TÉCNICO LISBOA

Pioneer 3-DX with EKF-SLAM

Eufémio Marques | 91154

Ivan Figueiredo | 93386

Miguel Roldão | 93405

Pedro Santos | 93411

Autonomous Systems

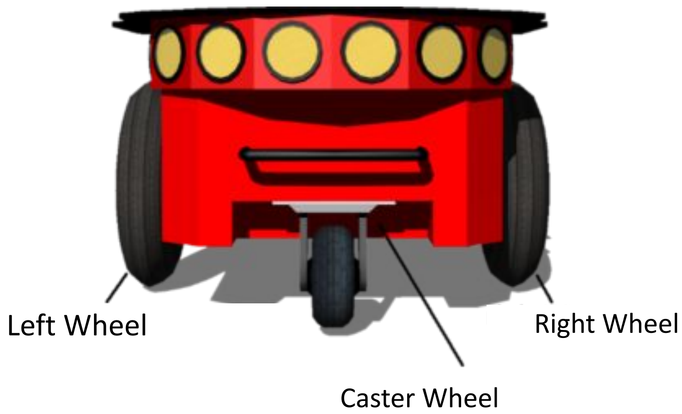
Group 17

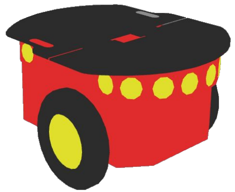
June 2022

The project

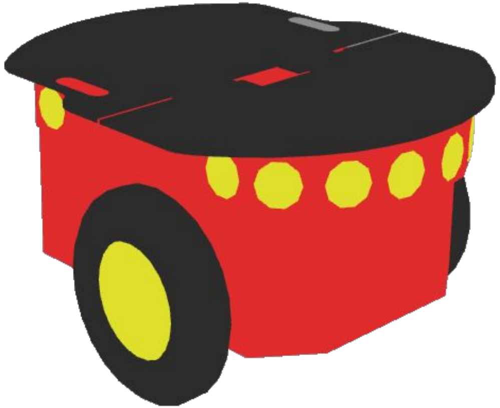
Implement a **S**imultaneous **L**ocalization **A**nd **M**apping (**SLAM**) algorithm

using an **E**xtended **K**alman **F**ilter (**EKF**) on a **Pioneer 3-DX** robot.





Sensors

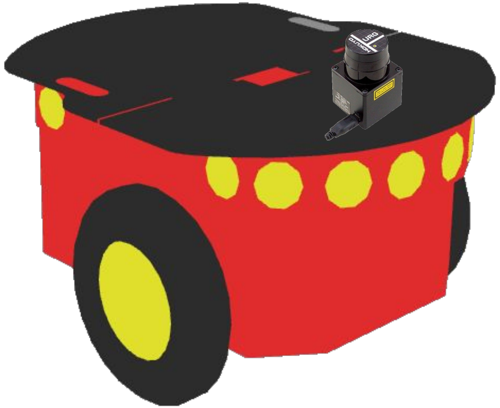


Incorporated in the robot

Odometer (left+right wheels)

8+8 sonars (front and back panels)

Sensors



<u>Incorporated in the robot</u>

Odometer (left+right wheels) ✓

~~8+8 sonars (front and back panels)~~

<u>Connected through USB</u>

Hokuyo Laser ✓

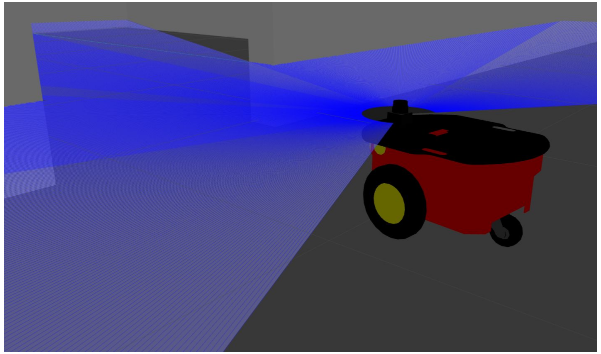
Hokuyo laser

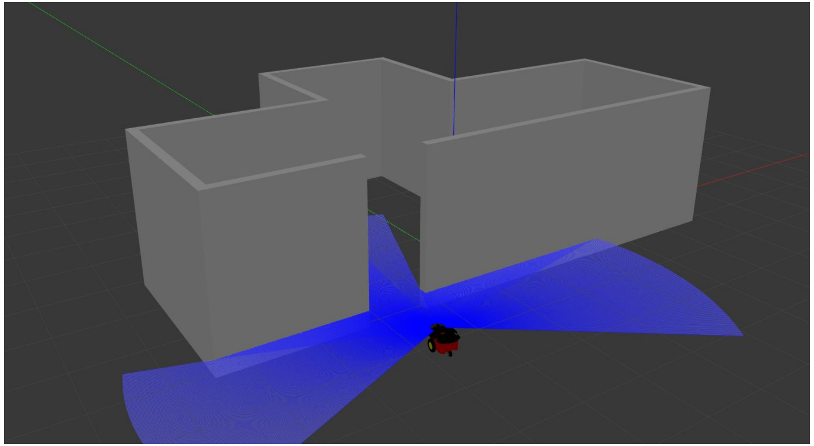
Model No.	URG-04LX-UG01
Power source	5VDC±5%(USB Bus power)
Light source	Semiconductor laser diode(λ=785nm), Laser safety class 1
Measuring area	20 to 5600mm(white paper with 70mm×70mm), 240°
Accuracy	60 to 1,000mm : ±30mm, 1,000 to 4,095mm : ±3% of measurement
Angular resolution	Step angle : approx. 0.36°(360°/1,024 steps)
Scanning time	100ms/scan

Microsimulation

GAZEBO

Gazebo creates simulated environments to test a robotic system





GitHub mario-serna / **pioneer_p3dx_model** Public

The algorithm

$$\mu_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ m_{1,x,t} \\ m_{1,y,t} \\ \cdots \\ m_{N,x,t} \\ m_{N,y,t} \end{bmatrix}$$

The robot moves and the new predicted state estimate is computed from the previous estimate

Algorithm 1 Extended Kalman Filter $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$

Prediction step

1: $\overline{\mu_t} = f(u_t, \mu_{t-1})$

2: $\overline{\Sigma_t} = G_t \Sigma_{t-1} G_t^T + R_t$

Correction step

3: $K_t = \overline{\Sigma_t} H_t^T (H_t \overline{\Sigma_t} H_t^T + Q_t)^{-1}$

4: $\mu_t = \overline{\mu_t} + K_t(z_t - h(\overline{\mu_t}))$

5: $\Sigma_t = (I - K_t H_t)\overline{\Sigma_t}$

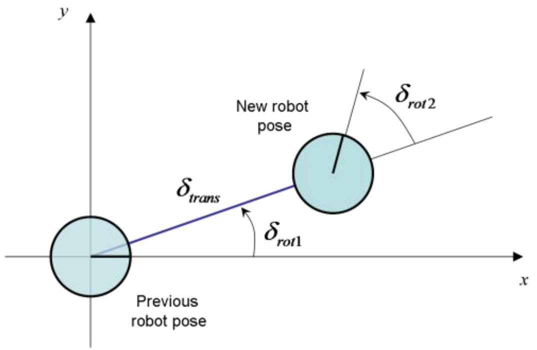
6: return μ_t, Σ_t

The laser identifies the known landmarks and the previous drawn map is corrected

Odometry and prediction step

Odometry

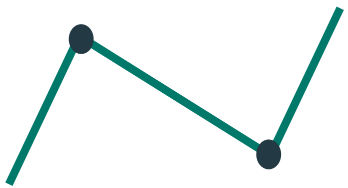
- The use of motion sensors to calculate the change of position of the robot in relation to a known position.
- 1st rotation + translation + 2nd rotation
- The motion model assumes that δ_{rot1}, δ_{trans} and δ_{rot2} are influenced by independent noise.



Prediction step

- The new estimate of the the predicted state is calculated from the previous estimate.

Landmarks



- We have used a mixture between lines and corners

Lines

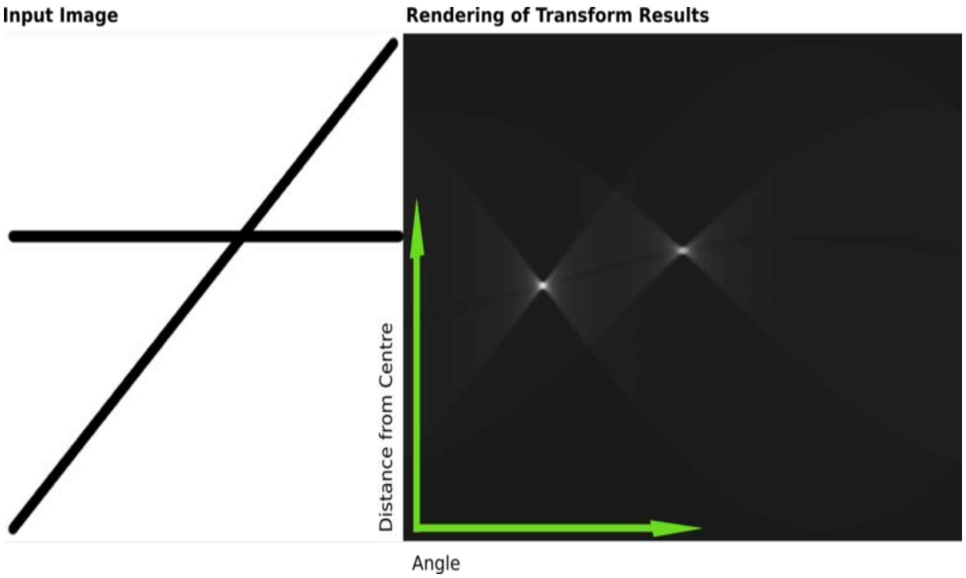
- Easy to identify
- Broad idea of the space around the robot
- Unexpected behavior in other more irregular environments

Corners

- Minimize the number of parameters exchanged between nodes
- Susceptible to noise
- The dots have to be connected to each other

Landmark detection and Hough transform

- Hough Transform is used to identify and isolate certain features presented in an image such as lines or circles.
- The points in Hough space have the form P(ρ,ϴ).

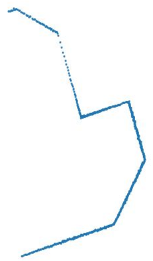


Hough transform

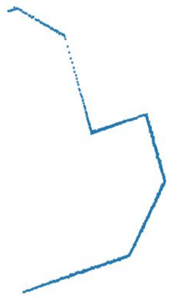
- It was implemented using the library in openCV

Probabilistic Hough Transform (used in order to detect lines/corners)

- Gives more accurate results
- Gives directly the starting and ending points of each line, which are the exact points the will be assumed as corners



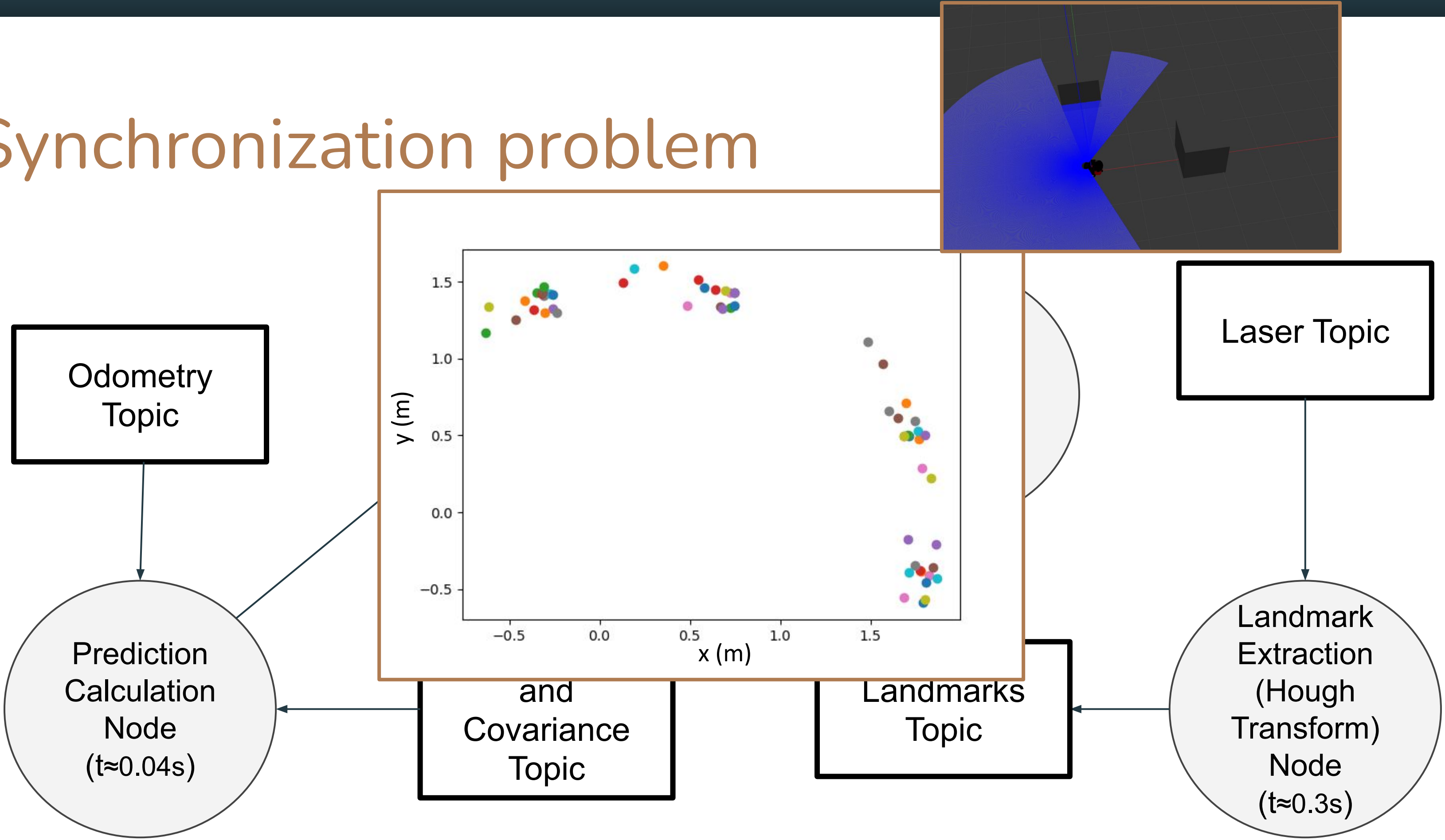
Standard method



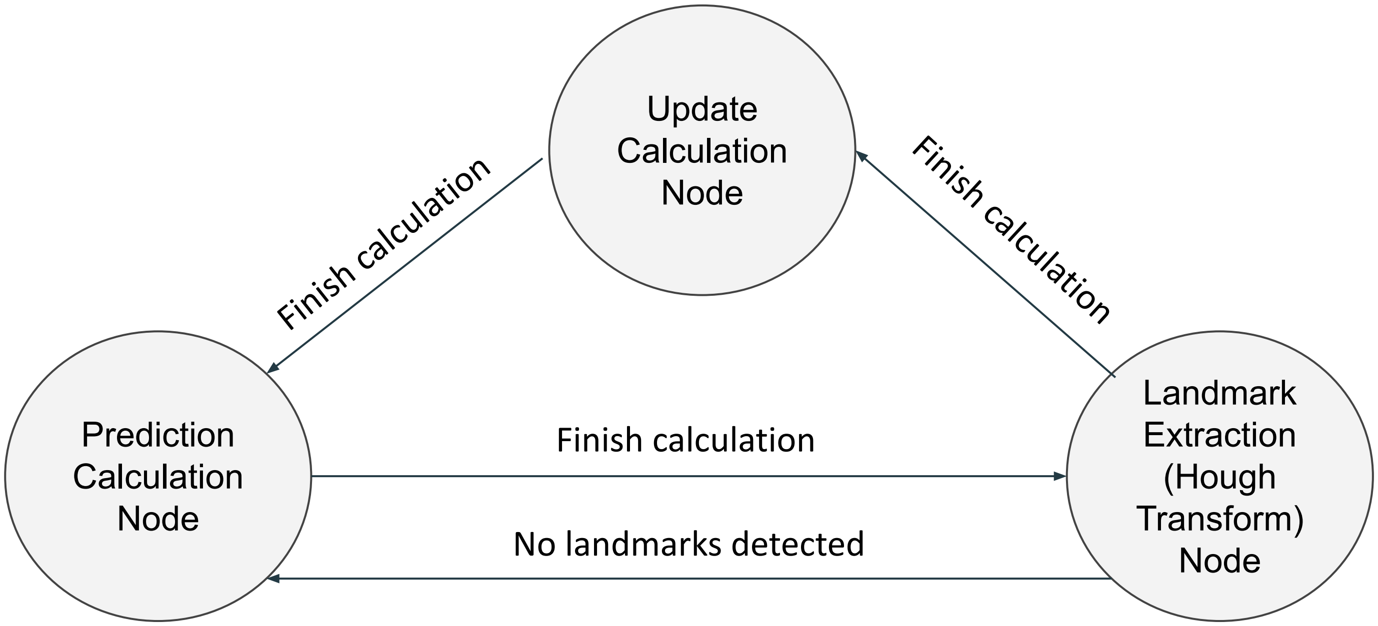
Probabilistic method

Threshold: the minimum number of intersections in the parameters space to detect a line/corners

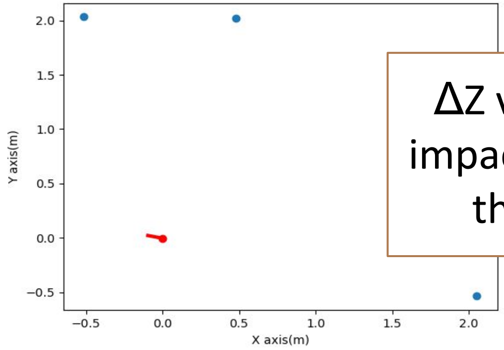
Synchronization problem



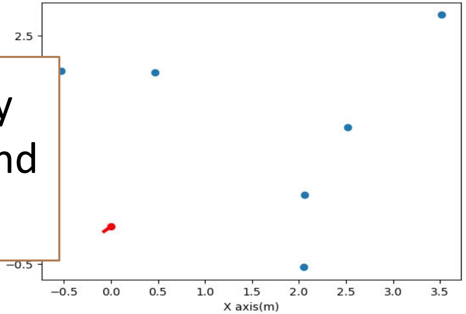
State machine implementation

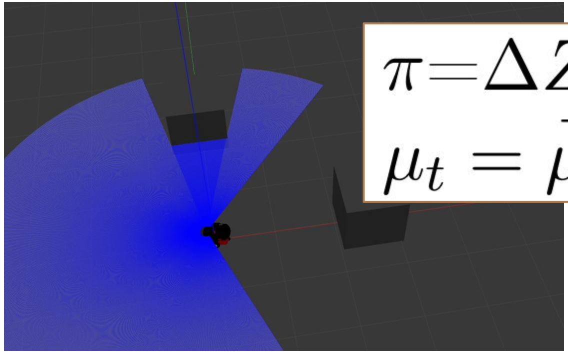


Discontinuity Problem



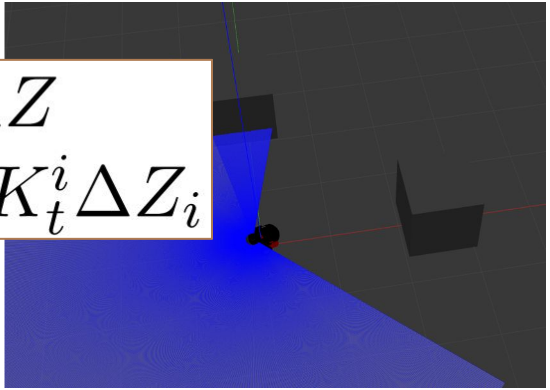
ΔZ vector discontinuity impacts the matching and the correction step





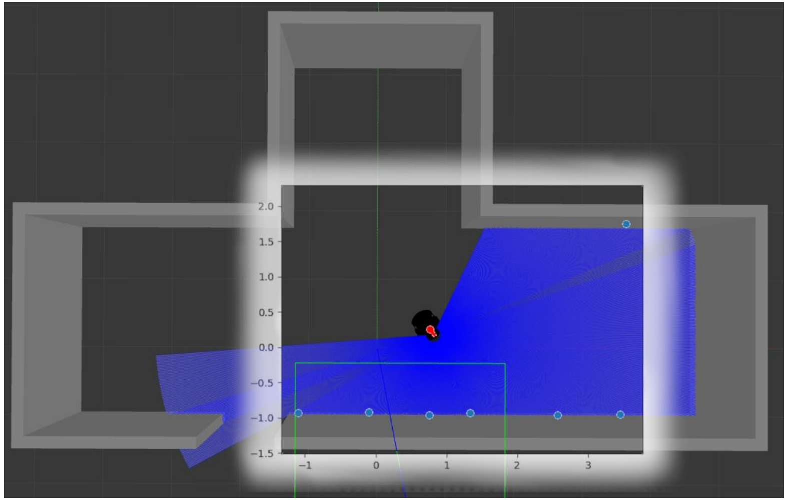
$$\pi = \Delta Z^T \Psi^{-1} \Delta Z$$

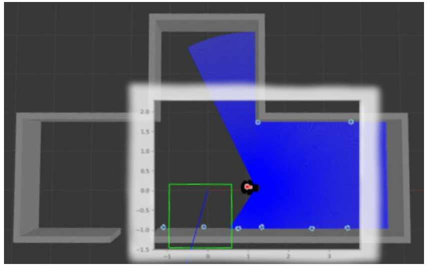
$$\mu_t = \bar{\mu}_t + \sum_i K_t^i \Delta Z_i$$

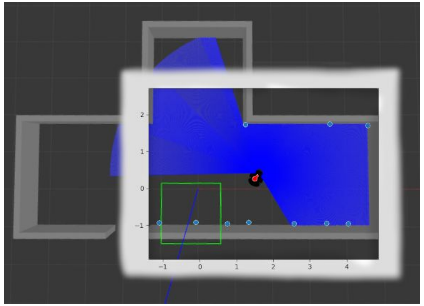


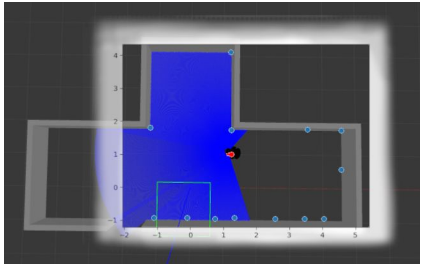
Microsimulation results

Using Gazebo to run and test our model



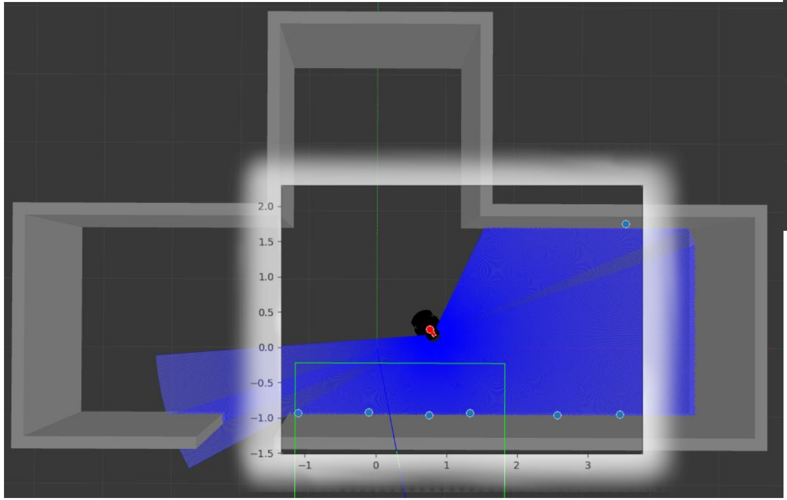


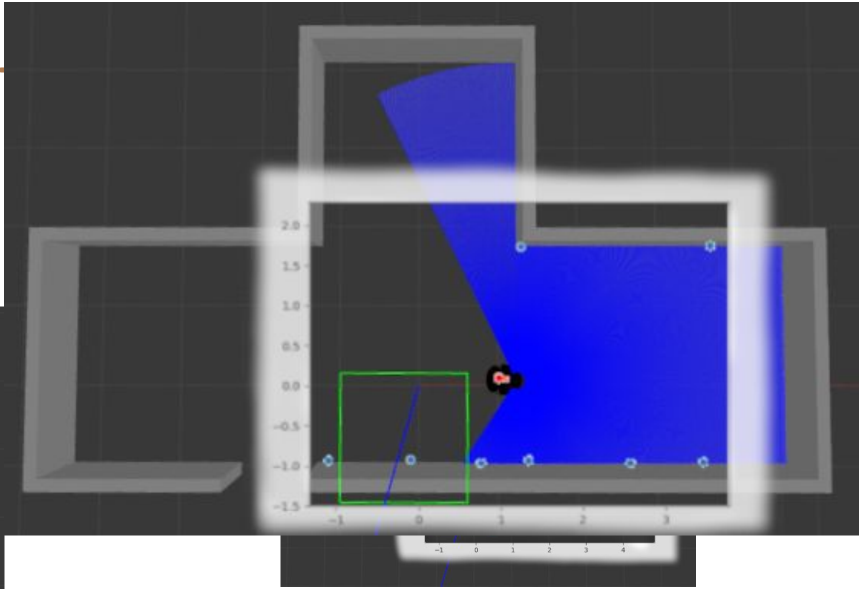


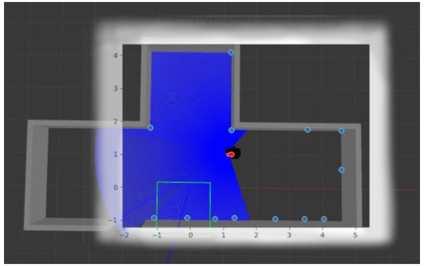


Microsimulation resul

Using Gazebo to run and test our model

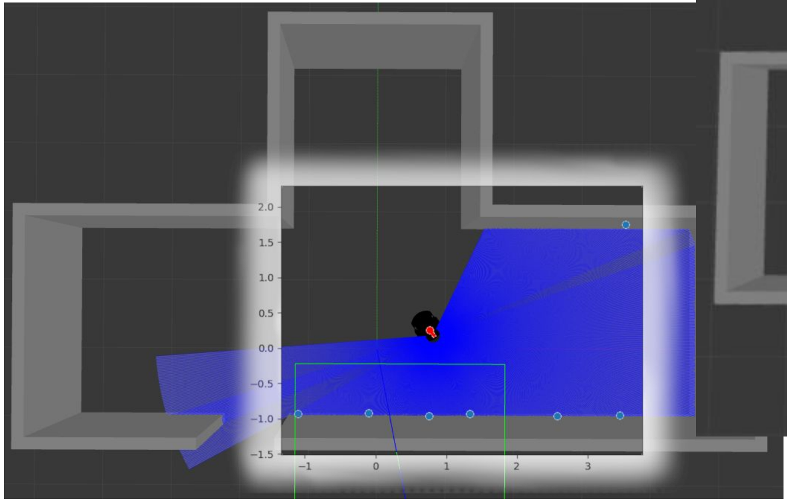


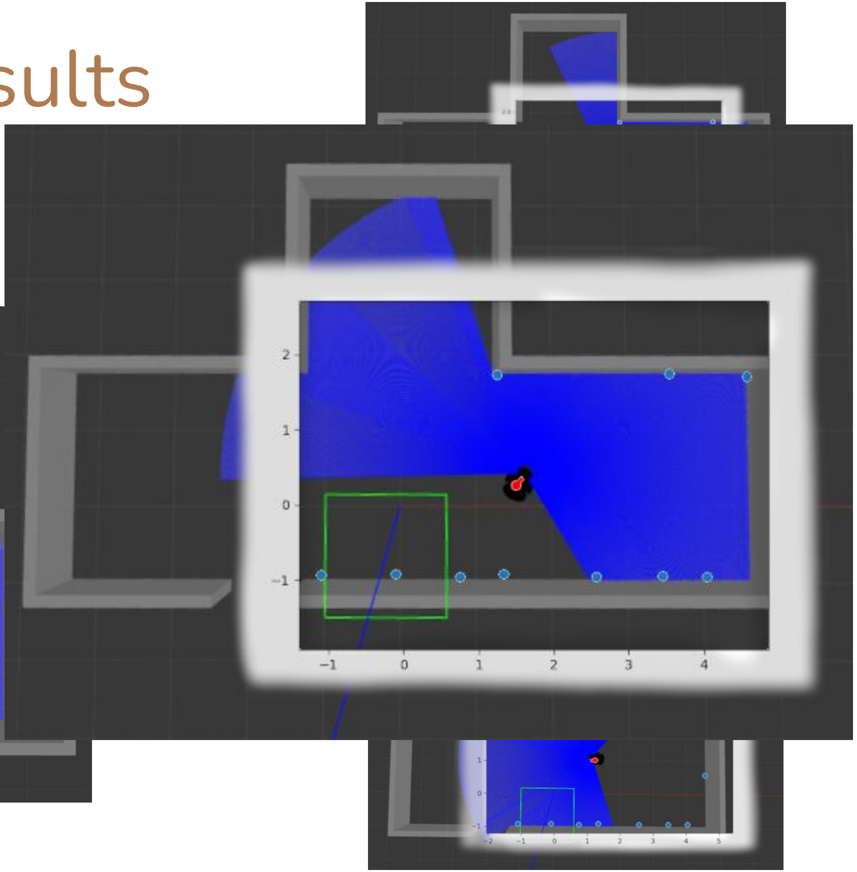




Microsimulation results

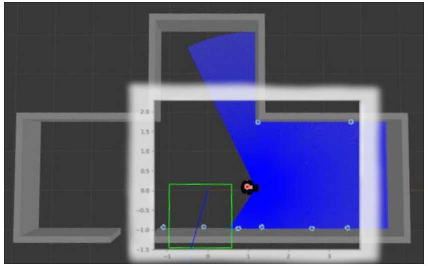
Using Gazebo to run and test our model

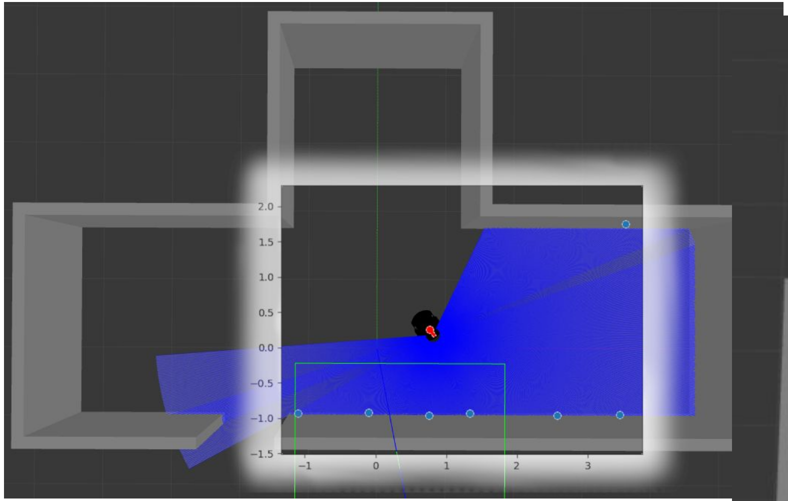


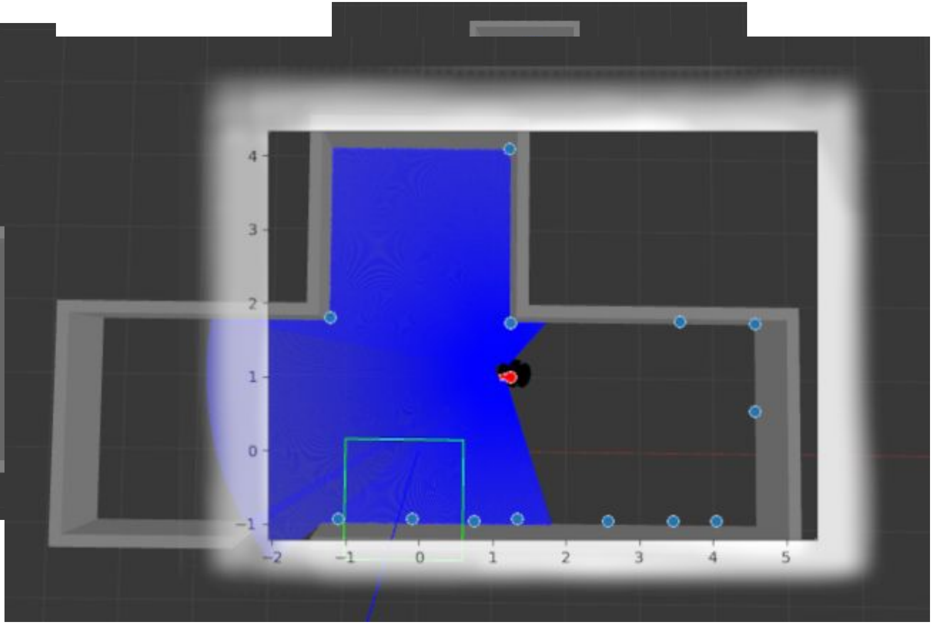


Microsimulation results

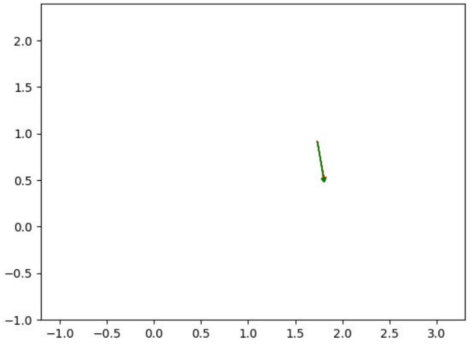
Using Gazebo to run and test our model





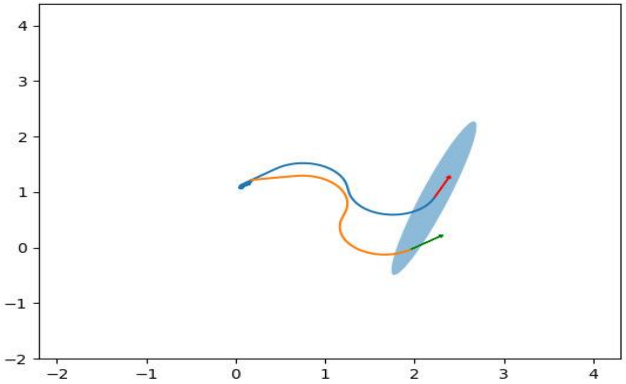


Other results

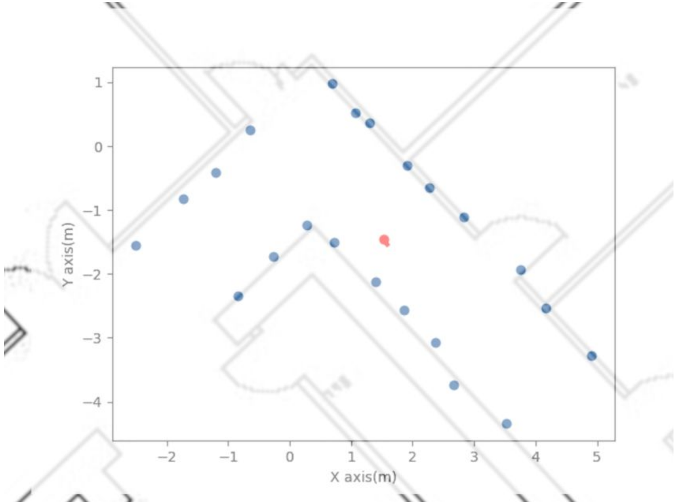


The pose prediction of the prediction method with noise influence (blue **path,** red **arrow**) and the truth position and orientation (orange **path,** green **arrow**).

The covariance matrix of the predicted pose can be represented as an ellipse



Experimental results



Results

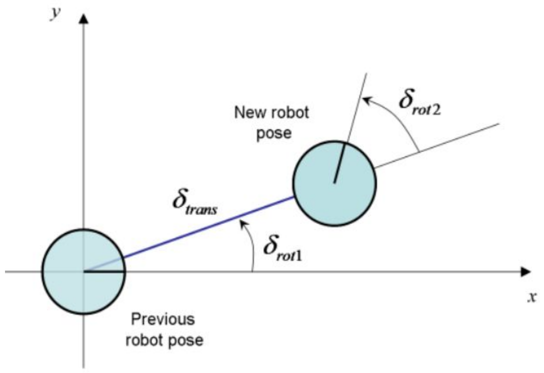
- The algorithm is stable even after approximately 30 cycles.
- Seems to represent correctly the state vector
- Correction step is too slow (can take up to 40 seconds if there are 20 landmarks stored)

Future improvements

- Change our **Hough transform** algorithm:
  - Find a way of **automating** the choice of the **optimal parameters**.
- Optimize the measurement update algorithm:
  - Only apply the update with the landmarks **near the robot.**
  - Re-analyse whether **using lines** instead of corners could benefit **landmark matching**, reducing the computational load.

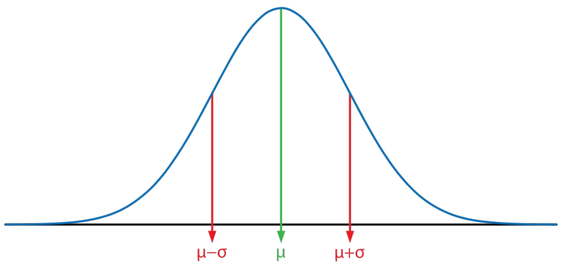
Thank you!

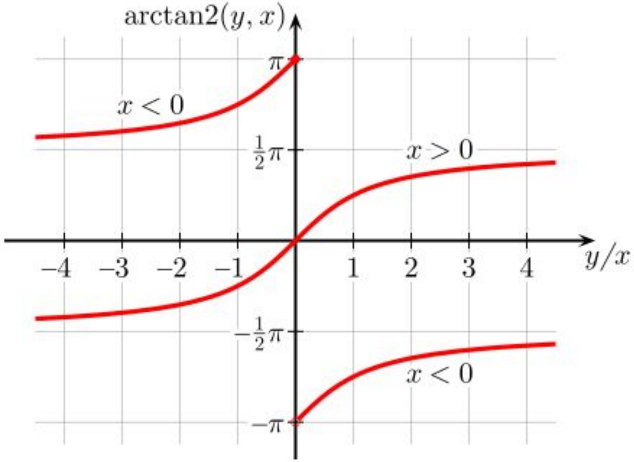
Adding noise to the simulation

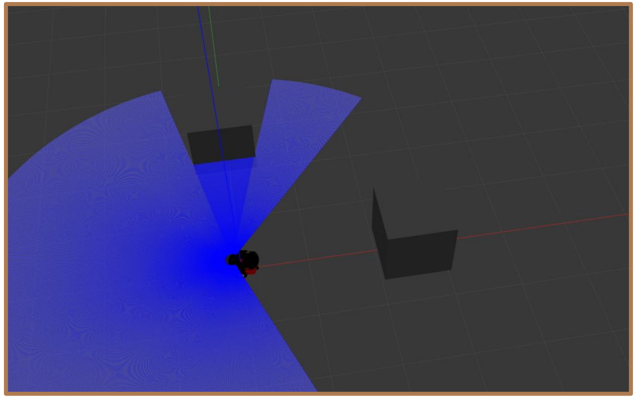


$$\begin{cases} \sigma_{rot1} = \alpha_1|\delta_{rot1}| + \alpha_2|\delta_{trans}| \\ \sigma_{rot2} = \alpha_1|\delta_{rot2}| + \alpha_2|\delta_{trans}| \\ \sigma_{trans} = \alpha_3|\delta_{trans}| + \alpha_4(|\delta_{rot2}| + |\delta_{rot3}|) \end{cases}$$

$$\begin{cases} \varepsilon_{rot1} \sim \mathcal{N}(0, \sigma^2_{rot1}) \\ \varepsilon_{rot2} \sim \mathcal{N}(0, \sigma^2_{rot2}) \\ \varepsilon_{trans} \sim \mathcal{N}(0, \sigma^2_{trans}) \end{cases}$$




arctan2(y, x)
π
x < 0
½π
x > 0
−4 −3 −2 −1 1 2 3 4
y/x
−½π
x < 0
−π



Covariance tests