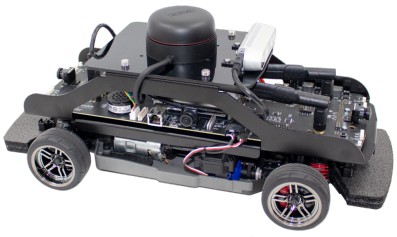
POLYTECH NANCY

Lab 8

Path and lane following control for the self-driving Qcar



Tom Grandjanin*
Hugues Garnier
Mayank Jha

September 2023

*5A IA2R SIA Student in 2023-2024

Acknowledgements

Thanks to QUANSER for providing basic details on QCAR.

Contents

1 Introduction

Autonomous vehicles are smart vehicles that have the capability to have automatic motions and navigate itself depending on its environments and scheduled tasks. Autonomous vehicle systems may differ depending on the environment it is operating on. Flying and aerial vehicles are autonomous vehicles that operate above ground in higher altitude which usually known as unmanned aerial vehicle (UAV). This lab work focuses on the autonomous vehicles that are operating on the ground which also known as autonomous ground vehicle (AGV).

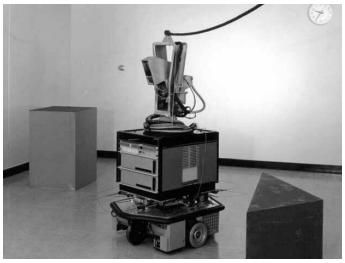


Figure 1: Shavey : Mobile Robotics Pioneer

Figure 2: Curiosity Mars Rover

Autonomous driving has been a real subject of study for years now. Autonomous highway system test were already conducted back in 1950 in America. In 1977, pioneering Japanese driverless car from Tsubuka Laboratory achieved tracking white street lines at 30 kilometers per hour.

Figure 3: Tsukuba Mechanical Engineering Lab, Japan, 1977.

Figure 4: Tesla self-driving car

In developing a successful autonomous ground vehicles, few typical typical challenges need to be addressed by answering the following issues:

- Where am I?
- Where do I want to go?
- How do I get there without hurting myself?

The approach can be summarized in a general manner using the "See-Think-Act" cycle as depicted in Figure 5. This figure also demonstrates the interaction between various functional modules that are involved in successful functioning of an autonomous vehicle.

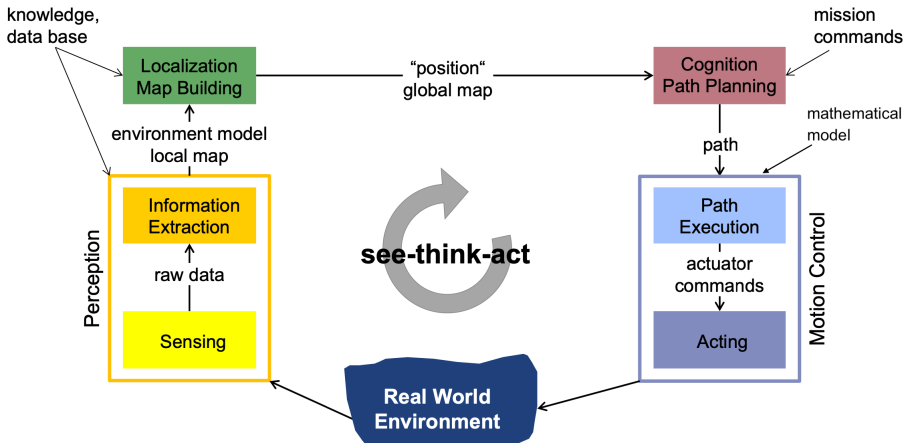


Figure 5: See-Think-Act Cycle, See here for more details: `https://asl.ethz.ch/education/lectures/autonomous_mobile_robots/spring-2021.html`

The various tasks involved can be broken down as shown in Figure 6.
There are three basic stages and modules in the system which are:

- Sensing and Perception: provide real time data to let the system know the real time location and environment around the vehicle and prepare the raw data in format that is feasible for the system to process.
- Planning: use the data provided by Sensing and Perception to dictate the safe and feasible path for the vehicle to follow.
- Control: contain control strategies to move vehicle to the desired path. This include actuator control of each sub-system.

In this lab, the main focus will on:

- **Trajectory tracking** control in the control phase of the autonomous vehicle system and vehicle model. To that end, the autonomous vehicle will be required to follow a specific path and trajectory given by an on-board planner.
  - **Steering Control** using Pure Pursuit: Path following or trajectory tracking controller is usually developed to ensure the vehicle to follow a predefined path and trajectory by determining and calculating the desired actuating input for the vehicle to follow. This can be a correctional steering input to adjust the vehicle's position in lateral direction or correctional braking or throttle setting to adjust vehicle's motion in longitudinal direction.
  - **Speed Control**. The vehicle is supposed to accomplish the steering using desired speed. The speed control will be effectuated using Proportional-Integral (PI) based controllers.

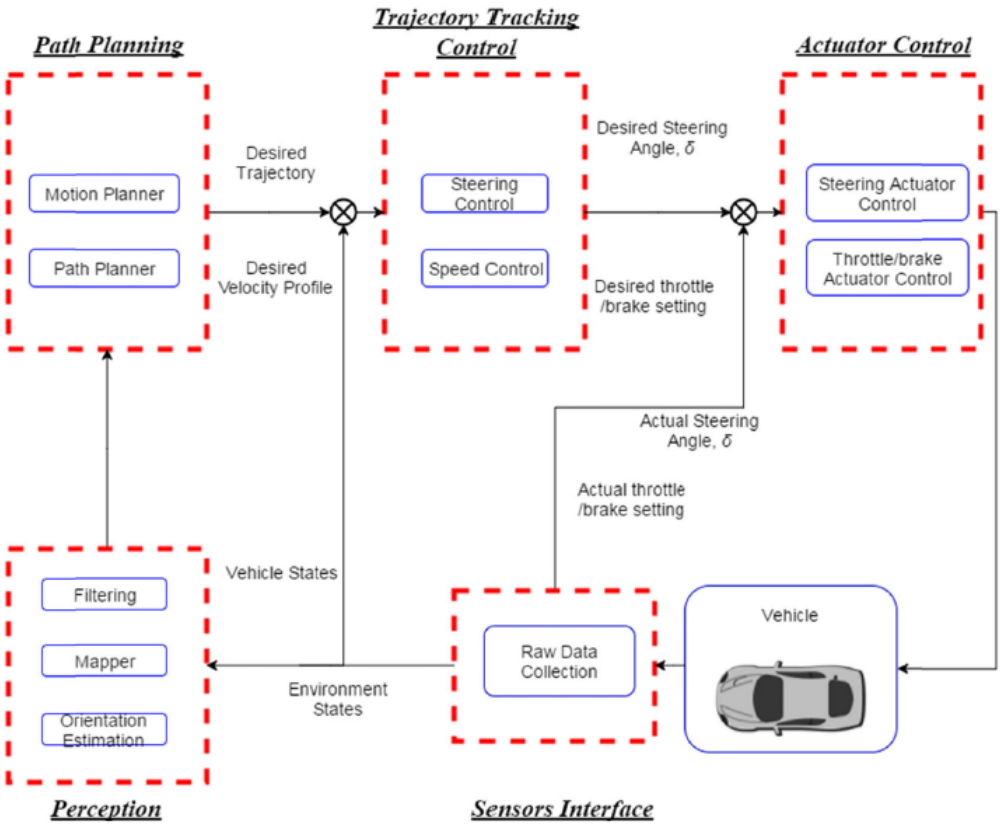


Figure 6: Overview of Interacting Functioning Modules in an Autonomous Vehicle System

  - **Perception** based tasks that will include Lane following and obstacle sensing using LIDARs.

- **Perception** based tasks that will include Lane following using cameras and obstacle sensing using LIDARs.

Here is a sneak peek of what the Qcar can do:
`www.youtube.com/watch?v=47cUdBbczUI`

2 The Quanser Qcar mobile platform

The Qcar is a scaled model vehicle equipped with a wide range of sensors, dimensions of a real car on 1/10 scale and weighs 2.7kg.

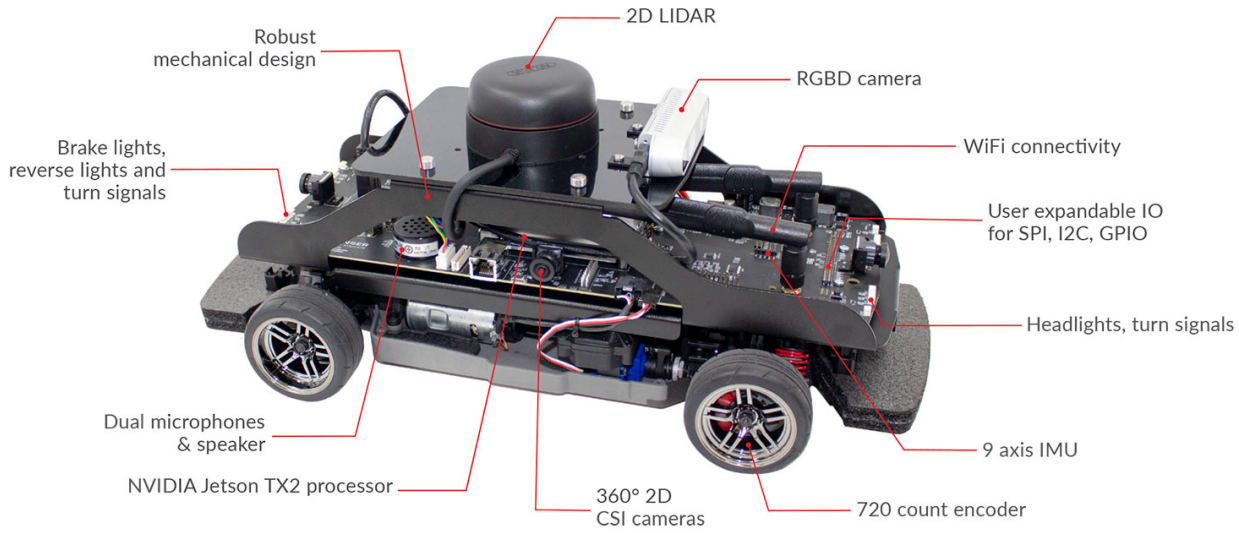


Figure 7: Quanser Qcar

Item	Value
Length	0.425 m
Width	0.192 m
Height	0.182 m
Wheelbase (1)	0.256 m
Front and rear Track (2) (3)	0.170 m
Maximum steering angle	±30° (0.5236 radians)
Tire diameter	0.066 m

Figure 8: Qcar Dimensions

The Self-Driving Car Research Studio comes with a high performance Computer and a router for Wifi connection. It is pre-configured to use both 2.4GHz and 5GHz bands for multiple PCs and autonomous vehicles.

Figure 9: Communication between a computer with QUARC installed and the QCAR

Connectivity Test :

- Make sure the yellow ethernet cable is linked to the computer. Do not use the yellow port.
- Turn on the router. After a few minutes, the lights on the front of the router should star flashing with a white light which indicates to the user that the particular ports are active
- Power on the Qcar.

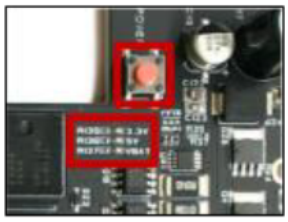
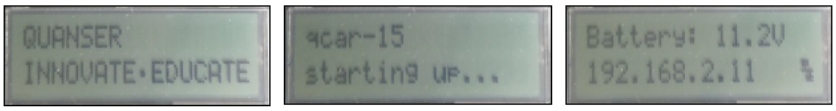
and LEDS.PNG 

Figure 10: Red Power Buttons and Green power LEDs on the Qcar

Note : Refer to Power.pdf in User Guide to know everything you have to know about connecting or charging the battery

- Use the ping test to confirm connectivity between the Qcar and the Computer. Type "ping 192.168.2.XX"" in the Command Prompt using the IPv4 address of the Qcar which can be found on the LCD display



a. First bootup message b. Second bootup message c. Default display message

Figure 11: LCD startup messages

With the Qcar comes QUARC, MATLAB Simulink Libraries which provides blocks that will be used to interact with Qcar sensors and actuators.
Note the Qcar battery has a 2 hours autonomy. Make sure the battery is not empty before running programs and do not let the Qcar ON if you do not use it. Refer to (II-Power.pdf)

Instructions :

- Do all your tests first with the Qcar on its socle not to damage the Qcar if command issues occurs. Or check if the battery is not empty or connection not lost.
- Refer to the Troubleshooting part at the end of this document when you come through errors.
- Take care of the Qcar when you handle it.
- To avoid unfortunate damages, do not input too high speeds to the Qcar.

3 Getting started

This section aims to make you browse and understand the different inputs and outputs of the QCar and the operation in relation to the Simulink programs.

3.1 MATLAB 2021a

1. Open MATLAB 2021a and NOT any other version.
2. From **Data Disk D** » go to the Folder `"2-TP_QCAR"` then `"TP_QCAR"` and finally `"TP_2023_2024"` .
   All the files required for this lab are stored in the folder named "TPfiles".
3. Create a new folder named: `NOM1_NOM2_NOM3`
4. Copy ALL the files from folder "TPfiles" and paste them in the just created folder `NOM1_NOM2_NOM3`
5. Add the folder (and subfolders) `NOM1_NOM2_NOM3` to the path of MATLAB in MATLAB workspace.
6. In MATLAB workspace, set you working directory to: `NOM1_NOM2_NOM3`
   From now on, in the rest of the document this will be your "working-directory".

3.2 Configuration : How to generate, deploy and run your Simulink model on QCAR

To understand various steps for code generation, deployment of executable code on QCAR and running QCAR using Simulink, Open the file *BasicIO.slx* in the Folder *I.Explore*

All the tasks asked using the BasicIO.slx has to be done with the Qcar on its provided base.

Here are the instructions to follow EVERY time you need to run a model. Come back to it as soon as you need.
First, make sure both Qcar and Router are on.
In the **Modeling** section of your simulink program, click on Model Settings and then write down the IP written on the **back of your QCar** as shown below:

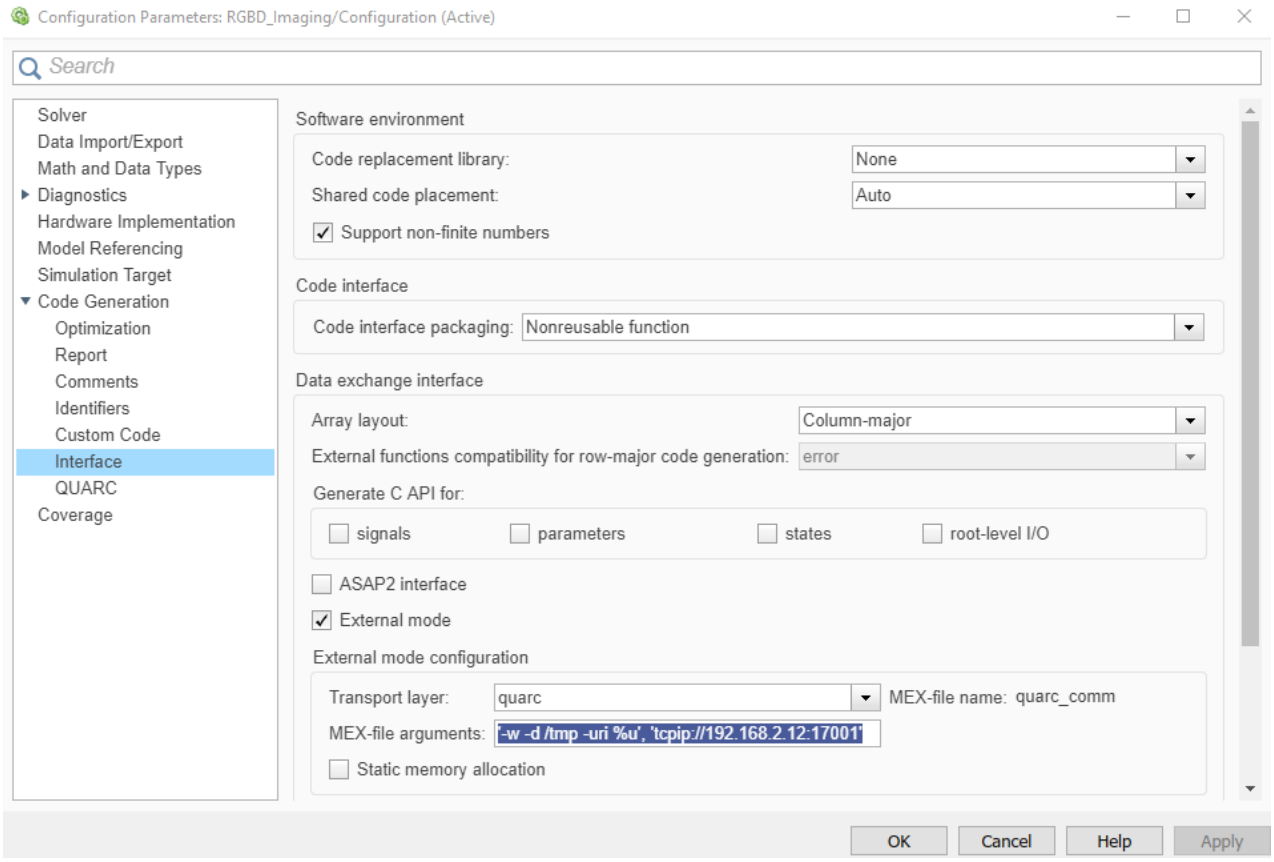


Figure 12: Configuration Parameters

The MEX-file arguments field should be filled with this:

'-w -d /tmp -uri %u', 'tcpip://192.168.2.XX:17001'

The XX number should be 11, 12 or 13. You can see it on the back of your QCar when it is turned ON (See LED display of QCAR).

To build code, deploy and run the Simulink program on your QCar, follow the instructions:

1. To build, download, connect, and start the model for real-time external operation, click the Monitor and Tune button on the HARDWARE tab (see Figure13)
2. If the model is successfully built and downloaded to the target. Simulink will automatically connect and start the real-time code on the target and the Monitor and Tune button will automatically change to a Stop button (See Figure 14)

See complete details (Step by Step procedure) here:
`https://docs.quanser.com/quarc/documentation/quarc_procedures.html#external_mode_operation`

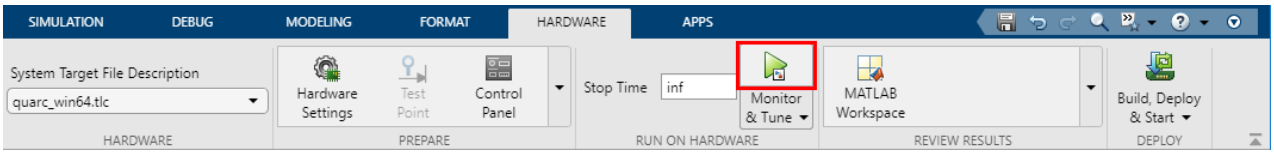


Figure 13: One step process for code generation, code deploy and Run Simulink (QUARC+QCAR) Model in External Mode

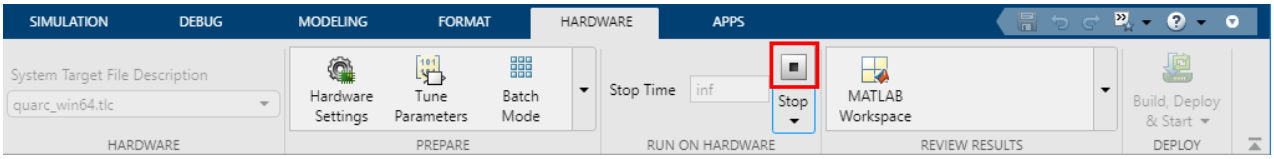


Figure 14: Running Simulink (QUARC+QCAR) Model in External Mode

If an error occurs, please refer to the Troubleshooting Section at the end of the document.

Refer to these steps to FOR EVERY NEW MODEL YOU RUN

More details available here : `docs.quanser.com/quarc/documentation/qcar.html`

4 Path following

4.1 The bicycle model

This section is based on Chapter 4 *Mobile robot vehicle* of the book by Peter Cork entitled "Robotics, Vision and Control Fundamental Algorithms in Matlab".

To follow predefined paths or trajectories accurately, the Qcar needs to know with precision where it is. To do so, different car-like vehicle model does exists, with more or less complexity. In our case, we are going to use the bicycle model.

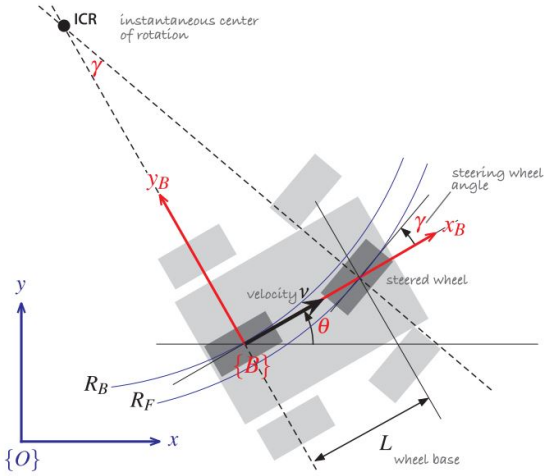


Figure 15: Bicycle Model

The bicycle model is a dynamic/kinematic model of vehicles, in which the two front and rear wheels are replaced by one front and one rear wheel respectively. The vehicle moves on the plane and its coordinates are described by the vector (x,y,θ), where x, y is the position of the centre of gravity and θ is the orientation of the vehicle. Steering angle of the front wheels is denoted by γ. A basic assumption of the bicycle model is that the inner slip, outer slip and steer angles are equal.
Using such a model aims to know with precision the Pose of the vehicle using known input parameters.

Pose means Position and Orientation.

i.e. the Position in world frame coordinates (X,Y) where X and Y are the axis of the horizontal plane. The angular Orientation of the vehicle θ which is the rotation around the Z axis.

Here, the parameters that are known are the speed of the ' V_{Qcar}, and the steering of the front wheels γ.

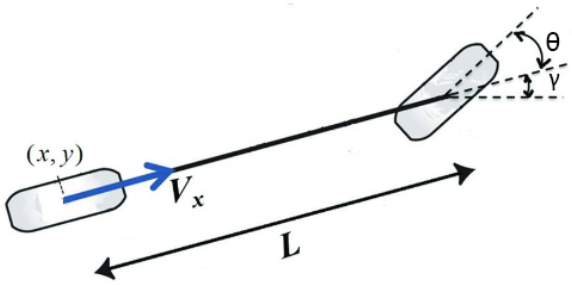


Figure 16: Bicycle Model

In world frame coordinates, we write the equation of motion as :

$$V_X = V_{Qcar} \cdot \cos\theta - (L/2) \cdot \dot{\theta} \cdot \sin\theta$$
$$V_Y = V_{Qcar} \cdot \sin\theta + (L/2)\dot{\theta} \cdot \cos\theta$$
$$\dot{\theta} = \frac{V_{Qcar} \cdot \gamma}{L}$$

Where L is the distance between the front and the rear wheel. X,Y and θ can be known by integrating V_X V_Y and $\dot{\theta}$
In this model, the position (X,Y) is the position of the middle of the bicycle, that is why there is a L/2 argument.

This is called a kinematic model since it describes the velocities and not the forces or torques that causes the velocity.
The orientation θ can be known with sensors like gyroscopes or can be deduced from the steering angle.

The orientation Ψ can be known with sensors like gyroscopes or can be deduced from the steering angle.
The following simulink block implements the kinematics equations of a bicycle model to simulate the motion of the vehicle.

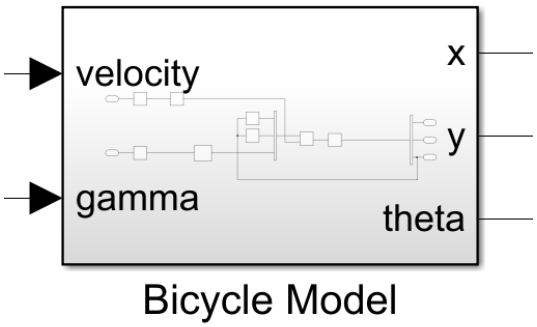


Figure 17: Bicycle Model Kinematics simulink implementation

Knowing the linear velocity of the Qcar and the steering angle of the wheels, with the approximation of a bicycle model, we are able to know the Pose (X,Y,Ψ) of the Qcar.

4.2 Drive Point

When the aim is to move towards a goal point, there are two elements to control. First we need to control the speed of the Qcar.
To do this, we'll use a speed proportional to the distance between the Qcar and its goal. Velocity can then be expressed as follows:

$$v^* = K_v \sqrt{(x^* - x)^2 + (y^* - y)^2}$$

Then we have to control the steering all the way to the goal. To do this, we use the relative angle between the Qcar and the target. This angle is obtained using the formula :

$$\theta^* = \arctan \frac{y^* - y}{x^* - x}$$

We then determine the steering angle as follows :

$$\gamma = Kh(\theta^* - \theta), Kh > 0$$

It's important to note that we're not using simple subtraction, but rather Matlab's "diffang" function, because we're talking about angles here.

Now, you'll need to open and run the file named "setup_TP.m". Then, open the file named "Qcar_DrivePoint.slx".

Task 1 : Complete both "Distance to goal" and "Steering to goal" on Simulink. Then, test your program (be sure that the Qcar is well positionned on its socle.)

Task 2 : Now change the path so that the car passes through the points[1] : (2;2), (0;2), (-2;2) and then returns to (0;0). What happens ?

The problem here is that Matlab's "atan" function can only return values in the interval [-π/2,π/2], which can quickly cause problems. To solve this problem, we need to use the "atan2" function, which returns values in the interval [-π;π].

Task 3 : Try the same path but using "atan2" instead of "atan".

Task 4 : Now change the Kv and Kh gain values to obtain better results.

[1]Note here that the (x;y) pair is inverted compared to what we are used to. It should therefore be read as (y;x)

The speedController block in blue is used to convert the speed command (m/s) into throttle command (%) for the Qcar's engine. What does the Kff gain represent?

Task 5 : Tune the parameters Kff, Kp and Ki in order to have better results.

Task 6 : Once you think that all the gains have been set correctly, place the Qcar on the ground, note its location, then run the program again. What happens and why ?

Task 7 : Change the Simulink program so that the current heading of the Qcar that is used is the one from the gyroscope instead of the one from the bicycle model. What do you notice ? Can you explain the difference between the values from the gyroscope and the ones from the model ?

As you've noticed, the results under real conditions are nowhere near as satisfying as the simulation results. **What suggestions for improvement can you make?**

5 Lane following

For this section, you will be using the "Qcar_LaneFollowing.slx" file.

Line following has two main components. Firstly, we need to apply image processing to the data we receive in order to detect the line. Secondly, we need to implement a control strategy to drive the Qcar.

5.1 Block explanation

Figure 18: colorThresholdingHSV block

This block is very important because it is where the whole image processing process begins. Firstly, it converts the image from RGB format to HSV format.
The RGB format is the better known of the two, and consists of representing images with three values corresponding to the presence of red, green and blue.
The HSV format corresponds to hue, saturation and value, making it much more practical for extracting certain elements from an image. Hue can thought of as the "main color", the saturation can be seen as the intensity of the color and the value can be considered as the brightness.

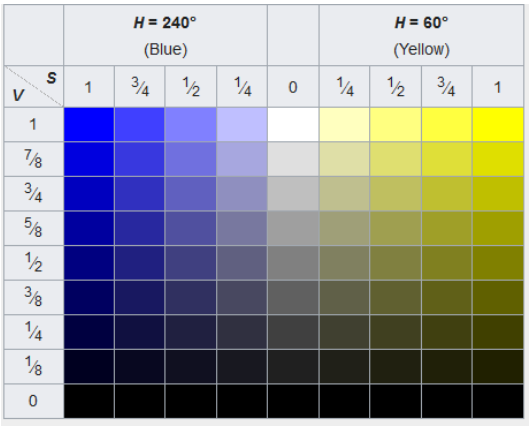


Figure 19: Example of HSV parameters for blue and yellow

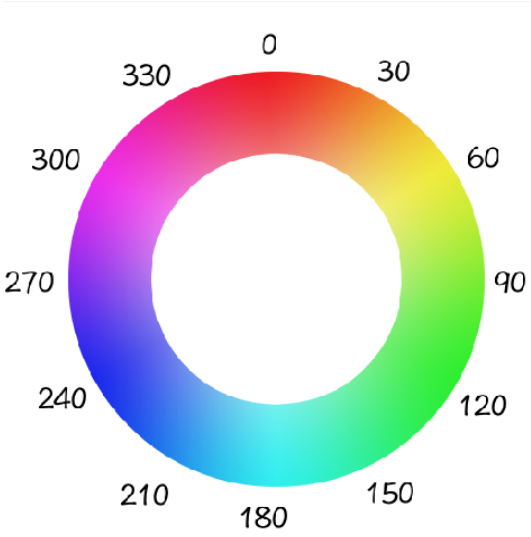


Figure 20: Representation of the hue wheel

Once the conversion is complete, the image is passed through 3 thresholds, to be configured later in the lab. A logic operator ”and” is applied to these three thresholds and the result is then filtered and sent to the output of the block.

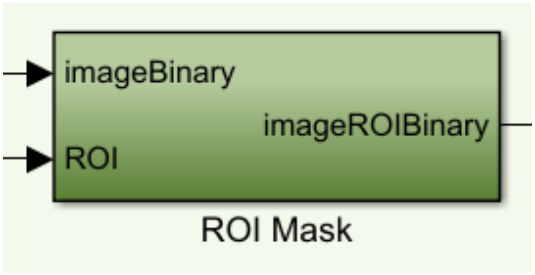


Figure 21: ROI Mask block

This block is used to apply a mask to the image so that only part of it is taken into account, which we call the "Region Of Interest". The ROI is given but you are free to change it if you wish.

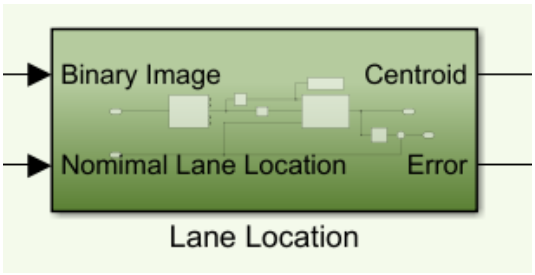


Figure 22: Lane Location block

This block locates the centre of the yellow lane and calculates the error between it and a nominal lane given as an input.

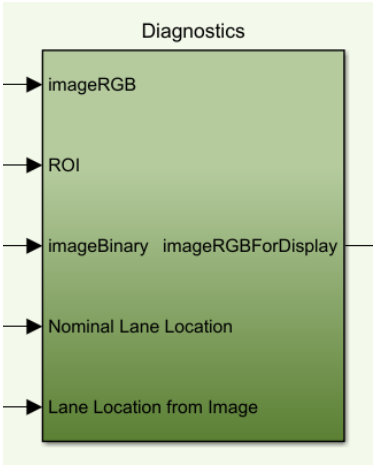


Figure 23: Diahnostics block

This block is purely visual and provides a visual interface with all the information such as the ROI or the detected line.

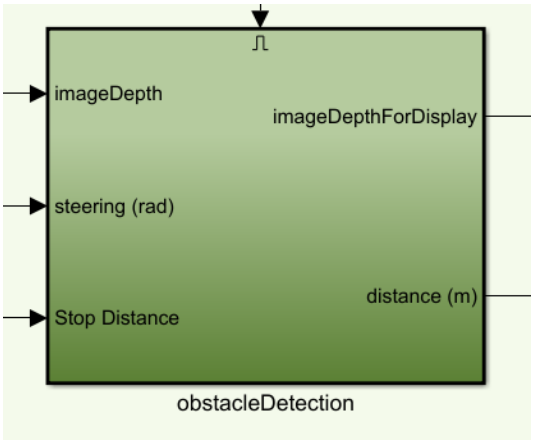


Figure 24: Obstacle Detecion block

This block uses the RGBD camera to detect the nearest obstacle and sends the distance to the obstacle as an output.

5.2 Control strategy

The control of the Qcar's trajectory is a closed-loop system. The aim here is to minimise the difference between the yellow lane and the Nominal Lane. To do this, we can use a simple P controller to generate a steering command.

Both speed and steering commands are then sent in three blue blocks :

- The "automatedDriving" block manages speed to avoid collisions.
- The "turnSpeedHandling" block allows the car to slow down when turning.
- The "speedController" block is used to switch between speed command in m/s and throttle command in %.

5.3 Practical part

Before doing anything, make sure the speed is set to 0.2m/s and the QCar Start/Stop is set to 0.

Task 1 : Go in the "colorTresholdingHSV" block and tune the parameters min hue, min sat, min val, max hue, max sat and max val in ordrer to detect the yellow line. **Be careful**, as lighting conditions vary depending on the location in the room, it is essential to place the Qcar in several locations when setting the parameters.

Task 2 : Once you're happy with your line detection, try a full lap. To do this, place the Qcar in the middle of the road and double-click on the yellow area in the "QCar Start/Stop" block.

Repeat the first two tasks until you are able to make a complete lap.

We now have a car that can drive on a single line. But in the real world you have to drive alongside it. How can we change the "Nominal Lane Position" parameter so that the car drives to the right of the line ? How do you get the car to drive to the left of the line ?

Task 3 : Change the position of the "Nominal Lane" so that the car is in the middle of the right side of the road when driving straight ahead. You may have to change the tuning of your parameters in order to complete a whole lap.

What can you say about both the line detection and tracking methods ? Can you suggest any improvements?

5.4 Bonus part

As a bonus, you can try your hand at one or more of the following activities.

5.4.1 Obstacle avoidance

Using the method of your choice, find a solution that allows you to avoid an obstacle by changing sides of the road for a short period of time, without compromising the lane following.

5.4.2 STOP sign implementation

Using the method of your choice, find a solution that allows you to detect the presence of a STOP sign and take it into account by stopping briefly.

5.4.3 Other

You can also try implementing the enhancement of your choice.

6 Troubleshooting

How to fix the issues that may appears :

- If the following error message appears : It was not possible to connect to the specified URI. Refer to "How to run your model" section. Your car may be switched off or you have entered the wrong IP address
- If the following error message appears : "The file could not be found" here is the method to follow : Make sure no special symbols or spaces are found in your repertory. Delete the NAMEOFTHEFILE Quarc Linux Nvidia executable and the Simulink Cache of that same NAMEOFTHEFILE
- If a Message Box pops up, saying the file is not added to the path : Click Add To Path
- If the following error message appears : "Error occurred while executing External Mode MEX-file 'quarc_comm': One of the arguments is invalid." : Make sure the Qcar ping with the computer and is not connected to another router.
- If the following error message appears : Error reported by S-function 'hil_initialize_block' in 'Stop/Figures/HIL Initialize': The "Active during normal simulation" feature is only licensed for QUARC Home use. Due to safety and liability concerns, this feature is not enabled, and will not be enabled, for full QUARC installations.
  Try again.