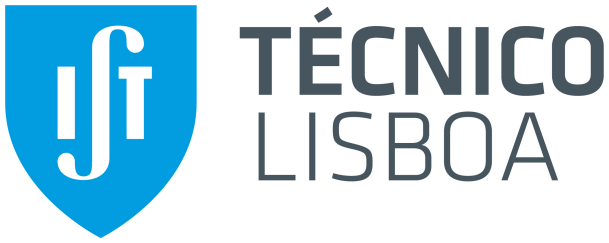


Instituto Superior Técnico

Robotics

2^{nd} Semester 2020/2021

Autonomous Car

Study conducted by:	*Number:*
Gonçalo Maia	87001
Gonçalo Morais	87004
Tiago Brito	87125
Alice Rosa	90007
Aprígio Malveiro	90026
Beatriz Pereira	90029
Francisco Galante	90073
Gonçalo Matos	90545

Professor: João Sequeira

June 11, 2021

Contents

1 Introduction **2**

2 Supervisor **3**

2.1 Path Planning . . . 3

2.1.1 Map Image Processing . . . 3

2.1.2 Visibility Graph . . . 4

2.1.3 Dijkstra . . . 4

2.1.4 Adding a node to the graph . . . 5

2.2 Trajectory Generation . . . 5

2.3 Event Handling . . . 6

2.4 Supervisor Conclusions and Appreciations . . . 6

3 Control **7**

3.1 Position tracking . . . 7

3.2 Choice of values for the control laws . . . 8

3.3 Energy consumption . . . 8

3.4 Controlling the car to handle events . . . 9

3.4.1 Event E1 - Stop Traffic Sign . . . 9

3.4.2 Event E2 - Speed Limit Traffic Sign . . . 9

3.4.3 Event E3 - Pedestrian Crossing Traffic Sign . . . 9

3.4.4 Event E4 - Pedestrian Crossing . . . 9

3.5 Control Results and Comments . . . 10

4 Navigation **12**

4.1 Car . . . 12

4.1.1 Car Model . . . 12

4.1.2 Car simulation . . . 13

4.2 State prediction . . . 13

4.2.1 Sensors and Data Acquisition . . . 13

4.2.2 State estimator through Sensor Data . . . 13

4.2.3 Sensor Failure . . . 14

4.3 Navigation Results and Comments . . . 15

5 Graphical User Interface (GUI) **16**

6 Conclusion **18**

1 Introduction

This project aims at developing a simulation of an autonomous car operating inside the defined environment of the IST Alameda campus.

The car must manage to move between any two arbitrary points, given an adequate energy budget, along trajectories entirely contained inside the admissible road areas and comply with certain predefined events, such as traffic signs and interaction with pedestrians.

The development of this lab assignment was a collaboration of three groups, one responsible for the high level control and handling of events (supervisor) and other for the low level control of the motion of the car. The third group was responsible for connecting the previous two, that is, the navigation.

2 Supervisor

This section and corresponding code was made by the group composed by Gonçalo Maia No.87001, Gonçalo Morais No.87004 and Tiago Brito No.87125.

On the following subsections, the group will explain what was made related to the Supervisor. The main focus points will be the decomposition of the map image to better work on the possible locations in which the car could circulate, the elaboration of the graph, how the trajectory was calculated and computed and the addition of a point to the graph.

2.1 Path Planning

2.1.1 Map Image Processing

In order to map the area of circulation and optimize the trajectories that the car can make in the designed area, the map of *Instituto Superior Técnico* should be on a image that can be processed and used to print the trajectory that the car will take.

Firstly, a *google maps* screenshot of the campus was taken, and with the help of *photoshop*, a binary image of the map was created, with the roads in white and the rest of the image is shown in black. The function *imcomplement* creates the variable *map*. This variable stores the binary value of each pixel of the image, accordingly with its color. In other words, a pixel has the value of 1 if it is part of the road, or 0 if the car cannot circulate in that area.

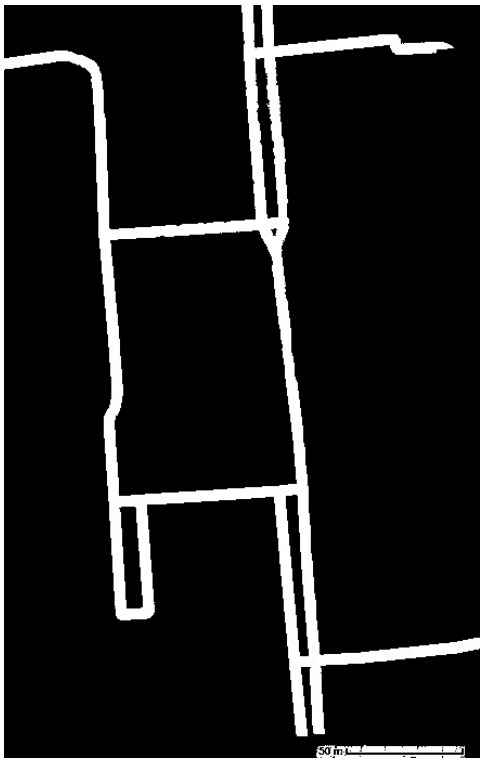


Figure 1: Binary map of the circulation area

Having the information of which parts of the map are valid, the map image is put in scale and the coordinates of each pixel can be estimated allowing to specify map's points with a specific location to better calculate the car's trajectory.

2.1.2 Visibility Graph

The visibility graph represents all the map references that are used to make the path of the car. These references need to be put in a specific location, in order to cover all the possible movements that the car will make. In this sense, to compute the possible ways to move on a intersection, a point was put on every way out of that intersection on the circulation area. For the parcels of the road that represent a straight line or a curve, nodes were included between the beginning and end of that section. This ensures that the car does not diverge from the path that is valid for circulation and guarantees that the trajectory has the same form as the road on the map. Adding more nodes to the graph would increase the accuracy of the path selected. The coordinates of these nodes are specified on the variable $grafoCoord$.

The visibility graph obtained is composed by 57 nodes, as can be seen in figure 2.

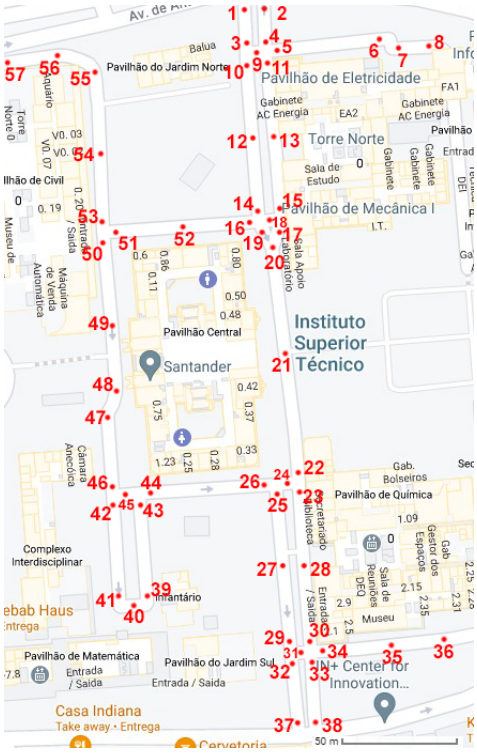


Figure 2: Visibility Graph

2.1.3 Dijkstra

In order to better optimize the path definition through the nodes of the graph, it was necessary to chose an algorithm that better suited the necessities of this search.

In this sense, the *Dijkstra* Algorithm was chosen, which will be briefly explained in detail.

This algorithm takes into considerations a given graph, that in this case consists in the nodes and connections that were explained in the last subsection. With that given graph, *Dijkstra* computes the shortest path from a given source to all vertices. However, in this project, only the path from the selected source to the selected destination will be considered, in order to avoid harming the performance of the program.

To compute the referred shortest path, a Shortest Path Tree (SPT) is generated with the selected source node as the root. Then, two sets are maintained: the one with the nodes included in the SPT and the other with the nodes not yet included in the SPT. At every step of the algorithm, the neighbour nodes of the ones

belonging to the set of nodes that take part of the SPT are analysed and the one with the smallest relative distance to the origin is added to that set. This process is repeated until the set of nodes that don't belong to the SPT is empty, i.e., all nodes are present in the SPT.

The function computed to implement the *Dijkstra* algorithm will, at the end of the execution, return an array that will contain all the nodes from the graph that take part on the discovered shortest path.

2.1.4 Adding a node to the graph

To allow the user to choose any point of the map as a starting or finishing position it was necessary to develop an algorithm that can take the coordinates of the selected point and add it to adjacency matrix, with the correspondent connections. The developed algorithm will be explained in detail in the following paragraphs, for a simpler reading of the algorithm please refer to the flowchart in figure 3.

The first step consists on accessing if the chosen point is already in the graph, if so the algorithm ends and the program proceeds to the its next steps with the information that the start node or the end node is already in the graph.

Then it is necessary to verify if the user chose a point that lies in the road. If it is a point of the sidewalk the algorithm terminates with the invalid point handler, meaning it notifies the user that the chosen point is not valid. If the point is on the road a line and a column are added to the adjacency matrix representing the connections of this new node.

Having a valid node to add to the graph it is necessary to compute which nodes it can connect to. To perform this evaluation a all the graph nodes must be studied. The first criterion to decide if a node is a neighbour of the new node is the distance, if the node is above a defined distance it is discarded and the algorithm moves to the next one. If the node is below the distance threshold, a straight line between the new node and the node being studied is computed. The node is considered a neighbour only if a straight line between the to points is composed only of road points. This process is repeated until all the nodes have been studied.

After computing all the neighbours of this new node the algorithm terminates with the adjacency and coordinates matrices updated.

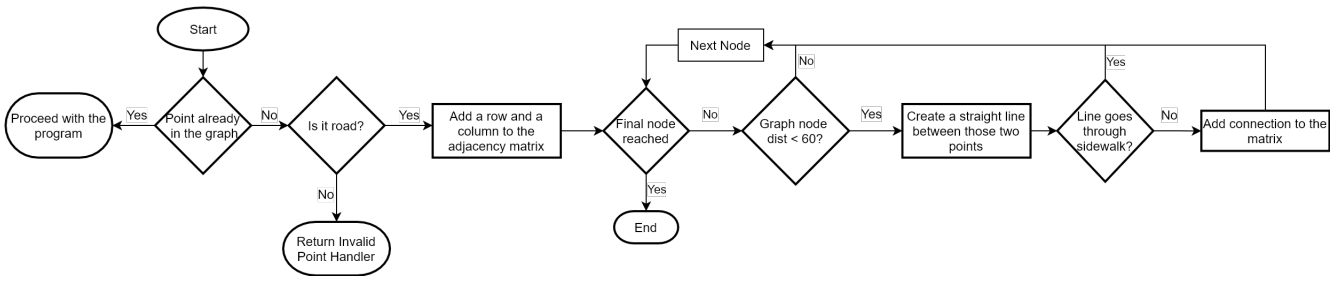


Figure 3: Adding a node to the graph flowchart

2.2 Trajectory Generation

The trajectory that the car should follow, and that will be fed to the low level controller is computed using the path defined before. This trajectory avoids collisions and is smooth enough so that the car can follow, specially in corners.

The trajectory is interpolated from the points of the path using the *pchip* function, a cubic interpolation

function presented in the lectures. This provides a smooth trajectory for the car to follow. In order to apply this function to the interpolation of points from a trajectory in a map, the x and the y coordinates should be dealt with separately and using as independent variable time.

After implementing the *pchip* function and using the output, an array of the x and y coordinates, the trajectory is printed on the map so that the user can see it and later on compare it with the results of the car.

2.3 Event Handling

On the area of circulation, events occur that require a change of behaviour from the car. These events are stop traffic signs, speed limit traffic signs, pedestrian crossing traffic signs and pedestrians crossing.

The location of these events is fixed and pre-specified, because of this, some nodes from the visibility graph were chosen arbitrarily to represent these events.

Firstly, to define this specific nodes, three columns are added to the variable $grafoCoord$, that relates respectively: to the type of event occurring on the location of the node, to the velocity of the car or the time duration of the event and to the simulation time when the event starts. The values 0, 1, 2, 3 or 4 are attributed according to the type of event that happens on that specific node: 0 if does not exist a event in the node, 1 if the node represents a stop sign, 2 if the node represents a speed limit sign, 3 if the node represents a pedestrian crossing sign and 4 if the node represents a pedestrian crossing. The velocity limit of the car in a event or the time duration of the event depends of the event: for the event type 1 the time duration is set to 5 seconds, for the event 2 the velocity limit is set to 30 km/h and for the event 3 is set to 20 km/h.

Having the trajectory computed, the points that composes it are returned to the main function in the variable *trajectory*. This variable contains the coordinates (x, y), the event type, the event handler that contains velocity limit or time duration and the event start time for each point of the trajectory. The data of the trajectory is all returned and the processing is made point by point by the Control.

2.4 Supervisor Conclusions and Appreciations

The implementation of the supervisor allowed the group to put to practice the concepts of trajectory planning and the development of a shortest path algorithm.

The results obtained were in accordance with the expected. The trajectory obtained was well defined and in accordance with what would be a real-life scenario. This same trajectory was also within certain specifications defined. This specifications include the traffic orientation of each parcel of road, the computation of the shortest path possible and the definition of the road boundaries where the car should circulate.

Overall, the project allowed the group to get a better grasp of the basic procedures necessary to implement an optimized trajectory.

It's also relevant to mention that with the automation of the decomposition of the map's image into valid and invalid circulation space and node generation, the basic concepts applied here could also be applied in a much more advanced broader sense, that could be used in a real autonomous car.

3 Control

The study of the Control of the autonomous car was conducted by the group composed by Alice Rosa No. 90007 and Beatriz Pereira No.90029.

3.1 Position tracking

The control law used was the "position tracking" law. The problem can be described geometrically, as illustrated in Figure 4.

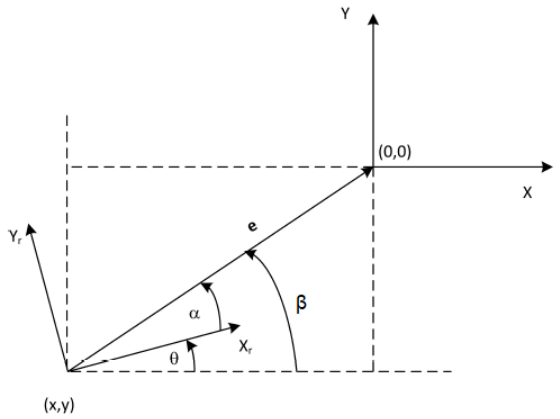


Figure 4: Geometric representation of the problem, [1]

Where,

- e= $\|e\| = \sqrt{x^2 + y^2}$, represents the length of the error vector (0,0)-(x,y) and (0,0) is the final position
- $\theta = \theta + \phi$, given that ϕ is the steering angle
- β= atan2(-y,-x) is the ideal orientation of the car
- $\alpha = \beta$ - θ is the orientation error

Thus, the control law is given by the equations (1) and (3), that can be shown to asymptotically stabilize e=0.

$$v = v_{max} \tanh(K_1 \times e_{final}) \tag{1}$$

$$\omega = \omega_{max}((1 + K_2\frac{\beta}{\alpha})\frac{\tanh(K_1 \times e_{final})}{e_{final}} \sin(\alpha) + K_3 \tanh(\alpha)), \;\; K1, K3 > 0 \tag{2}$$

In order to describe a realistic model, certain limitations had to be applied. The derivative of the steering angle, ϕ, defined as the angle between the front of the vehicle and the steered wheel direction, is the angular velocity, ω, restricted as ω<ω_{max}=$\frac{\pi}{4}$.

The orientation error, α, was limited between $[-\pi,\pi]$ to avoid heavy oscillations in angular velocity, ω.

The gains K_1, K_2 and K_3 influence the system. The gain K_1 controls the weight given to the position error, e. The gain K_2 determines the weight of the ratio between the ideal orientation of the car, β, and the orientation error, α. K_3 limits the hyperbolic tangent of the orientation error.

To obtain a smoother angular velocity control, the function *tanh* was added to the equation (3). Therefore, the control law used for the angular velocity of the car is,

$$\omega = \omega_{max} \tanh\left((1 + K_2\frac{\beta}{\alpha})\frac{\tanh(K_1 \times e)}{e}\sin(\alpha) + K_3 \tanh(\alpha)\right), \;\; K1, K3 > 0 \tag{3}$$

Also, in the control law applied there are two types of position error computed, e. One is the error between the current position of the car and a defined final position, e_{final}, used in the linear and angular velocity control law. The other one is between the current position and a near intermediate point in the path, e_{int}, used to compute the wanted orientation of the car, β.

3.2 Choice of values for the control laws

The gain K_1 was chosen so that the car decreases its velocity in curves and increases it in straight paths. For this effect, the expression of the gain K_1 is given by,

$$K_1 = 0.02 \times e^{-|90 \times \alpha|}, \tag{4}$$

where the values 0.02 and 90 where adjusted during the tests made to the system.

The gains K_2 and K_3 were also determined and adjusted during these tests, resulting in the final values of $K_2 = 1$ and $K_3 = 10$.

In order to better represent a real car and avoid big oscillations in the linear velocity of the car, it was defined a maximum linear acceleration that the car can have through out the path.

3.3 Energy consumption

The energy of the car is monitored along the mission so that if, at any moment, the car goes out of energy it stops immediately. The energy spent up to a certain instant is given by,

$$E_k = \sum_{i=0}^{k} \Delta E_i. \tag{5}$$

Where ΔE is given by,

$$\Delta E = (M|\dot{v}(t)||v(t)| + P_0)\Delta t. \tag{6}$$

The available energy for the remaining of the mission is given by,

$$\Delta E_{remaining} = E_{budget} - E_k \tag{7}$$

In order to include the energy consumption in low level control we started by computing the available energy per step given by,

$$E_{step} = \frac{\Delta E_{remaining}}{N_{steps\ remaining}}, \tag{8}$$

where the $N_{steps\ remaining}$ was estimated as 2 times the number of steps of the pre-defined trajectory. Then we set the maximal linear velocity to maximum value of the solution of the equation,

$$(M\dot{v} + P_0) \times v = \frac{E_{spent}}{\Delta t}. \tag{9}$$

The solution of the equation (9) was used as a conservative estimate for the maximum velocity that allows the car to reach the end of the mission, and is given by,

$$v = \frac{E_{spent}}{\Delta t \times P_0}. \tag{10}$$

3.4 Controlling the car to handle events

We start by verifying which events will happen during the mission and sort them taking into account the time in which they happen.

3.4.1 Event E1 - Stop Traffic Sign

In this event, we want the car to slow down until it reaches a position near the stop traffic sign, and then stop completely for a certain interval of time defined by the user.

For this effect we define the final position as the position of the stop traffic sign. This way we make sure that the controls of the car will make its linear velocity tend to almost zero as it reaches the position of the stop sign.

Then, we verify if we are near the stop sign and, if that's the case, define the linear and angular velocity to zero for the determined period of time. After that we reset the final position to the final position of the pre-defined path.

3.4.2 Event E2 - Speed Limit Traffic Sign

After passing through a speed limit traffic sign, the velocity of the car as to be under that limit the rest of the mission or until the car finds another traffic sign.

Therefore, we define the maximum velocity and verify if the velocity control is above that value, then we set it to the maximum established.

3.4.3 Event E3 - Pedestrian Crossing Traffic Sign

In this event we want the car to slow down when passing through the pedestrian crossing traffic sign. Thus, we have to set the final position of the car a little ahead of the position of the sign.

3.4.4 Event E4 - Pedestrian Crossing

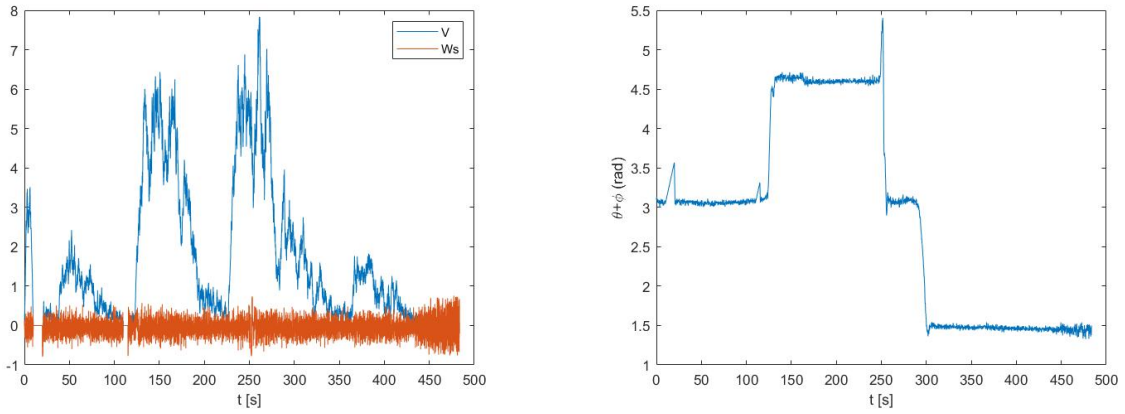
In event E4 we need to verify if we are going to be in the same location at the same time as the pedestrian is crossing the street. If so, the car has to slow down and stop the car so the pedestrian can pass through safely.

The prediction of the time of the simulation that corresponds to being in the same location as the pedestrian is calculated through the distance to that point and the velocity of the car as specified in (11).

$$\Delta t = \frac{d}{V} \Rightarrow t_{E_4} = t_{current} + \Delta t \tag{11}$$

If t_{E_4} is near the time specify for the start of the event E_4, then we slow down and stop the car at a safe distance from the location where the pedestrian is crossing the street.

3.5 Control Results and Comments



(a) Controls - Linear and angular velocity of the car

(b) Orientation of the car

Figure 5: Controls and orientation of the car during a mission

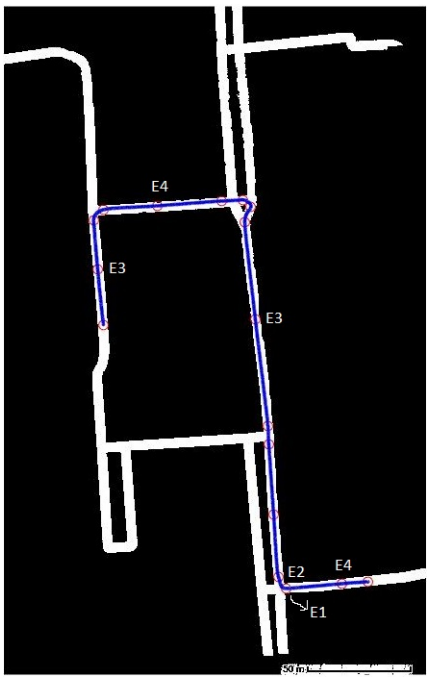


Figure 6: Trajectory of the car

To better show the results obtained, we defined and simulated a path where all events occurred.

In Figure 5(a) it's represented the linear velocity of the car in ms^{-1} and the angular velocity in $rads^{-1}$. It's possible to observe that both control signals present some noise, given that the information used comes from sensor's data (localization, orientation and velocity) that have associated noise.

By analysing the results illustrated in Figure 5(a), comparing it with the order that the events occur and with the path taken by the car we can conclude that event E4 is detected right at the beginning of the mission and the car correctly stops for 10 seconds to let the pedestrian cross the road.

Next, the car approximates a stop traffic sign (E1) and stops for 5 seconds. Right after, the car passes through a speed limit traffic sign (E2), thus the maximum velocity is defined as 30 kmh^{-1} (8.33 ms^{-1}) until

the end of the trajectory.

In between events E1 and E3 the car accelerates and then decreases the velocity again when is approaching the pedestrian crossing traffic sign. Then, the car drives through an accentuated curve that results in the variations of velocity shown.

The second time event E4 occurs isn't detected, because the time of the simulation for when the event starts (10 seconds) doesn't match the position of the car in that time instant, thus it isn't near the pedestrian.

The second event E3 is detected and the car slows down again when passing through the pedestrian crossing traffic sign and then accelerates. Since, by now, the car is close to it's defined final position doesn't accelerate a lot.

In figure 6, the blue line represents the path taken by the car, thus we can conclude that the trajectory was identical to the predefined one.

Tests were also carried out in which the total energy available for the road was reduced in order to verify the car's behavior. Although the maximum speed decreases due to the energy decrease, it is not enough to allow the car to complete the set path, since the variations in the acceleration consume a lot of energy. The equation (10) would have a bigger impact if the car's acceleration were close to 0 along the entire path. However, it allows the car slow down rather than make a sudden stop, when it runs out of energy.

4 Navigation

The Navigation section of the project was developed by the group composed by Aprígio Malveiro No. 90026, Francisco Galante No.90073 and Gonçalo Matos No.90545.

The main goals for this section is to model the functioning of the car, model the sensors used and to predict the state of the car.

4.1 Car

4.1.1 Car Model

For this project, the representation of the autonomous car was considered to be similar to a mobile robot with car-like kinematics. This approach facilitates the understanding and representation of the variables.

Regarding the car model, it was considered to be constituted by the variables presented in Fig. 7, where it is possible to separate these variables into two different sets.

One consists of the ones that remain constant throughout time and that correspond to the car dimensions. These are the following: L, represents the distance between the axis of the rear wheels and the axis of the front wheels and has the value of 2.2 m; L_r, corresponds to the distance between the rear end of the car and the axle of the rear wheels and its value is 0.566 m; L_f, expresses the distance between the front end of the car and the axle of the front wheels and it has the same value as the previous one; d has the value of 0.64 m and symbolizes half of the width of the car and finally r is the radius of the wheels (0.256 m).

The other set is composed of the variables that will change according to the time and situation. These are called state variables and are the following:

- θ is the angle that defines the current orientation of the car relatively to its previous position;
- ϕ represents the angle that defines the current orientation of the steering wheel;
- ω_s indicates the steering wheel angular velocity;
- x is the position along x-axis;
- y is the position along y-axis;
- V represents the linear velocity of the car.

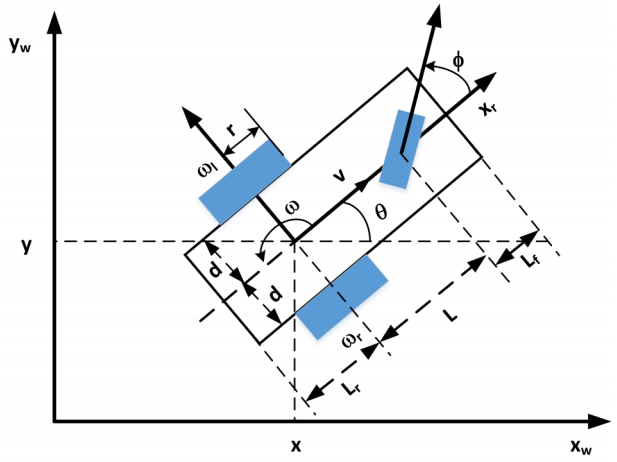


Figure 7: Kinematics of the car [2].

This car-like kinematics can be described as shown in equation 12.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sen(\theta) & 0 \\ \frac{tan(\phi)}{L} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V \\ \omega_s \end{bmatrix} \tag{12}$$

To get a better representation of the variation in the state, it was decided to use the Trapezoidal rule that results in the model equation 13.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ \phi_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ \phi_k \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} cos(\theta_{k+1}) & 0 \\ sin(\theta_{k+1}) & 0 \\ \frac{tan(\phi_{k+1})}{L} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{k+1} \\ \omega_{sk+1} \end{bmatrix} + \begin{bmatrix} cos(\theta_k) & 0 \\ sin(\theta_k) & 0 \\ \frac{tan(\phi_k)}{L} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ w_{sk} \end{bmatrix} \right) \tag{13}$$

4.1.2 Car simulation

Concerning the car simulation, in order to improve realism to the simulated environment some non-idealities were added to the system. This is important because it enables the user to study the situations that can occur in real life that may cause some problems with the system's performance. Thus, noise was considered in the liner velocity and in the angular velocity of the steering wheel. There was also considered an *offset* that represents the errors caused by the mechanical components of the vehicle, for example, the motor, the wheels axis, etc.

4.2 State prediction

4.2.1 Sensors and Data Acquisition

In order to get the most realistic possible state of the car, some sensors were used and modelled, so the correct measurements of each one of the sensors can be obtained. The 6 sensors that were used are the following: GPS, IMU, Speedometer, Encoder, Potentiometer and Tachometer.

The value that the GPS ("*Global Positioning System*") gives is the location, in coordinates, of the car. With the IMU ("*Inertial Measurement Unit*") it is possible to get the acceleration, angular acceleration, angular velocity and also the orientation. The speedometer measures the linear velocity of the car. The encoder measures the angular velocity and position of the steering wheel. The potentiometer reads the angular position and the tachometer estimates the angular velocity, also of the steering wheel.

From the values of the state of the car obtained through a simulation, for each iteration, together with the sensors, some parameters are then estimated and some noise is added. This process of data acquisition from the sensors is implemented in the function *sensor_kit*.

4.2.2 State estimator through Sensor Data

To obtain the estimation of the system's state, the information from the sensors was divided into 3 groups, which are the following: velocity and position, orientation of the car and its variation and, lastly, orientation of

the steering wheel and its variation. For each one of these groups, a Kalman filter was designed and implemented. This corresponds to the function *State_predict*.

Regarding the first group, in order to estimate the velocity and position, the value of the acceleration and the previous state of the car were used. Here is where the Kalman filter takes action, the values of the GPS and the speedometer are also used, to correct the deviations of the estimation of the model. The code for this is function *pos_tracker*.

Considering the second group, the estimation of the orientation of the car and the values of the angular velocity are obtained by using the angular acceleration provided by the IMU sensor. Then, the Kalman filter is applied, so that deviations of the estimation of the model are minimized. The code is in the function *teta_tracker*.

Now on the third group, the orientation of the steering wheel is an extremely important measure. As such, in order to minimize the errors, both position and its variation were obtained in duplicate from different sensors, the encoder, potentiometer and tachometer. In this case, from the values measured for the velocities of the steering wheel's rotation, two values for the orientation of the steering wheel are predicted. This is where the Kalman filter is applied. This filter minimizes the difference between each predicted value and the average of the measures from the sensors. It also tries that the predicted values are as close as possible, that is, the subtraction is equal to zero. The code corresponds to the function *phi_tracker*.

4.2.3 Sensor Failure

In order to make the simulation as close to reality as possible, some sensor failures were taken into account. This failures were implemented probabilistically. The probability of failure is very low, being 0.001%, except for the GPS sensor where the probability of failing is 70%, due to the fact that the GPS provides information in a slower rate and that the GPS might not provide signal because of its surrounding environment.

Also, some switches were implemented, so that it is possible to simulate eventual temporary or permanent breakdowns in the sensors.

The prediction of the state can be affected in different ways by the failures of the sensors, which are the following:

• Position Estimation:

If one of these sensors fail, the speedometer or the GPS, and the other is working, the correction of the state is made only with the values from the functional sensor. If both of these sensors fail, there is no correction made and the state is calculated only by the model of the car. Beyond this failures, if the IMU also fails, meaning that the linear acceleration cannot be known, it is impossible to calculate the state of the car and the signal to stop the car is emitted.

• Car's Orientation Estimation:

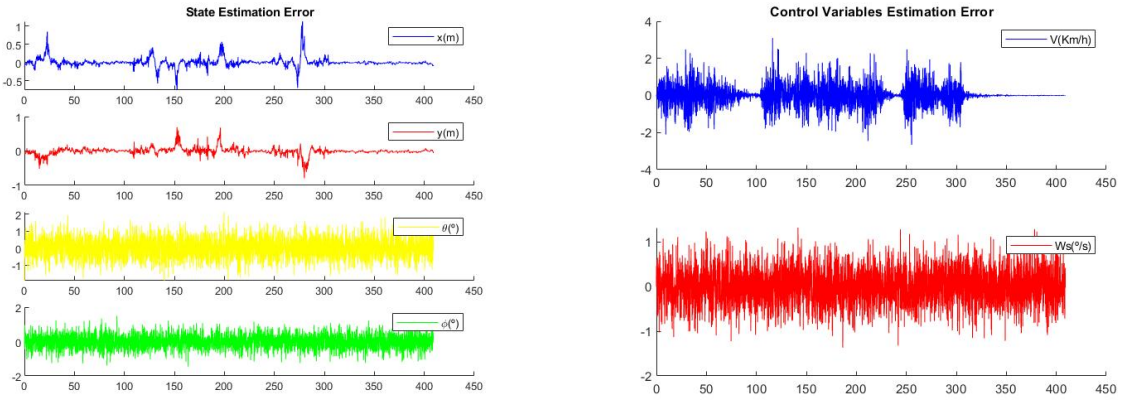
In the case of IMU failure, the state of the car is estimated using the θ row of the model of the car equation 12. Thus, there are no values to correct the estimation and, as the iterations increase, the estimation's quality decreases.

• Steering Wheel's Orientation Estimation:

As in the position estimation section, when none of the sensors relative to the steering wheel are functional, a signal to stop the car is emitted. Otherwise, the state of the car is calculated and corrected using the available

information from the sensors that are operational.

4.3 Navigation Results and Comments



(a) State estimation error

(b) Control variables estimation error

Figure 8: Navigation results

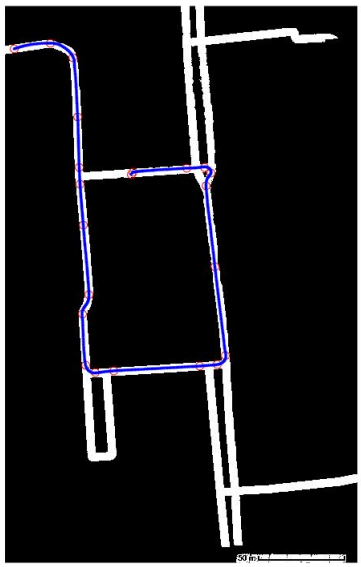


Figure 9: Trajectory of the car

To get a better understanding of the results and to test the estimation of the state, it was established a path for the car, shown in Figure 9, that has some challenges, that is, variations in the state and in the control variables. It is important to notice that these simulations were made not using the switches implemented in order to create more sensor failures.

In Figure 8, it is possible to see that the estimation error is very close to zero, so it can be concluded that a good estimation of the state and (V , ω_s) was obtained. The error of the state variables x and y present some peaks that are quickly attenuated.

Overall, the system is working as expected and good navigation results were obtained.

5 Graphical User Interface (GUI)

This section and corresponding code was made by the group composed by Gonçalo Maia No.87001, Gonçalo Morais No.87004 and Tiago Brito No.87125.

In this section, the creation of the user interface that allows to a smother presentation of the program and its usage will be explained.

As stated before, in order to ease the use of the program as a whole, a Graphical User Interface (GUI) was created resorting to the "App Creator" tool of Matlab.

An image of the implemented GUI is showed below.

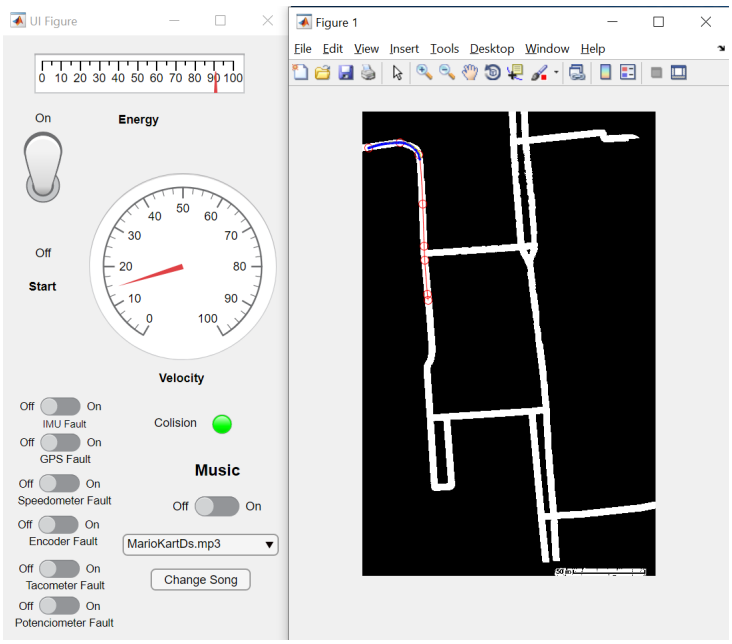


Figure 10: Graphical User Interface

In order to boot start the program, the user should turn on the toggle switch to the "Start" position.

After this, the black and white map of Técnico facilities (explained before) will pop-up. At this point, the user should use the pointer to select a starting and finishing point. Then, the program will show within the same map, the planned trajectory with the respective nodes of the graph in red, and the "real time" movement of the car will be appearing in blue.

At the same time as the car is "moving" in the map, the values of energy and velocity are constantly being updated. On the energy bar on top, which is a simple energy gauge, the percentage of remaining energy is showed. The current velocity at which the car is moving is displayed on the speedometer, which is a velocity gauge element.

To restart the program and select a new trajectory, the user should simply turn off and on again the "Start" switch.

In order to better demonstrate the potentialities of the project, the sensors' failures feature was made optional. To turn them on, the user simply needs to turn the Sensor Fault Switch correspondent to the sensor failure that is intended to be simulated on. This feature can be turned on/off at will. If by any chance, the car has a collision, the "Collision" light will turn red to indicate that. Otherwise, this light is defaulted to green. It is worth mentioning that this collision detection feature was made by checking the value of the current cell

of the grid in which the car is moving at the current moment, that may correspond to road or building.

Lastly, as a ludic feature, a music player was added to the console. The music player consists in a pre-added set of songs from which the user can select one and play it during the execution of the program. To turn on the music, the user simply needs to select a song from the list and turn on the "Music" switch. If the user intends to switch the song without turning off the music, he should select a song from the list and click on the "change song" button. To turn off the music, it's only necessary to turn off the "Music" switch.

6 Conclusion

In this project, the combination of the work performed by all groups allowed the creation of a system that emulates the basic real functioning of an autonomous car. This allowed all the groups to adapt the various theoretical concepts and ideas presented in class in a more practical manner. As mentioned before, the automation of the map information and decomposition were the only necessary steps left to a complete development of a almost fully autonomous car.

The excepted outcomes of the projects were met. We were able to demonstrate that the car moves between two admissible points within its energy budget. The trajectory also lies inside the road areas, without collisions. Nevertheless, if any collision occurs it is properly signalized. The behaviour of the car upon the detection of any of the 4 events is also adequate. Therefore, results were in accordance with the expected, making this project robust and reliable.

In a didactic perspective, this project offered a complete overview of the necessary work needed to put such project in place, as well as all its different subsystems and corresponding correlations.

References

[1] J. Sequeira, "2nd lab assignment, Autonomous Cars", Spring 2021

[2] J. Sequeira, "Lab Guidance", Spring 2021

[3] J. Sequeira, "Lecture 8, Robotics", Spring 2021

[4] J. Sequeira, "Lecture 10/11, Robotics", Spring 2021

[5] J. Sequeira, "Lecture 14, Robotics", Spring 2021

[6] J. Sequeira, "Lecture 17, Robotics", Spring 2021

[7] J. Sequeira, "Lecture 18, Robotics", Spring 2021

[8] J. Sequeira, "Lecture 19, Robotics", Spring 2021

[9] J. Sequeira, "Lecture 20, Robotics", Spring 2021

[10] Norvig, P., Russel, S. (2010) Artificial Intelligence: A Modern Approach, 3rd Edition. Pearson