Robotics, Lab 2

Magnus Knaedal, Francisco Sena Esteves, Martim Braga

Abstract—Development of a controller for the pioneer, based on the technique of Lyapunov stability analysis, adapted to a specific path using sensors to detect the state of doors. Obtained a accuracy of 95% in door position and state detection.

1 Introduction

THIS report addresses the work done in Lab 2, in the course Robotics at Instituto Superior Técnico. The main objective is to develop creativity related to the use of mobile robots, and become familiarized with problems related to sensing, actuating, and controlling a robot while executing a mission in an a prior known environment. The work is divided mainly in two parts: path following control of unicycle type kinematic model- and detection of doors.

Firstly, the two main parts and other methods used in the project will be described. Secondly, different experiments performed to test the implementation will be presented. At last, the results, a discussion, and a conclusion will be given.

2 Materials and setup

The Pioneer P3-DX robot (pioneer) has been used in this lab, and is shown in figure 1. It is a nonholonomic mobile robot with unicycle kinematics. Measurement of the pioneer necessary for implementation has been carried out by inspecting the pioneer itself directly in the lab. The pioneer provides the odometry (x, y, θ), and is controlled by setting the respective translation and angular velocity (v, ω). The pioneer is equipped with built in sonar, and a Hokuyo URG-04LX laser range finder (lrf) attached on the top of the robot 1. The laser has a $240°$ scan, with a $0.35°$ resolution. This implies that each laser scan will consist of a set S containing 682 measurements. The laser will be used for correction of odoemtry and detecting doors. The implementation is done in Matlab, with provided code for sensor reading and actuation of the described material.

- *Magnus Knaedal, nr. 91984,*
  E-mail: magnuok@stud.ntnu.no,
  Instituto Superior Técnico, Universidade de Lisboa.
- *Francisco Sena Esteves, nr. 78468,*
  E-mail:franciscosenae@tecnico.ulisboa.pt,
  Instituto Superior Técnico, Universidade de Lisboa.
- *Martim Gard Braga, nr. 78263,*
  E-mail: martim.gard.braga@tecnico.ulisboa.pt,
  Instituto Superior Técnico, Universidade de Lisboa.

10th of May, 2019

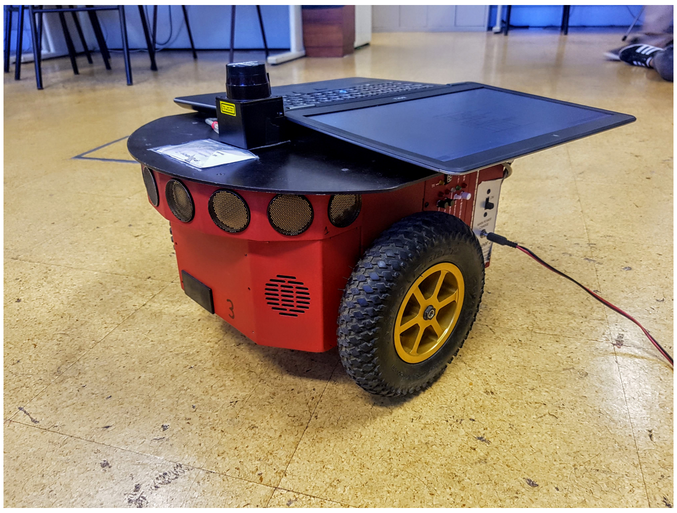

Figure 1: The Pioneer P3-DX robot with built in sonar, and a Hokuyo URG-04LX laser range finder (lrf) attached on the top of the robot

3 Path planning

First of all, a grid map has been made using `gmapping`-package implemented in ROS using the pioneer and the lfr, and a handout sheet of the floor plan [1]. By combining the two, a relatively precise grid map can be created, as we know the resolution in cells per meter from the map created by the `gmapping`-package and the exact measurements from the floor plan. The map is visualized in Figure 2.

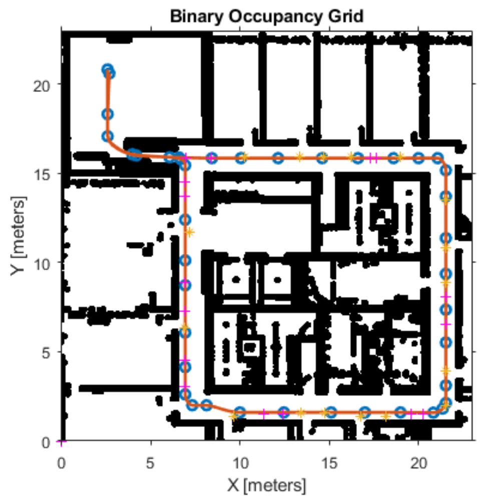

Figure 2: Grid map, where the blue dots denotes the interpolation points for creating the path, red line the actual path, and yellow stars the doors.

Using the built-in Piecewise Cubic Hermite Interpolating Polynomial in Matlab, a smooth path can be created [2]. The path is visualized in Figure 2. Since the pioneer will have initial odometry $(0, 0, 0)$, a transformation of coordinates needs to be done. By setting the first position to $(0, 0)$, and then apply the rotation matrix:

$$T = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi, \end{pmatrix} \tag{1}$$

, where ϕ is the angle relative to the x-axis between the two first elements in the trajectory list created by the interpolation, the path is set to start in $(0, 0, 0)$. ϕ can be found as:

$$\phi = \arctan\left\{\frac{y_r(2) - y_r(1)}{x_r(2) - x_r(1)}\right\}, \tag{2}$$

where x_r and y_r is the respective x- and y-coordinate of the reference trajectory list.

4 Controller

In developing a control strategy that generates the linear and angular velocities, and will make the pioneer follow the trajectory created in section 3, a position tracking approach suggested by Youngshik Kim and Mark A. Minor, has been applied [3]. Only the case for $v \geq 0$ is considered and implemented in this project.

4.1 Kinematics Model

The pose of the robot is (x, y, ϕ), and is attached to the moving robot with center in O, where ϕ is the orientation relative to the x-axis, positive counter clockwise. The reference pose is (x_r, y_r, ϕ_r) and is attached to the moving robot with center in R. Same definition and consideration applies to ϕ_r as to ϕ. Thus, we have the kinematics represented in Cartesian coordinates by:

$$\begin{aligned} \dot{x} &= v\cos\phi \text{ and } \dot{y} = v\sin\phi, \\ \dot{x}_r &= v_r\cos\phi_r \text{ and } \dot{y}_r = v_r\sin\phi_r. \end{aligned} \tag{4}$$

Using a polar representation, the error kinematics of O relative to R, can be represented as:

$$\begin{aligned} e &= \sqrt{(x - x_r)^2 + (y - y_r)^2}, \\ \theta &= \arctan(\frac{y_r - y}{x - x_r}) - \phi_r, \\ \alpha &= \theta - \phi + \phi_r. \end{aligned} \tag{6}$$

A visualization of the parameters are shown in Figure 3. Differentiating equation (5), applying trigonometric identities and substituting (3) and (5) into the differentiated equation, the state equations are obtained as:

$$\begin{aligned} \dot{e} &= -v\cos\alpha + v_r\cos\theta, \\ \dot{\theta} &= v\frac{\sin\alpha}{e} - v_r\frac{\sin\theta}{e} - \dot{\phi}_r, \\ \dot{\alpha} &= v\frac{\sin\alpha}{e} - v_r\frac{\sin\theta}{e} - \dot{\phi}_r. \end{aligned} \tag{8}$$

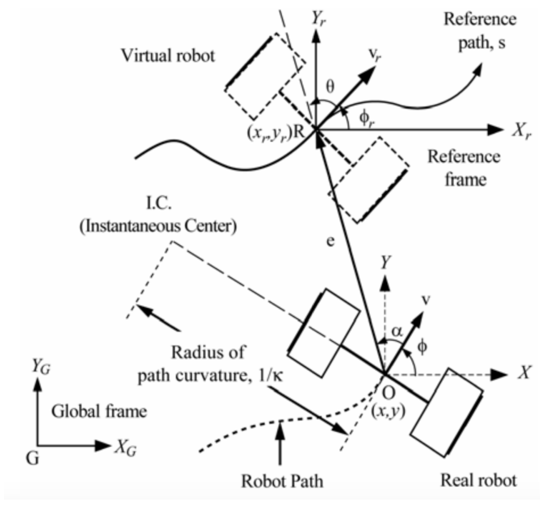

Figure 3: Kinematics for the unicycle robot considered in this project, as suggested by Youngshik Kim and Mark A. Minor [4]

4.2 Lyapunov analysis

The trajectory tracking problem can be solved by applying the well known nonlinear technique of Lyapunov analysis, of the state equations, stated in equation (9). We first consider the case for posture regulation, i.e when the reference coordinates are fixed, and then extend using waypoints to create a trajectory following controller. Posture regulation simplifies equation (5) to:

$$\begin{aligned}\dot{e} &= -v\cos\alpha,\\ \dot{\theta} &= v\frac{\sin\alpha}{e},\\ \dot{\alpha} &= v\frac{\sin\alpha}{e}.\end{aligned} \tag{10}$$

Considering the quadratic candiate Lyapunov function suggested by M.Aicardi et al. [5]:

$$V = \frac{1}{2}(e^2 + k_2\theta^2 + \alpha^2) \tag{11}$$

where $k_2 > 1$, is a constant. The candidate function is positive semi-definite, is smooth in the sense that the partial derivatives exists and are continuous, and $V(0,0,0) = 0$. The derivative of the candidate function with respect to time can be written as:

$$\begin{aligned}\dot{V} &= e\dot{e} + k_2\theta\dot{\theta} + \alpha\dot{\alpha}\\ &= -ev\cos\alpha + k_2\theta v\frac{\sin\alpha}{e} + \alpha(v\frac{\sin\alpha}{e} - \dot{\phi}),\end{aligned} \tag{13}$$

The proposed control law by Youngshik Kim and Mark A. Minor is [4]:

$$\begin{aligned}v &= v_{max}\tanh(k_1 e),\\ \dot{\phi} &= v_{max}((1 + k_2\frac{\theta}{\alpha})\frac{\tanh(k_1 e)}{e}\sin\alpha + k_3\tanh\alpha),\end{aligned} \tag{15}$$

where $k_1, k_3 > 0$. Plugging the control law in equation (16) into equation (12) we obtain:

$$\dot{V} = -v_{max}(e\tanh(k_1 e)\cos\alpha + k_3\alpha\tanh\alpha), \tag{16}$$

which is in fact negative definite for all values, except the equilibrium point $(0,0,0)$ (negative semi-definite). Using Barblat's theorem [6], and the fact that

$$\lim_{t\to\infty}\alpha(t) = 0, \tag{17}$$

the system is stable in the sense of Lyapunov around the equilibrium point. We can thus conclude that the error will tend to zero. For further details of deciding parameters an boundness of controller see Youngshik Kim and Mark A. Minor [4].

4.3 Trajectory following using waypoints

The Lyapunov analysis has been carried out for the posture regulation case. By considering the reference terms in the state equation (9) as disturbance, the controller can be extended to a trajectory following controller. Thus we can define a set of waypoints along our trajectory to follow. By defining a certain threshold, ε, for the position error stated in equation (5), we can constantly generate the next waypoint the controller should follow.

4.4 Correction of position

Several factors causes the odometry to integrate error over time, such as debris being collected on the tires or the two wheels not being exactly the same size. In order to correct the drift in the odometry tracking we take the distance to the wall that the robot reads with the lidar when it detects a door, and comparing it to the actual distance that the path should have. With the calculated error, we can add or subtract it to the paths coordinates. This will make the controller follow the new corrected path and eliminate the errors in the odometry tracking.

The measurements are taken before the doors and also in the spots defined to correct the robot orientation. To calculate the distance to the wall, 85 out of 682 measurements (29.75°) are taken from the desired side. After filtering out undesired values (e.g out of bound values), the minimum of these measurements is considered the distance to the wall. In this way, if the robot is not perfectly aligned with the wall or if the lidar is not correctly positioned, the distance is still well calculated.

4.5 Correction of orientation

The robots tend to drift in one direction because of the alignment of the wheels. To correct this behavior we defined spot where the robot measures the distance to the wall in seven different pair of points. Knowing that the wall is straight, we can calculate the angle of drift, θ_{error}, that the robot has by measuring the distance to the wall at the defined pair of points. θ_{error} can then be calculated as:

$$\theta_{error} = \arctan\left\{\frac{d^{(2)}_{wall} - d^{(1)}_{wall}}{d_{travelled}}\right\}, \tag{18}$$

where d^{wall} is the distance to the wall measured at point one and two. Knowing θ_{error}, we can rotate the the whole path around the first point defined in each pair. This way we correct any drift that the robot has. See Figure 5 and Figure 4 for a visualization.

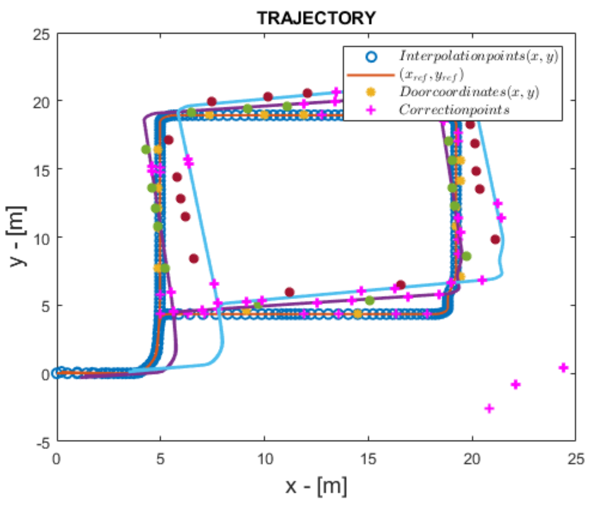

Figure 4: Grid map, showing the original trajectory and the several corrected paths, doors and important points.

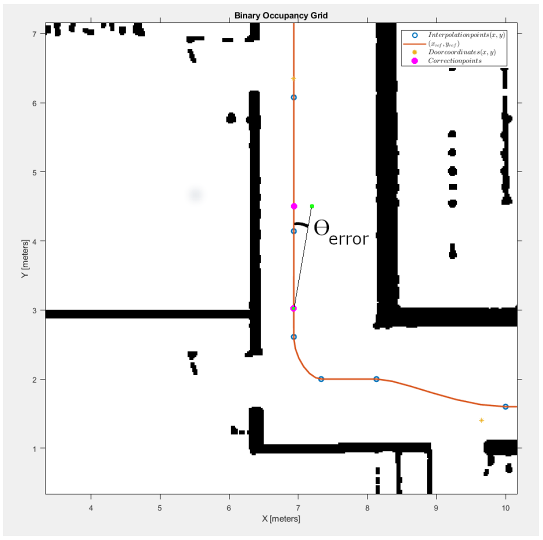

Figure 5: Schematic of measuring the drift in orientation

4.6 Timer

A timer has been implemented to assure a more or less constant time of updating and setting the controllers of the pioneer. After several rounds with the robot, the algorithm proved to be quite stable, with an average period of 0.0514 seconds and a standard deviation of 0.0437 seconds. The relative high value of standard deviation compared to the average period, is caused be the loop iterations where the lidar scans. This causes the average time period to be a bit higher than the reference value on 0.05 seconds.

5 Detection of doors

By transforming the input in polar coordinates to Cartesian coordinates through:

$$\begin{aligned} x_i &= s_i \cos\Big((\frac{240}{682})i - 30\Big), \\ y_i &= s_i \sin\Big((\frac{240}{682})i - 30\Big), \end{aligned} \tag{20}$$

where $s_i \in S$ is measurement i in S, and i is index of measurement, a better visualization is achieved. See Figure 6 for an example of a transformed laser scan.

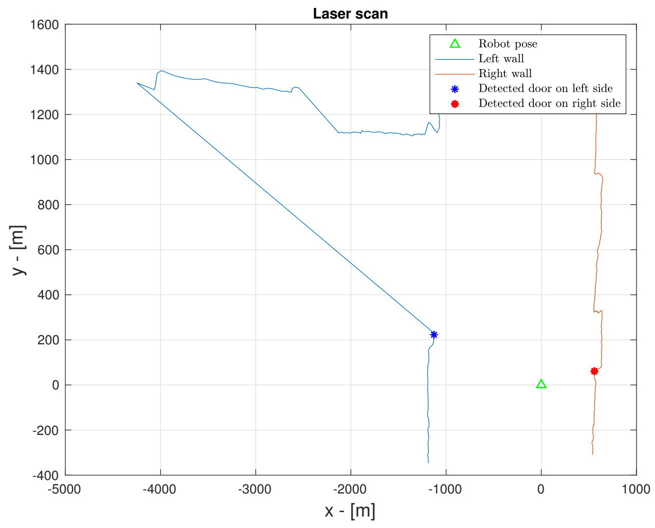

Figure 6: Example of a 240°-laser scan obtained from the lfr. The scan is transformed from polar to Cartesian coordinates. Green triangle shows the position of the robot. The blue and the red star indicates a detected door on left and right side, respectively. The left wall contains a open door, while the left contains a closed one.

5.1 Detection

To detect the doors, the respective norm between two following laser scans s_i and s_{i+1} is used. A straight wall without any disturbing elements will roughly have a constant value of the norm between two consecutive measurements. When the wall contains a door frame, a big change in value will appear. See Figure 7 and Figure 8 for a visualization. Knowing the global position of the door, we can start searching for it when the odometry, provided by the robot, tells us that we are close by. Additionally, considering the first norm that surpasses a certain threshold, we can detect the first door frame of each door. To reduce the measurement noise, only every second measurement. is considered.

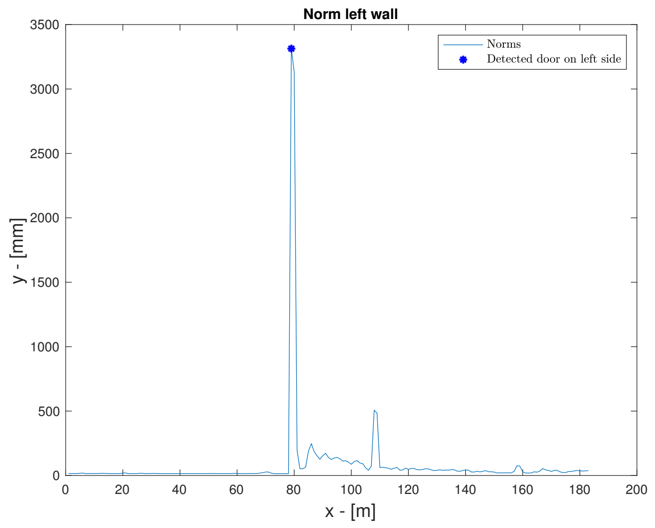

Figure 7: A plot of the calculated norms between measurements from the laser scan of the left wall in fig. 6. A clear peak, visualized with a blue star at the maximum, appears when a door frame is scanned.

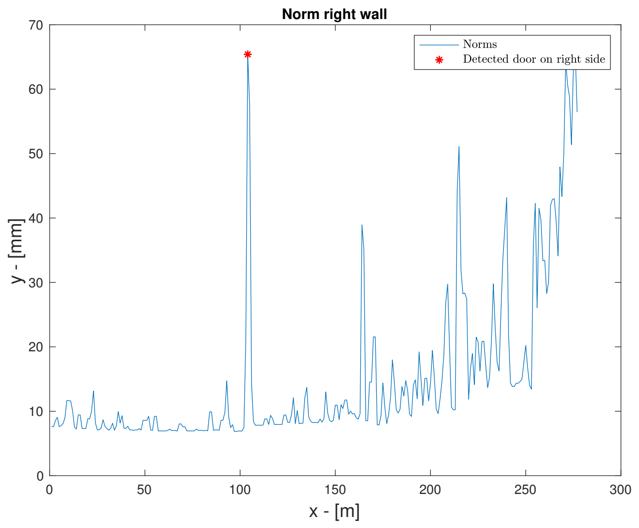

Figure 8: A plot of the calculated norms between measurements from the laser scan of the right wall in fig. 6. A clear peak, visualized with a red star at the maximum, appears when a door frame is scanned.

5.2 Determine if door is open or not

After detecting the door frame, the robot moves to the center of the door and turns 90º to assess its state. To detect the state of the door, which is either closed, open or half open, another laser scan is made in front of the door.

From this scan, using 40 measurements in the middle of the scan, the average value is calculated and used to classify the state of the door. If this value is greater than the previous distance to the wall (calculated upon reaching the door frame) plus a threshold, the door is considered to be open. If the average value is closer to the distance to the wall, the door is considered to be closed. All other cases mean a half open door.

6 Experiments and Results

In this section the experiments performed and the results for testing and tuning, will be introduced.

6.1 Tuning

In order to tune the controller, certain constraints is considered. First of all, the robot should finish the given task within a reasonable time, thus keeping as high speed as possible. Second, if the controller get lost, it should be able to recover quickly. Third, because of the physical constraints regarding limited space, the controller needs to follow the path precisely and can not perform too big curves when turning. The final parameters obtained after tuning is shown in Table 1.

Table 1: Final parameters and results obtained after tuning.

Final parameters
$v_{max} = 1.1$
$k_1 = 0.41$
$k_2 = 2.3$
$k_3 = 1.6$

A visualization using the final tuning parameters is plotted in Figure 9. The robot starts 2 meters away from target position, facing 180°wrong way. The plotting instance is run five times to assure consistency.

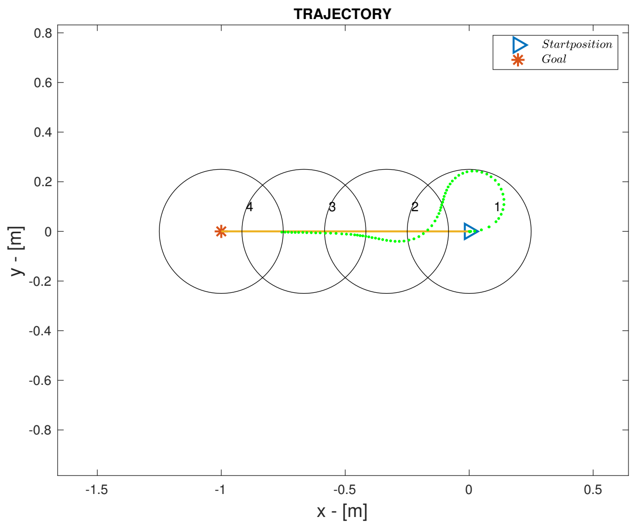

Figure 9: A plot of a tuning instance. The robot starts 2 meters away from target position, facing 180°wrong way. If the robot gets inside a black circle, it will update the reference point. The green dots indicates the retrieved odometry from the robot. The radius of the black circles is 0.2 meter.

6.2 Precision and recall of door detection

After running the robot 6 times we discover that the robot lost it self 5 times 2 without recovering in the space taht it had. In this runs the robot found the door state with of accuracy and found doors correctly except the times that it did not find the door first door frame.

Results
Left side doors detected correctly = 94%
Right side doors detected correctly = 96%
Doors state correctly identified = 93%
Recovered from getting lost = 3 out of 5

Table 2: The results is obtained after running the algorithm six times.

The accuracy is high and maybe not very precise because of the number of runs that we did.

7 Discussion

After testing we found that the robot sometimes would skip commands without any apparent reason, we were not able to find if it was a problem related to the connection between robot and computer, or a code problem. A suggestion to the issue is if there is read and write locks on the controller causing some commands being skipped.

7.1 Controller and correction of odometry

In our testing we saw that the controller was very good on recovery when it loses the reference point, as seen from fig. 9, but the way that it calculates the positioning causes some jitter/uneven speed, and this causes errors in the odometry reading. Our solution to correct the odometry is environment dependent, so if the some change in the space the system will not be able to correct the odometry correctly (eg. a person passing in front of tha lidar when the robot is taking measurements).

In order to get the robot to work flawlessly we could develop a system of continuous tracking of the distance to the wall in order the get a continuous correction. To implement this, we think it would be easier to implement in ROS, as we get a much faster program and greater rate of data from the sensors.

7.2 Detecting doors

Having reached an accuracy of 93% on detecting the state of the doors, we consider our algorithm good. The only errors we encountered, were given to a miscalculated reference value of the distance to wall, either by door frame not seen or an unexpected obstacle. With an incorrect value to compare to, the state of the door is more difficult to assess.

To correct this problem we think by implementing a Hough transform system we could get a more precise and robust system to detect the door state.

8 Conclusion and future improvements

The objective of this project was to develop creativity related to the use of mobile robots, and become familiarized with problems related to sensing, actuating, and controlling. In this project we have implemented a controller based on Lyapunov stability analysis, and executed the mission of detecting doors on the way, in a prior known environment. We have been introduced to and become familiar with problems related to a mobile robot, such as controller, timer, correction of odometry and so on. Last but not least, we have gained experience on how to approach such a problem in a smart and efficient way. For example, if the project would be done over again, several different controllers would be tested to see which one would perform better for our purposes.

Another improvement if the project were to be done again, is the choice of implementation. Using ROS (robot operating system) would be the preferred choice. This would gain better speed of e.g sensor readings. This would again lead us updating and correcting the error more often.

9 Code architecture

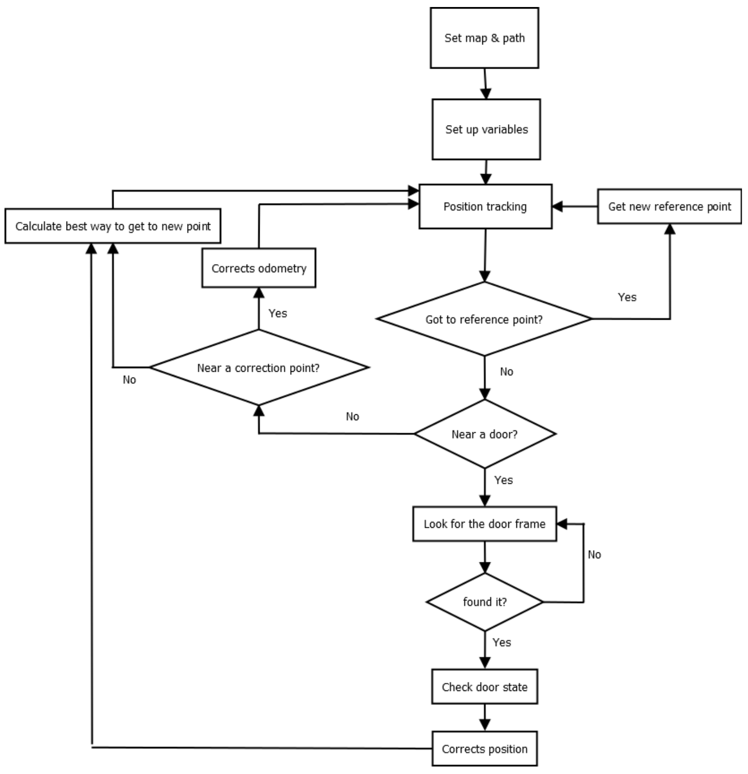

Figure 10: A flow chart explaining the code architecture.

References

[1] Gmapping
Available at: http://wiki.ros.org/gmapping

[2] Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)
Available at: https://www.mathworks.com/help/matlab/ref/pchip.html

[3] ”Bounded Smooth Time Invariant Motion Control of Unicycle Kinematic Models”. Youngshik Kim, Mark A. Minor. Procs. of the 2005 IEEE Int. Conf. on Robotics and Automation, April 2005

[4] ”Path Manifold-based Kinematic Control of Wheeled Mobile Robots Considering Physical Constraints”. Youngshik Kim, Mark A. Minor. Department of Mechanical Engineering University of Utah Salt Lake City. Procs. of the International Journal of Robotics Research Vol. 26, No. 09, September 2007
Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1010.8617&rep=rep1&type=pdf

[5] ”Closed loop steering of unicycle like vehicles via Lyapunov techniques”, M. Aicardi, G. Casalino, A. Bicchi, and A. Balestrino, IEEE Robotics Automation Magazine, vol. 2, March 1995.
Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=388294

[6] ”Applied Nonlinear Control”, Slotine and Li’s, p.122, 1990.

[7] ”Iterative Hough Transform for Line Detection in 3D Point Clouds”, Christoph Dalitz, Tilman Schramke, Manuel Jeltsch.
Available at: https://www.ipol.im/pub/art/2017/208/article.pdf