Minotaurus Design Report 2012

Presented to the 20th Annual Intelligent Ground Vehicle Competition

Capra Club
École de technologie supérieure

1100, Notre-Dame Ouest
Local A-1746
(514) 396-8800 x.7999
http://capra.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, ENng. Ph. D.

Chairman, Departement of Software and Engineering

Faculty Advisor, Capra

École de Technologie supérieure (ÉTS)

Table of Contents

1. INTRODUCTION

The club was founded in 1996 by a group of students passionate about the world of robotics. Capra is a student science club, which has as main goal the design and implementation of autonomous vehicles.

Last year, we built a new robot named Minotaurus. This robot is able to move around obstacles, flags and white lines painted on the ground. This year, Minotaurus has been improved so it could give a better performance at the competition.

1.1 Team composition

The scientific club's team is composed of students of the Baccalaureate level from École de Technologie Supérieure, which is an engineering school with five different departments. Capra is proud to have members from all of those departments and capitalize on every one's forces to accomplish the improvements of Minotaurus for the 20th edition of IGVC.

As shown in the team's structure (Figure 1), more than 20 active students are currently involved in the project. The captain meets frequently with the three team leaders (mechanical, electrical and software) to follow the process of the production in the different departments. The three main departments are divided in small teams to work on different objectives to improve Minotaurus. Each team reports their advancements to their team leader, who is constantly following the progress, motivating them to surpass their capacities.

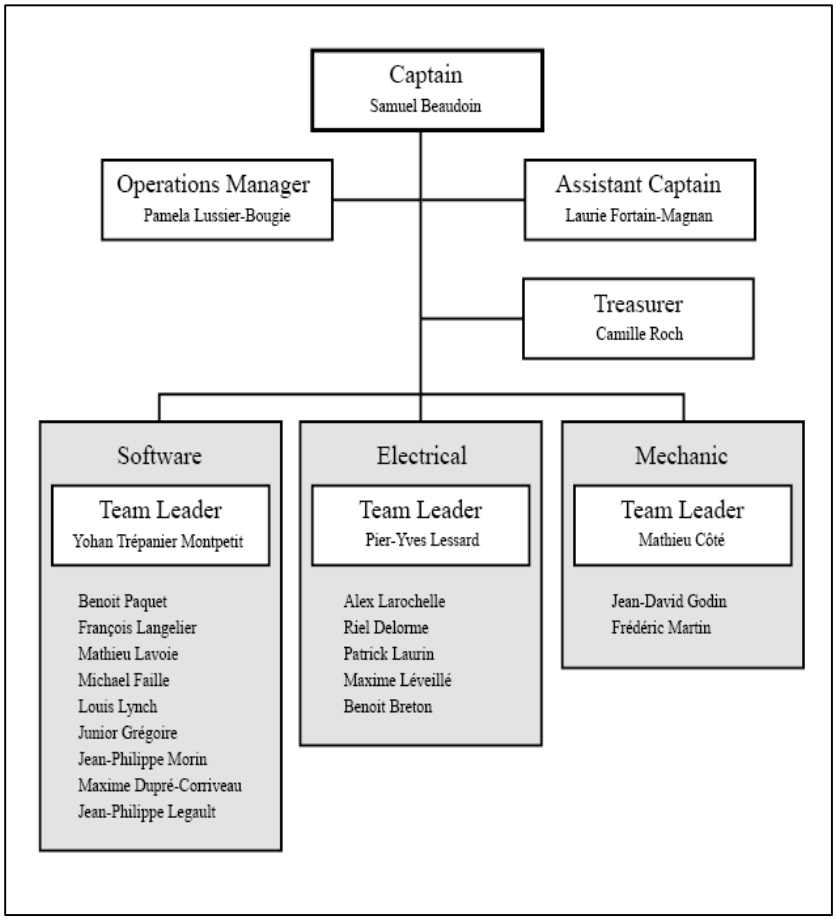


FIGURE 1: STRUCTURE OF THE CAPRA TEAM

Considering the schedule constraints of everyone, tests with the robot are usually done every three weeks (indoor in winter, outdoor in summer). This frequency allows the members to organize their time, meet each other in smaller teams and, later on, learn about the progress of the other departments during a full day of work, usually on the weekend, called *blitz*. During these *blitz*, most of the team is available and working on the robot. After those full days, people get back to work in their smaller teams.

1.2 Design Process

As any improvement process is based on the DMAIC improvement cycle (Figure 2). Capra is continuously using it to guide the innovations of the robot.

FIGURE 2 : THE DMAIC IMPROVEMENT CYCLE

Capra applies the DMAIC in the following way:

Define: The first step of any design process is to define the needs. Last year, since the team participated to the 19th edition of IGVC with a new robot, Minotaurus, the needs and possibilities of improvement were clear. Very 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 analyse the performance of the different aspects of the robot were already in place.

Analyse: These data were used to analyse 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 to produce results.

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

The cycle begins again after the tests; new objectives are defined, new measures are taken, etc. As an example of the application of the DMAIC cycle, we can consider two major aspects of the robot that were improved this year, following the last competition:

- The motors were not strong enough to drive the robot continuously in the grass. They would suddenly block and stop rotating, often requiring human intervention.
- Since the position of the range finder was not adjustable, the sensor detected the grass. This affected the robot in the way that it was not able to follow a path, because it detected obstacles everywhere around him.

The application of the DMAIC improvement cycle leads to the solving of these two problematic situations.

- For the motor problem, the gearboxes were changed and software modifications were done; Minotaurus is now powerful enough to accomplish the whole path within the time.
- The position of the range finder is now adjustable. The team can adjust its angle and height considering the environment; Minotaurus does not detect the ground and the grass as an obstacle anymore.

1.3 Innovations

Many innovations were added to Minotaurus this year. Every change is described in greater details in one of the following sections. During the 2011-2012 year, the team:

- Added a new power controller to manage the power and status of all sensors.
- Added a 48 volts to 24 volts DC converter to remove an extra battery and optimize the robot's weight.
- Built a GPS backpack, allowing for fast and efficient GPS testing and data collecting.
- Installed new gearboxes with an optimised ratio for the current motors.

- Removed 2 motors and their chains.
- Installed an adjustable range finder mount.
- Added a laptop charging docking station.
- Implemented new software tools to record data.
- Enhanced the robot application's user friendliness.
- Implemented a new algorithm to manage the robot's position.
- Implemented new artificial intelligence algorithms.
- Enhanced the mapping system.
- Created an Android application to remotely control the robot.

1.4 Cost

Since we are using the same robot as last year, we did not need the same amount of money. So, we made an estimate of the cost of Minotaurus including the components that we had last year and the new ones of this year.

Electrical	
Ethernet cable	$175.00
Programming cable	$31.00
Sensors	$679.00
Laser Scanner Sick	$2700.00
Control Board	$180,00
Mechanical	
Motors	$5579.97
Wheels	$350.00
Batteries	$2500.00
Batteries Box	$ 110.00
Range finder holder	$20.00
Other material	$2000.00
Software	
Hard drive	$400.00
Dock	$200.00
Laptop	$2000.00
Other	$300.00
Total	$17224.97

TABLE 1: COST OF MINOTAURUS

2. MECHANICAL

2.1 Mechanical design

During all the optimization phase, the team focus on the criteria of keeping the robot simple. In other words, we wanted to build a simpler robot than RS3 (Capra's previous robot). We have improved the reliability and the performance of Minotaurus compared to its predecessors. The fundamental concept of the mechanical design is a two-wheel drive vehicle with a swivel caster.

2.2 Structure of Minotaurus

The structure is made of 1” x 1” x 3/32” aluminium square tubes (6061-T6 alloy) and is completely welded together. It contains two parts: the platform and the motor rack. The dimension of the robot is 40 inches long, 30 inches wide and 30 inches high.

The platform is the upper part of the structure, and also the frame. On the top, a ⅛ inches aluminium plate is used to hold all the electrical components. Under the frame, two stoppers are used to block the polyurethane bump-stops located on the motor rack.

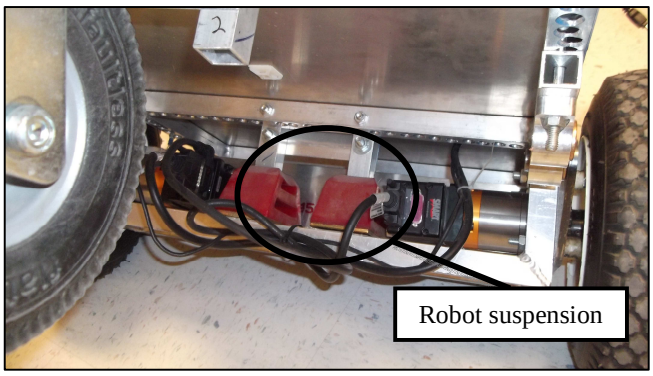


FIGURE 3 : MINOTAURUS SUSPENSION

The motor rack is mounted on the platform by pillow-blocks, and the brass bushing ensures their proper rotation. This system is similar to a swing arm suspension. As illustrated in Figure 3, there is a pivot, a bump stop and a stopper at the end of the course. This year, an aluminium plate was added at the back of the motor rack to strengthen the frame.

A lightweight shock absorption system was added on Minotaurus, because last year, the robot seemed to lose control after a speed bump or when it was running at full speed. This absorption system will also reduce impacts on electrical components.

The robot has two 12.5 inches lawn tractor pneumatic wheels for better traction on the field. Each wheel is driven by a smart electrical motor. They can each produce up to ¼ HP. One of the problems we faced last year was the lack of torque of the wheels. We had 4 motors (linked 2 by 2 with chains) and gear boxes of 20:1. We changed our propulsion system to only two motors and two high precision gear boxes with a 40:1 ratio to obtain the torque we needed to the wheels. These new gear boxes also give higher accuracy at lower speeds.

As usual, a range finder has been placed in the front of the robot. With the specific form of the frame, the range finder can see at a widest range of almost 220˚. Furthermore, we made a new and higher support with fine adjustment for angles for better positioning of the range finder. In addition, the body is now water resistant to protect all the electrical components in case of rain. The interior of the case was shot with easy liner to make it water resistant. The case is made of 0.063 inch 6061-T6 aluminium to have the best cost-value ratio and a minimal weight.

We are also working on a carbon fiber project in order to gain experience with this product. Our objective is to build the body of the next vehicle with it. The current project is a box to support and protect the camera, the remote kill switch and the GPS receptor at the end of the mast.

3. ELECTRICAL

Many changes have been brought to the electrical system of Minotaurus for this year’s competition. Most of these changes are meant to simplify the maintenance tasks and add new features.

3.1 Batteries

The robot used to have separate 48 volts and 24 volts batteries. The 24 volts battery has been removed and replaced with a "flyback converter", used as a DC/DC converter. This converter is basically a Buck-Boost converter that draws its energy from the 48 volts battery with a transformer for electrical isolation. Using the flyback instead of a battery removes a lot of weight from the robot. It also gives more space for mechanical or electrical supplements, and frees space for better heat dissipation. With this new converter, the robot can be used for at least 2 complete days of testing before batteries need to be changed. Also, when we bought the 48 volts battery, it was separated in two different cases. This year, we decided to build our own case to facilitate the transport and reduce the required space.

3.2 Sensors

The sensors are connected to the computer through a few different communication protocols. Figure 4 shows what protocols are used for the different sensors.

FIGURE 4 : COMMUNICATION PROTOCOLS BETWEEN THE COMPUTER AND THE SENSORS

3.3 Power supply

Minotaurus needs six different power sources that are distributed to the sensors, laptop PC and motors. To provide these power sources, we designed our own power supply that takes 24 DC volts at its input and provides 5 volts, 9 volts, 12 volts and 19.2 volts at its outputs. Most of these outputs are then sent to the power controller (see section 2.4) to finally be distributed to the peripherals.

3.4 Power controller

A layer of intelligence has been added over the power supply used by the robot. This system consists of a second PCB connected to the PSU board (Figure 5). This circuit is a microcontroller based system that allows a full control over the power distribution in the robot using a USB protocol.

The power controller receives five different power sources from the power supply which is 5 volts, 9 volts, 12 volts, 19.2 volts and 24 volts. These sources are then distributed to the sensors through simple relays and different connectors. The relays are software-controlled and the different connectors make wiring easier and fault proof. In addition to these power outputs, four general purpose outputs are available. One is used to control emergency stop and one is used for autonomous mode lights. The two others are for future uses.

Furthermore, the power controller monitors the 48 volts battery by reading its voltage and the current drawn by the robot. All the collected data is accessible to the onboard computer via USB.

Controlling power outputs with the computer gives the possibility to hard-reboot any individual sensor in the case of a malfunction and to simply disable them in order to save energy. Also, monitoring the battery voltage allows the power controller to protect the battery from being damaged. Since we use a $LiFePO_4$ battery, dropping its voltage under 39 volts will cause permanent damage to the battery. Thus, the power controller shuts the motors and sensors down when voltage goes under a configurable threshold of 46 volts.

This circuit adds new features to the robot but is not required for the robot to be fully functional. Since all the sensors are connected to it, a malfunction would cause the robot to be unusable. To ensure the stability and the safety of the robot, we have designed a bypass circuit that uses the same physical connectors as the power controller but that has no electronic component on it. In the case of a problem, we can quickly swap the two circuits to get back to a functional power distribution without any monitoring.

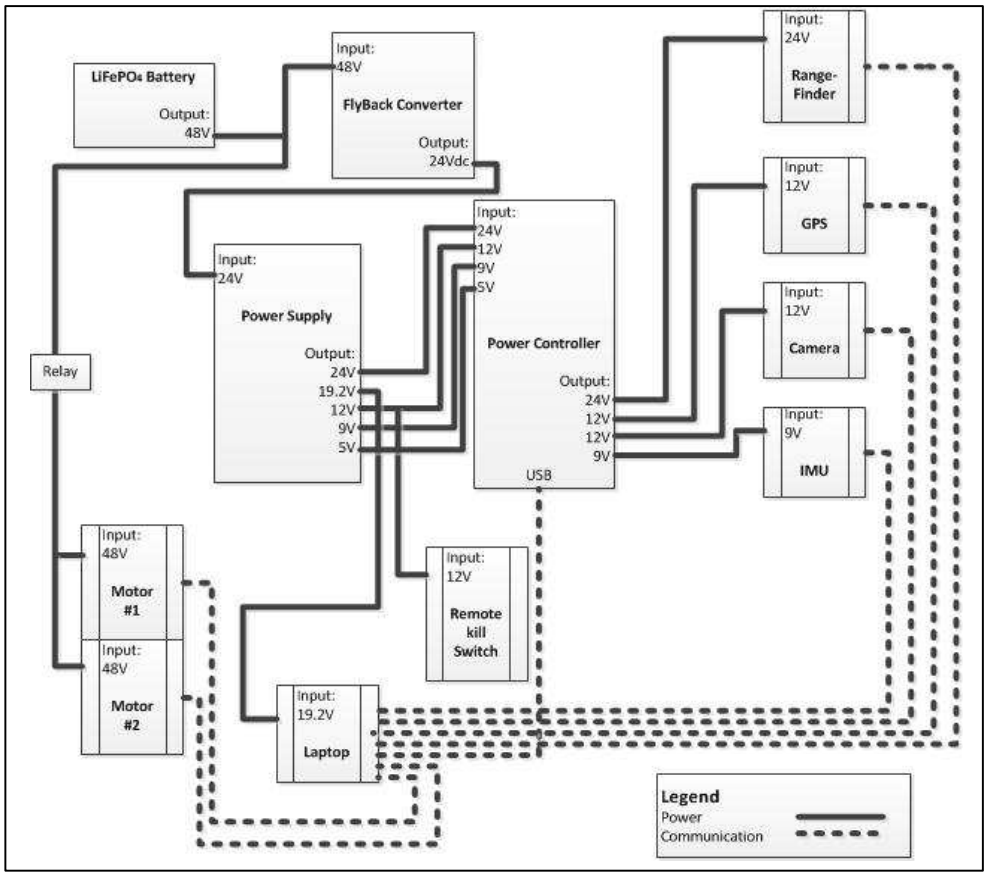


FIGURE 5: DISTRIBUTION OF THE POWER IN MINOTAURUS

As shown in Figure 5, the motors are connected to the battery by a relay contact. This relay is controlled by either emergency stop button, remote kill switch and power controller. This means that motors can be deactivated remotely or by pressing the emergency stop button and also by the software via a USB command.

This year, we have also added a switch on the exterior of the robot. This switch allows shutting down the power of the robot without having to open the robot and to disconnect physically the batteries.

3.5 Dock

This year, we installed a laptop docking station on Minotaurus. Every electric system that needs to communicate with the laptop is connected to it. This docking station allows the software team to rapidly start the robot. The docking station is powered by our power supply and can recharge the laptop battery.

3.6 GPS Backpack

A considerable innovation related to the testing of the robot is the GPS backpack. It has been built by using a simple used backpack and a basic wooden structure. The backpack and the structure hold a 12 volts battery, a Novatel Propak V3 GPS receiver and a Novatel GPS-702-GG antenna. These components are identical to the ones installed on the robot and ensure a perfect compatibility with our application. The backpack also holds a custom set of wires with a switch and a fuse to get power from the battery to the receiver. This set of wires replaces the original power input cable. With the backpack, we can collect

GPS coordinates by using only a laptop, without having to move the robot. It is a lot more efficient than going around with the robot and drastically reduces the time needed for test preparations.

4. SOFTWARE

At the software level, many innovations have been brought on for the upcoming competitions. First of all, several tools and software improvement have been developed in order to enhance the interaction between the user and the robot software. Robot tests will be more efficient and easier to accomplish. Second of all, new software tools allow to gather data from the robot and save it in order to later graphically representing the behaviour of each individual sensor. A new algorithm based on statistics and outlying data has also been implemented to manage the position of the robot. Finally, motor controls have been improved to use the full power potential of the motors and thus react more efficiently to software commands.

4.1 Computers and application

The application controlling Minotaurus runs on an HP Elitebook 8440p laptop (quad core i7 with 4GB of RAM) that is docked on the robot. Most of the sensors (USB, Ethernet and Firewire) are connected to the computer through a docking station. The computer runs Ubuntu Linux 11.04.

All the software used onboard the robot is free and open source. It incorporates third-party software libraries and applications published under various open source licences. For closed source software required to control the various devices (GPS, motors, etc.), custom implementations were made to replace them. Our vision library is OpenCV, a free open source library licensed under a BSD license. A few components, such as JAUS and the artificial vision, were even developed as shared projects between the 4 robotics teams of the university. Since our onboard computer uses Ubuntu, a popular, commercially supported GNU/Linux distribution based on Debian, our application should also run on any Debian derivative using 100% free software, such as Trisquel. Most of the tools and command line applications provided with the operating system are also free and open source.

One of the projects for the upcoming year is to release the code of our application to Github and make it a free, open source project. Other universities or robotic project teams could use it and improve it. All our code is in Java, except for the vision module, which is in C++.

4.2 Simulator

Our software contains a very useful tool, the simulator (Figure 6). The simulator can do the following:

- Replace real sensors and actuators by simulated ones.
- Test and run Artificial Intelligence strategies in a virtual environment.
- Output a variety of information concerning the robot's virtual status.

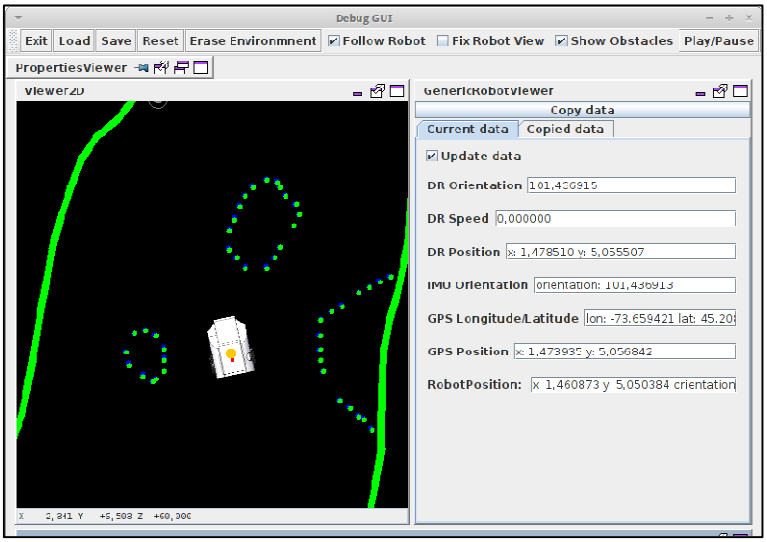


FIGURE 6 : THE SIMULATOR SHOWING THE ROBOT AND POSITION INFORMATION

4.3 Managing Versions and Dependencies: Maven

Since we often implement new drivers or modules, our main application ends up depending on many other projects and software libraries. Managing all the dependencies between the projects is quite hard. For this reason, a system is used to manage all of these automatically.

An application called Maven is responsible of downloading and configuring all the required dependencies of a project. The only element that has to be maintained is a file that describes these required dependencies. This system greatly simplifies the work of the developers since they do not have to worry about external projects or libraries.

4.4 Low-level components

4.4.1 Sensors

The robot uses a variety of sensors to guide itself through its environment. The most useful sensor is the rangefinder. The robot is equipped with a SICK LMS100 and it communicates with the computer through a TCP/IP interface. It has high detection capabilities by being able to see more than 7 meters away from the robot. It has a range of 270 degrees and takes one measure every half degree.

The robot has a PointGrey FireFly 2 Firewire camera that communicates directly with the laptop to be able to detect lines and flags. The feed from the camera is passed through a software filter that takes in consideration the angle of the camera and places the lines adequately on a 2D plane according to their position in the images.

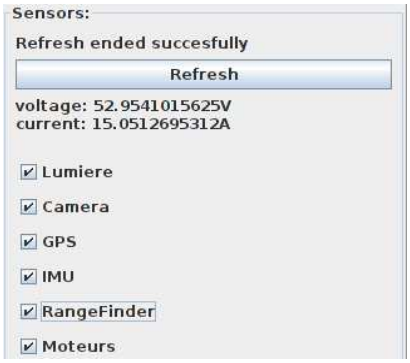
To follow the movements of the robots, encoders are present in every motor. They have a resolution of 4000 steps per turn, which allows us to know very precisely the position of the two motors. Our system uses the encoders to calculate the position and the orientation of the robot. Since the robot can physically drift and since the wheels can slip, we use a GPS and an IMU to correct the error.

The GPS has a precision of approximately 5 cm when it is placed in an open field like the one of the competition. It is used to reach GPS waypoints and to correct the position error calculated with the encoders.

An Inertial Measurement Unit (IMU) is used to determine the orientation of the robot. It is used at the beginning of a run for the robot to be able to move in the right direction towards a GPS position. It is also used during the run to correct the orientation error calculated with the encoders. The model is a MicroStrain 3DM-GX1.

An innovation from last year is the implementation of a tool able to dynamically provide the status of any sensor at any time during a run (Figure 7A). With the lights color, it is now a lot easier to tell if a sensor is running, in simulation mode or disabled. Also, the newly built power controller (see section 1.4) allows the user to reboot any sensor through the user interface of the application (Figure 7B). The user can also easily monitor the tension of the batteries.

A) SENSOR STATUS VIEWER DURING A SIMULATION RUN



B) SENSOR CONTROL INTERFACE

FIGURE 7 : SENSORS STATUS

4.4.2 Motors

The robot has 2 Animatics SM23165DT Smartmotor wheel motors. They are servo-motors, and allow us a very precise control over their rotation. We can communicate with them through an RS-232 interface. The motors are daisy-chained together.

The motors have the capabilities to let us know the state of each of them. We can diagnose problems, may they be electrical or mechanical. For example, detecting that a motor is stuck in a hole allows us to get out of trouble by using the other one.

4.4.3 Data saving

Since last year, many innovations have been implemented to optimise the speed of development and the efficiency of the robot in every competition run. Last year, the robot could not save the data acquired during a run. Right after the competition, functionalities were added to save the data from many of our sensors and to replay that data in our simulator. That is a crucial element to understand the reasons why the robot has failed during a run. A technique to view the recorded data in graphs has also been added to the system to debug sensor problems.

4.5 High-Level components

High-level components use the data of the low-level components to make calculations and take decisions.

4.5.1 Position Management

Our robot uses many sensors to manage its position. In our main application, we have a service called the “Position manager”. This service has the responsibility to get data from the GPS, the encoders and the IMU (for orientation) and analyze it to obtain the current position. It then sends that position to the artificial intelligence. The position manager is a key component of our software. This key service works with an algorithm that averages the position with a Z score. This algorithm starts by calculating the current position with data obtained from a single sensor. It then does the same with every others sensors. The next step is to calculate an average position and a standard deviation with all the obtained positions. Finally, it calculates a Z score for all the obtained positions together. If a position has a Z score below -2 or above 2, this position is considered as an outlier. It is then removed from the position list and a new average position is calculated. This average position is sent to the artificial intelligence. Figure 8 shows the statistical distribution of data. Any data included in the grey area is considered good.

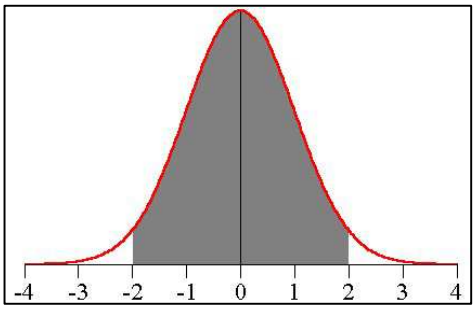


FIGURE 8: STATISTIC DISTRIBUTION OF SENSOR DATA

In future versions of the application, the position manager will use a Kalman filter as its positioning algorithm.

4.5.2 Computer vision

The vision system is a shared component between various robotics teams of our university. Every team implements the features that they require.

The system is composed of a server and a client. The server is responsible to grab images from the camera, apply filters and then to broadcast it to all connected clients. The clients are responsible of interpreting the results and send the results to the AI.

To avoid having to transfer all the images through the client-server channel, the server only transfers string representations to the client. These string representations, called "descriptors", can represent anything we want. For example, we have a descriptor to describe line positions and orientations and another one to describe flag positions. We are currently using Hough transformation to detect lines, and the information transferred is minimal (4 integers representing the coordinates of the endpoints of each line).

The vision execution time is optimized through this system compared to previous years because the treatment is done on the server-side (C/C++) instead of on the client side (Java). These modifications lead to a more reactive robot.

The filters that are mostly used are Hue-Saturation-Brightness (HSB) filters combined with Hough transformation filters. The C++ implementation uses the OpenCV library to access the filters. Since the camera, like the human eye, presents images with a perspective deformation, a filter is used to calculate the actual position of the pixels in a 2D plane. The Figure 9 shows how an image is processed. The initial image (left) is modified to compensate for the perspective distortion (center). A HSB filter is then applied and a Hough transform detects the line (right). In that last image, the red line represents the information that will be available for the AI.

FIGURE 9: IMAGE PROCESSING EXAMPLE

4.5.3 Mapping

The robot is able to remember the location of every obstacle it sees (Figure 10). To be able to reproduce a useful replay after a run, the positions of all perceived obstacles are recorded in a file on the computer.

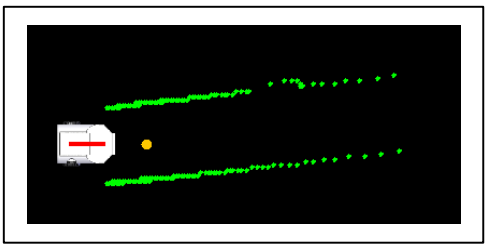


FIGURE 10 : MINOTAURUS MAPPING ITS ENVIRONMENT

Through testing, we have noticed that the robot can accumulate position error, especially indoors without the GPS. If the error gets important, a map of the environment could also become erroneous. Such a situation could result in the robot believing an obstacle is present where the path is actually free. A similar problem could happen with moving objects; if the robot remembered them as obstacles, a complete path could be blocked. For this reason, we have implemented a system in which obstacles disappear from the robot's visual memory after a predefined time.

4.5.4 Software Strategy

To avoid obstacle, follow lines and reach GPS waypoints, new artificial intelligence strategies were implemented this year. One of them uses an algorithm similar to "A star", modified to react in a stochastic (random obstacle position) and discrete (infinite amount of choices) environment. The robot progresses in its environment using a tree of decision (Figure 11). Every branch of the tree has a cost based on multiple heuristics: Detecting an obstacle (rangefinder).

- Detecting a line (camera).
- Following the orientation of a line (camera).
- Keeping the same orientation as the previous branch.
- Keeping the same orientation as the previous X seconds.
- Going towards a path that has already been visited.

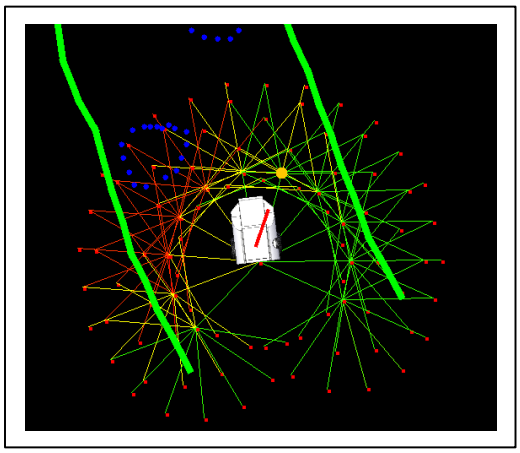


FIGURE 11 : MINOTAURUS AVOIDING AN OSBTACLE WITH A TREE STRATEGY

The robot receives new data about ten times per second. Using this information, it is possible to follow the lines and avoid obstacles even if lines are dashed, since the algorithm will still remember that it should orient itself the same way the lines do.

The heuristics, based on a variety of factors, allow the robot to take the right decisions even in complex situations, such as switchbacks, center islands, dead ends and traps.

Every artificial intelligence we design has parameters that can be modified at runtime and through configuration files. For the robot to be efficient during high-speed operations, we start by modifying the parameters in a virtual simulated environment to see how the decisions are impacted by a higher speed. Once the parameters of the artificial intelligence have been adjusted to make the robot efficient in the simulation, the settings are applied to the physical robot and tested in a real-world environment.

4.5.5 JAUS

For the JAUS challenge, we use a code library that is shared between 4 unmanned robotics teams of our university. All teams had to implement the JAUS specification for their respective competitions. To optimize the time we had, we decided to develop and maintain a single code repository and share it between us. From this library, every team is able to create a custom implementation of the specification in order to support the different JAUS services for their own competition. All the communications, messages and records are taken care of by the common library. Each team simply has to specify their subsystems, nodes and components and register the JAUS services for them. Those JAUS services, common to all teams, contain a delegate to which every team attaches its own method. When a JAUS message is received, the method attached to the delegate is invoked and the data from the JAUS message is sent from the common library to our application. It can then be executed by our vehicle. By using this approach, all the projects can reuse the same code for implementing the JAUS specification and they can all adapt it to their specific architectures or needs.

4.5.6 User Interface

Improvements have also been done in the user interface of our application. All tools have been regrouped at the same visual location and are now more accessible. Our configuration files have also been sorted in a tabbed interface, which is much more user friendly than the previous display system. Also, our simulation software now has play-pause capabilities, allows drag-and-drops of the robot and has a better implementation of the environment simulation.

4.5.7 Android Application

An Android application was developed to allow the members of the team to remotely control the robot. Bluetooth was chosen for the communication protocol because most smartphones possess a Bluetooth antenna. This also eliminates the need for a wireless network. To ensure the safety of the user and the public, the Bluetooth device uses a pulse system, continually sending the same commands. As soon as the robot stops receiving commands, it stops moving. This ensures the robot will stop moving as soon as it gets out of reach of the Bluetooth device.

5. CONCLUSION

With the hundreds of hours the team has invested in the project, many aspects of Minotaurus have been improved since last year. The team is expecting to obtain better results, especially on the autonomous challenge. The motors are stronger, the electrical components are more reliable, the software is easier to use and the artificial intelligences have been greatly improved.