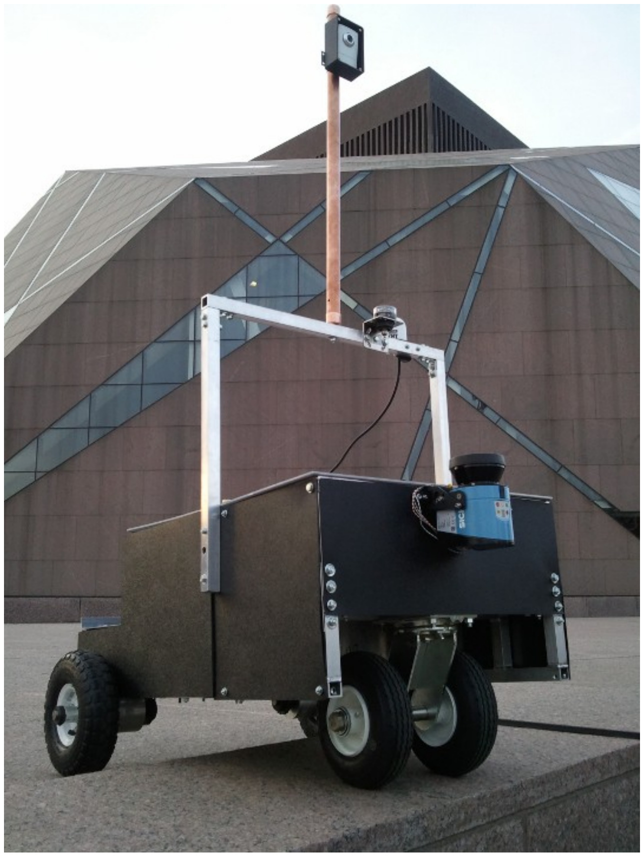
2013

The University of Minnesota - Twin Cities and GOFIRST present:



Ground Squirrel

Student Members:

Bryan Stadick – Junior, Electrical Engineering
Max Veit – Junior, Computer Science & Physics
Emal Alwis – Junior, Electrical Engineering
James Crist – Senior, Mechanical Engineering
Alex Dahl – Freshman, Computer Science
Terry Furey – Freshman, Math
David Haugen – Senior, Computer Science
Adam Juelfs – Freshman, Chemical Engineering

Faculty Advisor:

Dr. William Durfee
Professor of Mechanical Engineering

[UNIVERSITY OF MINNESOTA IGVC DESIGN REPORT 2013]

I certify that the engineering design present in this vehicle is significant and equivalent to work that would satisfy the requirements of a senior design or graduate project course.

Signed, ____, Dr. William Durfee

1. Team Organization and Design process

1.1 Introduction

This year, GOFIRST and the University of Minnesota are proud to enter the 21st annual Intelligent Ground Vehicle Competition with their entry Ground Squirrel. Ground Squirrel was designed and built by a self-motivated, self-run conglomerate of students at the University of Minnesota - Twin Cities. The team, under the student organization header GOFIRST, consists of students from several different majors across the University. This is the group's first year entering IGVC and the latest team the U of M has hosted at the competition since 2007.

1.2 Team Organization

The core team consists of eight members from varying majors. The team organization consists of a project leader and two sub-team leaders covering hardware and software respectively. The project leader's responsibilities consist of keeping the project schedule, organizing meetings and members, tracking the team's financial status and doing any other administrative tasks. The hardware and software leaders are then responsible for assigning tasks to members, providing guidance and assistance in their respective specialty area and ensuring the progress is being made toward the final product.

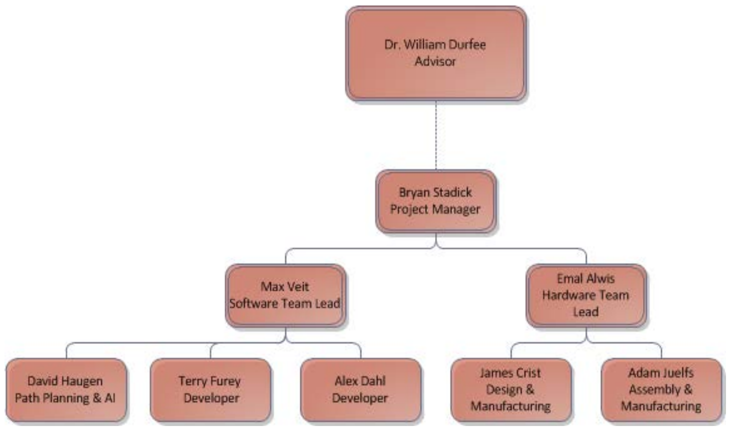


Figure 1. Hierarchy of team leadership.

1.3 Design Process

In the design and construction of Ground Squirrel, an agile development methodology was followed. Usually used in software development, we adapted the agile development process to work just as well for hardware as it does for software. Of the various agile development models we utilized the SCRUM model by running a series of "development sprints." Each sprint consisted of a goal/schedule stage, design and review stage, development and production stage and finally a testing and review stage. A condensed structure of a sprint is shown in Figure 2. In total, nine sprints were carried out over the development of Ground Squirrel, and the content of each sprint is displayed in Figure 3.

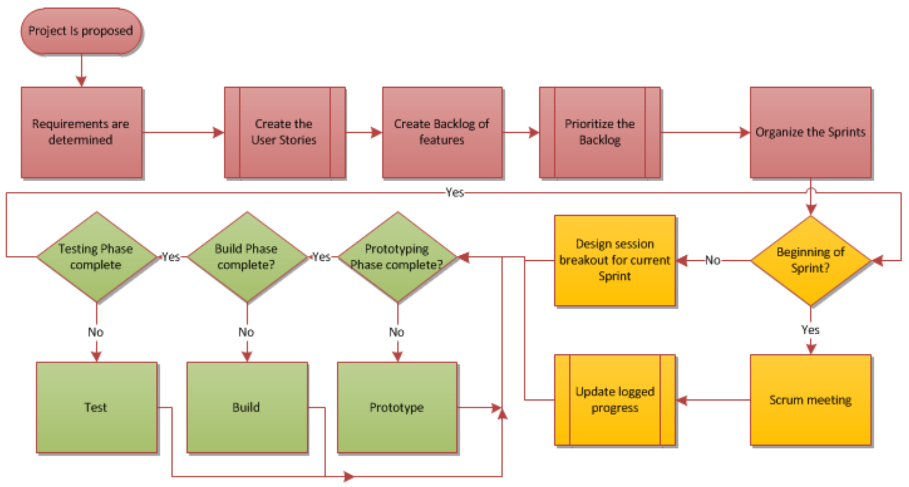


Figure 2. Structure of a SCRUM sprint.

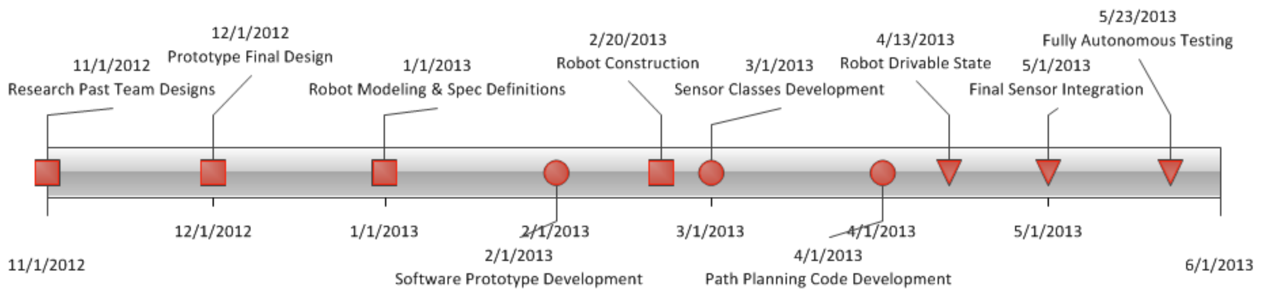


Figure 3. Timeline of sprints.

2. Hardware (drive and chassis)

The chassis consists of two levels, an upper non-load bearing and lower load bearing level. This leads to one of the innovations of Ground Squirrel:

- Upper chassis is completely reconfigurable and bears no structural significance (section 2.1)

2.1 Chassis

The chassis of Ground Squirrel was designed for manufacturing simplicity, as well as reusability. Aluminum extrusion was used wherever possible due to its high strength-to-weight ratio. The chassis is composed of two pieces: the load-bearing base, and the nonstructural top. A ladder geometry was used for the base; allowing for easy construction while maintaining rigidity. All joints on the base were welded to maximize the chassis strength. The remaining frame members are all non-loadbearing, and were bolted on to form the frame.

The advantage of this design is that it is reconfigurable. Since the base layer includes all structural components, a new top layer can be bolted on for a different robot configuration. Further, this minimizes the amount of welding required, reducing the cost of manufacturing.

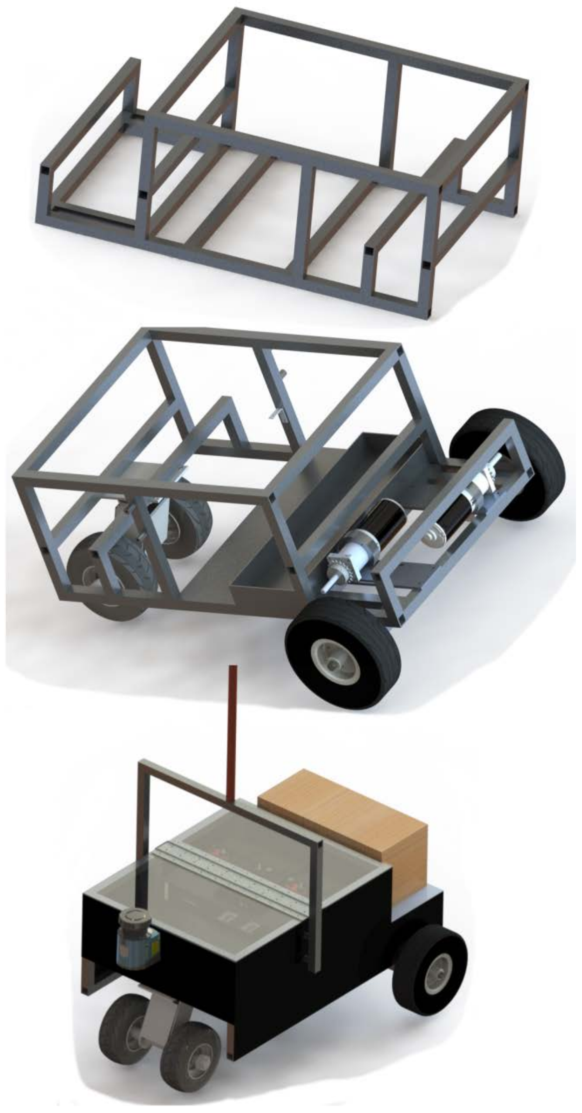


Figure 4. CAD models of the robot chassis, drive system and fully assembled.

2.2 Drive

A differential drive was chosen due to its simplicity. The two drive wheels each are powered by a brushed DC motor. Turning is accomplished by driving one wheel faster than the other, allowing for a simple control algorithm.

The third wheel is a dual wheel industrial caster. Pneumatic tires were used for all three wheels to allow Ground Squirrel to traverse rough terrain. Due to the inertia of the caster, the robot drives in a leading caster configuration to prevent fishtailing.

To determine the ideal drive configurations, a nonlinear model of a differential drive robot was developed and simulated using a Python program written by the team. This allowed the effects of hardware changes to be simulated before building commenced. It also aided in control development, as discussed in Section 7.2.

2.3 Waterproofing

To keep everything dry the robot body is fully enclosed and waterproofed. Thin ABS sheeting is used for the skins due to its low weight and low cost. Weather stripping was used between all connections to keep the enclosure waterproof. The side access panels were attached with Velcro, and the top panel was hinged to provide for easy access. All other panels were bolted on.

3. Hardware (electronics, controls and sensors)

Ground Squirrel consists of three layers of electronics: control and interconnections, sensors, and power distribution. Each subsystem is broken down further in the following subsections. Some key points of innovation include:

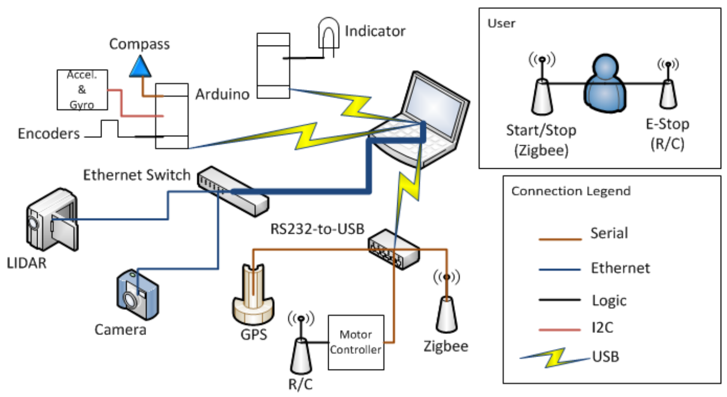
- Modular control board and connectors for easy removal from robot and for use outside of robot (section 3.4)
- Multi-level control board system for wire runs and isolating subsections (section 3.4)
- Separate breakouts for odometry data and status data/indicators (section 3.1)

3.1 Controls and Interconnections

The brains of the robot consist of a Lenovo IdeaPad Y580 laptop running an Intel i7-3630GM at 2.4GHz, an NVIDIA Geforce GTX 660M GPU and Ubuntu 12.04 LTS. It has been upgraded with 16GB of DDR3 RAM and a 128GB Samsung SSD. The main decision for using this particular laptop was the price point and the ability to run CUDA on the GPU in future competitions. The choice of upgrading RAM and hard drive was so that we gain a large amount of caching in faster volatile memory and to have a completely solid-state system that is capable of handling large shocks on rough terrain.

For interfacing to the laptop, a combination of Ethernet, USB and RS232 is used. A Cisco 200 series gigabit Ethernet switch is used to connect multiple Ethernet-enabled devices together. A StarTech 4 port USB to DB9 RS232 adapter is used to breakout a single USB port into four RS232 ports for direct communication with most sensors on the robot. USB is then used to connect the devices capable of USB communication such as Arduino.

Arduino's are used to breakout to lower level sensors and indicators such as lights, encoders and the accelerometer. Two Arduino Mega boards are used, each with a specific subset of data to handle. One Arduino is used for collecting and formatting odometry data while the other is used for sending and receiving less critical status data and control signals.



All data and control signals are ultimately collected, processed and generated on the laptop. The control signals are then distributed to the various other parts of the system. Figure 5 shows the controls and communications layout.

3.1.1 Motor Control

Ground Squirrel uses a Dimension Engineering's Sabertooth 2x60 motor controller. This motor controller provides many different features and contains two channels capable of running two separate motors at 60A on each channel with a peak voltage of 120A at 30V. It is internally fused via a thermal breaker and diode protected for overcurrent and reverse current protection. A 5V supply is also present on the controller. More about how this is utilized is described in section 3.1.2 below.

This motor controller allows for several forms of control including PWM, analog voltage, R/C and packetized serial data. We are utilizing the packetized serial control which allows for the most customization and direct communication from the laptop. This method of control then enables the Sabertooth to use a built-in E-stop as described in section 3.1.2 below. The Sabertooth has built in regenerative drive as well, meaning that when the motors are commanded to stop or slow down, the Sabertooth will instead redirect current stored in the motors back into the battery, thus saving power and extending the runtime of our batteries. Other features of the motor controller include built in speed ramping, differential turning and stopping.

3.1.2 Safety Features

Safety is a main priority in the construction and functioning of Ground Squirrel. The Sabertooth motor controller when in packetized serial mode allows for a built-in E-stop. By simply grounding the second signal input (S2) on the motor controller the motors will come to a complete stop. Two forms of locking E-stops are present on the robot. The first is the on-board E-stop which, when pressed, grounds S2 bringing the robot to a full stop in hardware. The second E-stop is the wireless E-stop which connects to the robot via an R/C radio. By simply having the wireless E-stop turned on and pressing the button, a relay is flipped on the robot and S2 is grounded. If the wireless E-stop is not on or loses signal, S2 will be grounded by default. While the stop is in hardware, the software will also pick-up on the changed state and go into shutdown mode, stopping any further program execution.

Ground Squirrel also contains a watchdog timer. If it goes for some time and does not receive a signal from the start/stop switch or teleoperated control, depending on the mode of operation, it will go into a stand-by mode until communication is reestablished or the robot is shutdown.

The entire robot power system is also protected by a 150A thermal circuit breaker which is accessible from the outside and fully weatherproof. More about this is described in section 3.3. All electronics are also insulated from the chassis and each other, or they are properly grounded.

3.2 Sensors

Most of Ground Squirrel's sensors are located on the exterior of the robot or separate from the control board and are connected to the control board via a single connector. This allows for easy disconnection and removal of the control board from the robot. The only sensors attached to the control board are the OSEPP I2C accelerometer and gyroscope. The most crucial sensors are described in more detail below.

3.2.1 GPS

The robot uses an Ag Leader GPS 1500 integrated antenna/receiver WAAS DGPS. This GPS provides sub-meter accuracy at an affordable price point. The data is output at 10Hz and is formatted in the NMEA standard for positional data over a serial connection. Along with positional data, the GPS outputs a simulated radar speed reading. The unit is fully weatherproofed and easily mountable to any flat surface.

3.2.2 Compass

Heading data is acquired from a Digital Yacht HSC100 Fluxgate Compass. The heading is output at 10Hz via a serial connection and is formatted in the NMEA standard for heading data. The sensor is capable of 0.5° resolution and can output accurate headings with up to 45° of tilt on two axes. The compass also includes built in compensation and calibration for handling magnetic interference.

3.2.3 LIDAR

The primary obstacle detection used by Ground Squirrel is a SICK LMS100-10000 Laser Range Finder. The LMS100 provides 270° degrees of measurement with 0.25° of resolution, a range of 20m, and accuracy of ±30mm. Data is generated at 50Hz and can be queried from the device over serial or Ethernet.

3.2.4 Camera

Vision data for line detection and flag identification is gathered using an Axis M1013 Network Camera. This camera provides H.264 of MJPEG streams over an Ethernet connection for higher bandwidth. Images are streamed with a resolution of 800x600 at a rate of 30fps with a horizontal viewing angle of 67°. The camera has the ability for color, brightness, sharpness, contrast, white balance and backlight compensation built in. Multiple streams are also able to be broadcasted simultaneously from the camera.

3.2.5 Encoders

Attached directly to the output shafts on the drive wheels are two US Digital E4P, 360 count optical encoders. The encoders are used for distance tracking along with velocity estimations in conjunction with the accelerometer and gyroscope combination.

3.3 Power Distribution

Ground Squirrel's power supply consists of three levels of voltages all supplied by two 150Ah deep cycle marine batteries connected in series to provide 24V. The 24V main line connects to a marine rated, unfused terminal block. From there the 24V line is stepped down to 12V and 5V via a 20A and 10A DC-DC step down converters respectively. The power from the 12V and 5V lines is then distributed via marine rated fuse blocks with ground planes. The fuse blocks accept standard ATC automotive fuses which are available in sizes from 1A to 30A.

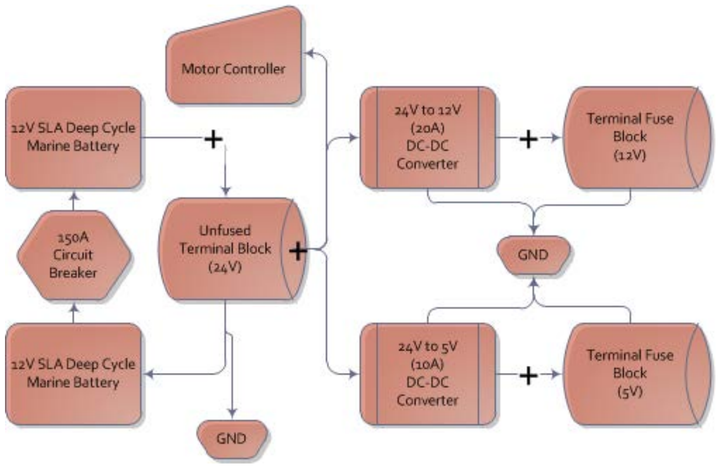

Motor Controller
12V SLA Deep Cycle Marine Battery
150A Circuit Breaker
12V SLA Deep Cycle Marine Battery
Unfused Terminal Block (24V)
GND
24V to 12V (20A) DC-DC Converter
Terminal Fuse Block (12V)
GND
24V to 5V (10A) DC-DC Converter
Terminal Fuse Block (5V)
+

Figure 6. Power distribution layout.

The entire power system is then protected via a 150A waterproof, marine rated thermal circuit breaker. The breaker will trip under a worst-case scenario motor stall where both motors are pulling 80A, and it will trip during a short. The protections built into the motor controller are then enough to protect against any transience that may occur in other situations.

The breaker is positioned between the batteries so as to allow for charging of the batteries in parallel with the breaker open or in series with the breaker closed, as shown in Figure 6. The battery charger used allows for the charging of the batteries in series and while they are in use, which enables for the robot to perform low power operations while the batteries are being charged. The battery also has a charging/testing port attached and a quick-disconnect connector for easy removal from the robot.

3.4 Control Board Structure

The control board containing the laptop, power distribution, motor controller, Arduino's and serial hub is modular in that it is easily removed from the rest of the robot. All connections on the board that run to other parts of the robot pass through quick-disconnect connectors and a single connector connects all sensors that do not use a simple-to-remove connector (e.g. Ethernet, USB, and DB9). The board itself is an aluminum frame with an ABS plastic top, bottom and shelf. All the power distribution blocks are located on the bottom of the board leaving the top for the laptop and MCU's. The layer between the top and bottom is for routing wire runs. This arrangement provides a cleaner area onto which the electronics are mounted.

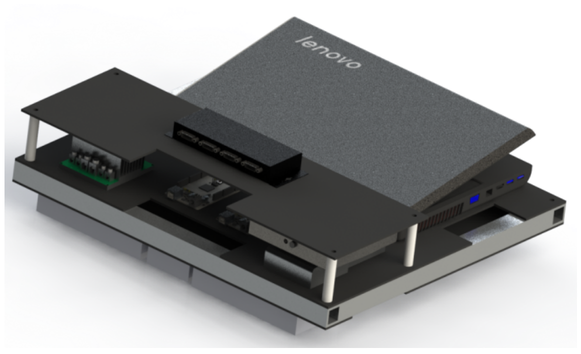


Figure 7. Control board structure.

4. Software (hierarchy and structure)

The software for Ground Squirrel is written in C++ and compiled using GCC. The code contains mostly custom classes along with libraries and some snippets of example code found online. The entire software hierarchy is based around the principles of object-oriented programming. The various components of the software are each developed independently and an API is created for each piece so that anyone on the team is able to use any part of the code without taking the time to go through the source code. An autonomous coordinator program regulates the various components and tells each part how to communicate with the other parts. This coordinator is how all the nodes work together to move the robot.

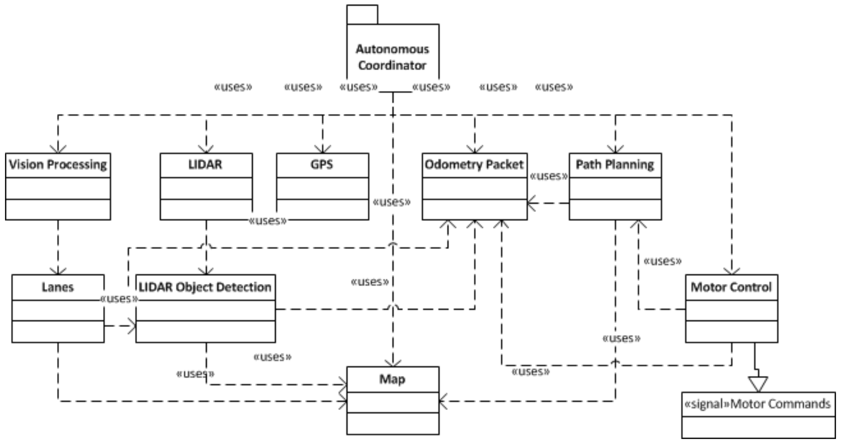


Figure 8. Software class structure and hierarchy.

4.1 Communications and Interconnections

For software configurations, a configuration file is provided with a class that contains all the key-value pairs and a configuration file parser. This allows the software to use a standardized, but easily changeable means to understand data and messages from different parts of the software.

5. Software (data processing, mapping and interconnections)

Every device connected to the laptop has a class associated with it. All of these classes share a common task in collecting and formatting the data and giving the user a way to access the data. This allows for the user to easily access the data from any of the sensors without needing to know how the data is formatted or transmitted. They only need to know what data is present. Metadata is also stored along with the data. This includes the age of the data, the sensor UUID, constants associated with the data or sensor, data resolution and accuracy. It also prevents the user from getting direct access to the data meaning that they cannot corrupt the original data.

5.1 Arduinos

Most of the data is collected on an Arduino and is then sent to the laptop in a custom packet format over the Arduino’s virtual USB serial port. The laptop then has a class that specifically reads in data from the Arduino serial port and stores the incoming data. The data, in addition to being formatted, is also conditioned before being sent to the laptop. Filters are applied and noise is removed from the sensor data, thus freeing up computational time and power on the laptop and making for easier use of the data.

5.2 Mapping

The mapping strategy utilized is based on SLAM (Simultaneous Localization and Mapping). A class contains a dynamically allocated map that is capable of growing and changing resolution. The user is able to then add various objects in the map stored as integers. This allows the user to define many different types of objects such as obstacles, lines, planned paths and previous paths in the map. The map class itself does not know how to place objects but is simply a container.

Objects are placed in the map depending on how they are detected. The LIDAR contains a higher-level class that takes the raw data and detects object positions relative to the robot's current location and places the objects in the map. Objects detected by the LIDAR within a certain width limit are assumed to be squares. Objects that are wider than the threshold are assumed to be one-meter deep rectangles.

Lines are placed on the map through image processing and camera calibration. Lane detection is described in more detail in section 6.

The map is also used for localization. The origin is set to be the starting GPS coordinates of the robot and from there everything else placed on the map is relative to this starting GPS coordinate. The location of the robot and its previous path is then mapped by correlating the encoder data with the accelerometer data and then checked against the GPS data. This provides for higher accuracy in placing the positions of objects and lines and in navigating to waypoints.

The map class allows for saving and visualizing the data. It can be saved to a custom file format (.gfm, short for GOFIRST Map) that contains the map data and associated metadata or it can be saved to a .csv file for easy import into Excel and other programs where it can be visualized. The map class also allows for loading from the .gfm file so as to load previous or modified maps into the program to debug and improve performance.

6. Software (vision and lane detection)

While the vision system is still under development as of this writing, the high-level approach has been developed and its effectiveness will be more rigorously tested in the final development phase.

The vision system is based on the OpenCV (Open-source Computer Vision) library. This has several advantages, including the availability of modular, efficient, well-tested implementations of the standard primitives and algorithms used in vision processing (such as edge detection and line transform algorithms) and the possibility of using CUDA-accelerated versions of these algorithms if performance becomes an issue. This library supports the building-block philosophy we intend to use in developing the vision system, which involves interfacing a small number of existing modular components to achieve the desired behavior. This philosophy supports the reuse of already existing software components and avoids creating unnecessary extra work and complexity in designing an already complex software system.

6.1 Lane Detection

Lane detection will be accomplished primarily by computing the (probabilistic) Hough line transform on a suitably preprocessed version of the input image. The preprocessing at this point consists of a Gaussian blur followed by an edge transform, although that has been found to lead to a large number of spurious lines in the output. This problem could be mitigated by tuning the edge-detection algorithm to be more conservative. Other mitigation strategies will be explored, such as adding random noise to the input image and keeping only those lines that remain constant over a large number of frames.

6.2 Object and Flag Detection

Object detection will be employed to complement the LIDAR's sensing capabilities. The object detection algorithm will primarily employ edge detection will extract features that will then be correlated with LIDAR distance data to reduce the error rate the overall system.

The LIDAR will likely not, however, deliver useful data as to the location of marking flags on the course. In this respect, the vision system must also stand on its own. Since the flags have known colors, intelligent thresholding can be applied by defining an acceptance range of colors that could be interpreted as either a blue or red flag. Performing contour detection on the thresholded image then yields the flag outlines, plus some possible spurious contours that can be eliminated by selecting only the maximum-area contour.

6.3 Object Localization

The location of detected objects relative to the robot will be determined, in the absence of three-dimensional depth data, by reference to their location on the ground plane. The perspective transformation between image coordinates and ground-plane coordinates will be determined experimentally once the camera has been mounted in a fixed position on the robot. This calibration will be done by placing a number of markers on the ground in front of the robot at known positions, then determining their positions in the camera image. From this data, we will be able to compute a transformation in the format that OpenCV understands.

This transformation will be applied specifically to detected lines in order to determine how far the robot is from a line on either side. Since the lines are on the ground, this application is relatively straightforward; for obstacles, it is necessary to take the lowermost point of the detected object (in image coordinates), which is assumed to lie on the ground. This approach also allows estimation of the object's real width, since the object's apparent size reduction can be computed from its ground-plane coordinates.

Determining the ground-plane location of the flags would be considerably more difficult, as the flag proper does not contact the ground. However, the only relevant information in this case is whether the robot is headed to the right or left of the flag, which can simply be taken from the horizontal image coordinate of the flag object, perhaps assisted by a more heuristic measure such as the flag's apparent size.

6.4 Performance Considerations

As the vision system will be running on the same processor as other computationally-demanding tasks such as the path-planning system, the limits of the available processing power and memory are a significant concern. There are several possible strategies for dealing with high computational demands of this vision system on the existing hardware. The simplest strategy is to decrease the resolution of the images before processing. The images obtained from the camera will be down-sampled to an optimal resolution (to be determined experimentally) that best balances computational requirements and ability to resolve the required features (the limiting factor most likely being the lines on the field).

A complementary approach is the utilization of the parallel-processing capabilities of the existing hardware, which includes an NVIDIA GeForce GTX 660M graphics-processing unit with CUDA 3.0 compute capability. As OpenCV contains a number of CUDA-accelerated operations available as modular building blocks, it should be fairly straightforward to incorporate parallel processing into our vision system.

7. Software (path planning, navigation and control)

For the actual driving of the robot, the software breaks down into two modules, the path planning and the path execution.

7.1 Path Planning

At the time of writing, the method for generating a navigable path is an A* search algorithm with some adaptation to account for waypoints off the existing map. As a form of best first search algorithm, A* will reduce search time by prioritizing search advancement in the direction of the destination. With minor modification to the A* algorithm, the path finding will be enhanced using a bug algorithm to handle traps and incomplete map situations. Since the mapping does not create potential fields, the bug element will also be used to adjust the path when passing close to physical obstacles.

In its current form, the search algorithm does a path to point search, rather than a path to direct route search. For this reason, the end point for the search is projected onto the current map if the endpoint is lying outside the currently recorded map.

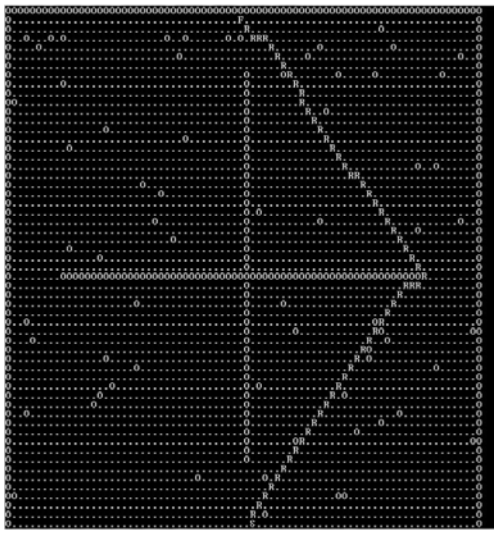


Figure 9. Demonstration of A* path planning on an example map with 1/4m resolution. The 'R' are the path and 'O' obstacles

7.2 Path Execution and Motor Control

Using the generated path, a desired heading and speed are calculated. These values are then compared to the current estimated state, and heading and speed errors are calculated. The errors are then sent via serial to an Arduino executing a MIMO PID controller. The corrected state is sent back to the laptop where the new motor commands are calculated and then executed.

State space methods were used along with the previously mentioned nonlinear model to generate a theoretical controller. This controller was then implemented and tuned to provide fast and accurate tracking.

8. Performance Analysis

8.1 Speed

Ground Squirrel has a top speed of 8mph on level ground, and it takes 10.2 seconds to reach this speed from standstill.

8.2 Ramp Climbing Ability

Due to Ground Squirrel's high torque motors and low weight, the wheels will slip before they stall on inclines. Assuming a coefficient of friction of 0.8, we calculate the max incline that can be traversed from a standstill to be 35°.

8.3 Reaction Times

The limiting factor in reacting to new environments is the path planning. From this, we predict that at map resolutions of ½ meter and ¼ meter we will see performances of

Predicted Performance: 0.35 Seconds (1/2 meter)

Predicted Performance: 3.33 Seconds (1/4 meter)

Times predicted are based on running the path finding algorithm on a simple generated test map. No optimization or parallelization has been implemented at this time.

8.4 Battery Life

With the deep cycle marine batteries, the calculated battery life of the robot is a nominal two hours at full power. After testing, this performance spec appears to match the calculated time. The time to reach full charge is about twelve hours.

8.5 Range of Obstacle Detection

Using the LIDAR for object detection gives a max distance of detection of 19.5 meters within our set resolution. Line detection is based solely on camera image data and is thus limited by the field of view of the camera and our region of interest which in turn is dependent on the mount height and angle. At the time of this writing the FOV and ROI intersect was unknown.

8.6 Accuracy of Arrival at Navigation Waypoints

At this time, the accuracy of GPS navigation is unknown. Early indications though from the GPS specifications and our capabilities in mapping would estimate the accuracy to be within one-half of a meter. With our current methodology for using the GPS, this estimate is solely based on the accuracy of the GPS data at the start of the run.

9. Budget and Time Breakdown

GOFIRST was limited this year to a budget of $9000. An approximate, high level breakdown of how this money was spent is outlined in Table 1 below. Over the course of two semesters, our team has placed in over 1700 hours of work and countless hours more in training. This estimate is based off of five hours of work a day, three days a week over 18 weeks and accounts for absences and breaks.

Table 1. Cost of Ground Squirrel

Item	MSRP	Cost to Team
Chassis Material	$470	$470
NPC Motors (x2)	$200	$200
Other Drive Components	$455	$455
General Hardware	$100	$100
Lenovo Y580 laptop w/ upgrades	$1200	$1200
Sabertooth Motor Controller	$190	$190
Laptop Interfaces	$490	$490
E-stop Button (x2)	$150	$150
Power Distribution & Batteries	$650	$650
Axis Camera	$200	$0[1]
Sick LMS100 LIDAR	$3900	$0[2]
Digital Yacht Compass	$280	$280
Ag Leader GPS	$995	$845[3]
Other Sensors	$330	$220[4]
General Electrical and Wiring	$150	$150
Totals	**$9900**	**$5740**

Notes:
1. Previously owned by team
2. Donated by SICK
3. University discount applied
4. Some parts here donated by Banner Engineering

10. Conclusion

This being our first year competing in IGVC, we believe we have developed an effective solution for the problem posed by the competition. We are excited to put Ground Squirrel through its paces and learn what worked and what did not in addition to gaining the experiences needed to be more successful next year.

11. Acknowledgements

At this time we would like to acknowledge Mr. Bruno Bittner at SICK for helping us acquire a LIDAR and Mr. David Anderson at Banner Engineering for his work in getting us the touch button in addition to several other sensors not used on this year's robot. We'd like to thank the University of Minnesota for providing us with the funding we need, Dr. Donath and Brian Davis for allowing us to borrow the U of M's 2007 IGVC robot (from a previous group with whom we had no direct contact) which gave us some inspiration, and Dr. William Durfee for supporting our group over the past few years.