Formula Student Driverless: The autonomous systems in the Skidpad event.

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Alvaro Linuesa Cabré

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATION ENGINEERING

Advisor: Albert Aguasca

Barcelona, June 2020

Abstract

In this thesis, it will be analysed some of the main algorithms used for one of the events in the Formula student Germany competition in the driverless category. This event is called Skidpad. It consists on an 8 shaped track and the car has to complete the track as fast as it can.

The algorithms that will be analysed are the SLAM, the path, the planner, the controller and the main Skidpad algorithm.

Resum

En aquesta tesis, s'analitzaran alguns dels algorismes principals utilitzats per una de les proves de la competició Formula Student Germany en la categoria Driverless. Aquesta prova s'anomena Skidpad. Consisteix en un circuit en forma de 8. L'objectiu es completar-lo en el menor temps possible.

Els algorismes analitzats son el SLAM, el path, el planner, el controller i el Skidpad algorithm.

Resumen

En esta tesis se analizarán algunos de los algoritmos principales utilizados para una de las pruebas de la competición Formula Student Germany en la categoría Driverless. Esta prueba se llama Skidpad y consiste en un circuito con forma de 8. El objetivo es completarlo en el menor tiempo posible.

Los algoritmos analizados son el SLAM, el path, el planner, el controller y el Skidpad algorithm.

Acknowledgements

This project would not had been possible without the people forming the ETSEIB Motorsport team. Specially Adria Lopez, Albert Gassol, Pablo de Juan, Pilar Martí, Jordi Espasa and Felix Martí who worked with me in the Autonomous Control section and developed many of the essential algorithms.

Albert Aguasca was also essential for the project. He is our project supervisor and is always eager to help us.

Revision history and approval record

Revision	Date	Purpose
0	14/05/2020	Document creation
1	27/05/2020	Document revision
2	13/06/2020	Document revision
3	27/06/2020	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Alvaro Linuesa Cabré	Alvaro.linuesa96@gmail.com
Albert Aguasca	aguasca@tsc.upc.edu

Written by:		Reviewed and approved by:	
Date	27/06/2020	Date	27/06/2020
Name	Alvaro Linuesa	Name	Albert Aguasca
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

1. Introduction

This thesis consists on the analysis of the main algorithms needed to complete the Skidpad event in a Formula Student competition. This thesis is made by Alvaro Linuesa, member of the autonomous control section of the ETSEIB Motorsport team. But first, let me introduce to you to the ETSEIB Motorsport team and to the Formula Student Germany competition.

All the algorithms described in this thesis will run on a single seater car named XALOC.

Figure 1- The car "XALOC"

1.1. ETSEIB Motorsport

This project is being developed in the ETSEIB Motorsport team. ETSEIB Motorsport was founded on 2007. It is formed entirely by students from different schools of the UPC, basically from the ETSEIB school and the ETSETB school. The team is 13 years old. Their goal is to design and build a single seater formula style car.

In their origins, they made combustion cars. In the 2011/2012 season, they built their first electric car. Finally, on the 2018/2019 season, they decided to jump into the Driverless category and made the first Formula Student Driverless car in Spain, the XALOC.

In the current 2019/2020 season, the team is formed by two sub-teams. The EV team and the DV team. This project is being developed in the DV team.

The team is formed by 20 students from ETSEIB and ETSETB. These students are distributed in 4 sections: management, perception, hardware and control.

Each section has some specific roles.

- Management: This section is in charge of the team organization, the treasury and the contact with the sponsors. They also are in charge of representing the team in the Business Plan event and the Cost event.

- Perception: This section can be seen as the eyes of the car. They use a LIDAR and two cameras to acquire the information from the track and format it to the required format.

- Control: This section can be seen as the nerves and the brain of the car. From the information obtained by the perception section, the section designs the algorithms for the control of the car.

- Hardware: This section oversees all the electronic and mechanical parts of the car.

1.2. Formula Student Germany

Formula student Germany is a competition which challenges students from all over the world to design and build a single seater racecar. From the organization it is described as:

"Students build a single seat formula racecar with which they can compete against teams from all over the world. The competition is not won solely by the team with the fastest car, but rather by the team with the best overall package of construction, performance, and financial and sales planning.

Formula Student challenges the team members to go the extra step in their education by incorporating into it intensive experience in building and manufacturing as well as considering the economic aspects of the automotive industry. Teams take on the assumption that they are a manufacturer developing a prototype to be evaluated for production. The target audience is the non-professional Weekend-Racer. The racecar must show very good driving characteristics such as acceleration, braking and handling. It should be offered at a very reasonable cost and be reliable and dependable. Additionally, the car's market value increases through other factors such as aesthetics, comfort and the use of readily available, standard purchase components.

The challenge the teams face is to compose a complete package consisting of a well-constructed racecar and a sales plan that best matches these given criteria.

The decision is made by a jury of experts from the motorsport, automotive and supplier industries. The jury will judge every team's car and sales plan based on construction, cost planning and sales presentation. The rest of the judging will be done out on the track, where the students demonstrate in a number of performance tests how well their self-built racecars fare in their true environment."

The competition consists on achieving the maximum points as possible. It has three categories: Electric Vehicles (EV), Combustion Vehicles (CV) and Driverless Vehicles (DV). The cars entering each category must follow a set of rules to ensure the security of all the competitors and to regulate noncompetitive behaviors.

The competition is divided into two main blocks, the static events, and the dynamic events.

In the static events, the economic and technical aspects are valued. The static events are the Business Plan event, the cost and manufacturing event and the engineering design event.

In the dynamic events, it is valued the behavior of the vehicle in the track. In the DV category, the dynamic events are the Skidpad, the acceleration, the autocross, the trackdrive and the efficiency test. The tracks of these events are made with cones from 3 different colors. The blue cones are always on the left side of the car, the yellow cones are always on the right side of the car. Finally, the orange cones represent the beginning and the endings of the track.

Each of the events have an own punctuation. They are shown in the table below.

	CV & EV	DV
Static Events:		
Business Plan Presentation	75 points	75 points
Cost and Manufacturing	100 points	100 points
Engineering Design	150 points	300 points
Dynamic Events:		
Skid Pad	75 points	75 points
Acceleration	75 points	75 points
Autocross	100 points	100 points
Endurance	325 points	-
Efficiency	100 points	75 points
Trackdrive	-	200 points
Overall	1000 points	1000 points

Table 1- FSG punctuations

In this thesis, it will be described the algorithms used in the Skidpad event. The Skidpad event consists on an eight shaped track. The car must perform two laps on the right circle and exit it, then it must perform two laps on the left circle and exit it. Finally, the car must stop in the designed space. The score is obtained by the average of the best time of each lap and using the next equation:

$$Skdipad\ score = 71.5 * \left(\frac{\left(\frac{T_{max}}{T_{team}}\right)^2}{1.25} - 0.8 \right)$$

2. State of the art of the technology used or applied in this thesis:

2.1. ROS

All the algorithms used in the car follow a ROS structure. But what is ROS? ROS stands for Robot Operating System. It is an open source, meta operating system for robots. It offers all the services expected from an operative system, including hardware abstraction, low level device control, implementation of commonly used functionality, message passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing and running code across multiple computers.

They way it works is really simple. There are some algorithms running in the called nodes. These algorithms can publish data or read published data via the topics. A topic is a “place” where data is published. The nodes can subscribe to the topics in order to get the information published in them.

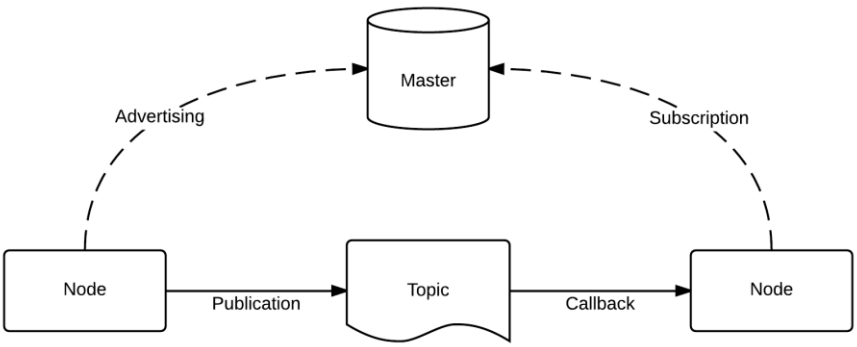

Figure 2- ROS behaviour

The algorithms running in the nodes can be programmed by python or C++.

The pipeline used in this project is really complex and requires many interpret nodes in order to format the information that is being published.

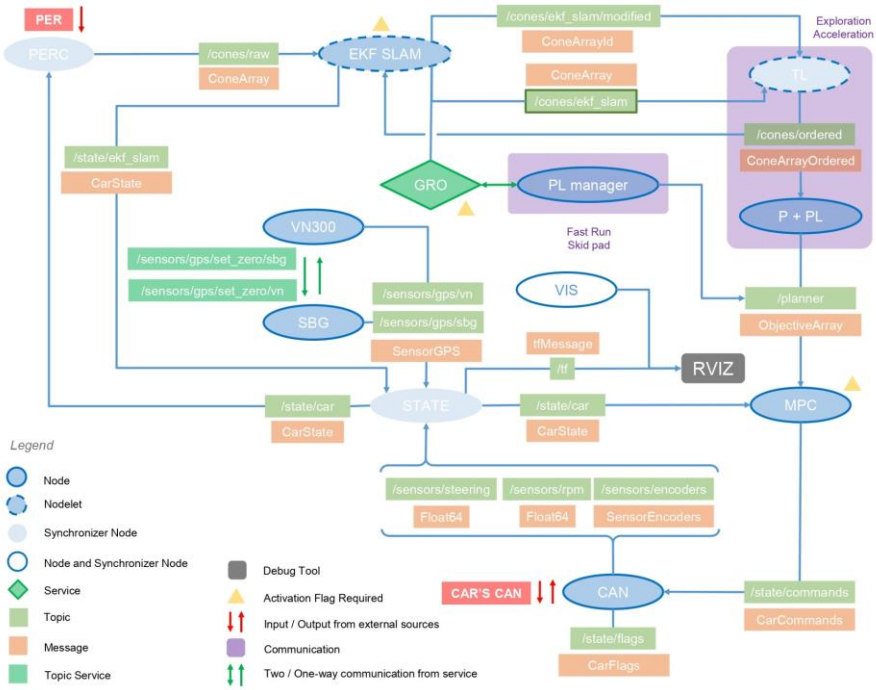

Figure 3- ROS control section pipeline

2.2. GAZEBO

Gazebo is a set of ROS packages that provides the necessary tools to simulate the ROS structures in a virtual environment.

For the development of our algorithms, we have pretty complete simulator that was developed during last season.

Without the simulator it will be impossible to develop any of the algorithms as it is mandatory to test all the programs before loading them into the car.

3. Methodology / project development and results:

The control section has a complex algorithm pipeline to make the algorithms work. There are many interpret nodes to format the information and do some manipulations to the data. For the Skidpad event, there are 5 important modules operating. These modules are the SLAM, the path, the planner, the MPC and the main skidpad algorithm. These 5 modules will be analysed in this thesis.

3.1. SLAM

3.2. What is a SLAM?

Simultaneous Localization and Mapping or SLAM is an intrinsic problem to mobile robots. It consists on mapping the environment of the vehicle while localizing it in the map that is being created. This can be seen as a trivial problem when having ideal data-inputs from the sensors. Unfortunately, noise free measurement doesn't exist in the real world. This noisy measurement complicates the solution to the problem.

An example could be a mobile vehicle in a landmark based environment equipped with some sensors such a LIDAR and some velocity sensors reading in both axes (V_x, V_y). The position of the vehicle $p_i = (X_i, Y_i)$ will depend on the last position $p_{i-1} = (X_{i-1}, Y_{i-1})$ and the velocity $v = (V_x, V_y)$:

$$p_t = p_{i-1} + v_i \Delta t$$

And if we add noise:

$$p_t = p_{i-1} + v_i \Delta t + w_t$$

As the second equation shows, the position estimation accumulates noise at every iteration. This does not only affect the location estimate but also the map created as well. Mapping is a function that totally depends on the current position even if we have a perfect perception system.

To solve this noisy velocity readings, there is a SLAM technique based on ignoring those inputs and use only the perception reading to find its location. This is acquired using an inverse observation function. Knowing a perfectly mapped landmark located in (X_j, Y_j) and we observe that landmark in (X_k, Y_k), we can determine our location accordingly. However, if the measurements are noisy, we have the same problem: accumulative error. Figure 4 shows the error accumulation described.

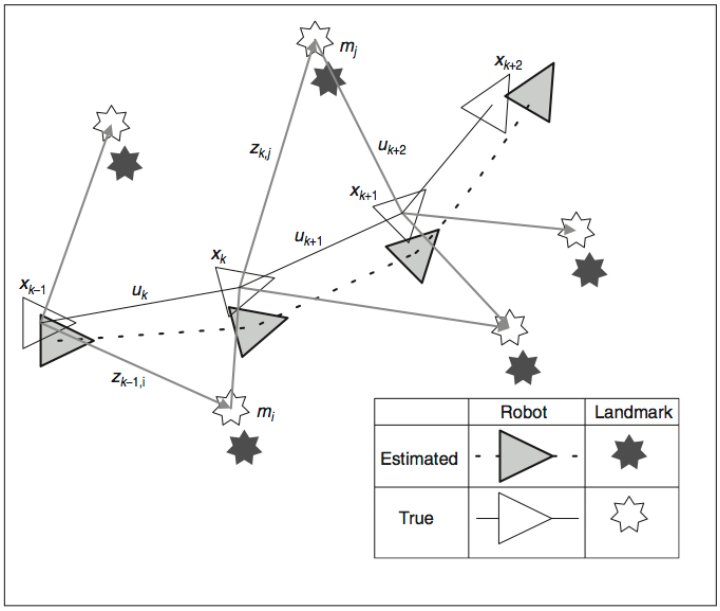

Figure 4- The SLAM problem

This system has some complex mathematics behind. To understand them, it is needed an introduction to some of the basic notation used.

The basic expressions are:

- μ_i: State of the system at the iteration i. This state contains all the information about the position of our vehicle and the map at the iteration i. Let`s assume that our vehicle is moving on a 2D environment formed by some punctual landmarks. The positions can be parametrized as $r_i = (X_i, Y_i, \theta_i)^T$ and the map $m_i = (l_0, l_1, \ldots, l_n)^T$ where each element is a landmark. The state vector will be:

$$\mu_i = \begin{pmatrix} r_i \\ m_i \end{pmatrix}$$

- u_i : Motion controls at the iteration i. They relate the last pose of the vehicle with the current pose. Let's assume that our vehicle moves in a 2D environment according to a cinematic model. The control vector would be $u_i = (v_i, w_i)^T$. The next pose of the vehicle would be:

$$r_i = r_{i-1} + \begin{pmatrix} -\frac{v_i}{w_i}\sin(\theta_{i-1}) + \frac{v_i}{w_i}\sin(\theta_{i-1} + w_i\Delta t) \\ \frac{v_i}{w_i}\cos(\theta_{i-1}) - \frac{v_i}{w_i}\cos(\theta_{i-1} + w_i\Delta t) \\ w_i\Delta t \end{pmatrix}$$

- z_i: Observation at the iteration i. This element provides information about the local environment. Let's assume that our sensors provide coordinates of all the landmarks in its range in polar local coordinates $z_i = (Y_0 \dots Y_n)^T$.
  The mapping process is named the inverse measurement model. The global coordinates of the landmark are a function of the current pose and the local observation.

$$m_i = h^{-1}(\mu_i, z_i)$$

SLAM algorithms can be classified into two main groups, the Full SLAM algorithms and the Online SLAM algorithms.

- Online SLAM: Estimates the posterior pose over the momentary pose along with the map. It provides a map and a pose result for every iteration. It is a real-time algorithm:

$$\mu_i = \underset{r_i, m_i}{argmax}\ [p(r_i, m_i | z_{1:i}, u_{1:i})$$

- Full SLAM: Calculates a posteriori over the entire path along with the map instead of just the current pose:

$$\mu_i = \underset{r_{i:t}, m}{argmax}\ [p(r_{1:i}, m | z_{1:i}, u_{1:i})$$

Our vehicle uses an online SLAM based on an EKF. To understand how it, it has to be described how an EKF works first.

3.2.1. Extended Kalman Filter

An EKF is an algorithm designed to work with Gaussian distributions and linear approximations via Taylor expansion.

The motion and measurement model in an EKF are non-linear functions:

$$\mu_i = g(\mu_{i-1}, u_i)\ , \bar{z_i} = h(\bar{\mu_i})$$

We can use this functions to compute the predicted pose and measurement, but we cannot use them to compute its covariance matrix as the output they produce is no longer Gaussian. We must use the Taylor expansion to linearize and solve this problem. When using Taylor expansion, we convert non-linear models into linear approximations of the models:

$$G_i = \frac{\partial g}{\partial r_i}\ V_i = \frac{\partial g}{\partial u_i}\ H_i = \frac{\partial h}{\partial \bar{\mu_i}}$$

Finally, the EKF algorithm:

$$
\begin{aligned}
&0: Extended\_Kalman\_Filter(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)\\
&1: \overline{\mu}_t = g(\mu_{t-1}, u_t)\\
&2: \overline{\Sigma}_t = G_t \Sigma_t G_t^T + V_t R_t V_t^T\\
&\\
&3: K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}\\
&4: \overline{z}_t = h(\overline{\mu}_t)\\
&\\
&5: \mu_t = \overline{\mu}_t + K_t(z_t - \overline{z}_t)\\
&6: \Sigma_t = (I - K_t H_t)\overline{\Sigma}_t\\
&7: return\ \mu_t, \Sigma_t
\end{aligned}
$$

3.2.2. EKF SLAM

Our EKF SLAM algorithm is inspired in the academic EKF SLAM algorithm from the *Probabilistic Robotics* book, by Sebastian Thurn.

We consider an environment formed by punctual landmarks, from which we receive information about their polar coordinates at every iteration i. Our sensors do not have infinite range, so we receive the landmarks that are in range of them. We also do not know how many landmarks form the environment, so we have a dynamic map that grows every time we map a new landmark. Finally, we do not know about the correspondences between the landmarks, so we have to perform a data association process in order to decide if the landmarks we are receiving have already been mapped or not.

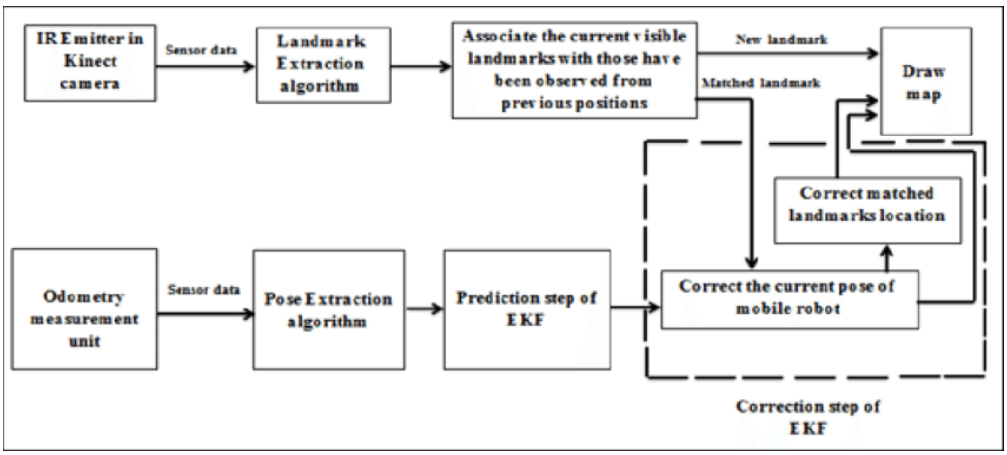

Figure 5- EKF SLAM structure

In our design, we assign and ID to each landmark to help with the landmark management process. This process is done in a peripheral class that contains a signature vector that associates each landmark of the state vector with a given id.

We also use a custom dynamic model to estimate the next translation and rotation at the iteration i. This dynamic model uses another EKF to fuse data from different sensors (phonic wheels, pneumatic pressure, IMU readings…) and a GPS observation (if available) and get an accurate estimation of the next pose. This estimation is re-filtered in the SLAM system to obtain the optimal estimation.

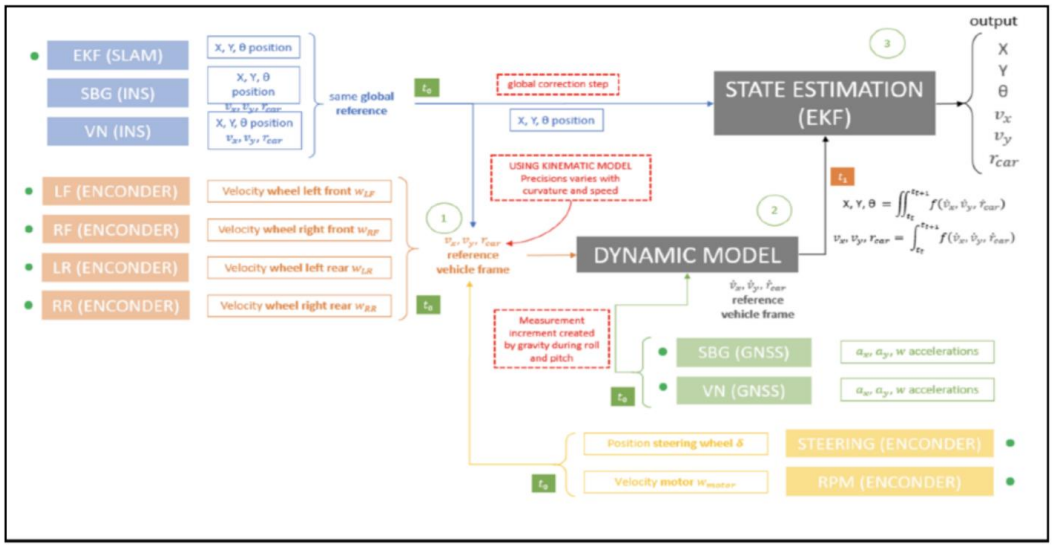

Figure 6- Block scheme of the Dynamic State Estimation system and SLAM system working together

3.3. Path algorithm

To get the correct path that the race car will follow is the next step after getting the limits of the track. This task is done by the Path program.

The purpose of the algorithm is to get the path that goes through the middle of the track limits. Therefore, the inputs of the Path algorithm are the cones that define the limits of the track. The output are the points of the computed path.

This algorithm assumes that the input is correct because the TrackLimits system corrected the possible errors that could had happened. In the Skidpad event that is being analyzed, the Tracklimits algorithm is not used. The reason is because the system used for this specific event, creates its own tracklimits making the Tracklimits algorithm redundant and therefore not necessary.

The main goals of the Path algorithm are:

- To design an algorithm that computes the path with a high probability of rightness.
- To design an algorithm that computes the right path in the following scenarios:
    - The perception system only sees cones of one side of the track.
    - The perception system only sees one cone.
    - The perception system only sees one cone from one side and many from the other side.
    - The perception system sees many cones.
- To achieve a fast and efficient algorithm.

Structure of the algorithm:

The Path algorithm consists on firstly doing a linear interpolation between the cones that define the limits of the track. The next step is to compute the Voronoi Diagram graph with the interpolated sites. And finally, the last step is to get the best trajectory with the information of the graph.

When the TrackLimits system only computes one side of the limits of the track, the Path algorithm gets the other side of the track and computes the best path. To get the other side of the track, the Path program projects the known side limits to the other side. In the Skidpad event that functionality is not used, and a lighter version of the Path algorithm substitutes the full version. This is because, as it will be explained later, the algorithm has all the cones of the track pre-initialized making it impossible to see only one side of the track.

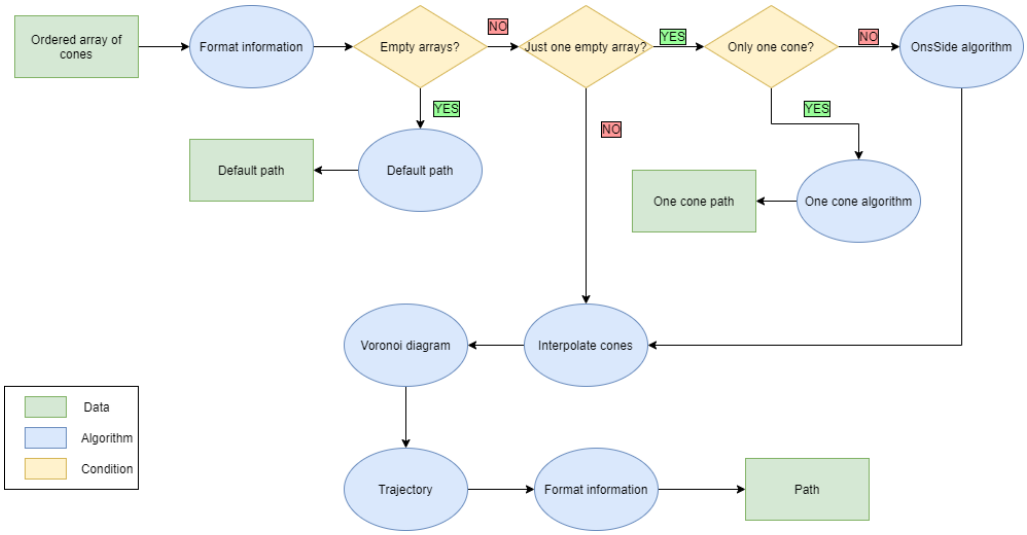

Figure 7- Full path algorithm

As the full algorithm will not be used, only the light version will be described.

The first step of the algorithm is to check if the cone arrays are empty. In the Skidpad event, this is an impossible scenario, so the arrays will never be empty.

After the array analysis, an interpolation must be performed. This is because the trajectory that the Path algorithm obtains is formed by many edges of a Voronoi Diagram graph. The edges of the limits of the track tend to be very large. If these edges are used to get the trajectory, the obtained path would not be smooth. To get a more accurate trajectory, sites are added so that the edges of the Voronoi Diagram graph become shorter. Once we join all the edges the trajectory is much smoother and accurate.

To build this smoother trajectory a linear interpolation of the cones is performed. The criteria used to do this interpolation is to have 3 cones per meter. To calculate the number of cones that have to be interpolated between two real cones, the following equation is used:

$$N = \lceil d(c_1, c_2) * 3 \rceil$$

Where N is the number of interpolated cones and $d(c_1, c_2)$ is the distance between two consecutives real cones.

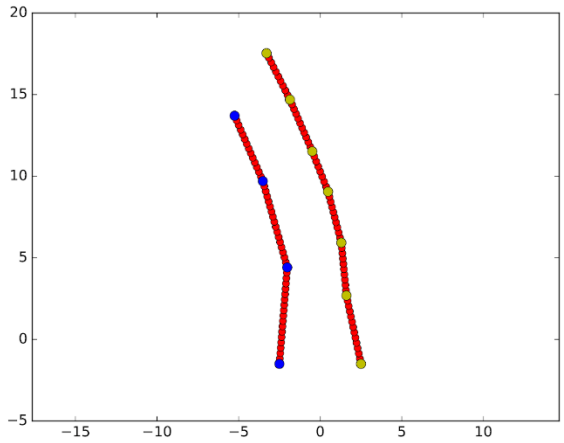

Figure 8-Interpolated cones (red) and input cones (yellow and blue).

After the interpolation process, the cones are numerated in ascending order starting with the blue cones.

After the interpolation, the algorithm needs to create a Voronoi Diagram. The Voronoi Diagram graph of the interpolated set of sites helps getting the path that follows the center of the track. The size of the graph increases with the number of sites. The interpolated cones graph is also much more precise regarding the trajectory we want to follow. This graph contains edges that follow the center of the track. If the edges are segregated and joined the together in the correct order, an accurate path will be obtained.

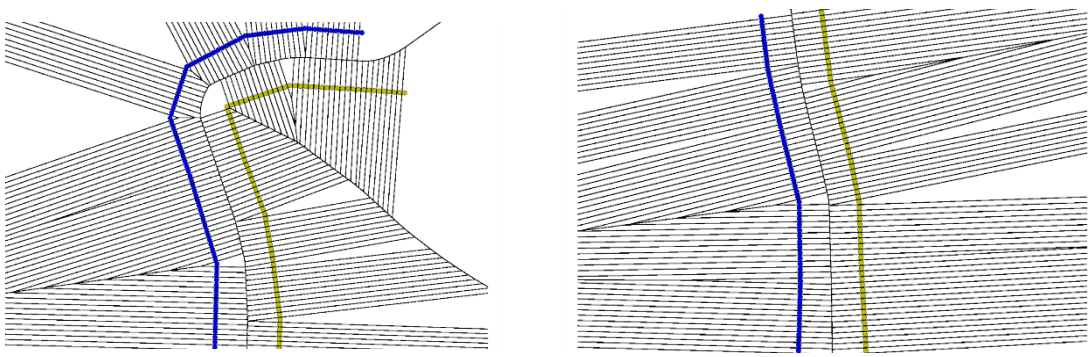

Figure 9- Voronoi diagram

In the images, it can be seen how the edges are combined and joined together to create a line following the center of the track. The next step will be to obtain this line by segregating the edges and obtain the final trajectory.

To do so, we must pay attention to the graph structure of edges, vertexes and sites. Two consecutive edges have a vertex in common and two consecutive vertexes have an edge in common as well. Now that this is known, the closest vertex to (0,0) has to be found and declare it as the initial vertex. Once the initial vertex is known, we find the initial vertex edge that is linked to two cones of a different color (blue and yellow) and with the highest index. The indexes are compared because there could be two initial vertex edges linked with cones of different color. The edges can point forward and backward. We want to get the edge that points forward. To do this, we compare the indexes of their linked cones and decide which is the one that is pointing.

Now, the first two vertex of the trajectory which are linked by an edge are known. From this point, we recursively find the edge of the last vertex found that is linked to two cones of a different color and that is different than the last edge found.

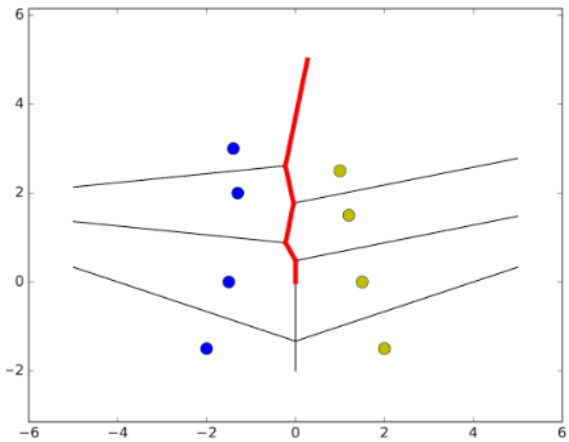

Figure 10- Trajectory algorithm

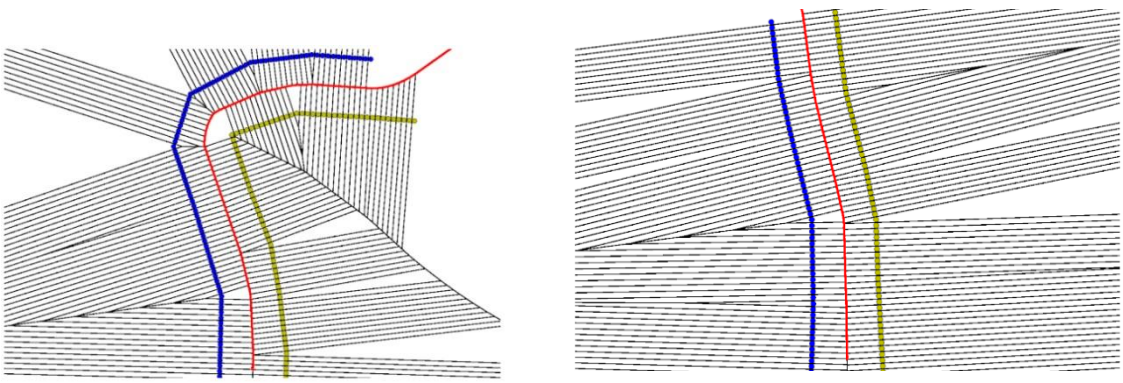

Figure 11- Trajectory output

3.4. Planner

Another of the algorithms involved in the Skidpad is the called planner. A planner is a routine of the vehicle that tells it where to go. It can be seen

The planner can be seen as a series of checkpoints that tells the vehicle what its next objectives are and how to get to them. The points contain information of the state of the vehicle like its velocity. These checkpoints must be equally spaced in time by ΔT, the control frequency. This means that even if the distance between two points it's not the same because the speed may vary, the time that passed between these two objectives will be always the same.

In this project, it is used a velocity variable planner. In comparison to other planners, the one used for the project does not look for the optimal trajectory as it is already given by the path module. Thanks to this, a lot of computational time is reduced as many of these processes are iterative. Due to the narrowness of the track, it would not make a big difference to cut through the corners. It is possible to obtain an optimized trajectory once the car has completed a lap but in the Skidpad event, this upgrade is useless and was discarded when designing the algorithm.

The planner performs two main tasks. The first task is to format the chaotic information received from the path module. Then, it decides the speed of the vehicle at every checkpoint.

To obtain the velocity profile, it will be used the lateral acceleration as a velocity restriction. A car can have some slip problems when turning at high speeds due to high lateral forces. Slowing down can help to reduce those forces. Using the lateral acceleration formula, longitudinal velocity and lateral acceleration can be related by the turning radius.

$$a_{lateral} = \frac{v^2}{R}$$

Fixing the maximum lateral acceleration, it is obtained the maximum velocity at a given point by knowing its radius.

$$v_{max} = \sqrt{a_{lateral}^{max} * R}$$

To obtain the velocity, the system must get the curvature at each position down the center line obtained by the path system. This center line may seem smooth but at closer look, it has some sharp edges. To get the velocity profile, those edges must be eliminated. This is done by applying a zero-degree spline.

Once the initial path has been smoothed, a new set of points have to be obtained. The number of points that will be obtained is governed by the sampling variable. This variable must be high enough to assure a continuity of checkpoints but without compromising the processing time.

The points in the spline line can be calculated by a parameter comprised between 0 and 1. The set of points is automatically obtained by a spline function. The values obtained are equally spaced using the sampling variable, but the product will not be.

Given three consecutive points of the smoothed path, it can be calculated the radius of the circle they define. The equation used to determine this value returns the curvature. By knowing that the curvature is related to the radius, it can be obtained the maximum theoretical velocity at a point via an inverse function.

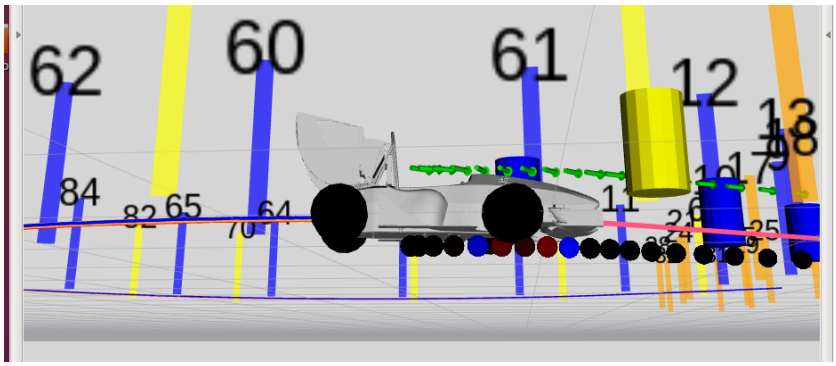

Figure 12- Planner output

The balls in figure 12 represents the velocity on each point.

3.5. MPC

This algorithm triggers once the vehicle knows its position (SLAM) and where it wants to go (PLANNER). Then the vehicle will need to know to steer to get to its desired position and ask the motor for the desired torque. This process is done in the controller node.

The controller used for this step is an MPC. Unlike many different controllers, the MPC objective is to get the position in which the steering should be to ger from A to B and how much throttle is required to achieve a reference velocity.

There are many different types of MPC. They have as an advantage the possibility to plan many steps in advance. They use optimizers to minimize the difference between the controlled system and its planned task. The optimizer finds a solution to a problem given a certain cost function and some restrictions. In the MPC, the restrictions define the plant of the system we want to control. In our case, it is a kinematic model of the vehicle. In this type of model, the forces acting on the system are not considered. The plant is defined by the velocities. This means that the wheels never loose grip with the ground and the vehicle goes accordingly to the steering position.

The difference between the solution obtained and the planned task can be measured by a set of indicators that will define the cost function. Before the execution of the solver, the MPC receives the planned task, which is composed by a series of reference points that contains information of position and velocity.

The kinematic model used in the MPC is a model based on the bicycle model. It is an efficient simple and fast system to control a driverless vehicle.

As our vehicle is a 4 wheeled car, an equivalent wheel has to be created for each pair of wheels. This is done by averaging the angle and speed of each pair of wheels.

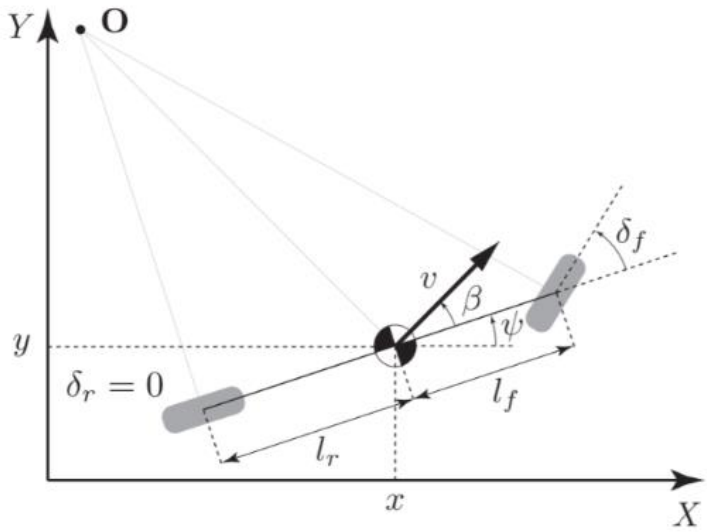

Figure 13- Biccyle model

The variables that compose the model are x, y, ϕ for the position of the vehicle and v for its speed. There are some control variables also, such as a for the acceleration and δ for the angle of the frontal wheel. The model also has some internal components and some constants. These constants, L_r and L_f, defines the distances between the center of gravity (COG) and the front and rear wheel. Finally, the model has a constant to dictate how often the controls can be adjusted, ΔT.

In the following lines, the equations that form the system will be introduced. All of them are expressed with a zero equality as they were introduced in the solver this way.

First, the next position of the vehicle based on its current position needs to be calculated. The factors used for this equation are the current position, the steering position and the velocity. To describe the steering position, it will be used the variable β.

$$\beta = \tan^{-1}\left(\frac{L_r}{L_r + L_f} * \tan\delta_0\right)$$

$$0 = x_1 - x_0 + v_0 * \cos(psi_0 + \beta) * \Delta T$$

$$0 = y_1 - y_0 + v_0 * \sin(psi_0 + \beta) * \Delta T$$

Then, it needs to get updated the velocity:

$$0 = v_1 - v_0 + a_0 * \Delta T$$

The next step is to update the angle of the car. The equation used is:

$$0 = psi_1 - psi_0 + v_0 * \frac{\delta_0}{L_f} * \Delta T$$

The physical constraints of the actuators need to be considered also. As sudden changes are not desired, we can smooth them by adding two inequations on the control variable:

$$maximum\ acceleration\ step \leq a_1 - a_0 \leq maximum\ deceleration\ step$$

$$-maximum\ steering\ step \leq \delta_1 - \delta_0 \leq maximum\ steering\ step$$

The cost function:

The objective of the cost function is to measure the validity of the solution s. The cost function error grows quadratically. This translates into a faster ascending rate as the vehicle derivates more from the planned trajectory. The error is also undefined, this means that a maximum error value is set. Bigger errors than this value grows at a faster rate, while smaller errors grows slower.

The cost function is formed by two modules, Q and dR. Q represents the error related to the state of the vehicle while dR regulates the changes in the control variables.

$$cost\ function = f = W_{slope} * (Q + dR)$$

Each module is regulated by a weight that varies from 0 to 1. Q and dR are weights of each module. All weights are unified. This means that the addition of complementary weights is equal to one.

$$Q + dR = 1$$

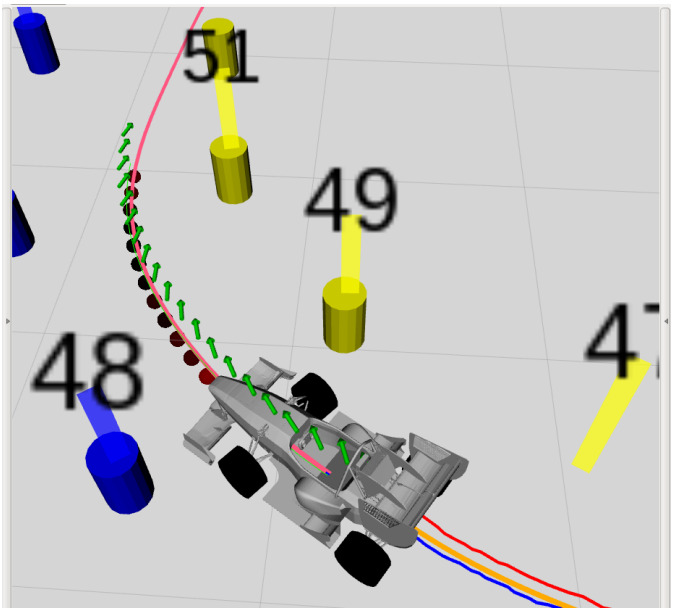

Figure 14- Arrows representing the desired steering angle

Q module:

The difference between the predicted state of the car at one instant and its respective desired state is the way in which the error is obtained.

First, we need to consider the lateral and longitudinal error of the vehicle relative to the planner. The error needs to be measured locally to each planned point. For this, a reference change from global to local will be performed.

$$diff_x = s_x - p_x$$

$$diff_y = s_y - p_y$$

$$diff_{lon} = diff_x * cos(p_{psi}) + diff_y * sin(p_{psi})$$

$$diff_{lat} = -diff_x * sin(p_{psi}) + diff_y * cos(p_{psi})$$

The angles are constraint from 180º to -179º. Even though the angle difference can be obtained by subtracting the planner and the robot predicted angles, the value might not be the expected one. For example:

$$diff_{psi} = s_{psi} - p_{psi} = 180^{\circ} - (-179^{\circ}) = 359^{\circ}$$

$$\sin(-1^{\circ}) = \sin(359^{\circ}) \approx -1^{\circ} * \left({}^{2\pi}/_{180^{\circ}}\right)$$

$$\cos(-1^{\circ}) = \cos(359^{\circ})$$

For the cost function calculations, the outcome is different. Absolute magnitudes are used in the cost function. That means that the smallest absolute value of the angle is used. This is obtained by small angles approximations using the sin function.

To solve this problem, atan2 is used. That function requires two parameters, the opposed and the adjacent sides of the triangle view from the angle we want to find. The reason why the atan2 is used for the small angle approximation is because its higher precision.

$$diff_{psi} = atan2\left(\sin(s_{psi} - p_{psi}), \cos(s_{psi} - p_{psi})\right)$$

Another part of the cost function considers the difference in speed between the predicted state and the reference, the planner.

$$diff_{v} = s_{v} - p_{v}$$

Once we have the differences measured, we can add them up in the Q module. All the differences will be squared as the cost function is quadratic. Each difference will be also multiplied by its weights.

$$Q = Q * (Q_{elon} * {diff_{lon}}^2 + Q_{elat} * {diff_{lat}}^2 + Q_{epsi} * {diff_{psi}}^2 + Q_{ev} * {diff_{v}}^2)$$

dR module:

The error in this module will be obtained by measuring the difference of the control variables between each control step. The objective is to smoothen the robot movements by minimizing the gap between each sequential action of the control variables.

$$step_{\delta} = s_{\delta}^{t+1} - s_{\delta}^{t}$$

$$step_{a} = s_{a}^{t+1} - s_{a}^{t}$$

$$dR = dR * (dR_\delta * step_\delta{}^2 + dR_a * step_a{}^2)$$

There is a need to translate the MPC commands, accelerations or decelerations and steering angles, to send them to the hardware elements as they do not use the same base neither units. We have three controllable elements in the robot: The main motor that receives a torque command, the brake that receives a value between 0 and 1, 1 signaling the hardest braking value, and the steering wheel.

To perform this translation, we will not consider the limitations of the pneumatics as our car does not work near their adherence limit. Using the radius of the wheel and the mass of the vehicle we obtain the main motor commands.

On the other side if the acceleration value is negative, we need to activate the brakes. Since it is plausible that we need to activate the brakes beyond the adherence limit we will consider the pneumatics friction. To determine the position of the brake, from 0 to 1, we have the following parameter: the maximum pressure in the brake line and the ratio between pressure and torque for the front and rear wheels

3.6. Skidpad algorithm

The main idea of the algorithm is to create a perfect map of the track and modify it on the run. Then, using a graph, a trajectory is created.

The algorithm has two main phases, the initialization, and the update.

3.6.1. System initialization

In this phase of the algorithm, the system will initialize all the necessary variables and systems needed by the algorithm. It has two parts: the perfect map and the graph initialization.

3.6.1.1. Perfect map

The first step of the algorithm is the creation of a perfect map. It can be done as we know exactly how the track will be and the dimensions of it as it is given to us in the rules. Using trigonometry, the position of each cone is determined.

Thanks to the possibility to know how the circuit is before the event, it is possible to know the separation, in degrees, and the radius of each circle.

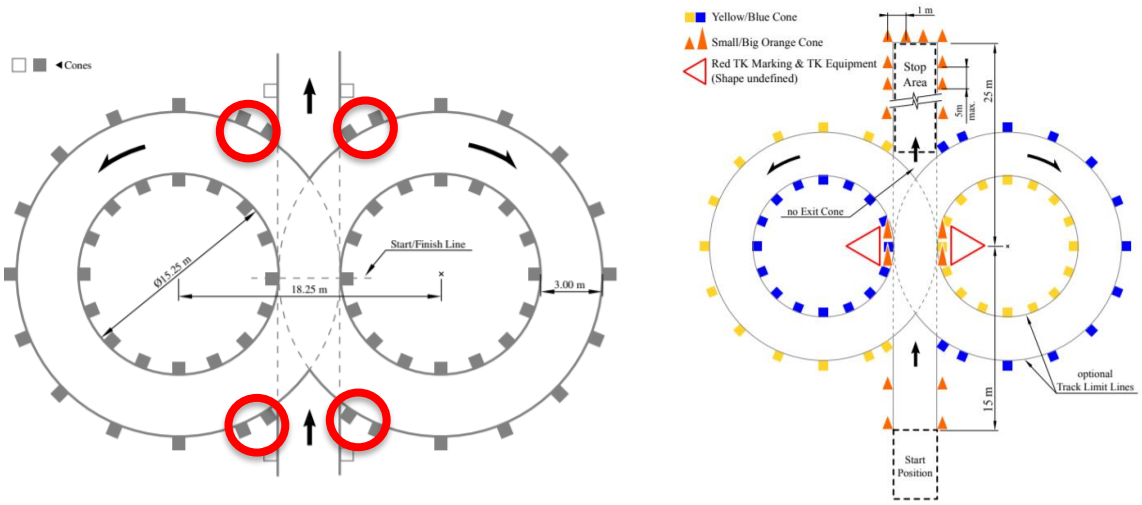

Figure 15- Skidpad layout

As it can be seen in the figure 15, the cones in the circles are separated by 22.5º, except the ones marked in red that have a separation of 9º. With this information, the dimensions of the track and using polar coordinates, the global position of each cone is determined.

To initialize each cone, it is used a struct named Cone. This struct has two attributes, X and Y, which are used for the global position in the map for each cone.

The cones are stored in a vector. The position of each cone inside this vector will be important for future steps of the Skidpad algorithm.

3.6.1.2. Graph creation

The idea of the graph is to create a gate between each pair of cones and connect these gates following a graph structure.

To create the gates, we must link the left cone to the right cone of each pair. As we know exactly where in the Cone vector is each cone, we can link them easily using loop structures.

Once we have all the gates, the next step is to determine the middle point of them. This is determined by performing the average of each coordinate of the pair of cones of the gate.

The next step will be to link the points. The idea is to connect each one to the previous point and the next. This will allow the car to have “objectives”. This means that, for example, when the car is in the gate number 4, it will know that it came from the gate 3 and its next move would be to get to the gate 5.

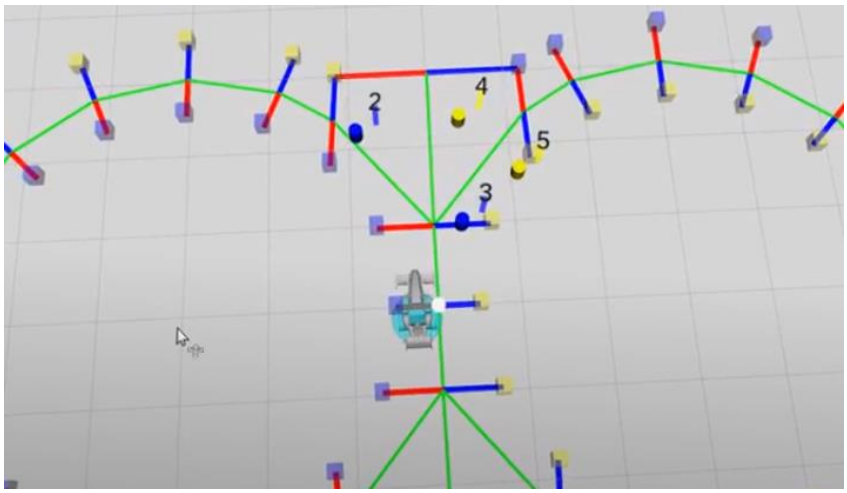

Figure 16- Gates and graph of the algorithm

In the figure 16, the gates are represented with blue and red segments and the connections between them, with green lines.

3.6.2. Update phase

In this phase, a new map will be created from the perfect map and the cones will be updated to their real positions.

While the creation phase only happens once during the execution of the algorithm, the update phase is executed each time a new cone is received.

To update the position of the cones and modify the graph, the algorithm must perform a data association to associate the real cones to the one in the perfect map.

From the SLAM system, the algorithm receives the cones seen by the car and performs a Hungarian Method based data association. This method consists on building a matrix with all the cones of the system. The rows of the matrix represent the cones obtained from the SLAM system. The columns are the cones from the perfect map. Each position of the matrix represents the quadratic distance between a perfect cone and a SLAM cone. The system will select the positions with the minimum distance between cones. Once two cones have been associated, they will not be eligible for future associations.

While performing this data association process, the system also searches for the center of the track. This is done by looking for a rectangle shaped figure formed by four big orange cones. The algorithm is robust enough to find this rectangle by only having two of the corners. When this figure is detected, it will automatically find the

coordinates of the central point and will relocate all the cones in the map to make the centers match. This will allow the algorithm to perform a better data association as both the perfect map and the updated map will have the same central point.

Once the data association is performed, the system must update the graph with the new cones positions. To do this, the algorithm will repeat the procedure followed in the graph creation but with the updated positions.

Finally, the algorithm will format all the information obtained to make it compatible with the next algorithm in the pipeline, the path.

After the path has processed the information, the planner will generate the velocity profile and will give it to the MPC

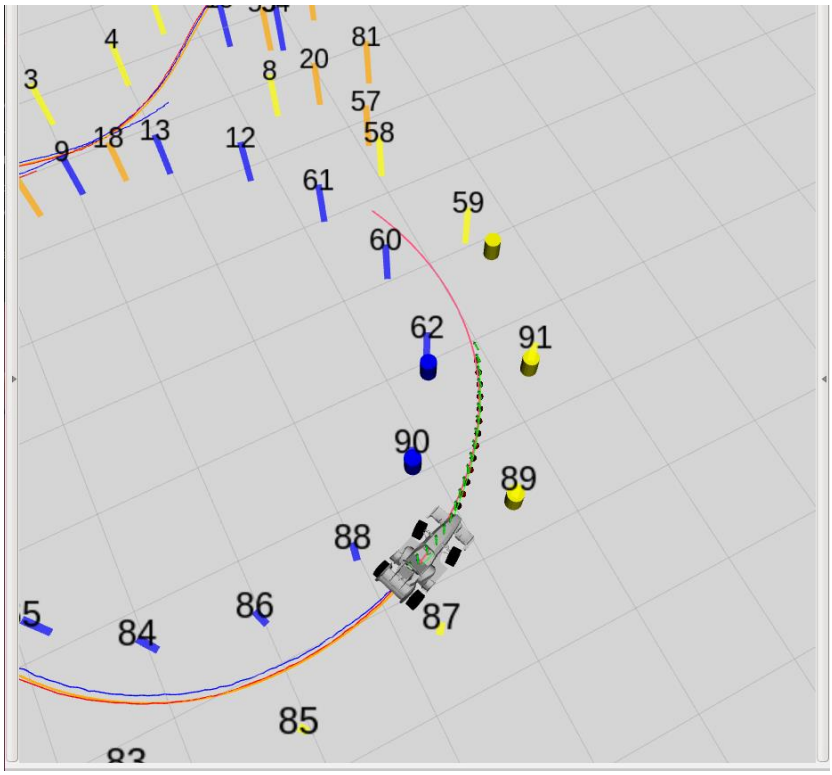

Figure 17- Algorithm running on the simulator

4. Budget

The project described in this Thesis is software based. All the software used is open code, so no licenses were paid.

As this project is a part of a bigger project, it will be included the approximate budget of the whole project for the 2019/2020 season.

DRIVERLESS BUDGET SEASON 2019/20				
	FORSEEN	**REAL COST**	**DIFFERENCE**	*SUBTOTAL*
PERCEPTION MATERIAL				*150€*
Supports	**150€**	8,90 €	141€	
SENSORS MATERIALS				*410€*
Speed sensor (Phonic wheel)	**120€**	0,00 €	120€	
Speed sensor (Optic sensor)	**200€**	0,00 €	200€	
Pressure Brake Pedal Sensor	**90€**	0,00 €	90€	
ACTUATORS				*1.800€*
Steering Wheel actuator	**600€**	0,00 €	600€	
Emergency Brake System	**800€**	484,90 €	315€	
Service Brake	**400€**	17,10 €	383€	
PROCESSING MATERIAL				*750€*
NVIDIA (Image processing)	**750€**	0,00 €	750€	
CAR ADAPTATION				*13.830€*
LEDs & Buttons	**50€**	0,00 €	50€	
Wiring	**300€**	0,00 €	300€	
Battery cells	**1.000€**	0,00 €	1.000€	
Electrical Components	**5.000€**	2.598,79 €	2.401€	
PCB	**400€**	0,00 €	400€	
Aluminum	**300€**	0,00 €	300€	
Carbon Fibre	**100€**	0,00 €	100€	
Milling	**2.000€**	0,00 €	2.000€	
Machinnings	**2.000€**	0,00 €	2.000€	
Vinyil	**200€**	0,00 €	200€	
Brake lines (x2)	**600€**	301,65 €	298€	
Brake calippers(x3)	**1.200€**	836,21 €	364€	
Brake spare parts	**200€**	635,87 €	-436€	
Wheel nuts	**480€**	81,07 €	399€	
OFFICE MATERIAL				*1.000€*
Roll-ups	**500€**	0,00 €	500€	
Office Material	**200€**	53,00 €	147€	
Copy shop	**300€**	0,00 €	300€	
TESTING MATERIAL				*1.100 €*
Cones	**200 €**	0,00 €	200 €	
Wheels	**900 €**	3,94 €	896 €	
TOOLS				*1.450€*
Quick Jack	**100€**	0,00 €	100€	
Push Bar	**100€**	0,00 €	100€	
Workshop tools	**500€**	0,00 €	500€	
Charger	**250€**	0,00 €	250€	
Battery cart	**500€**	0,00 €	500€	
COMPETITIONS				*15.220€*
FSS- Camping	**2.600€**	0,00 €	2.600€	
FSS-Inscription	**1.800€**	1.800,00 €	0€	
FSG-Inscription	**1.300€**	840,34 €	460€	
FSG-Camping	**3.120€**	0,00 €	3.120€	
FSG-Transport	**800€**	0,00 €	800€	
FSC-Camping	**1.300€**	0,00 €	1.300€	
FSC-Inscription	**1.300€**	1.400,00 €	-100€	
Fuel	**3.000€**	0,00 €	3.000€	
TRANSPORT				*1.300€*
Fuel	**1.300€**	0,00 €	1.300€	
TICKETS				*1.500€*
Others	**1.000€**	0,00 €	1.000€	
Workshop Germany*	**500€**	1.024,43 €	-524€	
Competition extras	**0€**	0,00 €	0€	
EXTRAS				
Safety cushion	0€		0€	
TOTAL EXPENSES	**38.510€**	10.086,20 €	**28.424€**	

Table 2- 2019/2020 budget

In the budget above, it is not taken in consideration the budget from the season 2018/2019. That is the reason why many of the sensors and the car itself are not included.

5. Environment Impact

Although this thesis is software related, the algorithms run in a real car, so it is relevant to analyse the environmental impact it has. This analysis was done by the ETSEIB Motorsport team the 2018/2019 season as a part of a requirement from the FSG organization. As the car for the season 2019/2020 has not changed, this analysis will represent in an accurate way the environmental impact of the car for this season.

In this section a complete analysis of the CO2 sustainability of the manufacturing of the car is presented.

Firstly, the overall composition of each component of the car is worked out, getting the kg of materials in each component. Thus, obtaining the total weight of the car filtered by materials.

Vehicle Component	Composition [Kg]								
	CFRP	Kevlar Composite	Aluminum	Steel	Tire Rubber	Adhesive	Copper	Electrical components	Plastic
Rims & Tires			23,22		13,5				
Suspension	2		5,5	2,5					
Monocoque	12		11			3			
Covers	1,5								
Ergonomy	5			3					
Aero-Sidepods	0,8								
Aero- Sides	2,8								
Main Hoop & Braces				6,3					
FH			2						
HV Battery Box		4,82							
Electronics								6,6	
Inverter	0,4		1,5						
Motor			4				10		
Motor&Gear box Support			0,5						
Gearbox			2,13	35					
Pedals	0,23		3						
PU + NVIDIA	0,5			2,5				3	
Steering actuator	0,8			2,5			1,5		
EBS+	1								

Service Support									
EBS				3,2					0,5
Service				2			1,5		
Lidar Support			0,85						
Firewall		0,6	0,2						
TOTAL	*27,03*	*5,42*	*53,9*	*57*	*13,5*	*3*	*13*	*9,6*	*0,5*

Table 3- Composition of the car

Decompositions

In the following tables a more detailed breakdown is given for the electrical components section.

Vehicle Electrical components	Composition
PCB	0,95
Plastic	0,03
Others	0,02

CFRP	Composition
CF	0,7
Epoxy Resin	0,3

Kevlar Composite	Composition
Kevlar	0,6

PCB composition	Composition
Antimony	0,0011
Arsenic	0
Barium	0,0035
Bismuth	0,00014
Chromium	0,000019
Copper	0,24
Gold	0,00031

Iron	0,023
Lead	0,028
Nickel	0,0018
Plastic	0,6428
Silver	0,00044
Tin	0,048
Titanium	0,0014
Zinc	0,0094

Table 4- Electrical components composition

In the following table the regular conversion from kg of material to kg of CO2 is given. It has been extracted from regular papers that cover that subject.

Material Consumption	CO2 / Kg [Kg]
CF	25,02
Kevlar	16,55
Aluminum	8,14
Steel	1,77
Tire Rubber	4,02
Adhesive	3,67
Copper	2,77
Antimony	12,9
Arsenic	0,3
Barium	0,2
Bismuth	58,9
Chromium	2,4
Gold	12500
Iron	1,91
Lead	1,64
Nickel	11,53
Silver	196
Tin	17,1
Titanium	8,1
Zinc	3,41

Battery Pack	CO2 / KWh [Kg]
Manufacturing	109,01
Materials	63,92
	CO2 / KWh [Kg]
Red Electrica España	0,28

Table 5- Conversion from kg of material to kg of CO2

Vehicle Materials		
	Material [Kg]	Material CO2-eq [Kg]
Adhesive	3	11,01
Aluminum	53,9	438,746
Antimony	0,010032	0,1294128
Arsenic	0,00024624	0,000073872
Barium	0,03192	0,006384
Bismuth	0,0012768	0,07520352
CFRP	27,03	676,5609
Chromium	0,00017328	0,000415872
Copper	15,1888	42,072976
Gold	0,0028272	35,34
Iron	0,20976	0,4006416
Kevlar	5,42	89,701
Lead	0,25536	0,4187904
Nickel	0,016416	0,18927648
Plastic	2,6888	9,303248
Silver	0,0040128	0,7865088
Steel	57	100,89
Tin	0,43776	7,485696
Tire Rubber	13,5	54,27
Titanium	0,012768	0,1034208
Zinc	0,085728	0,29233248
TOTAL		**1467,782281**

Table 6- Conversion from the car materials to CO2

Next on, graphics depicting the previous table are presented so as to get useful insights from it.

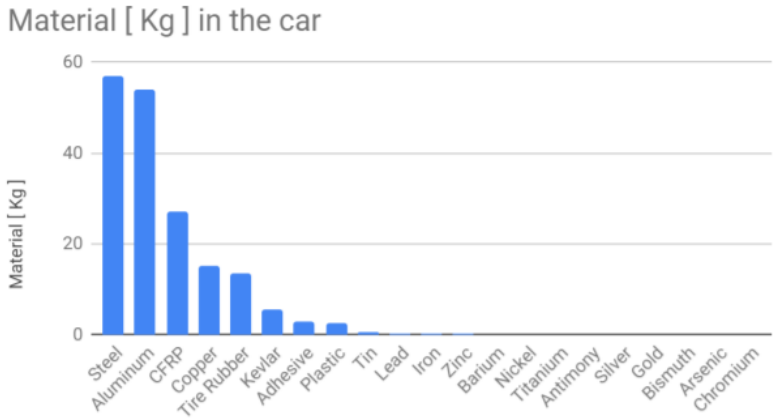

Figure 18- Material contribution to the weight

As it can be seen from the previous graphic the materials that contribute the most to the weight of the car are steel, aluminium and carbon fibre. Copper and tire rubber have also a meaningful influence on the final weight.

Regarding the CO2 emitted the two major contributions come from CFRP and Aluminium, with 676 and 438 kg respectively. Despite that steel is the material with more weight in the car its CO2-eq is lower in comparison with the one of Aluminium and CFRP.

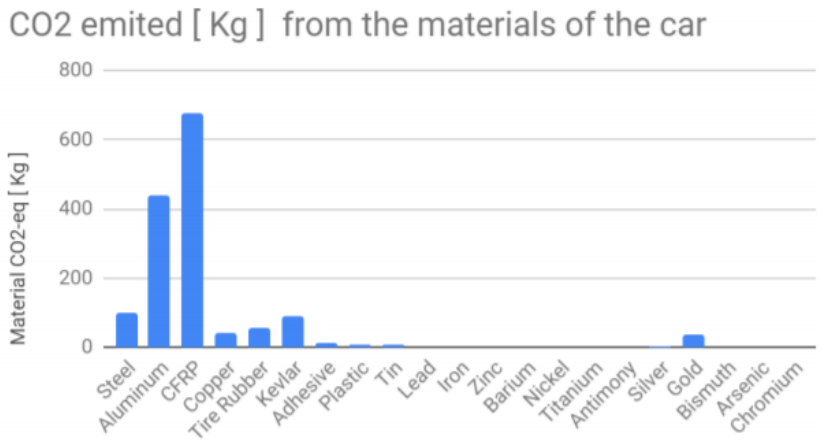

Figure 19- CO2 emitted

This tables summarizes the main processes to build the car and the energy consumed in kWh to carry through those processes. Then, the manufacturing CO2-eq of each process is founded by having the CO2 consumption of a kWh.

Vehicle Manufacturing						
	Process	Power	Manufacturing Time	Required KWh	Amount	Manufacturing CO2-eq
Steering actuator	Machining	15	4	60	1	16,8
Brake pedal	Machining	15	3	45	1	12,6
Monocoque	Autoclave	5	4	20	1	5,6
Steering actuator support	Vacuum Pump + 3D	0,51	32	16,32	1	4,5696
Aero	Autoclave	5	3	15	1	4,2
EBS+ Service Support	Vaccum Pump + 3D	0,51	23	11,73	1	3,2844
Main Hoop & Braces	Welding	5,3	1	5,3	1	1,484
Motor&Gearbox Support	Welding	5,3	1	5,3	1	1,484
Lidar Support	Laser Jet + Welding	5,3	1	5,3	1	1,484
Covers	Vacum Pump	0,15	4	0,6	4	0,672
Electronics	Soldering	0,08	24	1,92	1	0,5376
HV Accumulator	Vacuum pump	0,15	12	1,8	1	0,504
Firewall	Vacuum Pump	0,15	12	1,8	1	0,504
Ergonomy	Vacum Pump	0,15	2	0,3	4	0,336
Acceleration Pedal	Vacuum Pump	0,15	8	31.12	1	0,336
Inverter Support	Vacuum Pump	0,15	8	1,2	1	0,336
Pu + nvidia Box	Vacuum Pump	0,15	8	1,2	1	0,336
FH	Bending	5	0,15	0,75	1	0,21
Rims & Tires	None					0
Suspension	None					0
Inverter	None					0
Motor	None					0

Gearbox	None					0
PU + NVIDIA	None					0
EBS	None					0
Service	None					0
TOTAL						55,2776

Table 7- Energy consumption and CO2 eq.

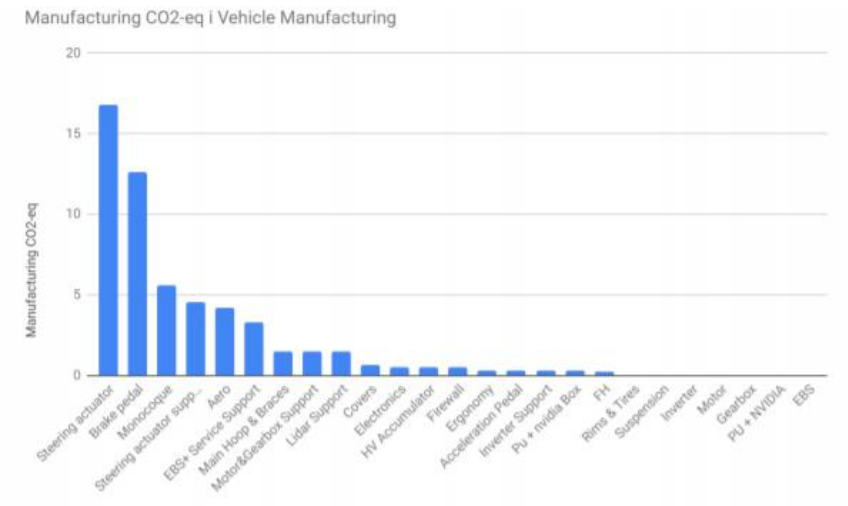

Figure 20- CO2 emitted in the vehicle manufacturing

In the previous graphic the parts that emit more CO2 are listed. These are the steering actuator and the brake pedal, mainly due to machining processes that are very energetically costly.

The same concept is applied for the battery assembling process.

Battery pack	S cells	P cells	Capacity	Voltage	TOTAL [KWh]
LV	7	3	6,35	4,35	0,5800725
HV	138	2	6,35	4,35	7,62381
					8,2038825

Table 8- Energy consumption and CO2 eq from the battery pack

Finally, the total carbon footprint of the manufacturing of the car, including the materials, processes and building required, is calculated.

Battery Materials & Manufacturing	
Battery Pack Materials	524,3921694
Battery Pack Manufacturing	894,3052313
TOTAL	**1418,697401**

Building		
	KWh	CO2-eq
Building Electricity Consumption	3996	1118,88

	CO2-eq [kg]
Processes	55,27
Materials	1467,78
Battery Pack Materials	524,39
Battery Pack Manufacturing	894,30
Building Electricity Consumption	1118,88
TOTAL GWP	4060,63

The main contributions to CO2 emissions are materials, the battery (materials and manufacturing) and the building electricity consumption.

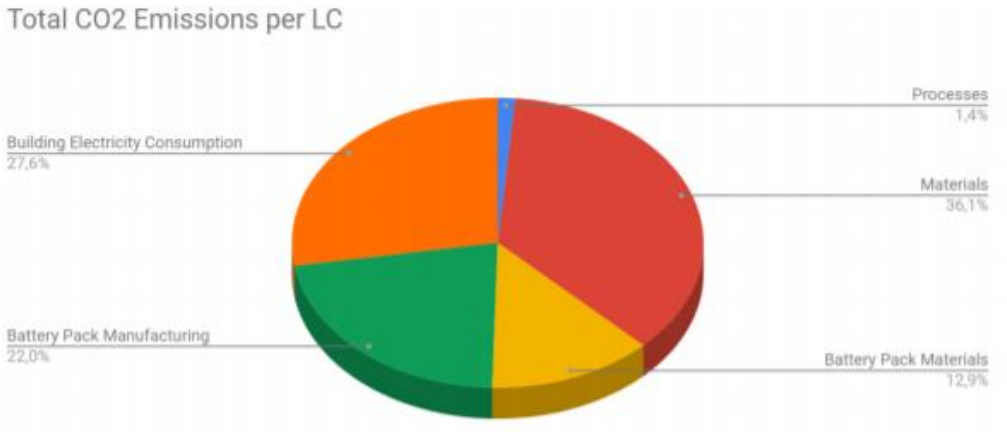

Figure 21- CO2 contribution

The TOTAL GWP takes sense when associated with a Life Cycle Assessment (LCA), which in the case studied is of 5 years.

If you actually consider in your scope the use phase of the vehicle (that is, the system boundaries here are from cradle to the end of the lifetime of the vehicle), then the carbon footprint of the vehicle fuel has to be taken into account. In our case, the fuel is electricity whereas in most cars the fuel is gasoline.

In the previous analysis the carbon footprint of the vehicle fuel has not been taken into account. However, in the case of electric cars the GWP of charging the batteries is pretty small when comparing it with the GPW of gas.

Moreover, the reduction in GWP of recycling parts or materials of the car has not been considered

6. Conclusions and future development:

The algorithms described in the thesis are only a little part of the whole project. It is very big and have many people working on it.

On the near future, the idea is to test all the algorithms in a real track. The algorithms were only tested in a virtual simulator. Because of the COVID19 all of the competitions were suspended and will take place next year.

The idea of using a modular pipeline makes the implementation of the algorithm much easier. Thanks to that structure, many of the designed algorithms can be reused with some modifications to perform in all of the events.

Talking about the skidpad, the central algorithm oversees formatting the data and interconnecting with other core algorithms. Thanks to this, future improvements will be much easier to implement as there will be no need to modify the whole code.

The main ideas to enhance the performance of the algorithms are to try to implement some variation to the algorithms to make them more efficient. That will give us more computational capacity for other algorithms. Another idea is to try to implement some neural networks to the algorithms and have much accurate models.

Also, due to an update to the rules for the 2021/2022 season in which It will be mandatory to have a car with manual and autonomous functions at the same time, the team will begin to organize the fusion of the cars (the not autonomous electric car and the driverless car) to build a rules compliant one.

Bibliography:

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. 2005. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press.

Joan Solà, 2014. Simultaneous localization and mappin with extended Kalman filter.

https://www.formulastudent.de/fsg/

https://www.ros.org/

Most of the references from this thesis are own made.

Appendices:

A video of a simulation of the Skidpad will be included with this thesis. The video shows some not desired behaviour of the algorithm due to the lack of computational power of the computer in which the simulator is running.

In the video it can be appreciated the different algorithms working together and how the objective is accomplished.

Glossary

EV: Electric vehicle

CV: Combustion vehicle

DV: Driverless vehicle

ETSEIB: Escola tècnica superior d'enginyeria industrial de Barcelona

ETSETB: Escola tècnica superior d'enginyeria de telecomunicacions de Barcelona

SLAM: Simultaneous localization and mapping

EKF: Extended Kalman filter

MPC: Model predictive controller

HV: High voltage

LV: Low voltage

PU: processing unit

FH: Front hoop

LCA: Life cycle assessment

CFRP: Carbon fiber reinforced polymer

GWP: Global warning potential