Basic navigation strategies for mobile robots

[1]Daniel Fortunato (81498), [2]Teodoro Dias (81723), [3]Pedro Santos (84162)

[1]Instituto Superior Técnico

June 7, 2019

Abstract — In this paper we present the results of the attempt to make a robot autonomously navigate the 5th floor of the North tower of IST and identifying relevant events along the way such as the state of all doors in the floor (open,closed or semi-opened) and people. We started from a simple state feedback using only the lateral and frontal sonar measurements as a reference to perform orientation but we soon found out that that was not enough, so we decided to add a contribution from the internal angle of the robot and we used the odometry to decide when to stop in each corridor. After some minor tweaks such as odometry resets we managed to not only navigate trough the whole fifth floor but also detect doors and the angles in which they are open/closed.

1 Introduction

Within the diverse areas that robotics embraces, mobile robotics has had great focus in the last decades from researchers around the world. In particular, issues like autonomous navigation, path planning, self-location, coordination and co-operative dynamics, mapping, exploration, surveillance, object detection, pursuit-evasion and coverage, which benefited from the progress of artificial intelligence, control theory, computer vision, real-time systems, sensor's development, electronics, communication systems and system integration.

This evident growth is extremely motivating for the development and contribution of new approaches for the community.

In this work, navigation strategies for autonomous mobile robots are studied, more specifically strategies for planning a robot's motion within an infra-structure. In order for a mobile robot to navigate in a structured environment, it needs to apply motion strategies.

In this study, most works consider a deterministic, finite, discrete and static environment, which only changes when the robotic agent moves.

Figure 1: Pioneer robot used in this laboratory

2 Path Planning

In this section is going to be addressed the approaches that were taken into account in the path planning of the robot. The path that was chosen can be seen in the figure (2).

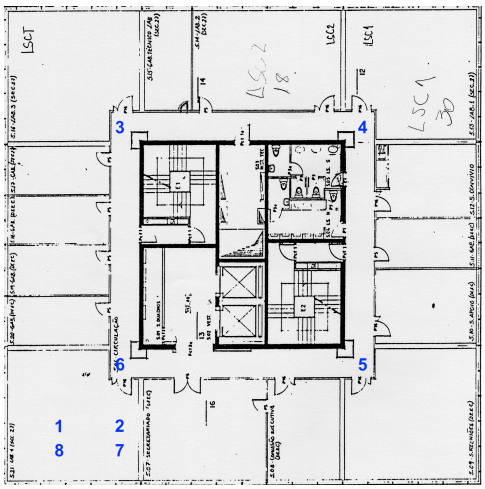

Figure 2: Chosen path for the robot

Looking at the figure (2), the chosen path was: 1- start in the middle of the laboratory room, 2- get out of the laboratory and go straight forward, 3 - go, clockwise, around the 5th floor, 4 - go back to the laboratory room.

2.1 Initial approaches

When trying to solve the path planning problem several approaches were tested before settling of the final solution. We first started with a control law of the following type:

$$\omega = K * (\frac{1}{d_{left}} - \frac{1}{d_{right}}) \quad (1)$$

Where K is a scalar gain, d_{left} is the distance measured by the leftmost sonar of the pioneer and d_{right} is the distance measured by the rightmost sonar of the pioneer. This initial feedback controller for the angular velocity has the property that when the robot is close to the wall the angular velocity increases exponentially in the opposite direction. The second proposed method is similar to the 1st one but does not "explode" when the pioneer is very close to the wall. We instead used the control law as follows:

$$\omega = K * (d_{left} - d_{right}) \quad (2)$$

With this setup we obtained better results but we still had some problems with irregularities in the corridors such as doors and their states (for example if a door is open on the left side of the pioneer, d_{left} is much greater that d_{right} so the robot will sway to the left with this control law)

2.2 Irregularity compensation

In order to compensate the irregularities shown above we decided that when we were passing trough a door we would make the static gain k=0 and when we finished seeing the door the static gain would return to it's initial value. The way we found that was best to do this is for each iteration we store 2 values of the sonar's measurements: the current one and the previous one. If the difference between these measurements is greater than 40 mm it means that we entered the space of a door (due to the depression between the wall and the door). Likewise if the difference is greater in module than -40 we left the door space. Whenever we are in a door space we made the gain K = 0.

2.3 Variant of the algorithm for corridors with benches and with elevator pit

For corridors with benches we have a problem: the compensation of the benches legs is done in the opposite direction of the doors but the underlying principle is the same. For corridor 4 (the one with the elevator pit) we used both walls for navigation until we reach the elevator pit and after we used only the left wall for orientation. using the equation:

$$right = 1600 - left - 269; \quad (3)$$

Where 1600 is the width of the corridor and 269 is the width of the robot.

Based on this we have, for corridors 1-4, the following setup: 1- compensates doors on both sides; 2- compensates doors on the left and benches on the right; 3- compensates doors on both sides until half of the corridor and then compensates doors on one side and benches on the other; 4- routine mentioned in the paragraph above

2.4 Final iteration of the algorithm

Despite seeing significant improvements in the navigation of the robot, it sometimes made very sharp turns in the orientation so we decided to add a term that had a compensation of the internal angle:

$$\omega = K_{dist} * (d_{left} - d_{right}) + K_{angle} * (\theta_{reference} - \theta) \quad (4)$$

Where θ is the angle measured by the robot when it is navigating the corridor and $\theta_{reference}$ is the angle that the robot has when it starts each corridor. This implies that the robot must start each corridor with an as straight as possible angle. For the rotation of the robot at the end of each corridor we first tried to do it by reading the front facing sonars but it proved unreliable in multiple iterations so we switched to odometry based turning: when the robot walks a certain distance it automatically turns.

2.4.1 Gain Values

For the corridor 1, the robot was supposed to move in the middle of the corridor, so the penalization for the distance measurements in the left side is the same penalization for the distance measurements in the right side. So, for this corridor, the only thing changed in the cost function (4) is the gain of the sonars measurements (K_{dist}) and the gain of orientation (K_{angle}). The values for the corridor 1 gains are in the table (1).

K_{dist}	K_{angle}
0.008	$K_{dist} * 1300$

Table 1: Corridor 1 gains

For the corridor 2, the robot was supposed to move more to the left due to the benches. In order to do that the cost function (4) was changed, as can be seen in the equation (5).

$$\omega = K_{dist} * (d_{left} + 400 - d_{right}) + K_{angle} * (\theta_{ref} - \theta) \quad (5)$$

The corridor 2 gains are shown in the table (2).

K_{dist}	K_{angle}
0.008	$K_{dist} * 1300$

Table 2: Corridor 2 gains

Basically, the cost function of the angular velocity is equal to the one used in the corridor 1 but now was added an offset in order to make the robot move 400 mm to the left of the center of the corridor, so it does not hit the benches.

Because there are benches in the corridor 3 aswell, the robot was supposed to move more to left. The approach implemented in the corridor 2 did not work for this corridor because, in this case, the benches are in the end of the corridor and not in the beginning as in the corridor 2. The approach taken in this case was, instead of having an offset, penalize more the distance measurements for the

left side of the robot. The cost function of the angular velocity for the corridor 3 is shown in the equation (6).

$$\omega = K_{dist} * (\frac{2}{3} d_{left} + 400 - \frac{1}{3} d_{right}) + K_{angle} * (\theta_{ref} - \theta) \quad (6)$$

Because there is a higher value multiplied by the distance measurements of the left side than in the right side, the control implemented penalizes more the distances from the left wall. The output of this approach is making the robot moving closer to the left wall, as wanted.

The corridor 3 gains are shown in the table (3).

K_{dist}	K_{angle}
0.01	$K_{dist} * 1000$

Table 3: Corridor 3 gains

Finally, for the corridor 4, because there are just doors in the left side, the approach taken was the same of the corridor 3 but with different regularization terms in order to move closer to the left wall and so there is no problem in the door detection.

The cost function of the angular velocity for the corridor 4 is shown in the equation (7).

$$\omega = K_{dist} * (1.7 * d_{left} + 400 - 1.3 * d_{right}) + K_{angle} * (\theta_{ref} - \theta) \quad (7)$$

The corridor 4 gains are shown in the table (4).

K_{dist}	K_{angle}
0.005	$K_{dist} * 1300$

Table 4: Corridor 4 gains

2.5 Odometry resetting procedures

In order for the robot to properly navigate the floor we needed to reduce as much as possible the errors due to odometry. One possible solution consists in resetting the odometry in each corridor to reduce the propagation of errors but since the inherent software of the robot to perform the resetting procedure does not work we needed to come up with out own algorithm. The algorithms consists on 2 simple features: saving the values of x,y and θ at the end of each corridor and subtract them in subsequent calls for the parameters. This way the navigation of the robot for each corridor is closely the same in terms of odometry and stopping values.

3 Door Detection

For the door detection it was used an approach using only data from the sonars, due to the limitations in terms of time of the laser range finders. This approach was based on saving the current and the previous distance measure of the sonar 1 and 8 (the leftmost sonar and the rightmost sonar, respectively). The scheme that shows the algorithm to detect doors can be seen in the figure (3).

We found out that, since we can find when the door begins and ends, by taking the distance measures of the sonars we can perform a linear regression between the pioneer position and the door,leading to finding the angle that the door makes when related to the pioneer path.

With this information we can find when the door is semi-opened or closed/opened by checking the angle (if the angle is close to zero the door is either fully closed of fully opened) and checking if the door is opened or closed by checking the average measured distances of the sonar on the door space: if the distance is greater than a certain threshold the door is opened and it is closed otherwise.

This yields very good results with a very low rate of mislabeled doors per iteration of the navigation algorithm. The only possible and tested source of error is sometimes the reflections of the sonar on the grids on the lower parts of the doors which leads to some mislabeled doors. In order to decrease even more the rate of mislabeled doors it was used the median instead of the mean of the vector of measurements, so this way the errors coming from the sonar reflections are not so pronounced.

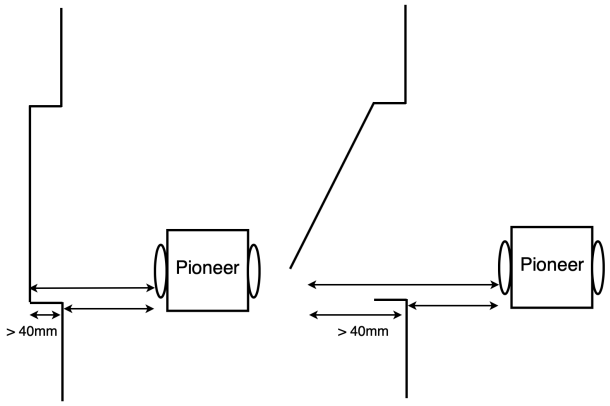

Figure 3: Door detection scheme

4 Results

For the navigation of the robot it was used a constant velocity of 200 mm/s for the whole floor. This was the velocity chosen due to the approach used for the door detection. In order to have as much points as possible to compute the linear regression of the distance measurements of each door it is necessary to have a low velocity.

In this laboratory, the issue of correcting the odometry was not address because it was taken of another approach, as explained in the previous sections. Due to that issue not being addressed the real path of the robot could not

be computed. The path that was able to be computed was the path given by the robot's odometry. This path can be seen in the figure (4):

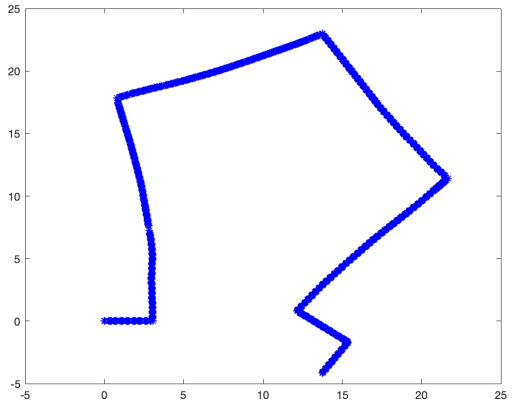

Figure 4: Path given by the robot's odometry

Note: The robot used in this experimental work was the Pionner nr. 6. This robot tended to turn a lot for the right side, comparing with the other ones.

In this figure (4) it can be seen (as experienced in the experimental work with the robot) that the path given by the odometry is wrong. Another thing that was observed in the laboratory was the wrong orientation in the turns. That can be observed in the figure (4) in the orientation of the robot in the corridor changes.

Finally, the last thing to be addressed in the figure (4) is the turning to the left in every corridor shown in the robot's odometry. That is due to excessive turning to the right that the robot does on its own when the control commands the robot to go forward. So, in order to compensate that turning to the right, the control that was implemented in this laboratory commands the robot to turn to the left. And, because the odometry just shows the path that the robot thinks he did, the figure (4) shows that turning to the left in every corridor.

In order to understand better the results, the plots of the distances to the left wall and the right wall are shown in the figure (5) and (6), respectively.

It can be seen in both figures (5) and (6) that the control implemented compensates a lot the position of the robot in the beginning of every corridor, due to the wrong position of the robot when it reaches the end of every corridor, and then tends to be more stable for the rest of the path. In the figure (6) it can be seen very abrupt changes due to the high distance measurements that the rightmost sonar does when the robot reaches the end of every corridor and, also, the elevator shaft and the hall that leads to the stairs.

It can also be seen that, in the every corridor, the robot moves a little bit away from the left wall. That is due to the gain that that is being multiplied by the orientation of the robot in the minimization function (4). Because that gain is higher than the gain that is multiplied by

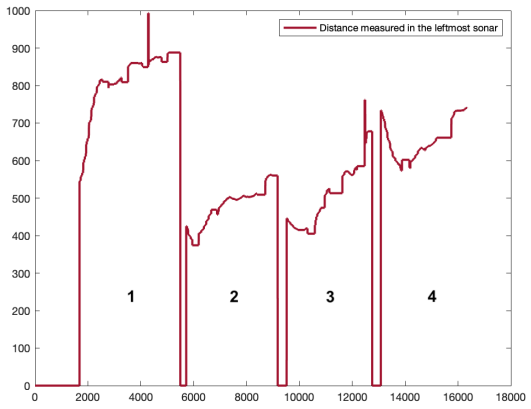

Figure 5: Distance measurements to the left wall

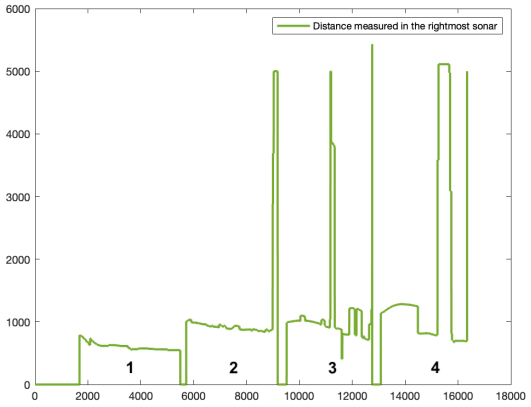

Figure 6: Distance measurements to the right wall

the difference between the measurements of the sonar 1 and 8 so the robot does not change the angular velocity abruptly, the robot still tends to turn a little bit to the right, but with the implemented control the risk of hitting a stationary object is almost null.

In order to prove that the control implemented does not give commands that makes the system unstable, the figures (7), (8), (9) and (10) show the control of the angular velocity for the corridor 1, 2, 3 and 4, respectively.

Note: The positive values of the angular velocity make the robot turn to the left and the negative values make the robot turn to the night.

It can be seen in the figures (7), (8), (9) and (10) that the control of the angular velocity is never abrupt. It has a maximum of $2^{o}/s$ and it's never for much time except in the beginning of the corridor 1 when it compensates the wrong position of the robot when it gets out of the laboratory room.

It can also be seen that the control implemented commands the robot to turn much more to the left side than to the right side. That is due to the fact that the robot used in the laboratory turns excessively to right and the

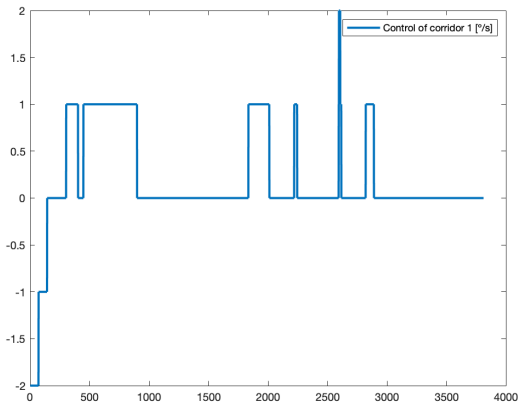

Figure 7: Control of the angular velocity for the corridor 1

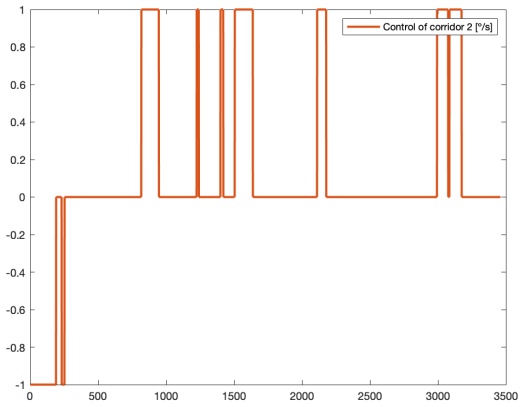

Figure 8: Control of the angular velocity for the corridor 2

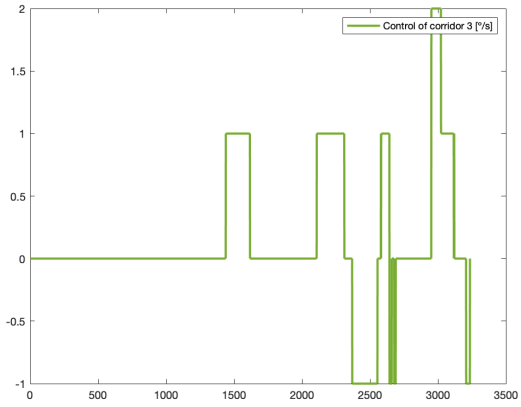

Figure 9: Control of the angular velocity for the corridor 3

control has to compensate that.

Finally, it can be seen in the figures that sometimes the

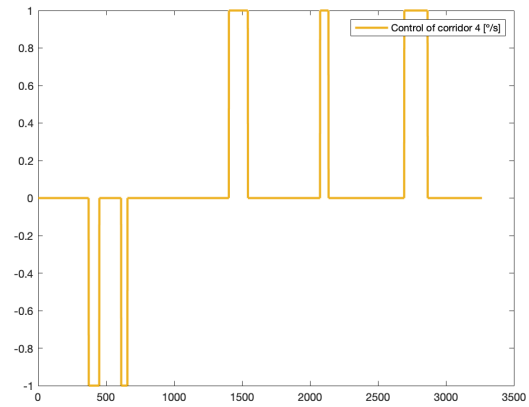

Figure 10: Control of the angular velocity for the corridor 4

control is null. This happens when the robot detects a door. When it detects a door, the gain that is multiplied by the difference of the distance measurements of the sonar 1 and 8 and the gain that is multiplied by the orientation of the robot go to zero in order to put the robot as parallel as possible to the door so there is no error in the computation of the linear regression of the distance measurements for each door.

5 Code Explanation

The function *get_rob_pos* is responsible for getting the robot's position given by the odometry. This function can also be used to reset the odometry if the input of this function is the path travelled because it subtracts that path to the reading given by the odometry.

```
function [pos_x,pos_y,pos_theta] = get_rob_pos
(reg_x,reg_y,reg_t)

    odo = pioneer_read_odometry();
    pos_x = odo(1)-reg_x;
    pos_y = odo(2)-reg_y;
    pos_theta = wrapToPi((odo(3)*360/4096)*pi/180
    -reg_t);

end
```

In terms of implementation, there are just two kinds of cycles in the code: the ones that are responsible for making the robot move to the destination and the ones responsible to make the robot turn.

The ones responsible for the movement just make the robot move with constant velocity and the respective control until the stop condition is reached.

The ones responsible for making the robot turn just put to zero the linear velocity and a constant angular velocity until the necessary rotation is reached.

In order to run the program it is necessary to have the *mp3* files that are responsible for making the door state

classification sounds. These files will be sent in attachment aswell.

6 Conclusions

In this study, even though it did not addressed the issue of correcting the odometry, the navigation of the pioneer through the 5th floor was made very smoothly. In order for this to happen it was needed some adjustments throughout the work, such as the gains and the regularization terms of the angular velocity cost function for every corridor.

As can be seen in the results section, this approach did a pretty good job in the robot navigation and showed stability in the control of the system. The only disadvantage of this approach is the fact that the real path of the robot cannot be shown. Due to this a video of the movement of the robot around the 5th floor was recorded.

The door detection approach was a little bit difficult to implement since, at the beginning, it was difficult to get the robot to move parallel to the doors and, due to that, the focus of this work was mostly oriented to the movement and the path of the robot. But, in the end, with all the adjustments, the robot was able to go around the 5th floor without hitting anything and detecting all the doors, as can be seen in the recorded video.

In the evaluation day, there was a problem in one door state classification and it did not detected one. Both of this problems appeared in the corridor 3. The non detected problem was due to, in the implementation, the robot only detects doors in the right side of the corridor 3 until a certain distance, due to the benches. So, in order not to confuse doors with benches it was put a limit until the robot can detect doors in the right side. The door that was not detected was after this limit. The door state classification problem was in the end of the corridor 3 and was due to the robot reaching the end in a crooked position, and when it started the corridor 4 it was not parallel to the door, which was the cause of the mislabeled door.

In general, it can be said that the robot showed enough robustness and stability in the navigation and in the detection of doors.

Two videos will also be attached to show two different situations where the doors are arranged differently.

7 References

[1] Mark W. Spong, Seth Hutchinson, M. Vidyasagar (2nd edition), Robot Dynamics and Control

[2] Portugal, D. "A study on local planning techniques for mobile robot navigation" University of Coimbra, 2010.