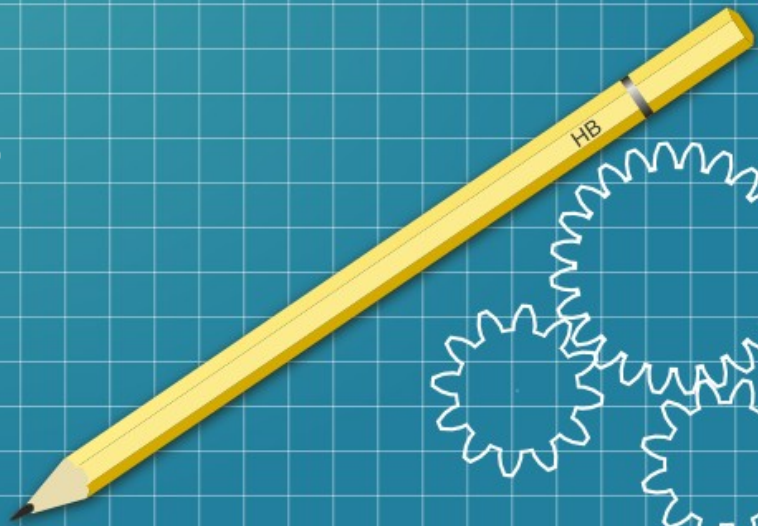
Last updates on Master Thesis

Lidia Fargueta Pelufo



Objectives set

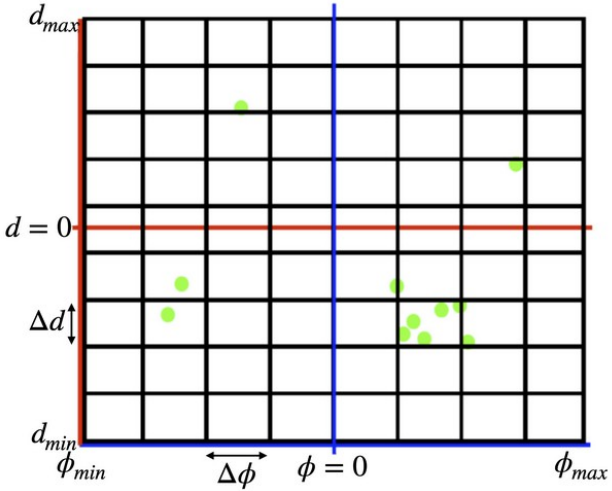
- Try the IMU sensor → **Done**
- Track the coordinates of the robot while it follows the lane → **Done**
- Move the robot from coordinate to coordinate (given array of coordinates). → **Done**
- Scale the path prediction in an image to real coordinates and make the robot move through these.
- Try the second algorithm (TransPath .vs. Neural A*)

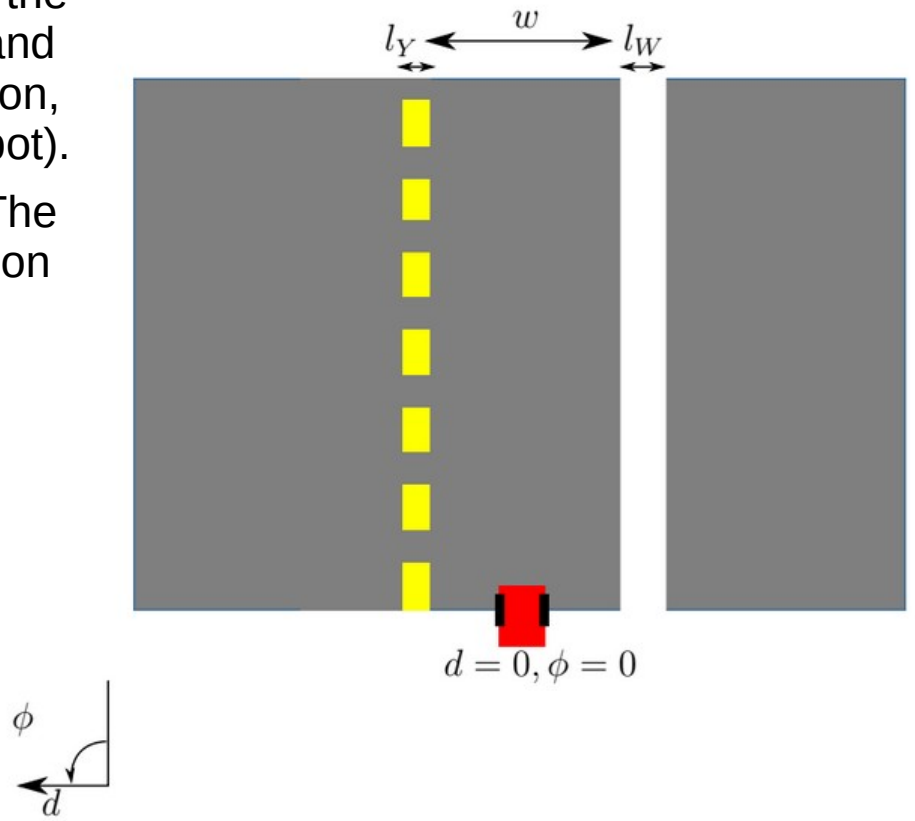
Using IMU

- The IMU gave us the linear and angular acceleration and the orientation. The objective was to integrate the linear acceleration to get the speed and, then, integrate the speed to get the position. Unfortunately this takes so many ideal assumptions, such as that the gravity acceleration is exactly at position z and doesn’t involve other axis and discard any possible noises such as vibrations on the surface where the robot is moving or others.
- The conclusion was that, after displaying the calculated position there was a huge amount of noise, so I discarded using the IMU as a sensor for tracking position.

Track coordinates while following lane

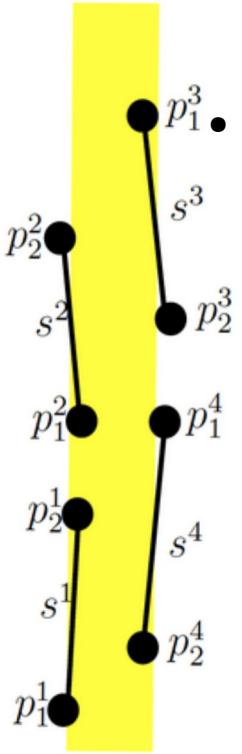
- When executing the demo for lane following, the robot tracks the position of the lines (yellow and white) with respect to itself (so, relative position, assuming the center of coordinates is the robot).
- In order to do this, it uses a histogram filter. The histogram is a 2D grid representing the position and orientation of the robot.





Track coordinates while following lane

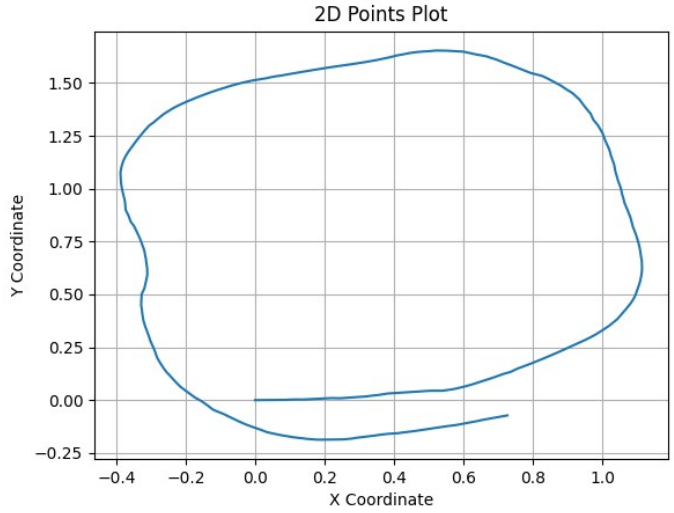
- Motion model:
  - Calculate Odometry Increments (Δd and Δθ) (by using delta time and wheel encoders).
  - Propagate each centroid forward. Loop through each cell and calculate the current position and orientation of the cell.
  - Accumulate the Mass in the New Belief Grid.
  - Apply Gaussian Blur to Add Noise.
  - Normalize the Belief.



- Measurement Model:
  - We have a list of segments. Based on it we estimate the position.
  - We combine the prior belief and measurement likelihood to get the posterior belief:
    - Posterior_belief = Belief *measurement_likelihood

Track coordinate while following lane

- We decided to use the wheels encoders to track the position of the robot while following the lane. When following the lane the robot linear and angular velocity, directly related with the encoders, are selected depended in the calculated d and θ. So the initial thought was that, indirectly we were using a histogram filter for the calculation of the position by using the wheel encoders and that this way it would be more precise (robot doing a circuit loop):



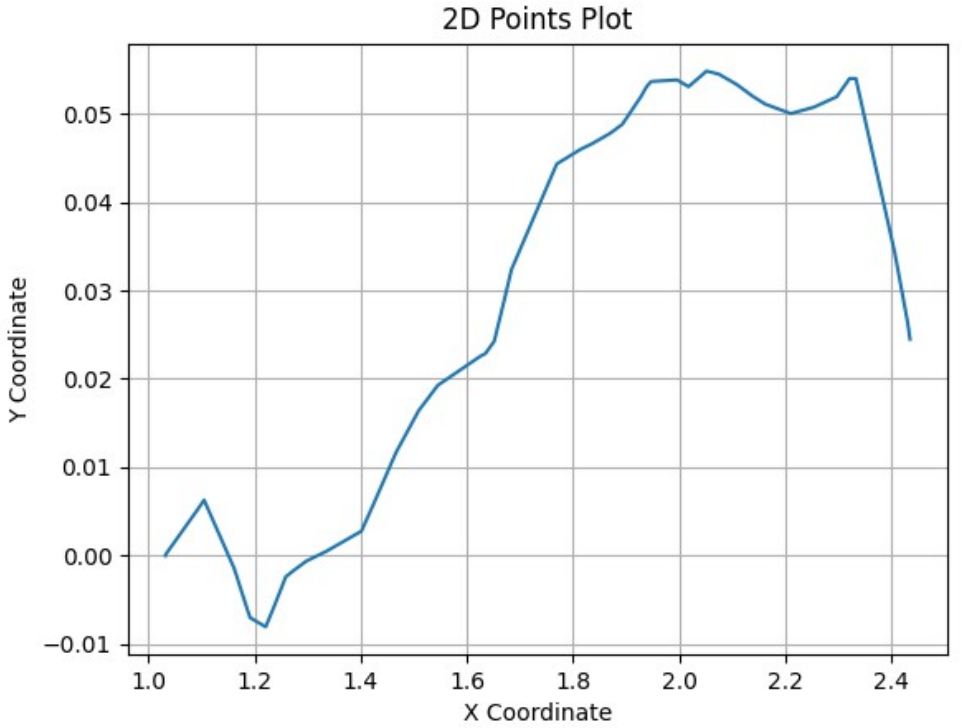
- There is always going to be an error. When trying to close a loop (begin and end position should be the same) this error is appreciated. But in comparison to previous results, this error looks not such a problem.

Some observations of the wheel encoders

- The ground can make the wheels slip. If we test it leaving the robot just move on the black surface, the error is less than if the robot moves through an irregular surface (through white and yellow stickers).
- When we move the robot manually (by using the arrows of the keyboard) we don’t get a good representation, probably because the autonomous driving has a more precise movement and there are more abrupt and strong changes in speed and orientation when we do it manually.
- The encoders are initialized at 0,0 when the robot is turned on, but from then on it keeps accumulating ticks.

Some observations of the wheel encoders

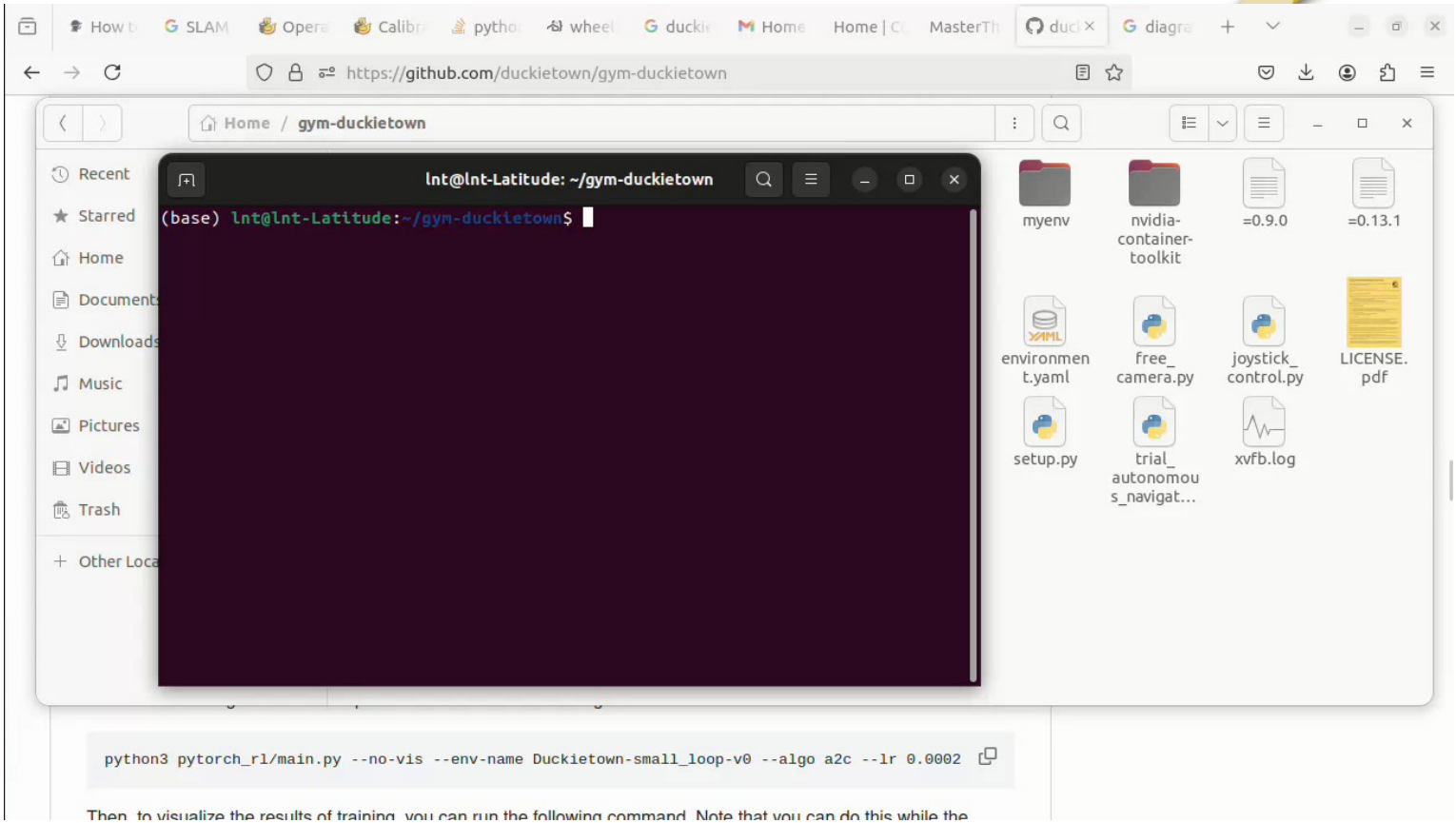
- We moved a robot on a straight line for approximately 1.5m to check what was the estimated error. It was approximately 5 cm max. Unfortunately this error keeps accumulating when the robot makes more complicated movements (turns, backwards, etc).



Move the robot from coordinate to coordinate

- As the output of the path planning algorithm is an array of coordinates, we want for the robot to be able to move from one coordinate to another. The idea is to calculate the distance from the current position to the first point of the array, then, the orientation vector and this way track the linear acceleration and the angular acceleration based on the robot orientation towards this point. We already got good results in simulation previously, so now the objective is to make it work in real life, which is always more complicated.
- The points to move will be [0,0], [1,0], [1,1] and [0,1], so a square if 1m x 1m.

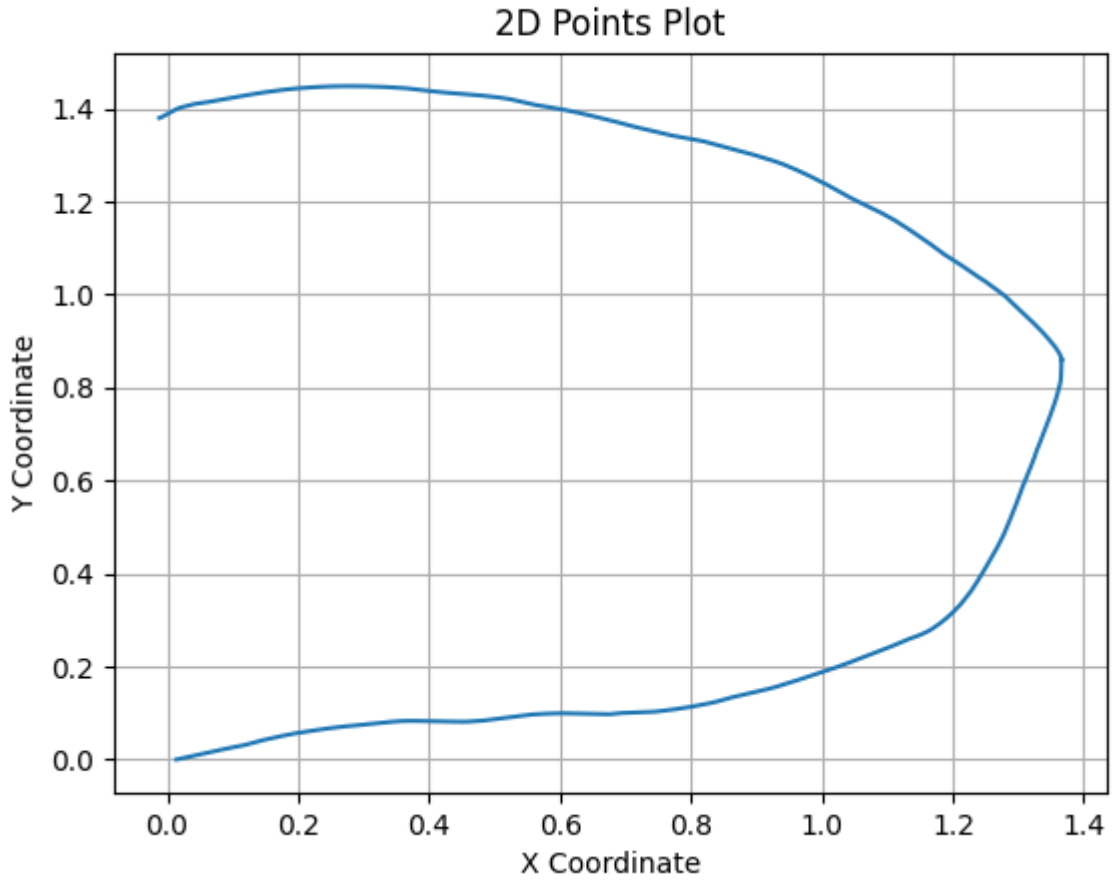
Simulation results



Real life results

- I attached a video in the mail.

Graphic of the registered coordinates



Questions:

- Is this error okay? Should I proceed by fusing together everything (so, do the path prediction for a circuit and make the robot moved through the predicted coordinates)?
- Is the lane following with coordinates useful for something? Should the robot follow the lane for the predicted path algorithm? Do I build a more complex circuit (add an intersection, so that in the path planning it has to decide the root, not just follow lane)?

End