ÉCOLE NATIONALE SUPÉRIEURE DE TECHNIQUES AVANCÉES BRETAGNE
FORMATION D'INGÉNIEUR SOUS STATUT ÉLÈVE

BERNARDO HUMMES FLORES

Visual Robot Localization Using Floor Tiles

Report presented in order to obtain a partial grade
in the Projet discipline

Prof. Dr. Luc JAULIN
Advisor

Prof. Dr. Fabrice LE BARS
Coadvisor

Brest
05 2021

CONTENTS

1 INTRODUCTION

Robot localization is an active field in mobile robotics research, being essential for the safe and correct implementation of autonomous and remotely controlled missions, where many tasks heavily dependent on information about the robot's position. In this context, progress is constantly being made on lowering costs of sensors and technology necessary for robotic operations, and methods based on camera sensors are becoming more and more common, including in this project, as advancements have made them considerably cheap when compared to other alternatives joined by the amount of information available from them.

Within the scope of those methods, probabilistic ones are frequently used as they offer an approach that is closer to the usual understanding used in other areas, with many concepts that can be re-purposed for the different applications. However, there exists also the interval analysis approach, which offers guaranteed approximations where probabilistic ones may diverge, such as particle filters, and are capable of dealing with non-linearities that traditional methods would struggle to operate without too much compromises, such as would be needed for a Kalman Filter.

In this work a method for localizing a robot equipped with a single downwards facing camera is presented, using interval methods to deal with the hardly linearizable constraints that are presented when considering the tiled floor environment. The source code with instructions for execution and further development is available in the following address: https://github.com/birromer/tiles-localization

A general overview of the project is done at chapter 2, followed by the theoretical background necessary for understanding the method in chapter 3. Chapter 4 shows the details in implementation, such as the robotic architecture, the image processing and the localization algorithm. Some analysis of the problems and results encountered are presented in chapter 5. Finally, some words of conclusion are presented in chapter 6.

2 GENERAL

2.1 Overview

This project had as objective the development of an algorithm capable of localizing a robot equipped with a single camera facing the floor, using as well an estimation of its movement. The following elements were assumed true for the execution of the tasks:

1. The environment's floor consists of identical looking tiles;
2. The robot is equipped with a downwards facing camera;
3. The input **u**, composed of the heading and speed, is available;
4. The parameters **y** can be extracted from the tiles;
5. Localizaton is possible by merging the odometry data with the estimated tile parameters.

2.2 Starting Material

As a continuation of a previous internship at ENSTA Bretagne, some materials were already made available from the start. Most of it came from the first student to tackle with the problem, Bertrand Turck, and consisted of his internship report and the source code of his implementation. Even though those files were available, it is important to point that the robot's motion was not taken into consideration for the localization, and ultimately his algorithm did not work.

A considerable help also came from professor Luc Jaulin, who oriented both iterations of the projects and provided the mathematical basis for the method. He contributed with an article explaining how he envisioned the method to work and python scripts with starting examples.

2.3 Objectives

For the development of this project, the original idea was to correct the previous implementation, getting it to work on the real robot and perform extensive testing for

improving the method, however several problems were encountered on the way and the objects shifted to the ones presented in the following list:

1. Study and understand the mathematical background;
2. Study and understand the existing implementation;
3. Study of the improvement of the existing implementation;
4. Refactoring of the existing architecture;
5. Re-implementation of the method according to new understanding.

There were roughly 12 weeks available for the project, and the usage of the time was done as presented in the following table. It is also relevant to note that many activities overlapped and some of them were returned to after a while, but this context switching is not displayed and only a procedural development of the work.

Table 2.1: Usage of the weeks available for each of the activities

	1	2	3	4	5	6	7	8	9	10	11	12
1	X	X										
2		X	X									
3			X	X	X							
4					X	X	X	X	X			
5								X	X	X	X	X

3 THEORETICAL BACKGROUND

3.1 Interval Analysis

The usage of intervals to represent uncertainty guarantees that results will not diverge, something common in probabilistic methods. They are also capable of handling heavily non-linear systems and make available interesting operators that will be used in this work.

The non-linearities don't have to be simplified in order to be processed and this is one of the key reasons interval analysis has been chosen for this project, as the tiles parametrization, dealt in section 3.2, creates a series of non-linear constraints that would be hard to solve otherwise.

Using this approach also makes available the contractors, which are operators that remove unfeasible solutions according to a set of constraints, never removing a valid one. This operator will be used for the localization algorithm.

3.2 Tiles

The application scenario comprehends a floor composed of tiles, regular sized squares with uniform distribution and clear delimitation marks.

3.2.1 Parametrization

Tiles may be represented by the sets satisfying $T_0(a_1, a_2) = \sin \pi a_1 \cdot \sin \pi a_2 = 0$.

They may have a set of parameters $\mathbf{y}$ extracted from a view, and they correspond to the horizontal displacement, vertical displacement and rotation applied to them, respectively. If those parameters are applied in reverse on the image, the original orientation and origin centered tiles would be the result. Because of multiple ambiguities on their view, the domains are defined as follows, $y_1, y_2 \in [-1/2, 1/2]$ and $y_3 \in [-\pi/4, \pi/4]$. With those, the tiles may be defined as

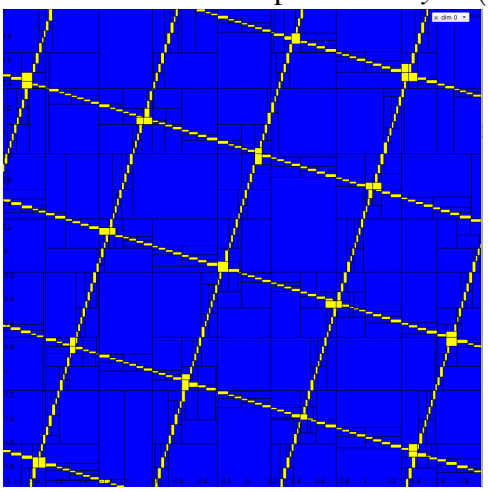
$$\mathbb{T}_y = \{a | T_y(a) = 0\}$$

where

$$\mathbb{T}_y(a) = \mathbb{T}_0 \begin{pmatrix} y_1 + a_1 \cos y_3 - a_2 \sin y_3 \\ y_2 + a_1 \sin y_3 + a_2 \cos y_3 \end{pmatrix}$$

Which can be illustrated in the following figure:

Figure 3.1: Frame with associated parameters $y = (0.1, 0.2, 0.3)$.



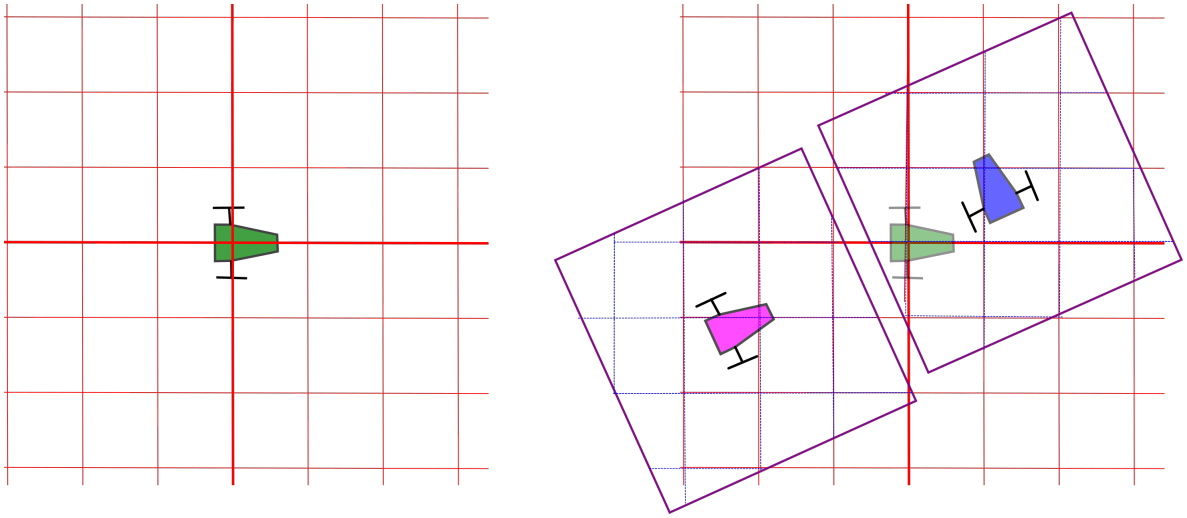
3.2.2 Equivalency

Multiple views of the tiles may generate parameters that are considerably close, and an equivalence relation can be established such as:

$$\mathbf{y} \sim \mathbf{z} \Leftrightarrow \mathbb{T}_\mathbf{y} = \mathbb{T}_\mathbf{z}$$

This equivalence can be seen in the following image.

Figure 3.2: The blue and pink robots see the same frame.



Because of that, an equivalency between two sets of parameters **z** and **y** can be expressed as follows:

$$y \sim z \Leftrightarrow \begin{cases} \sin\pi(y_1 - z_1) = 0 \\ \sin\pi(y_2 - z_2) = 0 \\ \sin(y_3 - z_3) = 0 \end{cases} \quad or \begin{cases} \sin\pi(y_1 - z_2) = 0 \\ \sin\pi(y_2 - z_1) = 0 \\ \cos(y_3 - z_3) = 0 \end{cases}$$

As mentioned in section 2.2, this mathematical intuition was a contribution of professor Jaulin.

4 IMPLEMENTATION

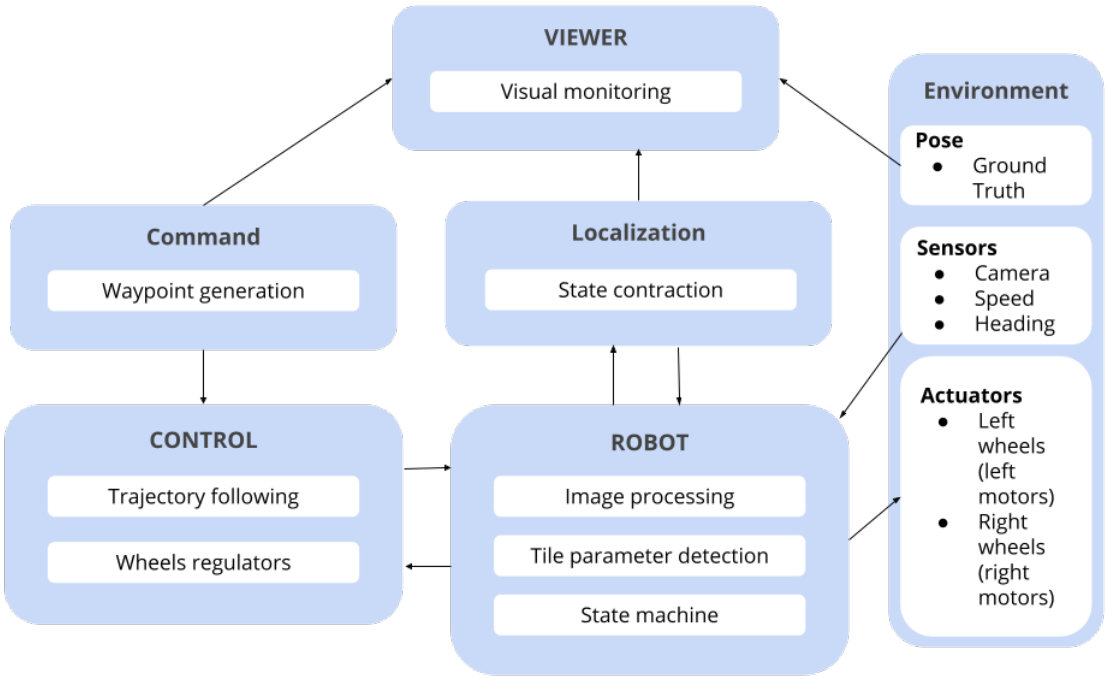
4.1 Architecture

The project has been re-implemented with a clearer organization of the responsibilities between the nodes and from the new understanding of the problem, which can be seen in the following architectures. The robot node was thought to center most of the connections, making the interface between the environment (simulation) and the multiple parts of the robot. The commands sent to physical robot come from a controller node. The true pose, the waypoint and the estimated state are sent to a viewer node for better visualization of what is happening.

4.1.1 C2 Architecture

The image below represents what has been implemented of the project, reflecting what was mentioned at the begining of the section.

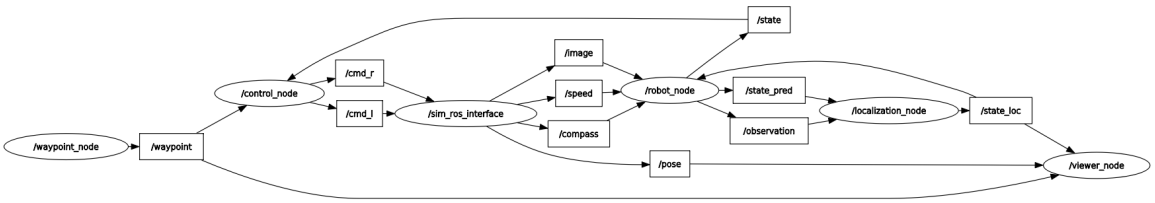
Figure 4.1: Architecture of the developed system.



4.1.2 ROS Architecture

Custom messages were added in order to represent an interval state $\mathbf{X} \in \mathbb{I}^3$, observation $\mathbf{y} \in \mathbb{R}^3$ and the input $\mathbf{u} \in \mathbb{R}^2$. The same implementation as above can be seen in the following node graph.

Figure 4.2: Node graph of the developed package.



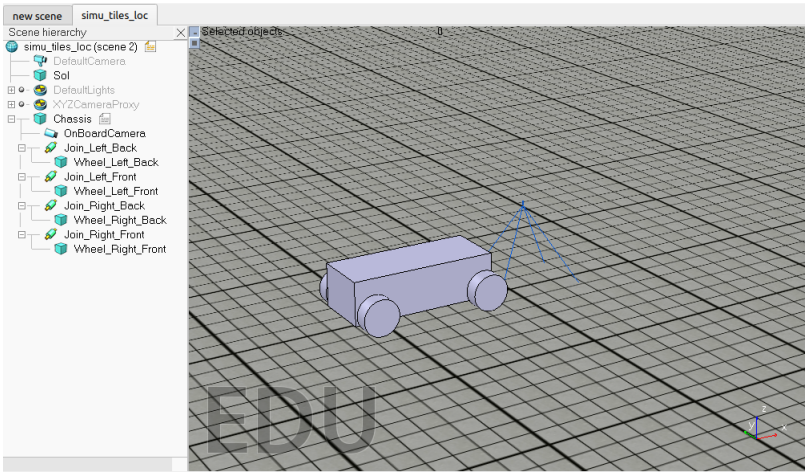
4.2 Robot Representation

4.2.1 Simulation

A simulation was provided from the previous student's work, and it has been adapted to better suit the project. Topics were added to provide the compass and the speed of the robot, as now the movement is considered to improve the localization. Those values could be estimated from an optical flow method or additional sensors, but as that is not the topic of the project, they were simply extracted from the simulated environment.

The simulated environment can be seen below.

Figure 4.3: Simulation of the robot on a floor with tiles.



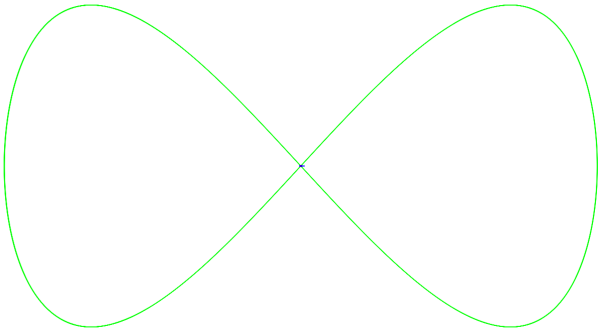
4.2.2 State Equations and Movement

As mentioned before, the current speed and heading of the robot are extracted from the simulation. This lets us use the simple Dubin's car model to describe the movement:

$$\begin{cases} \dot{x}_1 = u_1 \cos x_3 \\ \dot{x}_2 = u_1 \sin x_3 \\ \dot{x}_3 = u_2 \end{cases}$$

A target setpoint is generated for every time t, attracting the robot to follow a Lissajous curve, as represented below.

Figure 4.4: Lissajous curve used as setpoint over time.



A PD control was implemented relative to the error between the orientation of the robot and the line connecting the robot and the setpoint. The robot was controlled via a differential steering, where each pair of wheels on each side had a different input.

4.3 Image Processing

The image processing has been divided in two main steps: the pre-processing of the image, for removing noise that may cause problems for the line extraction, and the parameter estimation for the remaining lines.

4.3.1 Pre-Processing

During the pre-processing step, multiple filters are applied on top of the image, as described below:

1. Conversion to greyscale;
2. Derivative computation for enhancing limits;
3. Canny filter for edge detection, creating a binary image;
4. Morphology operations for cleaning noise and merging close borders belonging to the same line;
5. Hough transform for line detection;
6. Horizontal and vertical line detection and orientation filtering;

Steps 1, 2, 3 and 5 are performed via simple filters, already implemented in OpenCV, cvtColor, Laplacian, Canny, HoughLineP (the probabilistic version has better results), respectively. Step 4 was done via the application of a closing operation followed by a dilatation. Step 6, which is part of both the pre-processing and the parameter extraction, removes the lines that have an orientation too far away from the median of all lines.

The result of the steps can be seen in the figure 4.5.

4.3.2 Parameter Extraction

The parameter **y** is extracted primarily by information on bags of vertical and horizontal lines.

Because of the fact that all angles that are rotations of $\pi/2$ of one another, representing the same parameters, an equivalency between two angles can be seem with

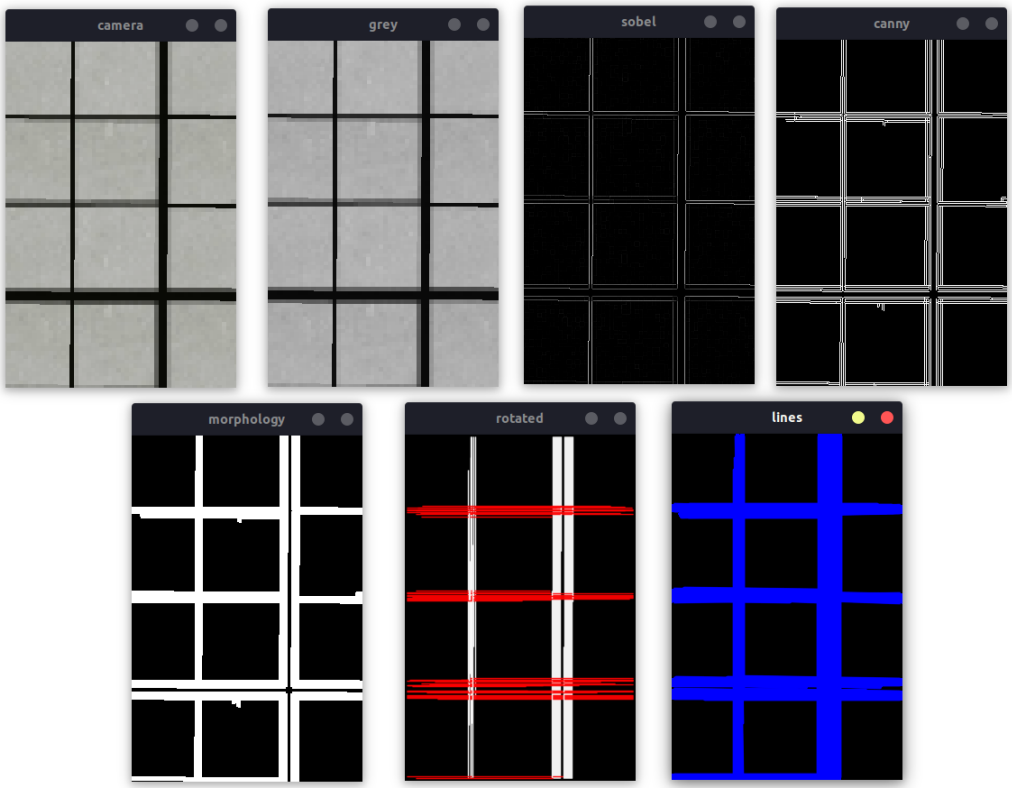
$$\begin{cases} cos\ 4\alpha_2 = cos\ 4\alpha_1 \\ sin\ 4\alpha_2 = sin\ 4\alpha_1 \end{cases}$$

which leads to the disambiguation function

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} cos\ 4\alpha_i \\ sin\ 4\alpha_i \end{pmatrix}$$

With the cloud point composed of x_i and y_i we can then get the angles of the lines via

Figure 4.5: Steps of pre-processing applied in the incoming image. First row: original image, grayscale, laplacian filter, canny filter. Second row: morphology operations, horizontal/vertical detection, final lines detection.



$$\hat{\alpha} = \frac{1}{4} \arctan 2(\hat{y}, \hat{x})$$

Those lines can be separated according to their angle into a bag of vertical and horizontal lines, if the cosine or sinus are close to 0, respectively.

Finally each of the bags can generate a displacement of the tiles view by taking the median of the displacement between lines, normalized by the size of the tiles (l), as bellow:

$$\hat{d}_{horizontal} = 1 \cdot \text{median}(\tfrac{i}{l} - \text{floor}(\tfrac{d_i}{l})$$

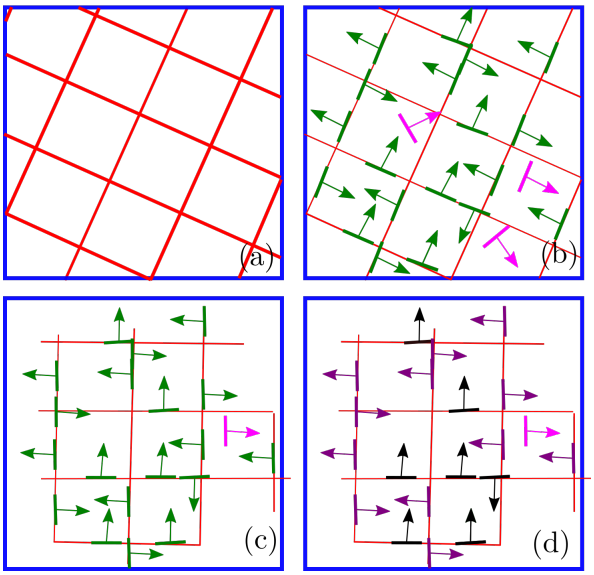
with

$$(d_i, \alpha_i) \in \text{horizontal lines}$$

The same is repeated for vertical lines and the parameters are found as in $(\mathbf{y} = (\hat{d}_{horizontal}, \hat{d}_{vertical}, \hat{\alpha})$

This process and the image were suggested by professor Jaulin as well and modifications are still being tested for performance.

Figure 4.6: Steps of parameter extraction. (a) lines seen; (b) orientation computation (c) and (d) orientation filtering, correction and bags formation.



4.4 Localization

The localization is done in two main steps: the state prediction and the state contraction. The first one simply evolves the model of the robot according to the state equations and the available measurements of speed and orientation, while the second one contracts the new state according to the constraints from the tiles equivalencies.

4.4.1 Contractor Network

A contractor network is created using the equivalency relation explained in the item 3.2.1. The network receives three dimensional interval vector with the current predicted state and the parameter vector from the tiles view. As the observation from the robot is equivalent to its state, both elements are inserted as the vectors **z** and **y**, as explained in 3.2.2.

4.4.2 State Estimation

As mentioned before, the first step of the localization is the state prediction using the state model explained in 4.2.2, which does not provide a reliable estimation as noise is present and the quality of the values may vary. In the implementation, a Gaussian noise

is added to every sensor and the state is always inflated by a factor to consider the added uncertainty.

This process is better depicted in the following algorithm.

Algorithm 1 Main loop for the state estimation

Result: Estimation of the current state of the robot.

```
while true do
    state ← integrate the state in order to evolve it
    parameters ← process incoming image
    state ← contract state checking equivalency between it and the parameters
    if state is not empty then
        send updated state
    else
        skip cycle
    end
end
```

5 ANALYSIS

5.1 Current Situation

At the end of the writing of this report, the localization algorithm does not work, but many objectives have been completed. The entire project has been rewritten, as the previous implementation did not work and multiple details weren't clear or documented.

5.1.1 What works

Considering the previously stated objectives in section 2.3, multiple of them have been reached, such as the full understanding of the mathematical specification and objectives, the entire infrastructure in a ROS package, and part of the image processing.

The localization method has been implemented into a ROS package, with custom messages for the interval state, observation and command to the robot. The nodes have been divided according to their functionality, centering the operations around a robot node, as described in section 4.1. The image processing part related to the pre-processing of the tiles view seems to work, as the resulting lines rarely present any aberrant orientations. The quality of the parameters generated is still being investigated, but the equations presented in 3.2.2 allow us to see how similar are the observation to the predicted state, and they are being used for such.

5.1.2 What does not work

First, the localization does not work, as the estimated parameters and the predicted state do not have any intersection when the constraints are applied, and an empty set is always achieved, meaning that the robot is always lost. That behaviour is believed to come from incorrect parameters, as the state model is quite simple and has already been re-implemented to fit the robot's sensor's acquisitions. It is also important to point that the development has not left the simulation, although the Saturn robot is available for later experimentation.

5.2 Problems found

Multiple problems were found at many steps of the development, starting from the understanding of the original project. Some documentation had been made available, but unfortunately there were parts of the code that had little to no explanation and some extra time had to be used for understanding it and later re-implementing it. The usage of a camera as sensor lead to the necessity of dealing with the quality of the image, as some noise was present (and even introduced, in order to have a scenario closer to real life). Also, with the image processing using the OpenCV library, multiple functions needed were already implemented, however a considerable time was spent optimizing parameters for the correct results that were expected from their usage.

Finally, the parameter extraction was problematic, as initially a hybrid implementation was done, reusing part of the previous project's code, and it was later completely re-done. However, the values presented in the equivalency equations show that they do not match, succeeding only in finding the correct orientation.

5.3 Future Work

In order to finish the project, the parameter estimation and state evolution must continue to be debugged, as their contraction is probably the source of he problem. Multiple articles were also found on powerline detection, a problem that seems fairly similar to the tiles and there are methods to be tested for its detection and tracking. Also with the research found, more constraints may be added to the contractor network that could improve results, such as the ones relating the state evolution equations and the ones linking the lines to their predicted position, considering that the movement is sufficiently well predicted.

6 CONCLUSION

The work done on this project has been very interesting as it tackled a very educational case where the usage of interval analysis is evidently beneficial, considering the heavily non-linear constraints that the tiles pose for the system and the ease for representing the problem. An interest has been sparked as well on the functioning of algorithms in real world conditions, as the simulation offered too perfect of a world and several noises were added in order to find a more accurate representation.

An implementation of the work done is made available at the repository indicated at the beginning, where branches started by "release_" contain the stages of development in between new attempts of fixing the method. At the moment the master branch contains the latest development stage.

7 BIBLIOGRAPHIE

[1]P. Bao, L. Zhang, and X. Wu, “Canny edge detection enhancement by scale multiplication,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 9, pp. 1485–1490, Sep. 2005, doi: 10.1109/TPAMI.2005.173.

[2]W. Gao, X. Zhang, L. Yang, and H. Liu, “An improved Sobel edge detection,” in 2010 3rd International Conference on Computer Science and Information Technology, Jul. 2010, vol. 5, pp. 67–71. doi: 10.1109/ICCSIT.2010.5563693.

[3]L. Jaulin, Mobile robotics. 2015. Accessed: Jul. 17, 2019. [Online]. Available: http://www.sciencedirect.com/science/book/9781785480485

[4]P. Mukhopadhyay and B. B. Chaudhuri, “A survey of Hough Transform,” Pattern Recognition, vol. 48, no. 3, pp. 993–1010, Mar. 2015, doi: 10.1016/j.patcog.2014.08.027.

[5]T. Santos et al., “PLineD: Vision-based power lines detection for Unmanned Aerial Vehicles,” in 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Apr. 2017, pp. 253–259. doi: 10.1109/ICARSC.2017.7964084.

[6]B. Turck, “Localisation d’un robot à partir de la mesure des carreaux d’un carrelage,” p. 27.

[7]J. Zhang, L. Liu, B. Wang, X. Chen, Q. Wang, and T. Zheng, “High Speed Automatic Power Line Detection and Tracking for a UAV-Based Inspection,” in 2012 International Conference on Industrial Control and Electronics Engineering, Aug. 2012, pp. 266–269. doi: 10.1109/ICICEE.2012.77.

[8]“Image Transforms - Hough Transform.” https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (accessed Apr. 23, 2021).

[9]X. Wu, M. Xu, and L. Wang, “Differential speed steering control for four-wheel independent driving electric vehicle,” in 2013 IEEE International Symposium on Industrial Electronics, Taipei, Taiwan, May 2013, pp. 1–6. doi: 10.1109/ISIE.2013.6563667.