation radius had to be slightly raised, the wheel control had to be modified to take into consideration their new radius. The robot footprint/radius was not defined but it should probably be taken into consideration to avoid possible collisions.

The performance of the customized robot were a little worse than the first one, probably because of its loss of one symetry, that made it harder to handle.



Customized robot toward goal, Vincent F. 2018 (C)

Conclusion

The project focused on the localization problem, although the navigation was clearly not optimal and too slow to be considered implemented in real time situations.

The next step to focus on, would certainly be about navigation to try to reach position as quickly as possible.

In a controlled environmment with clear path, and flat ground, for example in a plant with defined path for robots, this kind of algorithm would work perfectly. It is important to underline that only one robot was handled at once in those experiments, for industrial purposes, it would be interresting to extend those experiments to several robots.

The use of swarm bot could help localize the robots if they communicate data with each other. In this particular case, the processor could probably be deported to handle a larger amount of data.