

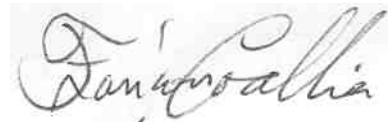
CAPRA DESIGN REPORT

École de technologie supérieure
François Langelier
Camille Roch
Gabriel Kenderessy
Jérémy Savage
Philippe Delisle

Department of Software Engineering and Information Technology

capra@ens.etsmtl.ca

I certify that the engineering design in the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.



Prof. François Coallier, Eng. Ph. D.
Faculty Advisor, Capra
École de Technologie supérieure (ÉTS)

ABSTRACT

This design report documents the features, specifications and innovations found in Club Capra's Capra6 autonomous vehicle. An introduction to the report and its goals will help the reader understand the context of this design report. A quick overview of all the important innovations for 2014 is also included to give the reader a general idea of what the Capra Team achieved since last year's IGVC. Details are given concerning these innovations in 3 sections: Mechanical Design, Electrical Design and Software Design. Finally, a concluding statement is available for the reader specifying objectives for next year.

BRIEF HISTORY

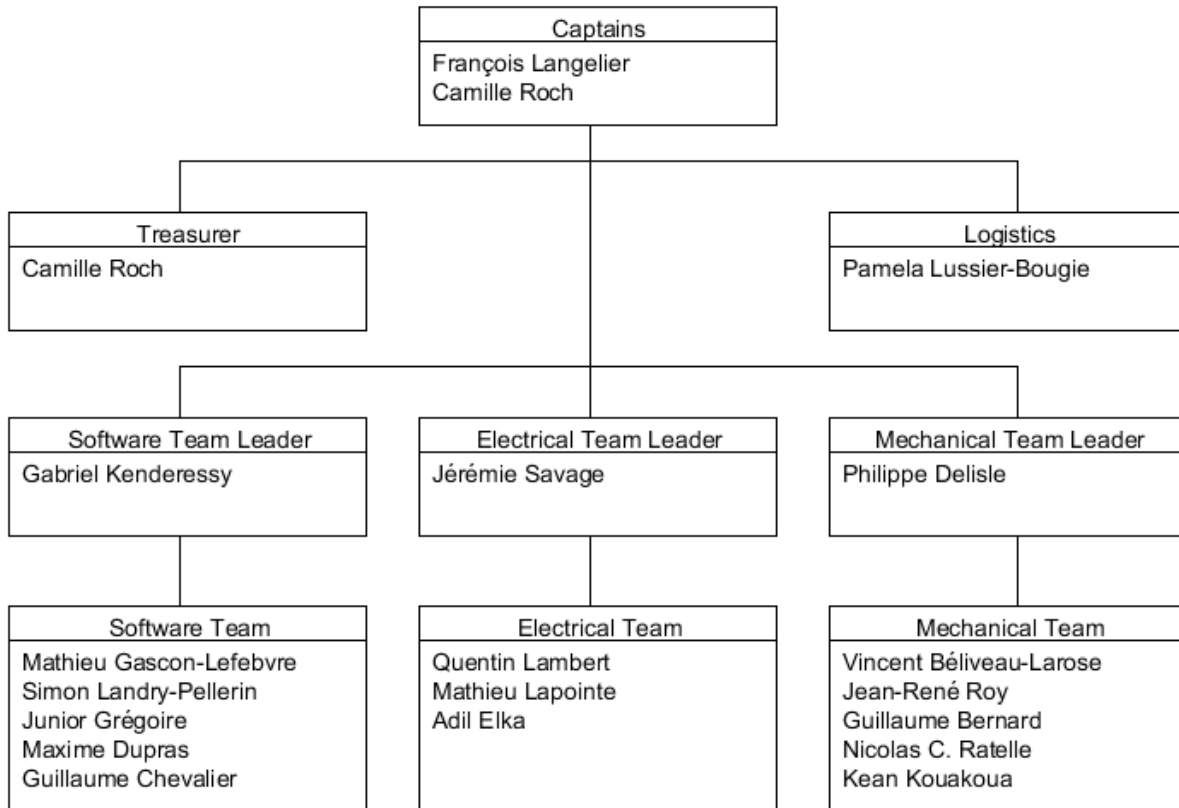
Club Capra was found in 1999 by a group of engineering students passionate about robots. Capra is a scientific club that has as a main goal to design and implement autonomous vehicles.

This year, the team worked hard to improve Capra6, the robot from last year, in order to achieve a better ranking at the competition.

TEAM ORGANISATION

The team is composed of students of the baccalaureate and master level from the École de technologie supérieure which is an engineering school with five different departments. Capra's team is proud to have members from all these departments. The variety of knowledge allows the team to explore different possibilities to improve Capra6 so that it can perform well at the 22th edition of IGVC. Figure 1 shows the team structure of Capra.

Figure 1. Capra Team Structure



Every month, each department leader meets up with the captains to share the progress of their teams. Also, a general meeting takes place every first Monday of the month so that the members can share their ideas and the captain can inform the whole team of any changes concerning the robot or the club. The members are divided into three departments who work on different aspects of Capra6.

DESIGN PROCESS

Capra is continuously using the DMAIC improvement cycle to guide the team members' work on the robot. The Capra team applies the DMAIC in the following way:

Define: The first step of any design process is to define the needs. Last year, as the team participated in the 21st edition of IGVC with a new robot, Capra6, the needs and possibilities of improvement were clear. Specific objectives were then defined to guide the team's work.

Measure: During the last year, the team recorded measures, results and statistical data. Methods to analyze the performances on different aspects of the robot were already in place.

Analyse: The data was used to analyze the gap between the past situation and the future improvements to fix achievable objectives.

Improve: This section initiates the changes. The team leaders encourage their team members to act and yield results.

Control: Small teams can test their improvements and control the changes.

The cycle begins again after the tests; new objectives are defined, and new measures are taken.

INNOVATIONS

Club Capra has been hard at work this year to perform well this year at IGVC 2014. All departments in the student club achieved an important number of innovations to help the robot perform better than ever.

Table 1. Quick Facts About Capra6

Weight: 155 lbs	Top speed: 2.07 m / s in a 15° slope
Dimensions: 37" x 25"	LIDAR range: 20m @ 270°
Battery: 48 V and 30 Ah	Camera range: 5m @ 170°
Battery life (autonomous): ~5h	Camera frames processed by second: 38
Battery life (full load): 2h	Reaction time: ~0.05 second
Total value: \$44 310 USD	Waypoint accuracy: ~0.4m

The Mechanical team built a new robot frame and was able to change various aspects including, but not limited to:

- A new independent suspension system to mitigate the effect caused on the sensors from rolling on rough terrain
- The robot now follows a sealing standard to protect internal components from rain
- Optimization of the interior space
- Flexible pole for easier transportation and storage
- Increasing wheels diameter to improve stability

The Electrical team improved on the components and brought up new solutions:

- Optimized and merged the remote controller PCB with the USB / Serial Hub to optimize space
- Brand-new software for the remote controller including speed selection and E-Stop

Finally, the Software team redesigned the entire software suite used by the robot by switching to a more reliable framework, Robot Operating System (or ROS):

- Software built from the ground up using ROS as the base architecture
- Better robot positioning strategy using updated algorithms
- New Artificial Intelligence to make better decisions based on a bigger set of data
- Revamped line detection software now more efficient

MECHANICAL DESIGN

Capra6 has two motorized wheels at the front and two rear swivel wheels. Capra6 also has an independent suspension system. The front suspension is made of two pneumatic shocks. These shocks are not directly fixed to the wheels, but are installed on a suspension arm fixed to the wheels. The suspension protects the frame and all the internal components of the robot. It makes Capra6 extremely reliable and durable even in extreme environments where the terrain is bumpy and uneven. Since last year, due to stability problems, the motorized wheels of Capra6 have been scaled up from 12 inches to 16 inches.

Figure 2. New Robot Design



This year the robot's sealing follows the IEC60529 standards. The camera is protected by a cover that provides rain protection. The team designed a new battery case which has a built-in connector that reduces the amount of wire inside Capra6.

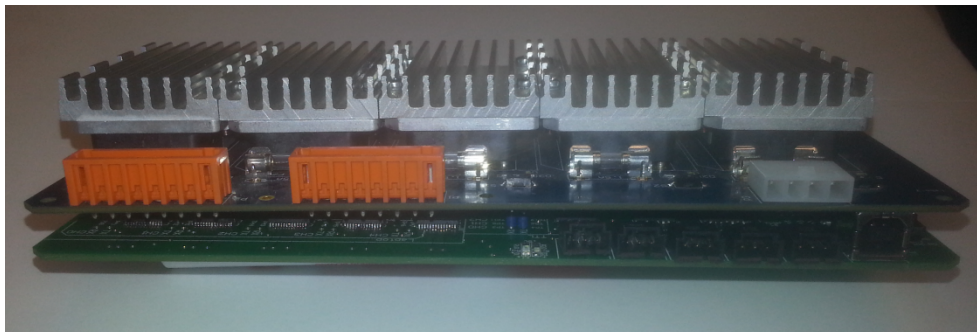
ELECTRICAL DESIGN

General Concept

Capra6's electrical system is designed to be modular, flexible, and easy to use and maintain. The tests have proven the system to be reliable and robust. The power is provided by a LiFePo4 48VDC battery and is brought to each component through an intelligent distribution circuit.

The electrical system is composed of three printed circuits boards that each have specific tasks to accomplish. Two boards are stacked in a mezzanine configuration (see Figure 3), except for the control panel which is mounted at the rear of Capra6 and act as the « nervous center » of the vehicle. See Appendix I for a general electrical plan.

Figure 3. Electrical Stack



Power Distribution

The battery is directly connected to the motors and to the electrical box. Inside the electrical box, the power lines are connected to a power supply. This power supply has been designed by Capra's team and is based on Vicor Corp micro DC-DC converter. The power supply produces different voltage rails to supply each sensor. It generates 5V, 9V, 12V, 19.5V and 24V. Each rail is fused at the input and the output.

The power supply is connected to the control panel through a power cable and to the other PCBs through a mezzanine card connector. The control panel splits the power line to every sensors/actuators. The control panel allows the user to disable/enable any sensor/actuator individually either manually with a switch or with a software command. A power-on LED indicates the state of each sensors/actuators.

RS232 Hub And Wireless Controller

As a 2014 innovation, Capra's electrical team has designed a printed circuit board that allows easy scaling of the system. This circuit has mainly two features:

- It provides five RS232 ports over one USB cable.
- It includes a wireless controller circuit which can take control over the motors.

This circuit uses five MAX3232 and FT232R to generate the USB serial port. A custom made USB hub based on TUSB2077 IC allows all these five ports to be accessible through one USB cable connected to the laptop.

A circuit based on a PIC18F4515 microcontroller is also included on the board to decode a wireless Playstation 2 controller and take control of the motors which are RS232 standard.

Simply by turning a switch on the control panel, the microcontroller can take or release the control over the motors. The control of the motors is either held by the wireless controller or the laptop. This feature has the advantage of easing the transportation of the vehicle because no computer is required to use the controller. It is as simple as using a remote controlled car.

The wireless controller uses a 2.4GHz communication and has a range of 80 feet (line of sight). The wireless receptor uses a typical Playstation 2 protocol, which is based on SPI protocol.

SOFTWARE DESIGN

The Capra Software team has been hard at work this year to update the software used by the robot. Besides changing hardware to improve the general performance of the various algorithms used, the software architecture has been completely revamped. The software development team migrated from the old custom-built software to a ROS solution that proved to be more stable, more efficient and more maintainable.

Hardware

At IGVC 2013, issues related to the line detection software were reported. Since line detection is a resource-intensive task, a more powerful laptop was acquired in order to support both the line detection software and the new ROS architecture.

Two nearly identical Lenovo Thinkpad W530 laptops were acquired to solve some performance-

related issues. The two laptops can be used in parallel while the robot is in use in order to fix problems during runs.

Table 2. Laptop Key Specifications

Processor	Intel Core i7-3630QM
GPU	NVIDIA Quadro K1000M
RAM	16 GB DDR3, 1600 MHz

The new laptop has a better processor, more RAM and, most importantly, a better GPU that's specialized for rendering, a feature that's heavily used in the new line detection software.

Table 3. GPU Key Specifications

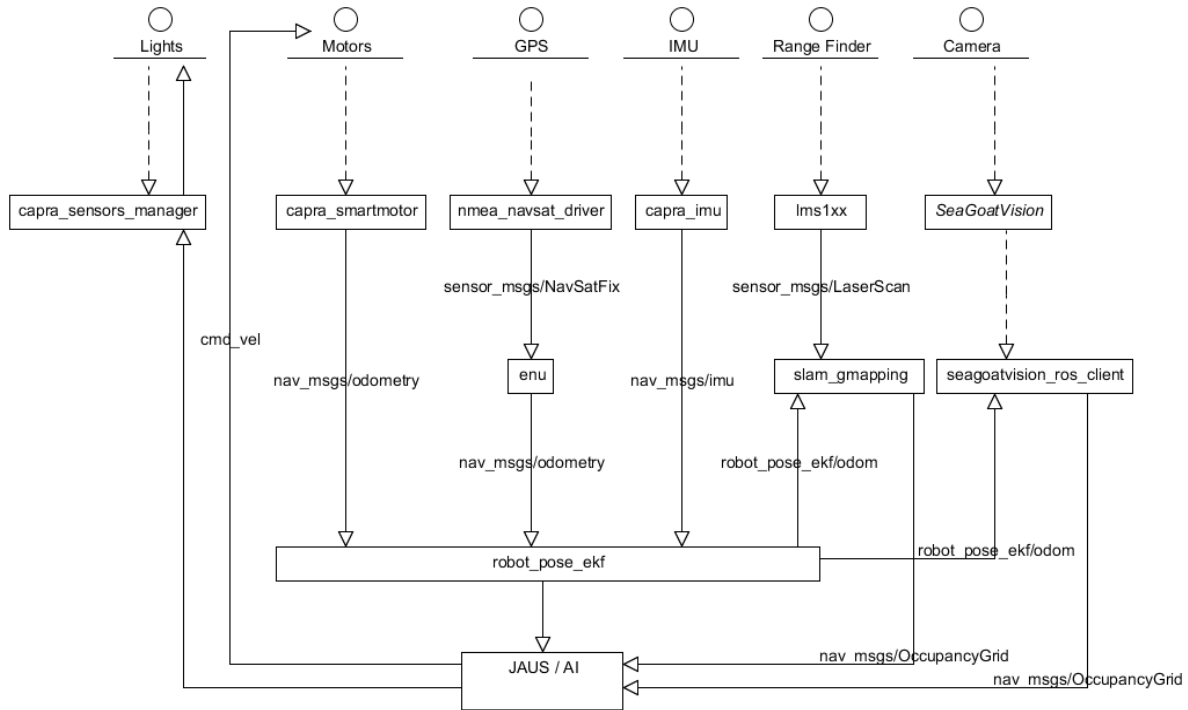
NVIDIA CUDA Parallel Processor Cores	192 @ 850 MHz
Memory	2GB (DDR3) @ 1800 Mhz
Memory Interface	128-bit
Memory Bandwidth	28.8GB/sec

ROS Architecture

The biggest innovation made by the Software Team this year was the migration from a custom Java-based robot management software to a more efficient and stable ROS-based solution. Since the old software was not on par with the solutions developed by the other teams at IGVC last year, the team decided to port most of the legacy software to ROS Hydro. ROS, or Robot Operating System, is an open-source platform using a system of nodes communicating with each other using particular message formats.

The biggest advantage of using a ROS-based solution is that some packages for popular sensors, for basic artificial intelligences and for other common tasks performed by robots (such as a Kalman filter node or Simultaneous localization and mapping node) are already available and tested by the robotics community since ROS is an open-source platform. The software development team used both custom and premade nodes to make the robot operational and autonomous.

Figure 4. The ROS Software Architecture



Every device on the robot has its own set of nodes that are independent one of another. This makes the code easier to maintain as each node acts like a software that receives and publishes messages (i.e.: raw sensor data, the robot position, etc.) from and to other nodes. Most custom nodes were implemented using ROSJava, a Java-based solution that wraps the ROS core. While the entire ROS architecture is new this year, this design report will only focus on custom nodes and improvements worth mentioning.

Robot Positioning Strategy

Last year's solution had an issue reading the encoded position which created an offset on each returned position. As the robot navigated the course, the offset got bigger and caused a *teleportation effect* which affected the implemented AI. This *teleportation effect* was taken into account while the AI was developed, but, the erratic nature of the bug often made the AI take bad decisions.

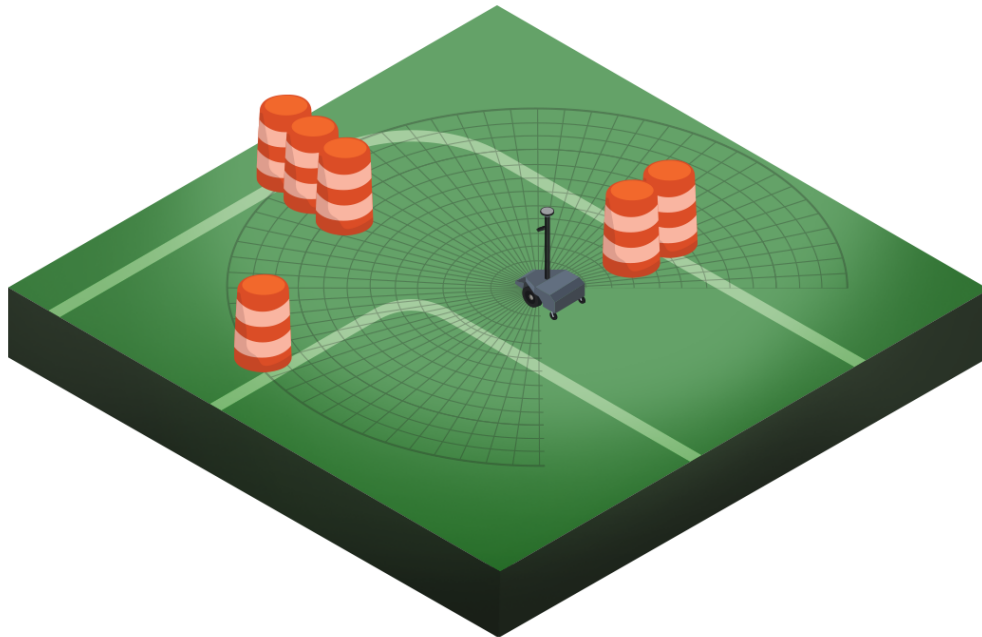
Great improvements were made when it comes to position the robot in a virtual environment. With the help of an extended Kalman filter (i.e.: the *robot_pose_ekf* package, available on the ROS package repository) and a better SLAM (i.e.: the *gmapping* package, also available on the ROS package repository), the custom ROS node that communicates directly with the motor is now more reliable and reactive.

The Kalman filter is able to return an accurate reading of the robot's position with the help of the encoded position from the motors, the robot orientation from the IMU and the global position from the GPS.

Artificial Intelligence

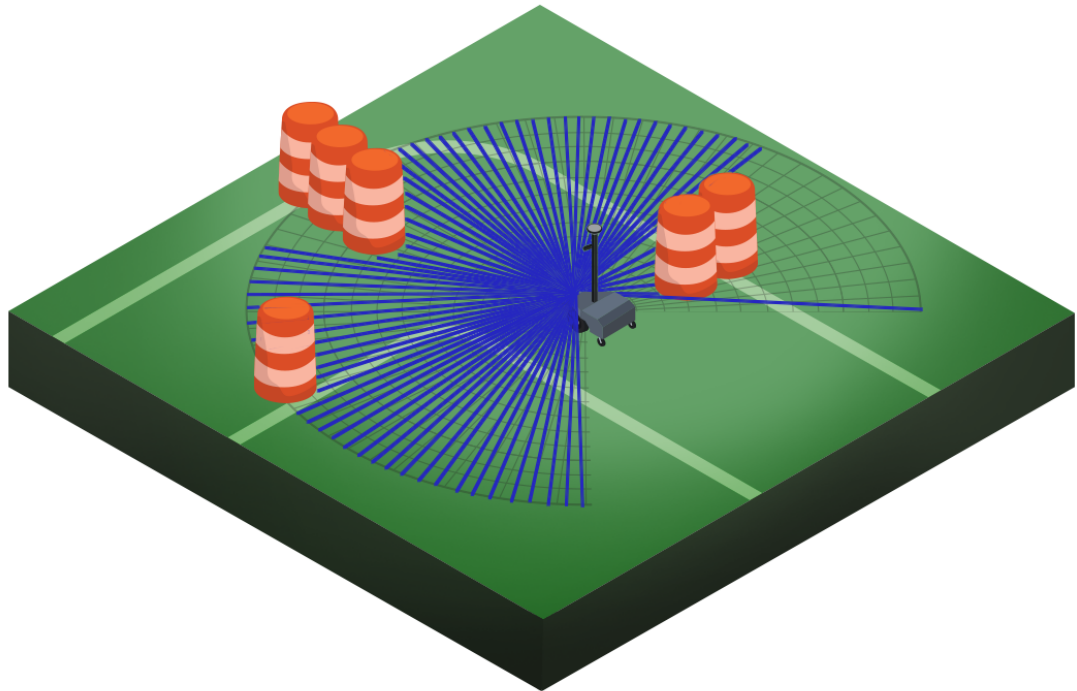
To avoid obstacles, follow lines and reach GPS waypoints, Capra6 uses a custom implementation of the A* algorithm, modified to fit the team's needs and support the stochastic (random obstacle position) and discrete (infinite amount of choices) environment.

Figure 5. Field Of View



The robot progresses by scanning its environment, taking into account all the accessible directions and choosing the one that will lead him to his waypoint with the shortest possible distance. There is always a path that is calculated to reach a waypoint considering the actual view of the robot. Since the robot does not know the location of all the obstacles on the course, the path will initially be imprecise. As Capra6 progresses and detects new obstacles, it will be recalculated multiple times per second and finally allow the robot to reach the waypoint perfectly. That strategy, combined with the large field of view of the robot and the possibility to remember what it has seen, allows it to take the right decisions even in complex situations, such as switchbacks, center islands, dead ends and traps.

Figure 6. Obstacle Detection



This year, the artificial intelligence has been integrated in the new ROS architecture. Since the artificial intelligence acts like a ROS node, it can react to the messages it receives through its subscribed topics. In this case, the AI listens for messages from the extended Kalman filter, the SLAM and the Line Mapper nodes. Those nodes aggregate all the information processed by the robot's sensors into simple messages that can be easily processed by the AI module.

The basic AI developed by the team this year can easily be modified and replaced depending on the conditions and issues encountered in production environments (i.e.: IGVC 2014).

Line Detection Software

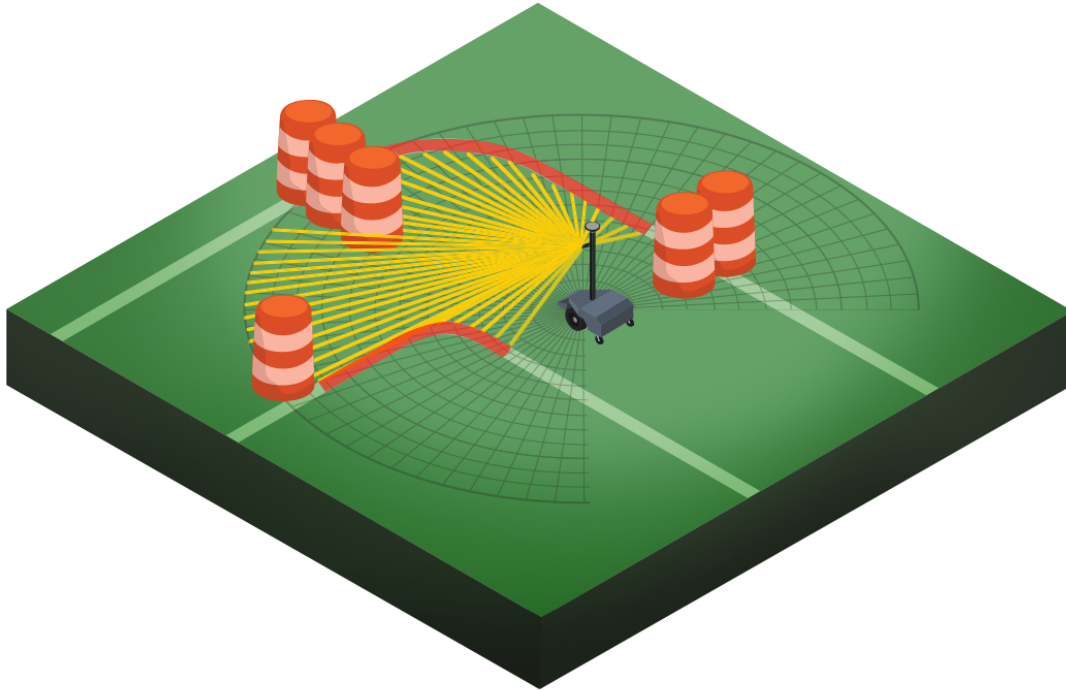
The line detection software has been vastly improved this year. In association with other student clubs at *École de Technologie Supérieure*, Club Capra was able to integrate a common vision server, named *SeaGoatVision*, to its software suite. A ROS node was also created to act as a client to the previously mentioned vision server.

The camera uses a new driver as a 2014 innovation and is able to achieve a throughput of 35 images per second with a 720p resolution. Since the camera has a field of view of 170 degrees, the robot can be on par with other teams without using 2 cameras. Using a single camera makes it so there's no need for hardware or software synchronization between multiple optical devices.

The goal set by the team this year was to be able to have an artificial intelligence that could take decisions at a rate of 20 Hz. In order to achieve this goal, the vision software had to be reworked entirely. With the help of multiple filters to sanitize the output from the camera, the line detection software can

now output lines at a rate of 30 Hz. Besides the optimized filters, the execution of the line detection algorithm was moved to the GPU in order to let the AI use as much of the CPU as possible.

Figure 7. Line Detection



The Theano library made it possible for the line detection software (in Python) to run on the GPU. Theano optimizes the Python code and converts it to native code to be interpreted by the GPU with the help of the CUDA library from Nvidia.

Table 4. Advantages and Inconvenients of using the Theano library

Advantages	Drawbacks
Transparent use of a GPU	Less flexibility for implementation of algorithms
Speed and stability optimizations	Encapsulated automatic optimization
Dynamic C code generation	Steep learning curve

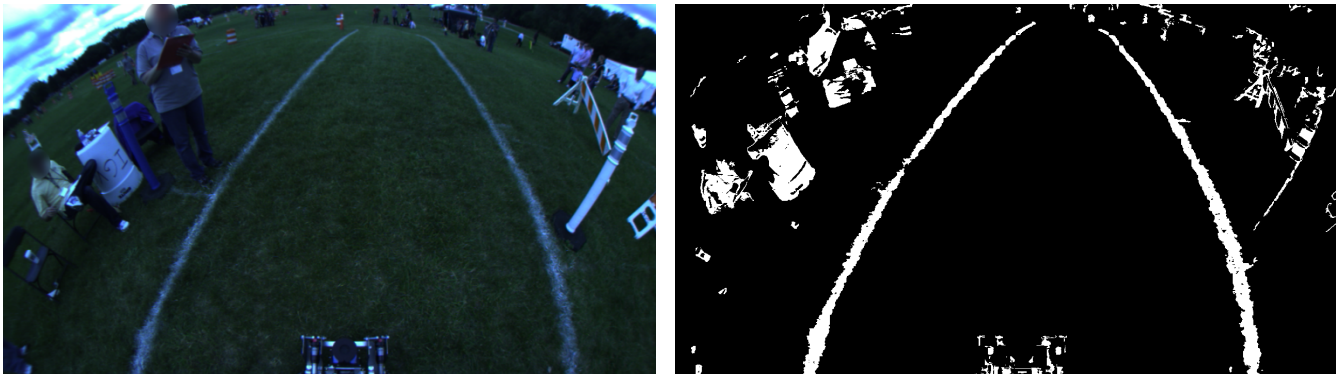
The team did extensive testing of the Theano library and figured out the main advantages gained from using the library were more significant than the main drawbacks. It's important to note that only a small part of the Theano library is actually used to optimize the software, however, future iterations of the robot management software will make use of the library to further reduce the load on the CPU. The Theano documentation¹ states that many computations can be optimized:

¹ Theano 0.6 documentation: http://deeplearning.net/software/theano/tutorial/using_gpu.html

- Only computations with float32 data-type can be accelerated. Better support for float64 is expected in upcoming hardware but float64 computations are still relatively slow (Jan 2010).
- Matrix multiplication, convolution, and large element-wise operations can be accelerated a lot (5-50x) when arguments are large enough to keep 30 processors busy.
- Indexing, dimension-shuffling and constant-time reshaping will be equally fast on GPU as on CPU.
- Summation over rows/columns of tensors can be a little slower on the GPU than on the CPU.
- Copying of large quantities of data to and from a device is relatively slow, and often cancels most of the advantage of one or two accelerated functions on that data. Getting GPU performance largely hinges on making data transfer to the device pay off.

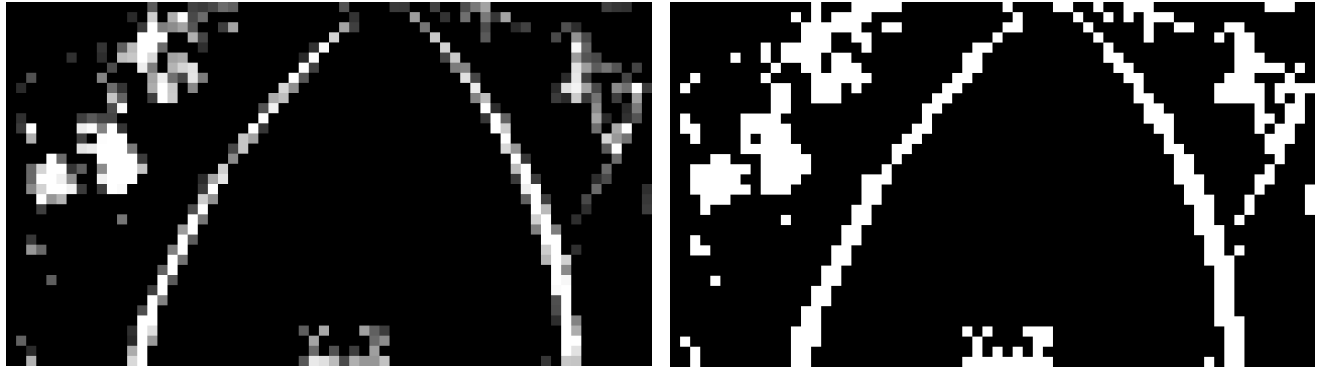
The line detection algorithm uses the RGB values and the difference between each of those values (the differences between R and B, B and G, and R and G) to efficiently detect the lines. Using a good camera calibration and a simple threshold to remove unwanted noise, the software can detect most of the lines on the ground and ignore obstacles even in extreme conditions.

Figure 8. Line Detection Software Output



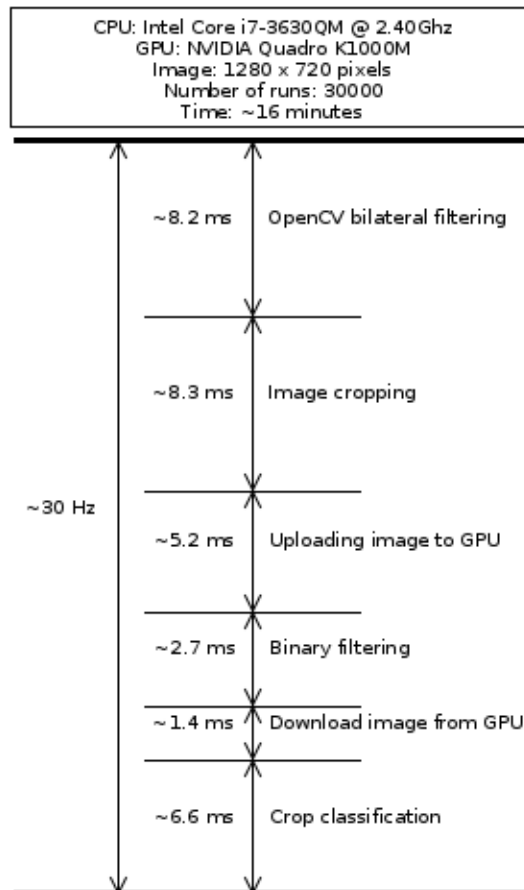
The detected lines are eventually sent to a ROS node which publishes them as a message that can be parsed by a SLAM node. On a higher level, the lines are treated as obstacles making it easier for the AI to take the best decision.

Figure 9. Line Data Formatted For Artificial Intelligence Module



As shown in figure 10, in-depth profiling of the line detection algorithm was executed in order to have the best performance as possible. A response rate around 30 Hz was achieved with this careful analysis.

Figure 10. Line Detection Performance Graph



Line mapping

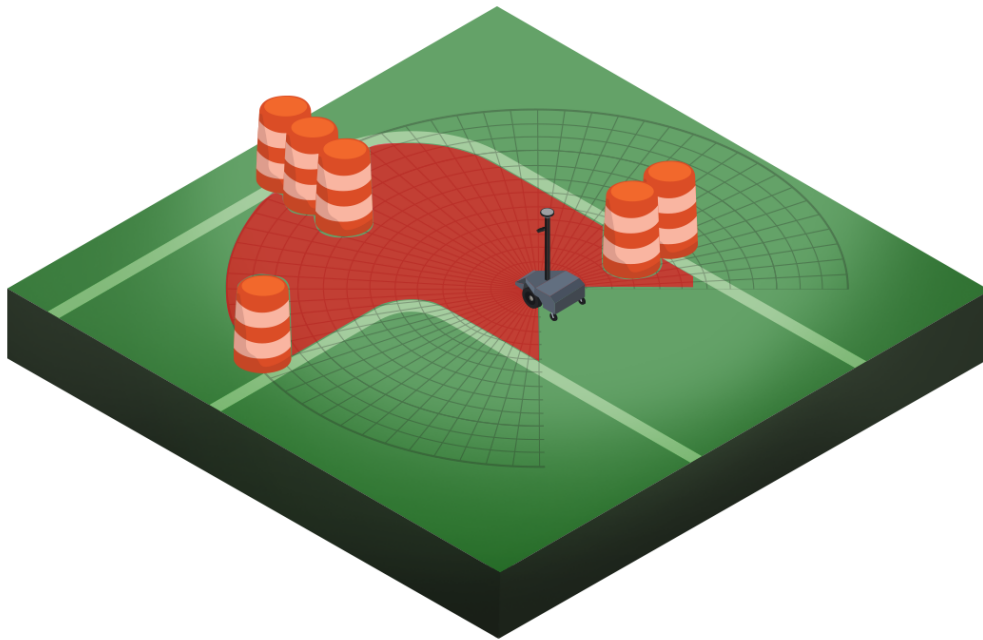
Unlike last year, it was decided that no “*undistort* filter” should be applied to the camera feed. Keeping the original image eliminates an important load on the CPU. Furthermore, no data loss occurs when keeping the original feed giving the robot a cleaner data set.

Each 1280x720 image is then transposed in a 64x36 matrix where every cell (called *element* in this context) represents a 20x20 pixels piece of the original image. Each element is characterized by the angle and the distance of the range finder. Since the range finder is used as the origin of obstacle mapping, the merging process of the line and obstacle maps becomes more accurate.

The value of each element is determined by the number of white pixels in the 20x20 pixels piece it represents. If the value is higher than a certain configurable threshold, the AI will consider this element like a line.

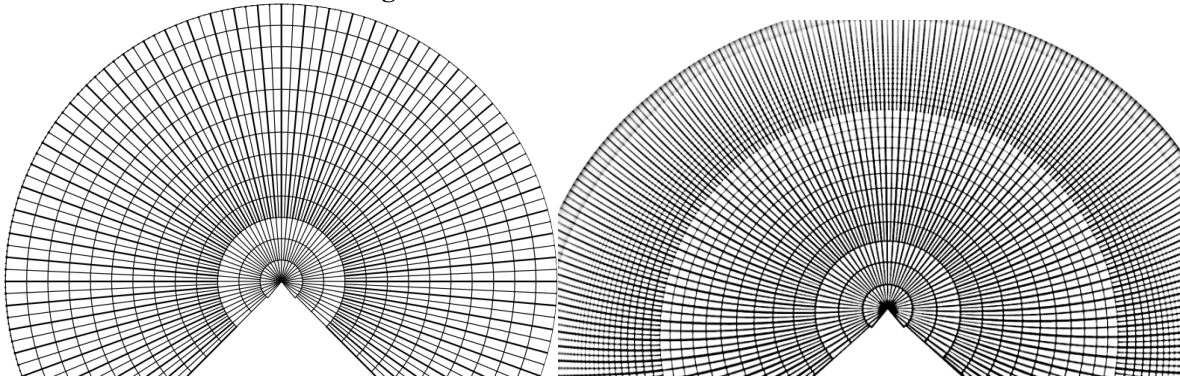
The end result of the line mapping process gives the AI a simple map to help it make decisions based on detected lines. This method works well for dashed lines as long as the spacing between lines is smaller than the robot’s width.

Figure 11. Obstacle And Line Maps Processed By AI



However, the camera needs to be calibrated after every focal change on the lens because of the distortion. This process can be simplified with the help of a *calibration poster* as seen on the left of figure 12. The right part of figure 12 demonstrates the distortion effect created by the lens in order to achieve a 170° field of view.

Figure 12. Camera Calibration Poster



The calibration is necessary in order to associate a distance and an angle to every element in the matrix. The distances between each element is computed after calibration using simple trigonometry. Using this method means the line mapping does not need to hog the CPU at runtime.

CONCLUSION

The main objective of this year is to finish at least in the top 6 in the 22th edition of IGVC. The team also wants to improve its position in the JAUS challenge. Last years, Capra6 reach the 5th place in this challenge. This year, the team will be able to show the judges how the ROS architecture improves the performance of the robot. The team is confident that the mechanical design of Capra6 will help it tremendously navigate the uneven terrain and to move around obstacles more efficiently Furthermore, the team is also expecting the electrical system to be reliable and that the software improvements will boost the reaction times of the robot in order to make swift and correct decisions.

APPENDIX I. Electrical System Plan

