INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Université Fédérale Toulouse Midi-Pyrénées

LAAS

Département de Génie Électrique & Informatique

4th year - Automatic Electronic - Embedded Systems

RESEARCH INITIATION PROJECT

Technical progress report

May 21, 2020

Xavier Bourlot, *Student, INSA Toulouse,*
Pauline Combes, *Student, INSA Toulouse,*
Alexandre Dimnet, *Student, INSA Toulouse,*
Romain Gary, *Student, INSA Toulouse,*
Thibault Jean, *Student, INSA Toulouse,*
Thomas Zennaro, *Student, INSA Toulouse,*

Tutors :

M. Patrick	TOUNSI
M. Sébastien	DI MERCURIO
M. Germain	GARCIA

***Abstract*—The NXP Cup is a competition organized by NXP where students design their own autonomous car. The winning car is the one that manages to finish the circuit without leaving the track. The main stake for NXP is to assess the candidates' ability to adapt to the constraints imposed while achieving a robust vehicle. Our objective is to program a vehicle that runs a circuit by automatically adjusting its speed according to the collected data from the track. We focused on software development by using C++ language. Specifically, we implement line detection and cruise control. We regularly test the performance of our programs and particularly camera algorithms. Indeed, image processing is complex because the camera is sensitive to light disturbances. We must be sure that the data collected is reliable, so we carry out filtering operations. Furthermore, we adjust the speed and direction engines with numerical control laws to ensure wheel rotation turns with adequate speed. Our car reacts effectively when applying speed and steering controls. The data collected from the camera allow us to precisely define the position of the vehicle. These results were confirmed during the "Dry Run" organized by NXP. Moreover, this project has an even broader objective. We are led to think about the concept of the autonomous car of tomorrow.**

***Index Terms*—NXPCup, autonomous car, race car, ...**

Remerciements

We would like to thank our two tutors M. Patrick Tounsi, M. Di Mercurio and M.Garcia for their unfailing support during these long months of preparation. It would have been impossible for us to have such a well-equipped car without the help of Mr. Di Mercurio who devoted a lot of time and energy to give us the keys to a good preparation for the competition. We would like to thank him for passing on his passion for innovation and improving our work. We also wanted to thank Mr. Tounsi for motivating us when the fatigue of the partials took over and for accompanying us in this adventure.
Of course, we would like to thank the whole of the management and secretariat of the GEI for having allowed us to buy all the material we needed and for having put a room at our disposal for a large part of the year.
We would like to thank the Fablab, which we have greatly solicited for all our experiences.
And finaly we would like to thank NXP Toulouse for welcoming us in their offices during the Dry Run. There we met some caring people who took a critical look at our work.

Introduction

The purpose of this report is to contextualize, document and explain our Research Initiation Project. We wished to work on a very particular subject which is the NXP Cup.

This competition, whose various aspects we will describe later, is organised by the company NXP for students from all over the world. The goal is to complete a circuit with a small autonomous car, where students design the different aspects: embedded code, hardware, etc... This report is not only for the jury who will have the pleasure to evaluate us but also for the students who will participate in the NXP Cup next year. We have recorded here snippets of our work this year. From low-level programming to memorization strategies, we wanted to faithfully retranscribe all aspects of this preparation for the competition. The health crisis we are going through has prevented us from completing this preparation and we are the first to be disappointed. However, we still wanted to present our work to you. After a quick contextualization, we will describe to you some of different subjects we have tackled during our preparation. We will then look at the question of the modeling of the car system because it seems essential to us for the future to be able to make more precise regulations based on efficient modeling. Finally, in a last part, we will describe the strategies that were considered for the continuation of the competition as well as the improvements that we recommend to our successors.

A. TODO

- The NXPCup is in several parts, we have a mandatory part where we must finish a lap without leaving as quickly as possible and an optional part which corresponds to additional tests for example : detecting an obstacle on the road and bypassing it. We started the project in October and the qualification phase was to take place at INSA on April 2. We worked on both the software part and the hardware part of the car.
- INSA participates in this competition every year so we were able to access the resources of students from previous year. However, we had a hard time at first understanding their software well. So we subsequently decided to review some of their software to make it more orderly and clear for our team. This year we also changed cars compared to the previous year, we switched to a smaller model, with electronic cards to review and changes to be made on this one.
- Thanks to M.DI MERCURIO we were able to have great cards, however there were some differences with the previous cards, for exmaple the ESC. Going back to the car, the new version had a wider steering shaft, however we did not use the camera advisor with the latest version of the car. The old car's camera remained much more reliable and we were able to use observations from previous years. The calibration pat of the cars is a very important part, it was different between the two models.

1. Description of the different aspects covered during the preparation for the competition

A. Software restructuration

1) A modular approach: The code supplied by the former project members provided us with useful examples , especially for the low level drivers. However, it needed a strong restructuration, as well as some major changes in order to fit the new boards and integrate essential aspects such as encoder regulation.

The first steps were therefore to divide the work into small modules. Fig.1 shows a simplified view of the modules and their interactions. To fully test the new board, the most basic modules were developped first, such as the motor and servo controller, as well as the user interface, which enabled us to efficiently debug later on the more complex modules.

The modules were developped first in C and then in C++, to take advantage of the object-oriented nature of the language. As an example, the Direction Controller contains two Speed Controllers (left and right), each containing a Motor object as well as an Encoder object.

To give more sense to the data handled internally, we tried to express all data in sensible units, e.g. mm/s, degrees, meters, ... Instead of manipulating raw counter or pixel values, we converted systematically in units that could later be used within models. This helped clarify the meaning of the data manipulated, and also allowed for a more methodical approach, not having to tweak unknown numbers by hand but relying on real-world measurements. Several examples of the usefulness of this method will be discussed later, like the digital differential or the position tracker, used for track modeling.

2) The regulation loops: There is three regulation loops in total. Two are used to regulate the speed of both the left and the right motors. They use encoders as speed and postion feedback, and PWM as command outputs. These two loops were made completely independant, as the motors characteristics are not well matched. Therefore, both controllers can compensate the slight different response. One particularity of our control loop is that it has variable sampling time. The issue is that encoder data is provided with digital signal edges, which are discrete time events. We found that at low speed those events were to rare to wait a long period of time before averaging to get continuous value. This meant our speed control would be very slow for low speed targets, in fact too slow to be manageable.What we did instead was to recompute the speed on every n^{th} new encoder signal edge, instead of doing it at a fixed frequency. This meant that the higher the speed, the most accurate the regulation is.

It is also important to input different speeds for the left and right motors when steering, to maintain a good road grip. This is called a differential, and it is usually a mechanical

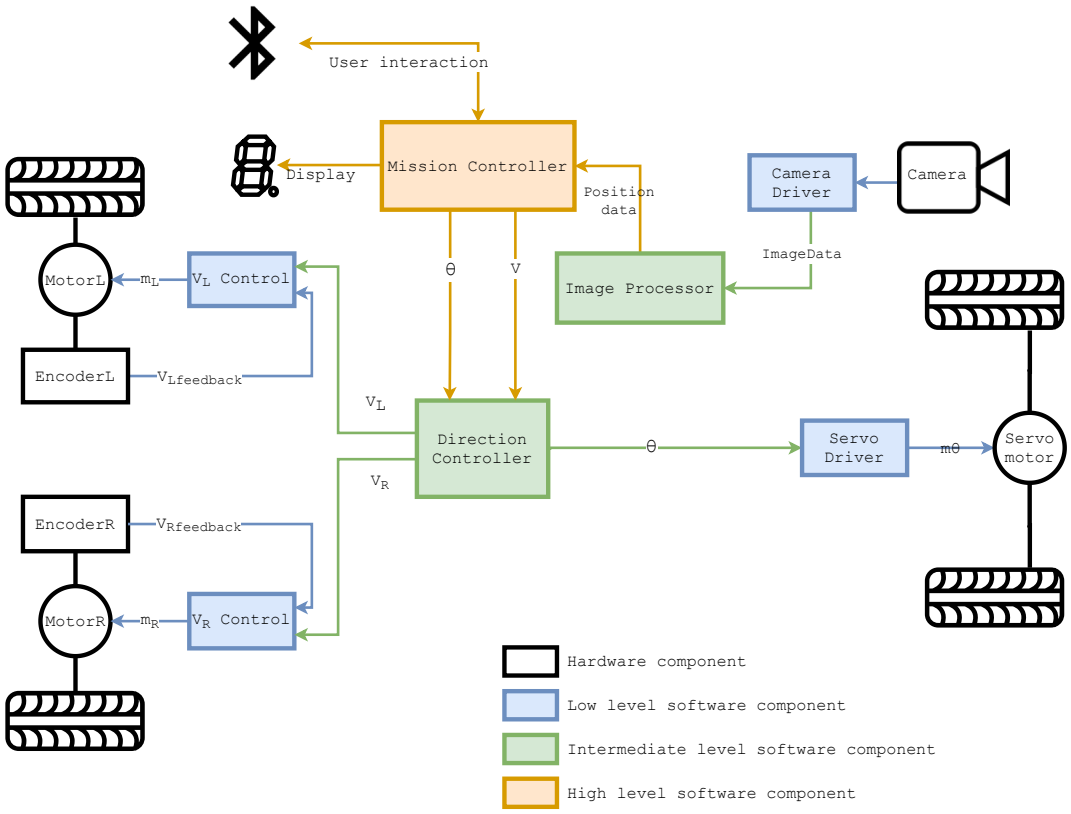

Fig. 1: System diagram

device on real-world cars. We implemented instead a digital differential, which will be explained in depth later on.
The modelling of the digital differential is critical for speed racing, as our car has no mechanical differential. We can even use the flexibility of the software to improve road tracking, especially on curves, by over-compensating the steering by example. This is one of the most basic application of torque vectoring (see [1]).
The third and most important regulation loop tries to maintain the car at the center of the track. This is a critical task, and we only have a narrow margin between being on track and out. This is why we took special care in all the components that interact with this regulation : the regulator itself, the drives, the steering mecanism, the sensors, ... The loop goes as follows :

1) The camera provides a snapshot of the track
2) The image processor extracts position information, in particular the deviation from the center of the track (this is trickier than it sounds).
3) The regulator in the mission controller compensates by changing both the angle and the speed target.
4) Wheel speed controllers kick in action to respond quickly and accurately to the changes.
5) The car responds, with its inertia.

As you can see, many components play a role in the command chain, which is the main reason why we worked on each individual component first, so we could validate its individual correct behavior before integration. This loop executed at 100Hz, while the speed regulation occured at around 1kHz. This is important because the outer control loop needs a much faster response from the inner loops to operate properly. That way we can make the assumption that the speed regulation is almost instantaneous compared to the direction regulation. Although it isn't true, it considerably simplified the control for the outer loop, and held up reasonably well during testing.

B. Finish line detection

An important rule regarding the race is that the car must be able to stop directly after crossing the finish line. This finish line is shown in Fig 2. Therefore we worked on this feature by trying two approaches :

- a first approach based on the calculation of each edge of the central rectangles (so 4 in all) that vary linearly according to the position of the left and right borders of the track returned by the camera.
- a second approach based on the detection of each edge on the left and right of both rectangles by scanning a

range of values centered around each edge. In the same time, we adjust the threshold which allows us to detect the brightness difference between a white pixel and a black pixel.

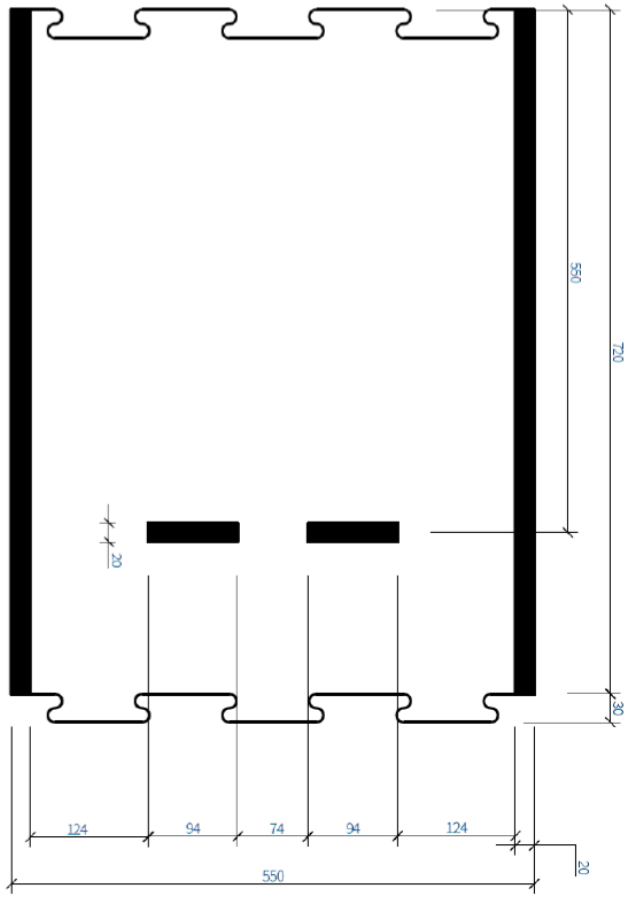

Fig. 2: Portion of track corresponding to the finish line with the corresponding metrics

1) First approach: This first approach needs to precisely determinate the position of each edge of the rectangles. However, we know that these positions change every time and particularly when there are turns. We can't know where will be located the finish line in the official track. That is why we decided to estimate these positions according to the limits left and right of the track captured by the camera. The calculations are as follow:

$$\begin{cases} BELR1 = BLL + \frac{(124)\times(BLR-BLL+1)}{530}; \\ BERR1 = BLL + \frac{(124+94)\times(BLR-BLL+1)}{530}; \\ BELR2 = BLL + \frac{(124+94+74)\times(BLR-BLL+1)}{530}; \\ BERR2 = BLL + \frac{(124+94+74+94)\times(BLR-BLL+1)}{530}; \end{cases} \tag{1}$$

We consider the metrics given in the official rules.

With these computed positions, we can search the matching value of the pixel difference in the array built with camera data and we compare it with our mnimum threshold. We used a variable which counts the number of found edges. If this counter is equals to 4 or greater than 4, we can stop the car.

Unfortunately, this methode was not effective. Indeed, because of light interference and having a static threshold, we had very variable results and it was common for the car to stop in the middle of the runway, which is not acceptable. Therefore, we focused on a new apporach.

2) Second approach: This approach was less difficult to put in place. First we program an algorithm which adjust the threshold for pixel comparison. Indeed, we use an array which contains the difference between the values of two close pixels, which help us to obtain a first approximation of the position of the black edges for each data acquisition by the camera. Then we count the number of black edges around the middle of the track. The ranges allow us to compensate for uncertainty about the exact position of the 4 black edges (which failed us in the first approach).

If we obtain a counter equals to 4, it means that we cross the line and so we ask the car to stop.

If we obtain a counter greater than 4, we need to increase the threshold.

Else, if we obtain a counter less than 4, we need to decrease the threshold.

This method gave better results in terms of robustness compared to the first method. During the tests, the car stopped 75% of the time after the finish line. Sometimes we had a sudden stop in the middle of the track when the portion of the track was too bright. Indeed, lighting plays a crucial role because it influences the sensitivity of the camera and therefore the relevance of the data. So it also affects the detection threshold.

We would have to continue on this idea to be able to improve it further and arrive at a reliable program.

2. System Modelling

To accurately control the car on the track, one of the most essential aspects is having a good model. Unfortunately, a good model can quickly become complex, and require many information we don't have. Most of the components of the car are not fully specified. Therefore we will present a basic modelling approch, first regarding speed control, then steering control. These models are simplistic, static models, but were also the ones used during developpement. One advantage is that we can more easily identify the model parameters with simple experiments, like step responses.

A. Wheel speed control

The goal here is to control the speed of the car in a straight line. We focus on implementing a control loop for each motor. Feedback is provided by an encoder on each shaft, and speed is controlled via Pulse Width Modulation (PWM).Both encoders allow us to convert the angular speed into a digital output signal.

1) Model: The simplified model of the control is given below :

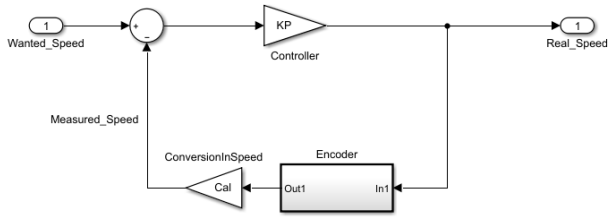

Fig. 3: Automatic engine control model valid for each wheel

For each engine control, the speed error corresponds to the difference between the value obtained by the encoder and the desired one (which depends on the steering of the wheels). The corrected output speed is then regulated according to the speed error via a proportional gain of value 1 to avoid having a value too large that could destabilize the system.

2) Control: The encoders convert an angular speed into a series of pulses at a given frequency. We make the calculations considering a speed of $v = 2m.s^{-1}$. The wheel diameter is $6.5cm$, so:

Step 1: Computation of the angular speed for both wheels.
$v_{angular} = \frac{v}{R_{Wheel}} = \frac{2}{\frac{6.5\times10^{-2}}{2}} = 61.54rad/s$

Step 2: Computation of the number of pulses per second for both wheels.
$N_{pulses} = \frac{v_{angular}\times180}{\pi} = 3526°/s$
$1pulse \iff 1° \implies 3526pulses/s$

The update of the encoders is done by interruption. Each time the interruption occurs we modify the pulse counter of each encoder. The difference between two captures is used to update the current speed measured by each encoder. A conversion coefficient is therefore necessary to move from a pulse delta to a physical speed. This is determined by considering the frequency of the CPU, the prescaler used and the various previous mechanical parameters:

$Calibration = 10 * \frac{f_{CPU}}{PSC} \times \frac{R_{Wheel}\times\pi}{180}$
$Calibration = 10 * \frac{48\times10^6}{128} \times \frac{3.25\times\pi}{180}$
$Calibration = 212712$

Remark : We multiply by 10 to obtain the speed in mm/s. This is done to avoid storing floating point values, as operations are much faster using integers, because the CPU we use has no floating point arithemtic unit.

The speed is deduced as follows: $speed = \frac{Calibration}{delta_{ccr}}$
Finally, the main control algorithm is as follows:

```
void Movement::regulate(void) {
    GPIOB_PTOR = DEBUG_RED_Pin;

    /////////////LEFT WHEEL ////////////
    int err=encoder.getLeftSpeed();
    if(err<0){  //detect invalid speed readings
        err=0;
    }
    err=targetSpeedL-err;//calculate error between the wanted speed and the current speed
    if(err>MOVEMENT_CORR_THRESHOLD || err<-MOVEMENT_CORR_THRESHOLD){//if error needs correction

        actualSpeedL=actualSpeedL+err*MOVEMENT_CORR_KP;//compensate real speed command
    }

    /////////////RIGHT WHEEL ////////////
    err=encoder.getRightSpeed();
    if(err<0){  //detect invalid speed readings
        err=0;
    }
    err=targetSpeedR-err;
    if(err>MOVEMENT_CORR_THRESHOLD || err<-MOVEMENT_CORR_THRESHOLD){
        actualSpeedR=actualSpeedR+err*MOVEMENT_CORR_KP;
    }

    applySpeeds();
}
```

Fig. 4: Overview of the main code for the speed regulation

We made some tests in order to check the robustness of our algorithm when we want to regulate a speed ranging between $0m.s^{-1}$ and $9m.s^{-1}$ in both straight line and turns.

a) In straight line: we notice that the car responds efficiently and quickly when the user asks for a certain speed. In addition, we have implemented a small interface allowing the programmer to increase or decrease the speed as he sees fit via the keyboard and using the bluetooth module integrated on the vehicle to transmit the changes to the microcontroller. By changing the speed to be reached several times in a row, the vehicle seemed to oscillate slightly around its target but without having a major effect on the track adhesion,.

b) In straight line with chicane: we see similarities in behaviour with the case in a straight line. The big difference is the oscillating character which is more important since the vehicle must adapt its speed at the same time as the steering of the wheels and this, alternating according to the direction (left/right).

c) In turns: in this case, the regulation is more delicate. Indeed, it depends on the current steering of the wheels knowing that they each have a steering limit that is otherwise different for each wheel. If the requested speed is high, the vehicle will tend to either take a wide turn or to deviate from the track until exiting definitively. The speed limit adjustment in turns was therefore carefully taken into account in order to find the limit value. This can vary slightly between a left turn and a right turn due to the different wheel asymmetry and maximum steering.

d) Speed limitation: Based on those observations, we decided to implement variable speed control, depending on the position on the track. The idea is simple : if the track is straight, and the car tracks well, we should go faster. In contrary, if the track is turning fast, or we are close to loosing control, we should slow down.
Therefore, we decided to set a target speed, which would be the maximum speed of the car in a straight line, and a slow speed, which would be the minimum speed, reached in turns by example. Then, the car would slowly raise its speed when in good conditions towards the maximum speed, and lower it quickly towards the slow speed in case of difficulties at tracking the edges of the road. After fiddling with different performance measurements, we found a combination of speed and thresholds which allowed us to go fast in straight lines and take conservative turns.

B. Steering control

1) Steering model: The steering geometry of our car is based on a mechanical arrangement called Ackermann steering. This geometry locates the center of the turning

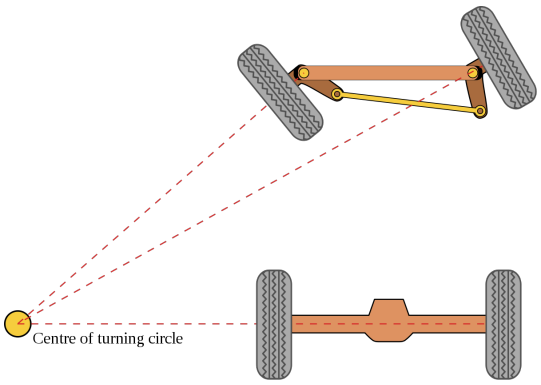

Fig. 5: Ackermann steering geometry

radius at the level of the rear wheels. Notice that the two front wheel angles are different. In fact, we can calculate the required angle of the wheels based on the desired turning radius with the folowing formula :

$$\begin{cases} \theta_r = arctan(\frac{H}{R+\frac{W}{2}}) \\ \theta_l = arctan(\frac{H}{R-\frac{W}{2}}) \end{cases}$$

Where :

- R is the desired turning radius
- H is the distance from the front to the rear wheels
- W i is the width of the car

Our actuator is a servomotor, which is mechanically coupled to both wheels. Because the linkage mechanism is quite complex, we did not try to produce a theoretical model. Rather than that, to identify the mechanical relationship, we experimented with various servo commands, and found the corresponding turning radius. This approach is much more precise than measuring the resulting angles of the wheels. That is mostly due to the $tan()$ in the formula, which make the turning radius vary a lot for small θ changes.

a) Basic control law: Plotting the results of the experiments, we were able to convert back the turning radius into precise angle values, and derive a basic relationship between the position of the servo and the direction angle. We found that a first order approximation of the relation would yield at the most extreme angles an error of less than 5%. This is good because it means that we don't have to spend a lot of time calculating , especially trigonometry functions, which can be quite time consuming, with no floating point arithmetic unit..

b) Mechanical improvements: Another takeaway fron this experiment was the influence of the mechanical linkage of the steering on the cars' turning radius. Once again, due to the $tan()$ in the formula, we noticed that a slight increase in the available wheel angle range yielded a far shorter turning radius. This is beneficial for us, because it gives more freedom to the car to correct its trajectory. As an example, the turning radius of the track is approx. $45cm$. During our experiment, we found that the minimum turning radius of our car was only approx. $40cm$. This meant very little margin to correct when we were off center.
To overcome this limitation, we took a look at the steering mechanism, and found that it could be improved by flipping a bracket upside-down. The angle increase was small, only a couple of degrees, but the minimum turning radius shrinked to approx. $33cm$, which improves our turning margin over the track by 160% ! With this improvement and the new corresponding control law derived, we could start pushing the speed of our car much further.

2) Digital differential drive: As mentionned previously, our car lacks a mechanical differential drive. Instead, the two motors drive the rear wheels directly, through a gear reduction. This means we need to implement a digital differential. This is done by calculating the required speed of both the left and the right motors from the target speed and the angle of the wheels.

$$\begin{cases} V_r = V(1 + \frac{WH}{2tan(\theta)})) \\ V_l = V(1 - \frac{WH}{2tan(\theta)})) \end{cases}$$

Where :

- V_r and V_l are the required right and left motor speed.

- H is the distance from the front to the rear wheels
- W i is the width of the car
- θ i is the target angle
- V i is the overall car speed target

Once again we simplified this equation using simple first-order approximation. The initial results were very good, with no tuning, thanks to the accurate internal representation of the data using real units (e.g. degrees and mm/s). We adjusted the coefficients to finally obtain a slight over-compensation, which we observed to give better tracking of the road during tight curves.

These calculations have been verified with experiments (see Fig.6), to give us a baseline of the results we should expect, as well as an intuition for the system behaviour. Notice that the experimental ratio of $\frac{V_R}{V_L}$ stays almost constant, validating a first order approach.

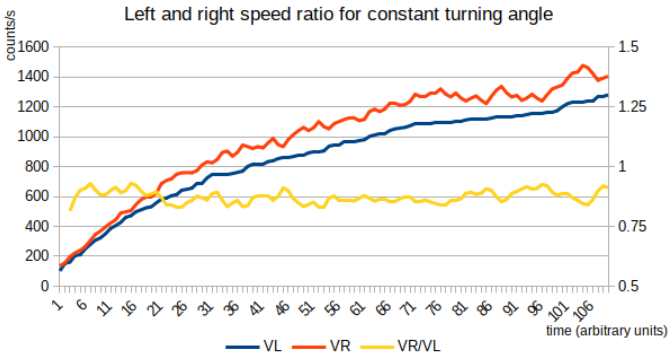

Fig. 6: Experimental insight at differential driving

Tests were carried by turning in circles of specified radius, and progressively increasing the target speed, until slip or radius enlargement was detected. The effect of slight changes in the differential was clearly visible, both in the maximum speed reached, but also in the road grip that could be obtained. Both clockwise and anti-clockwise rotations were tested to ensure the system's symmetry.

3. Results, Considered Improvements and Strategies

A. First results : the Dry Run at NXP Toulouse

On Thursday, 27 February 2020, NXP welcomed us on its Toulouse site for a Dry Run with the other teams close to the region. We had the pleasure to exchange with engineers working at NXP, with students from other schools and their teachers. The main event of the day was our overwhelming victory against the other teams (even against the NXP team !). Overall, most of the teams did not have fully functional cars yet. We were able to test different settings on our car such as the acceleration rate in straight line or out of corners. Prolific discussions with other teams and NXP's engineers were made, giving us plenty of new ideas and motivation. We came out of this event quite satisfied and confident for the future.

B. Strategies for the future

At the time of the confinement, we had a functional car, able to follow the circuit on the day of the competition, with a few mistakes ready. We were in the process of thinking, of starting to implement solutions to compensate for a certain number of hazards that could have cost us victory on the day of the competition. In the sections below, we will detail these solutions, the needs they meet and what our ideas were to implement them. If the future team comes across this document, this section should be of interest to them.

1) Circuit mermorization: We recall here that the aim of the race is to finish a circuit in a minimum time without going off the road. In case of failure (off road) it is possible to redo the circuit twice. If the circuit is completed it is not possible to try it again. We have imagined a scenario like this: we could program the car to do the circuit a first time slowly, recording the circuit, and then run off the road just before the end. Then, for the second test, the car would already know the corners and other chicanes and could accelerate on the straights and anticipate the corners. This implies a circuit memorization function at first. We had started to implement Flash writing but unfortunately we didn't go any further. The idea would have been to memorize the circuit "by pieces", for example: "straight ahead", "straight ahead", "40° turn"...

2) Adding a second camera: Sometimes, without really being able to explain it, the car seems not to "see" the bends, or it seems to see them too late. One of the reasons for this can be the difference in brightness between the shadows cast on the track and the strong brightness caused by the direct sunlight. To overcome this, the idea would be to add a second camera, oriented differently in order to "see" further, and thus anticipate the decision. This way the "near" camera could focus on instantaneous regulation, whereas the

"far" camera could forsee turns and decide to slow down approaching these difficulties. Another advantage is that the "near" camera would be less prone to picking up noise at it is looking closer to the car, hence better tracking the raod limits. The rules of the competition don't exclude this two cameras approach, so it's something to dig into for next year.

3) Addition of a modular LED panel: In the paragraph above, we discussed the problem caused by the difference in brightness on the track. This is the same problem that our predecessors encountered in 2019. One possibility would have been to add a red LED panel under the camera in order to get rid of the luminosity differences and shadows. By controlling these LEDs correctly, it would have been possible to subtract the brightness of the LEDs from the signals returned by the camera to obtain a cleaner and less error-prone signal, a basic backgroung cancellation algorithm. Such an LED panel was tested, but the major issue were the lights spots that resulted from the individual LED. To diffuse the red light from the LEDs we thought of wrapping this panel with an LCD backlight diffuser. To attach this structure to the mast, we fabricated a 3D printed piece that would clip onto the mast and in which two wires would be housed. These wires would stiffen the diffuser to hold it in place. At the time of the containment, we had printed a piece that was not quite suitable but we think it's a good start. You can see the design on figure 7.

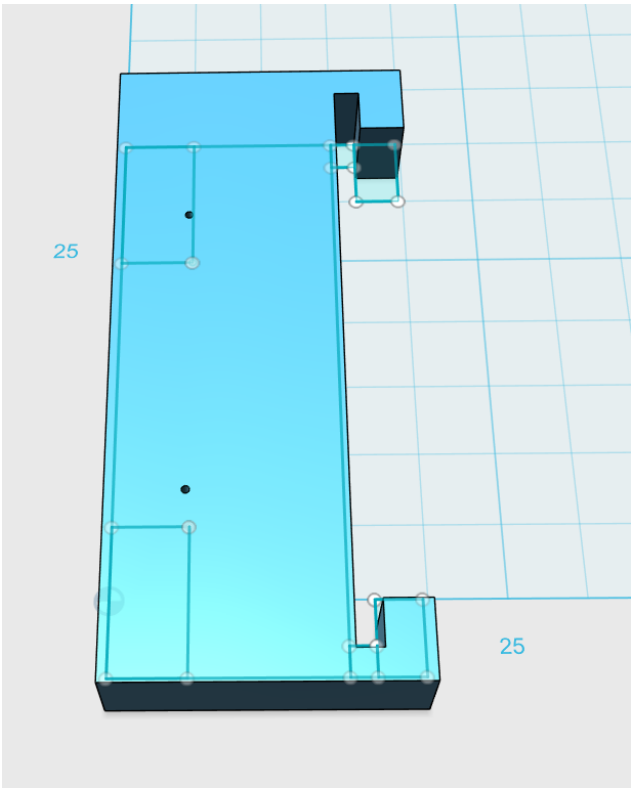

Fig. 7: First draft of the design of the add-on piece to fasten the diffuser to the led panel (side view)

4) Using the Pixie2 smart camera: The Pixy2 camera is a camera made for robot vision with an image processing directly embedded on it. This processing allows, among other things, to distinguish the center of a track or to detect several possible paths at a crossroad for example. At the very beginning of the project we were very interested in the implementation of this camera on our car. The PCB even included an input for this purpose. Unfortunately, while testing it on the top of the mast, we realized that it did not have a wide enough field of view to cover the whole width of the track. It was therefore impossible to see the middle of the track. During the Dry Run organised at NXP Toulouse on 27 February 2020, we noticed that some teams were using this camera. These teams seemed to have a problem focusing the camera but still got a good answer. Given the quality of the camera we are currently using, it might be worth trying the Pixy2 again. Note that some teams had a larger mast due to an oversight in the specification of the mast height in the rules of the 2020 competition.

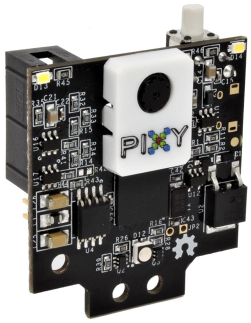

Fig. 8: The Pixy2 camera (image from Amazon)

C. Improvement in code architecture

Our current version of the code, although functional, does not have a clear distinction between the middle layer and the high level layer. Although the low layer is modular and can be changed from car to car, with the middle layer generic enough to adapt to different low-layers, the frontier between the regulations loop is blurry. The speed control resides in the Direction controller (see Fig.1), but the track following loop is partly located in the image processor and partly in the mission controller. This is not good for separating the modules. Therefore, an improvement would be to have a separate position controller which could receive generic position data as well as special orders from the mission controller, and output directional orders to the Direction controller.

D. Decision support for sensor reliability using fuzzy logic

1) About fuzzy logic: Fuzzy logic is a tool used in algorithms to make decisions based on data that are not

judged by a mathematical formula but by an index of confidence. The article by Jean-Pierre Rabbat, Mireille Rabat and Yves Lecluse "Examples of application of fuzzy logic: temperature control of a pilot furnace", is a very good start to understand the mechanisms related to this logic. We will briefly discuss the three key stages of fuzzy logic: the choice of membership functions, the elaboration of inference rules (or fuzzy rules) and finally defuzzification.

2) Usefullness of fuzzy logic and choice of variables: For this project, it seemed interesting to use fuzzy logic since there is no reliable and convincing model of the car. The servo control set up on the steering does not take into account all the parameters such as the various frictions for example. The choice of direction is then based solely on what the camera sees. However, we were able to show that the camera is not always reliable. A steering angle of the car directly derived from this data was dangerous.

As a reminder, the camera returns an array of 128 pixels representing the width of its field of view. The black stripes on the edge of the track are reflected by low values and the white of the track gives high values (between 0 and 1023). Where the difference between several consecutive values is greatest, the edge of the track is considered to be the edge of the track. To do this, a differential mathematical function is applied which positions two peaks where it is estimated that the two black bands are present.

From the positions of these two peaks, we can assign to these values a confidence index according to whether the peak is considered "too close", "at medium distance" or "far" from the car.

3) First approach: We started a first approach to regulation using fuzzy logic with the development of fuzzy membership functions, the drafts of which are presented on figure 9. The values of the functions are arbitrary because we did not develop this approach much further.

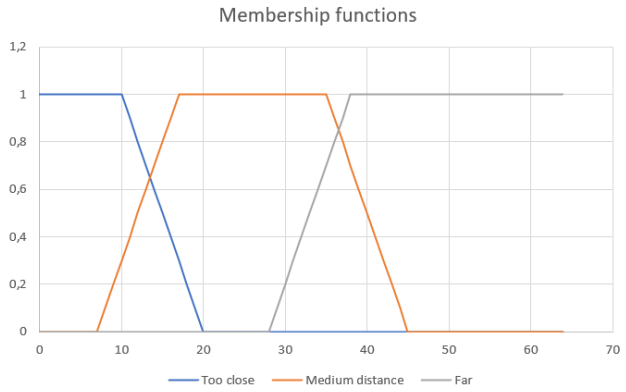

Fig. 9: Graph representing fuzzy membership functions (approximate values)

Then, it was necessary to define the inference rules (the fuzzy rules) that apply a character to the right engine or/and to the left engine. The characters are "max speed", "stop or very slow", "a little faster", "a little slower", "fast". The elaborated rules are presented below :

- IF RightLine is at medium distance AND LeftLine is at medium distance THEN RightEngine is max speed and LeftEngine is max speed.
- IF RightLine is too close AND LeftLine is at medium distance THEN RightEngine is a little faster and LeftEngine is a little slower.
- IF RightLine is too close AND LeftLine is far THEN RightEngine is fast and LeftEngine is stop or very slow.
- IF RightLine is at medium distance AND LeftLine is close THEN RightEngine is a little slower and LeftEngine is a little faster.
- IF RightLine is far AND LeftLine is close THEN RightEngine is stop or very slow and LeftEngine is fast.
- IF RightLine is far AND LeftLine is at medium distance THEN RightEngine is a little slower and LeftEngine is a little faster.
- IF RightLine is at medium distance AND LeftLine is far THEN RightEngine is a little faster and LeftEngine is a little slower.

4) Abandonning the fuzzy logic approach: When it was found that the engine control system worked very well and the runway following was very satisfactory, it was concluded that the use of fuzzy logic was not necessary. On the contrary, developing this approach was likely to waste our time. Moreover, finding a free embedded library deploying fuzzy logic was quite hard to find, and hand coding fuzzy logic can be time-consuming and unsatisfactory.

E. The research initiation project: a first experience in project management

In parallel to this project, we followed a project management course. Even if it would have been preferable to have this course before the project, we still learnt some lessons in terms of project management from our preparation for the NXP Cup.

1) Distribution of the team on the different subjects of the project: One of the advantages of our team for this competition was the diversity of profiles. The different experiences of each person allowed the emergence of new ideas among which we retained only the best. Very quickly we split up into pairs to be able to move forward in parallel on different aspects of the project. We used the Git tool to perform version management on our code as well as to keep all the documents (PCB plans, datasheet...) needed to produce the code.

2) What played against us: Despite this, we fell behind on the project as the first half of the year progressed. First of all, we didn't really start working on the project until very late in the year, when we could have started earlier. The start of the project fell at the same time as a very busy part of the

year, and taking on our free time during part time was not always easy. Knowing that the competition was to take place in April gave us the impression of a faraway event that didn't motivate us more than that. Finally, it was sometimes at the technical level that we were stuck, believing (sometimes wrongly) that we didn't have the necessary skills to develop such an ambitious project.

3) Our advice in terms of project management to the next team: Our first piece of advice is quite simple, and it was even given to us by the previous team: start earlier in the year! As soon as the teams are done, you have to start working on the project. Then, we advise you to set deadlines for the work to be done. Example: "The programming of the optical encoders must be finished before next week". Preparing deadlines forces us to work more efficiently and that's what we missed.

Conclusion

To conclude, we were satisfied to participate to the project. This challenge was difficult but interesting.We encountered issues regarding the hardware and the methodology to put in place but we succed to solve theses problems. The lack of documentation was another difficulty. A correct code given by the last team allow us to improve our software. Then, we were even more motivated by winning the dry run. We would have liked to be able to participate in the national competition. We hope that the next team will do its best to finish the competition.

Glossary

BLL Black Line on the left of the track. 10

BLR Black Line on the right of the track. 10

IEEE Institute of Electrical and Electronics Engineers. 10

PWM Pulse Width Modulation. 10

References

[1] V. T. T. Pawar, "Active torque vectoring for all wheel drive fsae electric car," Ph.D. dissertation, 2016.