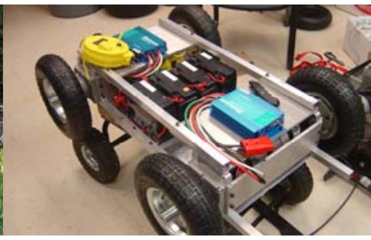


Penn State Lion One

IGVC 2009

Design Report

This design report documents the design strategies used during the development of Penn State's entry to this year's Intelligent Ground Vehicles Competition. Specific emphasis has been laid on the innovations conceived by the team during the iterative design process.

Design Team
Abhinav Choudhary
Amit Patel
Drew Logan
Joan Singla
Kshitij Jerath
Madhu Soodhanan
Pol Morral
Steve Chaves
Faculty Advisor: Dr. Sean Brennan

Table of Contents

Faculty Advisor Statement

I, Sean N. Brennan, certify that the design and development of Penn State Lion One has been significant and that each student performing this work is a registered student. This work, as part of a graduate class project and as an extracurricular project, represents a participation level equivalent to what would be awarded credit as a senior design project.

Sean N. Brennan, Department of Mechanical Engineering, Pennsylvania State University

1. Introduction

The design team from The Pennsylvania State University is proud to field the Penn State Lion One for the 2009 IGVC. The design effort was a continuation of a graduate-level Mechatronics course offered by the Department of Mechanical and Nuclear Engineering. The design team consists entirely of graduate students from this department.

The Penn State Lion One is a significantly improved robot as compared to Penn State's entry for the 2008 IGVC, and has had several innovative upgrades including a complete overhaul of its software platform to a MATLAB/Simulink-based architecture. Several other hardware and algorithm innovations have been applied to the robot, improving its performance significantly over the previous year's effort.

2. Design process

This is the second year that Penn State is participating in the IGVC, so we have focused this year on developing processes for continuous improvement. Further, from lessons learned last year, we have established three core design concepts as follows:

Use iteration to learn from mistakes quickly

To formalize an iterative design methodology, we used the V design approach shown in Figure 1. We analyzed the requirements and restrictions for the project to create a set of desired design objectives. Given these design objectives we decomposed them into subsystems and further to algorithms. The remainder of the report provides details of each decomposition and integration process.

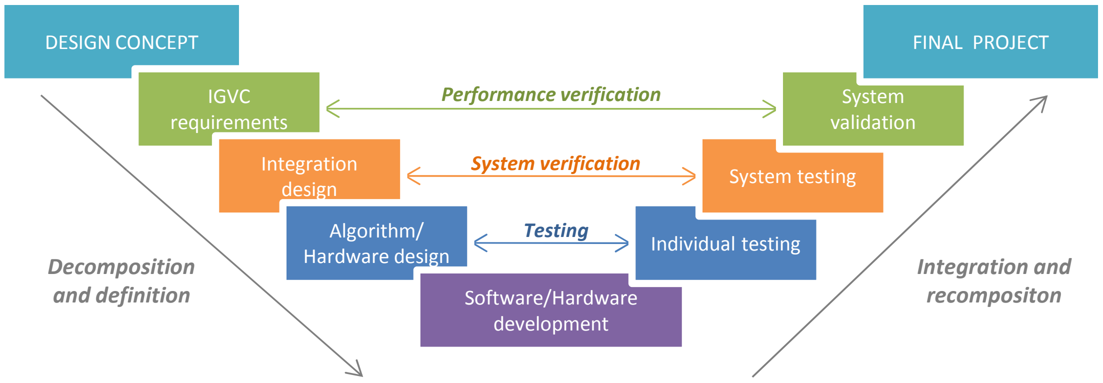


Figure 1: The "V" Systems Engineering Model

Prepare for competition by using competition

For each algorithm, we divided the team in at least two groups that compete to develop the best solution. This way, performance becomes a first priority, multiple team members learn and hence can scrutinize each other's code, and team members become proud of their code. After each competition, we compared all solutions so that the best methods are shared.

Fast algorithms are better than fast execution

IGVC robot software will always be a prototype, and last year's entry taught us that debugging bad algorithms to create outstanding code has more performance payoff than using complied language or fast processors on otherwise bad code. MATLAB/Simulink is a high level language that supports this design philosophy, and it is OS, computer and version agnostic. For this reason it is used throughout the engineering curriculum to the point that PSU no longer teaches C or C++ to most engineering students.

2.1 Team Architecture

Because of our competition design principles, each team member is at least an expert on two groups shown in the Figure 2. Logs have been kept of all design activity, and the team has devoted 1020 hours to date towards the development of Penn State Lion One.

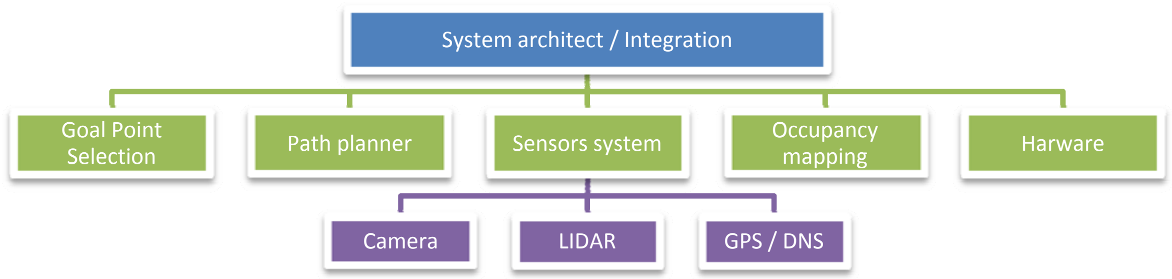


Figure 2: Team Architecture

3. Innovations

The design team has introduced numerous innovations into the robot for the 2009 IGVC. The foremost among them is the use of a MATLAB- and Simulink-based software architecture. This scheme allows for a transparent code development strategy. Further, the use of real-time control software (QuaRC) allows real-time gain tuning for our control algorithms. Several innovations have been introduced on the hardware side, such as a new modular design of the robot, an internal battery recharge system, the ability of the robot to pivot around any arbitrary point, and an on-board diagnostic scheme for detecting ground faults.

The software component of the robot also includes several innovations such as a fully adaptive image processing algorithm that adapts to lighting conditions, a boundary-based Veroni diagram goal point generation scheme for path planning, and on-the-fly map archiving and retrieving for automated map building. Further, data exchange operations now are performed with TCP/IP instead of serial. These innovations are emphasized in the report wherever they occur.

4. Vehicle Design

Several hardware lessons were learned from last year's competition: our robot was too slow, the drive chain and sprockets were unreliable, it was too difficult to recharge quickly and access equipment inside, it suffered from electrical faults, and was even unable to climb the ramp due to insufficient friction.

Starting from last year's design, we have followed an iterative design process to reach our present design, dealing with all previously known hardware problems and new ones found on the way.

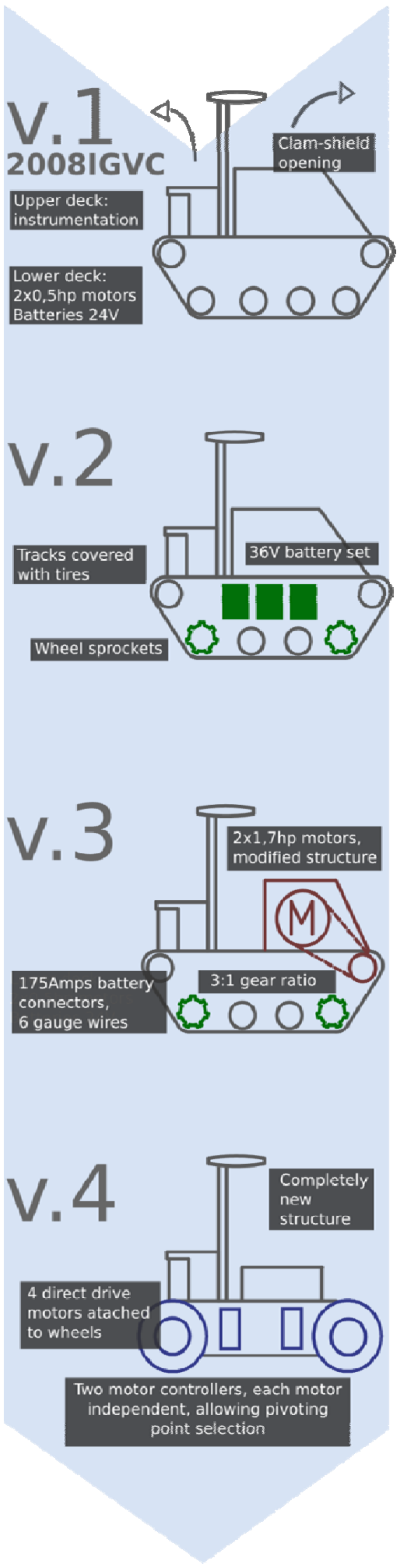
4.1 Iterative Design Steps

Last year's robot was "version 1", the first of many iterations. It is a track-driven, two-decked platform. The lower platform contains two 0.5 hp DC motors and four batteries, providing two separate 24V sources, one used by the motion system, the other by the instrumentation equipment. The upper deck holds all the electronics. The access to the lower deck is through a clam-shelled opening system, which proved to be impractical. We encountered problems with derailed tracks from the very first run and, by the end of the competition, we realized that we had a very slow machine - an issue we needed to solve for the following editions.

For version 2 we focused on solving the problems of insufficient power and traction. We increased the motor voltage to 36V, we added guide sprockets wheels to the track system, and added a rubber overlay to the treads. With these changes, the robot was still barely fast enough. Even worse, the guide sprockets significantly increased wear on the tracks.

Version 3 was used to test a new set of drive motors. These solved all the previous power issues, but further increased the rate of damage to our track system. Recognizing that design of a high-speed tank-drive UGV is a tradeoff between reliability and power consumption, and that reliability is always difficult to obtain in a prototype robot, we decided to migrate to a simpler and more effective wheeled robot.

Version 4, our final version, is a four motor direct-drive wheeled vehicle. Although this approach has more limited traction and is more expensive, it has proven to be the most efficient and robust. Further, we have designed the robot to have independent control of the front and rear axles. This allows guidance algorithms to choose the pivot point on a zero radius turn, a capability very convenient for a competition of this kind.


v.1
2008IGVC
Clam-shield opening
Upper deck: instrumentation
Lower deck: 2x0,5hp motors Batteries 24V
v.2
Tracks covered with tires
36V battery set
Wheel sprockets
v.3
2x1,7hp motors, modified structure
175Amps battery connectors, 6 gauge wires
3:1 gear ratio
M
v.4
Completely new structure
4 direct drive motors atached to wheels
Two motor controllers, each motor independent, allowing pivoting point selection

The **robot platform is modular** and allows quick change of equipment and batteries, enabling quick deployment

An **internal recharge system** helps avoid the need to open and unplug the robot and dealing with plugs every time the robot needs to recharge its batteries.

5. Electrical Systems and Electronics

5.1 Power and communications

Several measures have been taken to ensure efficient power distribution while minimizing interference. The drive train batteries are contained in the lower deck and are physically and electrically separated from the electronics that are located on the upper deck. Two 12V 18Ah lead-acid batteries in series are provided for the electronics, and two more are present for the motors. Appropriate fuses have been provided for the batteries.

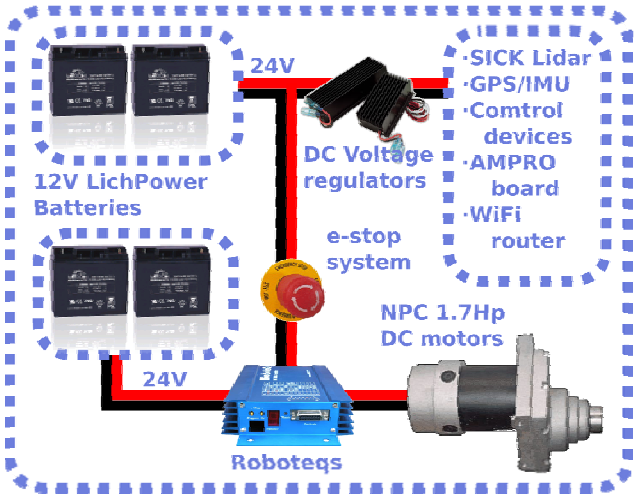


Figure 3: Power supply diagram

Figure 4 represents the data connectivity diagram, showing all major robot components. As an innovation, Arduino PICs have been installed as interfaces to all low-level hardware. Each subsystem is thus linked to all others via Ethernet, providing high data rates as well is isolating electrical ground between all devices. The Arduino used for motor control also can read signals from a RC receiver when the RC mode is selected, allowing us to drive the robot safely when autonomous mode is not desired.

5.2 Fault Detection and E-stop

In order to improve troubleshooting capabilities and enhance reliability, an on-board fault detection system has been implemented to track any electrical or communication failures. For the five main subsystems of the robot, an Arduino PIC monitors continuity between chassis and both high voltage and ground. Faults are indicated directly from the PIC using a LED array and also sent to a supervisory computer via Ethernet. Further, both on-board and wireless emergency stops are also built into the robot. The on-board emergency stop is located in an elevated location at the rear of the robot, per the competition rules; the wireless E-stop has a range of 300 ft, a distance we have verified several times.

An **on-board fault detection system** allows us to monitor robot health on the supervisor terminal.

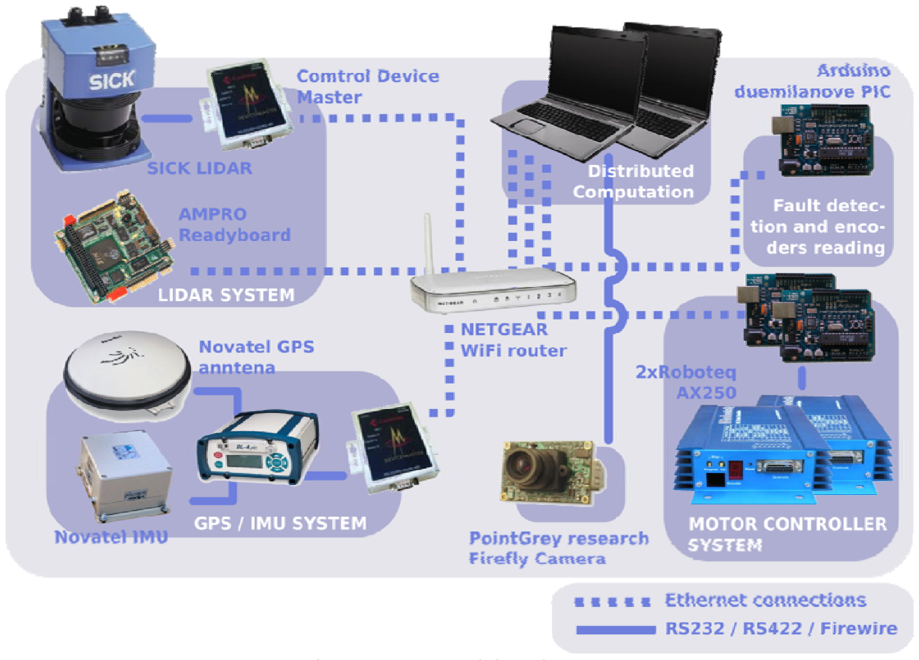


Figure 4: Connectivity Diagram

5.3 Sensors

The sensor systems used on the robot are described briefly in Table 1.

Table 1: Description of sensors

Sensor	Description	
Camera - Point Grey Research Firefly MV	▪ Resolution 720 x 480 ▪ 60 frames per second.	
SICK LMS	▪ Range – 30 meters ▪ Scan rate – 37.5 Hz ▪ 0.5 degree resolution	
NovAtel DL4 plus OEM4 dual frequency GPS receiver	▪ Dual frequency receiver ▪ Position accuracy of 2 centimeters	
Honeywell HG 1700 Military tactical Grade IMU	▪ Ring-laser gyro with laser-calibrated MEMS accelerometer ▪ Drift bias – 10 deg/hr ▪ Acceleration bias – 3 milli-g ▪ Velocity and sampling rate – 600 Hz	
US Digital s2 2048 Optical Encoder	▪ 2048 counts per revolution ▪ 5V supply (from Arduino board)	

6. Software Design

6.1 Software platform

One of our guiding design principles is that code transparency is critical. To enforce this, a unique and innovative software architecture based completely on MATLAB and Simulink was developed. Functions written in the native MATLAB language handle almost all of the necessary computations for the robot's path planning, map generation, and occupancy structure. Sensor data streaming and robot motion control are accomplished through Simulink block diagrams, with the help of a program called QuaRC. QuaRC is a real-time control software toolbox developed by Quanser Inc that directly compiles Simulink diagrams to code that can be executed and monitored in real-time. This type of architecture makes the hardware-software interfaces very streamlined, robust, and reliable. Further the MATLAB-based software architecture allows for powerful tools like MATLAB's Profiler to be used, where the robot's entire code can be analyzed instantly for computation time and function calls in mere seconds.

The software platform is **completely based on MATLAB and Simulink**, with real time control performed by QuaRC. This architecture enhances the code's transparency and reusability.

Further, the real-time control capabilities provided by QuaRC allow us to tune the control gains in real-time and monitor the performance of the robot as the gains are changed. This allows a quick design-test-verify iteration, speeding up the robot deployment schedule. Figure 5 shows one of our test runs where the control gains for orienting a robot according to a specific yaw command were tuned in real-time.

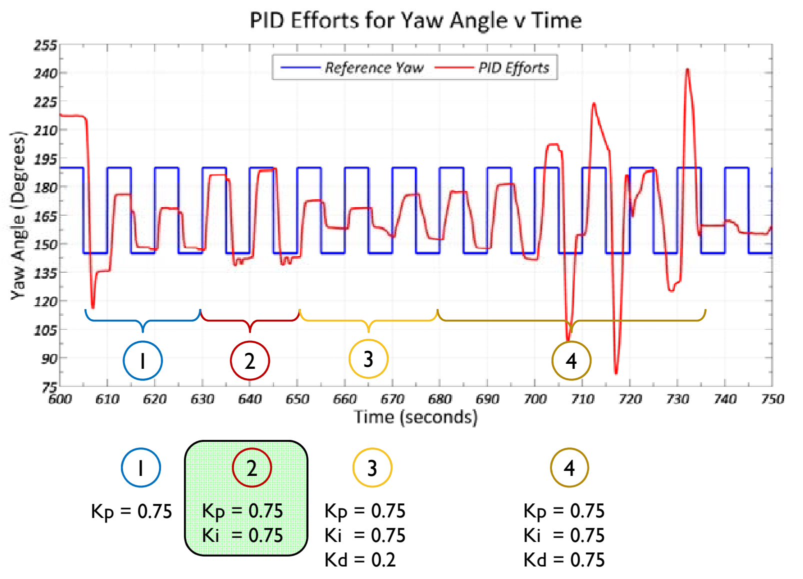


Figure 5: Real-time gain tuning of robot during a test run for orientation control

QuaRC allows a **real-time gain tuning** scheme that allows for extremely quick design-test-verify iterations allowing a speedy deployment of the robot.

6.2 Software architecture

The software architecture is indicated in Figure 6. The remaining subsections of software design in the report follow the same general order as shown in the figure. Path planning is discussed in 6.4 and 6.5, followed by a discussion about cameras and image processing in section 6.6. Occupancy maps are discussed in subsection 6.7 about sensor fusion.

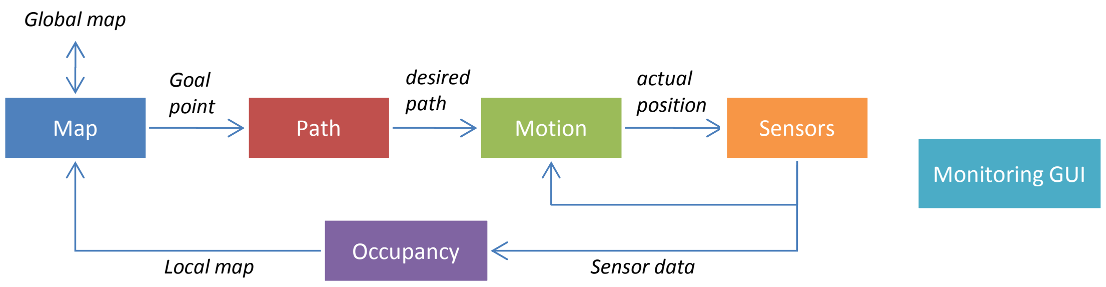
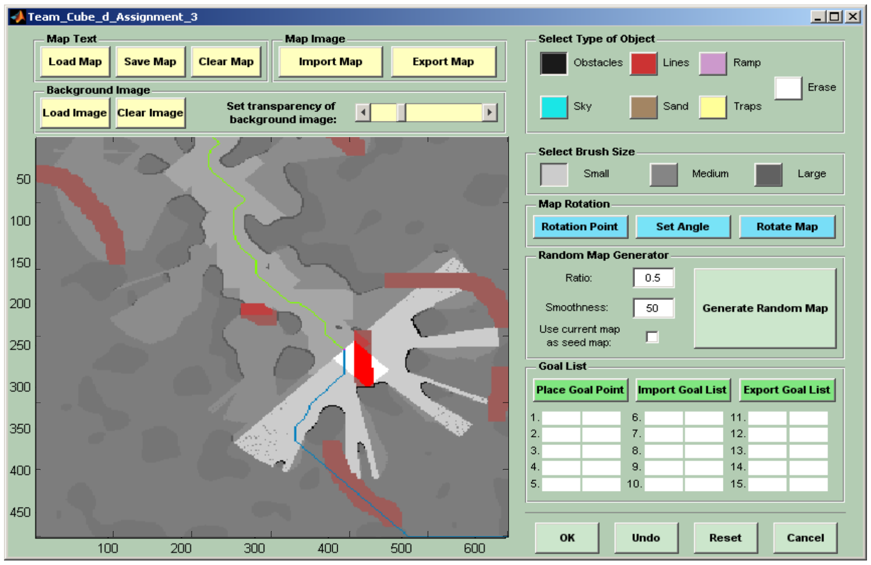


Figure 6: Software Architecture

6.3 Simulation tool

One of the major improvements this year is the development of a powerful GUI that allows us create simulated environments (objects and lines) to first virtually test the algorithms in different conditions before testing in a physical robot environment.

As shown in Figure 7, the robot's sensors, like the camera or range finder, are also simulated and follow closely the real behavior of these sensors. This simplifies debugging because we can control levels of sensor noise and faults. Further, with this tool, multiple developers can be testing the robot at the same time in different contexts.



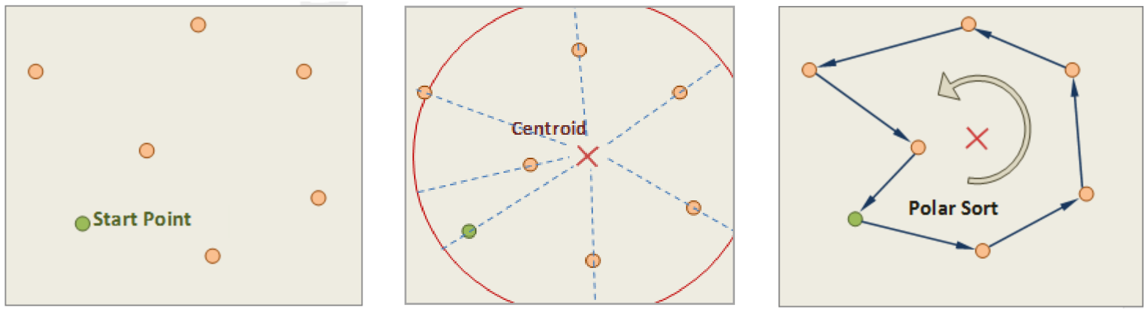
Ratio: 0.5
Smoothness: 50
Use current map as seed map: ☐

Figure 7: GUI with the simulation tool

6.4 Goal Point Generation

The path planning strategy for the Penn State Lion One has been divided into two segments: the first is a Goal Point Generator, which decides the order in which the given waypoints should be traversed; the second is a Path Planner, which decides which path to take to the next waypoint. Two Goal Point generators have been developed, one for the Navigation Challenge, and another for the Autonomous Challenge.

The Goal Point Generator for the Navigation Challenge decides the order in which the given GPS waypoints should be visited. This problem is essentially the celebrated Traveling Salesman Problem. Three techniques that were analyzed, coded and tested were a heuristic technique (polar sorting around centroid), a modified Monte-Carlo method, and a Genetic Algorithm. The Heuristic Polar Sorting technique was found to be faster than any other algorithm by an order of magnitude, while giving near-optimal routes for small number of waypoints. The technique is explained pictorially in Figure 8.



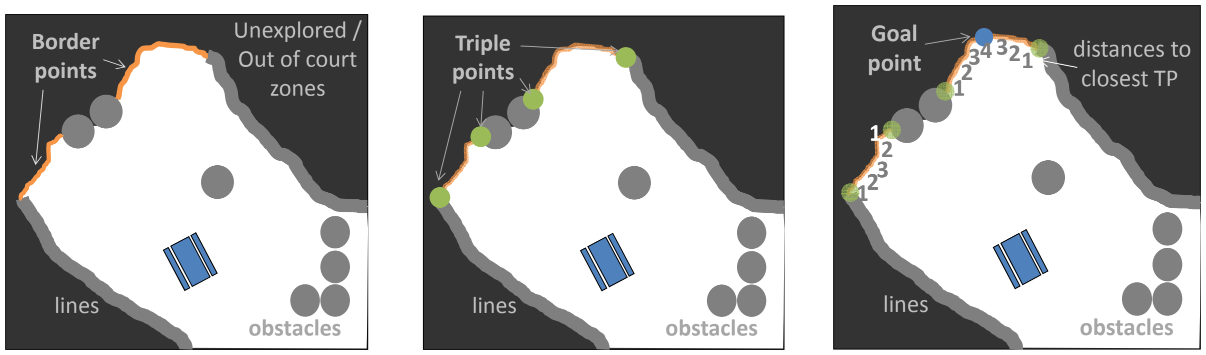
(a) Randomly distributed waypoints (b) Centroid Calculation (c) Polar Sorting

Figure 8: Heuristic Polar Sorting Algorithm yields optimal route for small number of waypoints

The Heuristic Polar Sorting algorithm is an **order of magnitude faster** than other global path planners. Further, it only runs once and has a deterministic computation time.

The Autonomous Challenge requires that the robot explore an unknown environment. Thus, there is difficulty in deciding which location to move to next. The classical approach is wall following, but our simulations of this algorithm showed it will fail in common circumstances.

To decide the next location, or ‘goal point’, we then investigated seeking points that lie at the intersections of unknown, open and obstacle spaces, areas which we named “triple points”. There are only a few triple points in any occupancy map, thus monitoring these points is quite fast. Simulations showed that wall-following is a sub-class of algorithms that seek such triple-points.



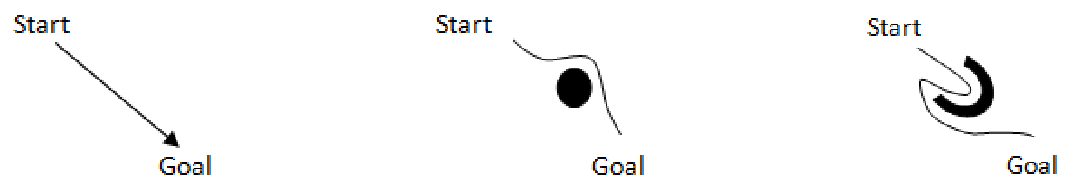
(a) Select border points inside explorable zone (b) Select Triple Points (TP) (c) Calculate distance of border points to the closest TP

Figure 9: Goal Point Generator algorithm for the Autonomous Challenge

For the 2009 IGVC, another triple-point algorithm was designed that was motivated by 2D Veroni diagrams, and which we refer to as Veroni 1D. Essentially, this algorithm seeks to move to the middle of the largest open, unexplored area between triple points on the 1D boundary. The computation process is explained in Figure 9, and is not only robust, but exceptionally fast to compute.

6.5 Path Planning Algorithms

Any robot traversing the course should be able to plan a path under conditions shown in Figure 10. A summary of the performance of different commonly used path-planning algorithms is provided in Table 2. In the table, under processing speed, N indicates the distance between the goal and start point, whereas "t" indicates calculations that must be performed at each time interval. There is clearly a tradeoff between the speed of the algorithm, and the complexity of the planned path.



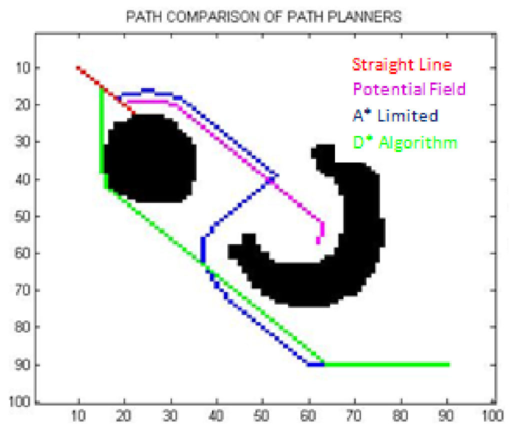
Condition 1 – Straight line   Condition 2 – Move away from obstacle   Condition 3 – Cul dé sac

Figure 10: Path Planning Conditions

Table 2: Capabilities of Path Planners

Algorithm	Condition			Processing Speed
	1	2	3	
Straight Line	✓	fails	fails	O(N)
Potential Field	✓	✓	fails	$O(N^2) + O(N).t$
A* Limited	✓	✓	✓	$O(N^2)t$
D*	✓	✓	✓	$O(N^2)t$

Our path planner switches between all four algorithms to exploit their benefits and still minimize processing time.



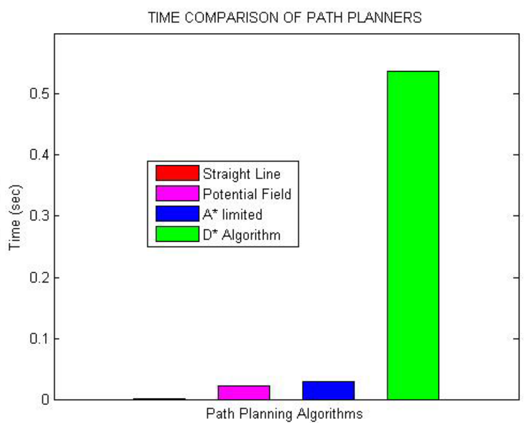
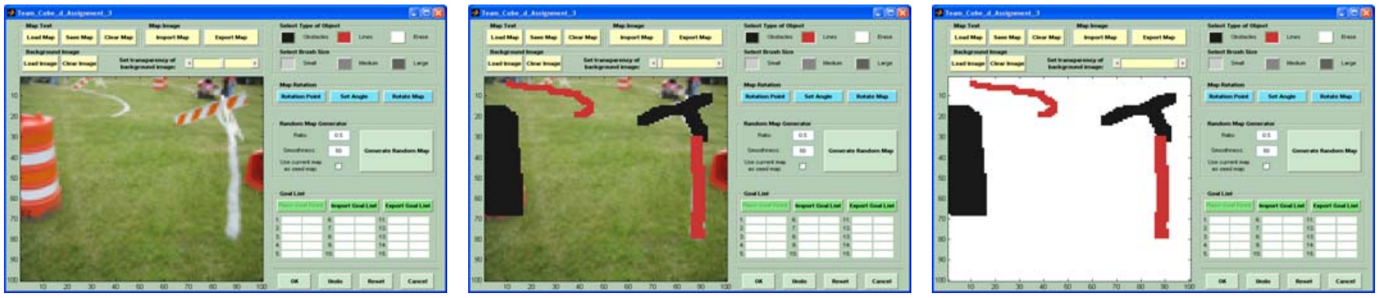


Figure 11: Comparison of Path Planning Algorithms

To achieve this type of dynamic scheduling, the simplest path planners are attempted first, and each algorithm self-monitors progress. In the case that a planner is unsuccessful, a supervisory algorithm switches to a more complex path planning approach and “milestone” conditions are calculated that allow switchover later back to a simpler algorithm.

6.6 Vision and Image Processing

One of the major problems faced during last year´s competition was inadaptability to adapt to changing lighting conditions such extreme sun versus overcast conditions. Last year’s algorithm was a simple intensity-based static thresholding algorithm and did not perform well on the course. Consequently, the main focus of the image processing effort this year was on introducing adaptability into the algorithm. To test the various new algorithms, we created a GUI (Figure 12) for manually tagging images to represent the ground truth. Last year’s algorithm was tested on this set of tagged images, and was found to fail on many occasions.



(a) Image loaded into GUI (b) Lines and obstacles tagged in GUI (c) Tagged Image

Figure 12: MATLAB GUI used to generate tagged images for testing purposes

For IGVC 2009, three different algorithms with varying degrees of adaptability were considered:

a) A simple **intensity-based algorithm on the RGB color space**. Similar to last year’s algorithm, except that that thresholds are intensity-scheduled, i.e. change with image intensity.

b) An **adaptive threshold algorithm based on the R^{2}GI color space**. By trial and error on images saved at IGVC 2008, it was found that the R^{2}GI color space yields a good color space basis. A new threshold is calculated for each image using averaging, making this algorithm adaptive.

c) A **fully adaptive algorithm based on use of Principal Component Analysis (PCA) and K-means clustering**. The PCA delivers the optimal color space (the principal component) that best partitions grass and non-grass pixels. We then use K-means to quickly calculate thresholds.

The three algorithms were compared against each other for speed and accuracy. It was found that PCA with K-means clustering performs the best in terms of accuracy, and does so at an acceptable speed.

Principal Component Analysis

As lighting and weather conditions change, the color space that helps best distinguish between white lines and grass also changes. Such a color space can be determined on-the-fly, by using only the principal component of the image, i.e. the eigenvector pointing in the direction of maximum change in the image information. As can be seen from Figure 13, the image data appears scattered. However, by proper orientation, it can be seen that most of the data is tightly clustered along the principal component axis (pixels.

Figure 14). The advantages of performing PCA include reduced processing time (by reduction of data dimensionality), adaptability with changing lighting conditions, and the option of extension to higher dimensions (using kernel functions) for higher accuracy.

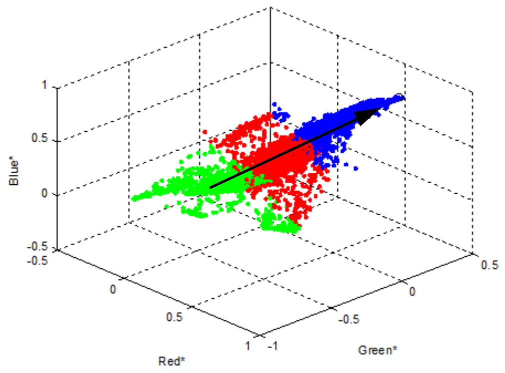


Figure 13: The image data as seen orthogonal to the principal eigenvector (PE). This direction provides the largest separation of line and grass pixels.

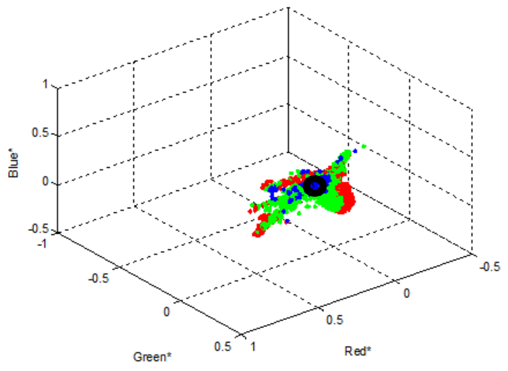


Figure 14: The image data as seen along the direction of the PE (circle). This view provides minimum separation of line and grass pixels.

K-means clustering

K-means clustering is performed on 1-D data to separate lines from grass. K-means minimizes the sum of distances across all points by putting them into appropriate clusters. Figure 15 shows the clustering for grass and lines.

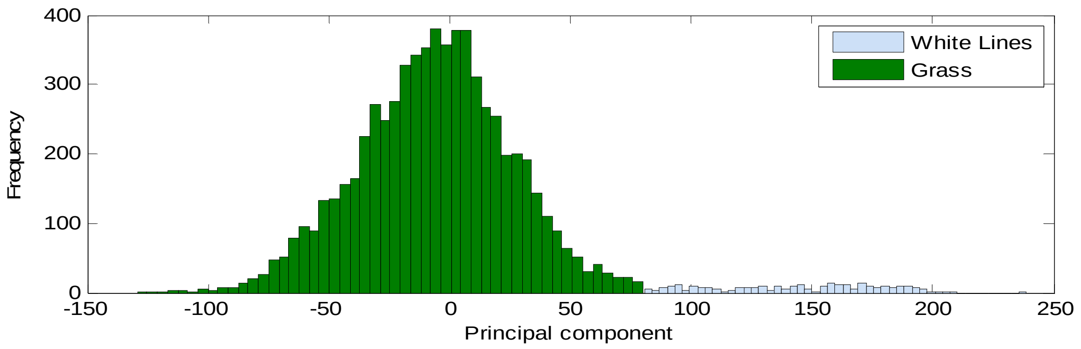


Figure 15: Clustering on the one-dimensional principal component

After the data points have been clustered in one dimension, the original image is reconstituted and a projective transformation is performed on the image to obtain a birds-eye view (Figure 16). The birds-eye view is then converted into an occupancy map, and is then fused with a LIDAR occupancy map to generate the world representation for the robot.

The image processing algorithm is **adaptive, and can modify the color space and thresholds on-the-fly**, based on the lighting conditions and weather.

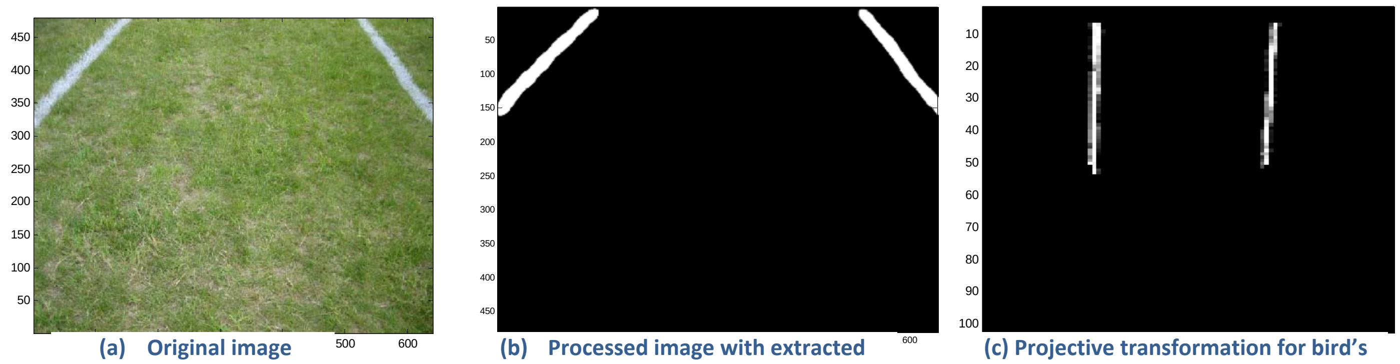
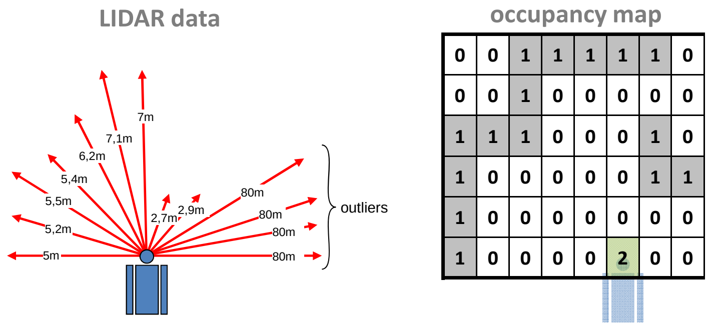

(a) Original image (b) Processed image with extracted lines (c) Projective transformation for bird's eye view for occupancy map

Figure 16: Image processing – From original image to bird's eye view

6.7 Occupancy Map Generation

Mapping the obstacles and lines

The data that comes from the LIDAR and camera sensors is processed to generate individual occupancy maps (obstacles or lines) of 1's (objects) and 0's (clear spaces).



0	0	1	1	1	1	1	0
0	0	1	0	0	0	0	0
1	1	1	0	0	0	1	0
1	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	0	0	2	0	0

Figure 17: Mapping LIDAR to occupancy

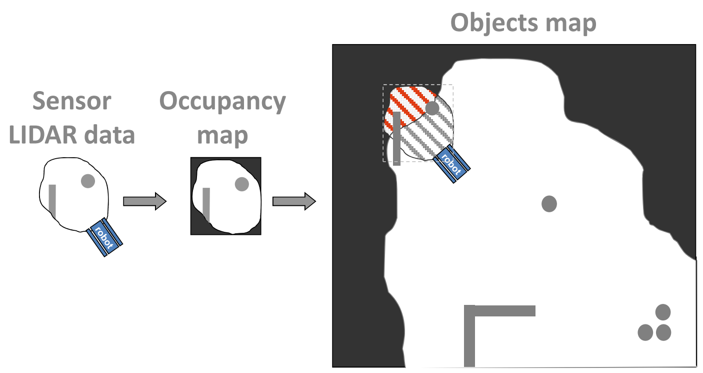


Figure 18: Converting occupancy to a global map

Using the GPS position of the robot, the occupancy map is merged with its corresponding map (obstacles or lines) using an IIR filter to remove noise. Each value is averaged according to previous readings and a user-defined threshold depending on the speed of the vehicle, level of confidence of the map, and sensor noise. The result is quite similar to a spatial Kalman filter.

Fuse occupancy data

Once the obstacles and lines maps are updated, the information is merged in what we called the "fused map". This map will be used by the path planner algorithm to define a route to the goal point. Special attention has been put on filtering white obstacles appearing in the camera to dose not count as obstacles in the fused map.

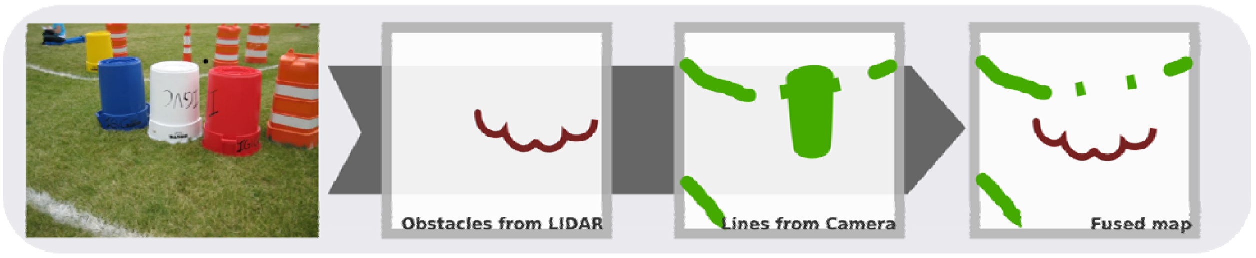


Figure 19: Data fusion process

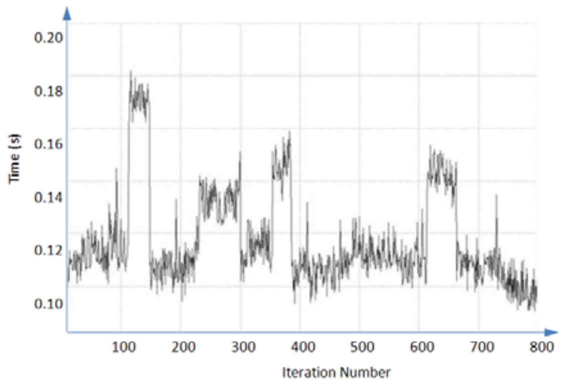
7. Systems Integration

7.1 Distributed Computing

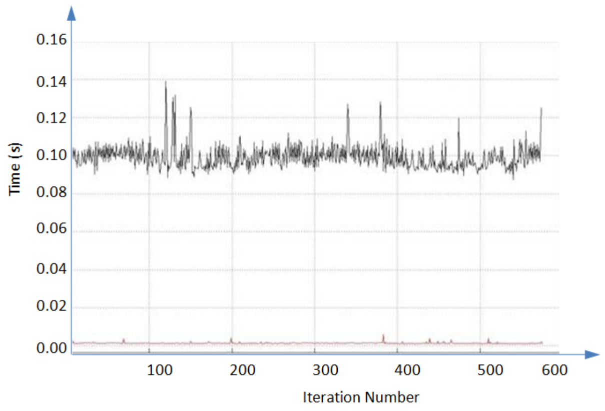
The innovative software architecture for the robot is built around a foundation of distributed computing and modularized systems. Using UDP commands, suitably modified from MATLAB's TCP/IP toolbox, all functions necessary for the robot are allocated to specific computers depending on their computational and inter-function communication loads. With this method, critical calculations and algorithms for the robot were optimized to run in parallel.

The concept behind distributed computing is quite simple. Each function for the robot is designated as a "foreground" function, with an associated "background" server that runs hidden beneath the main function. Parameters for each background server are specified during robot setup. These parameters govern the UDP communication to and from the server, the variables passed through the server, and their destinations. These background servers can be implemented on computers running MATLAB scripts or computers running Simulink diagrams with QuaRC real-time control. The foreground functions then pass variables to their respective background servers via variable flags; when a variable has been updated in the foreground, the background recognizes this update and passes the new variable to other functions distributed among different computers.

Optimization of the processing speed for various functions running on the robot can be done by allocating the function or diagram to any computer which has the capabilities of a background server (**Error! Reference source not found.** 20). For example, one computer can run an image processing algorithm while another computer plans the robot's path, with the background servers handling any communication between these two processors. Any number of computers can be declared with background servers, so this type of system integration is versatile, powerful, and scalable. As an added benefit, this method of distributed computing allows for a "supervisor" computer – a computer that logs the robot's performance real-time, monitoring critical information like position on a map, speed, power usage, etc.



(a) Simulation using only one computer – 10 to 20 Hz



(b) Simulation using two computers – all processes operate at 10 Hz reliably

Figure 20: With robust communication and distributed computing, the robot can perform the same functions in less time

8. Safety, Reliability, Durability

In addition to the fault detection system mentioned earlier, the construction of the IGVC robot incorporates proper engineering practices whenever possible to make the platform both safe and reliable. Per the competition requirements, the robot is equipped with a wired and wireless emergency stop (e-stop). In addition to these two e-stops, two Hela switches are in place to as a physical break the batteries powering the motors and the motor amplifier. Other than the wheels, there are no exposed moving parts within the robot, thus preventing injuries while servicing.

Each sensor system (laser scanner, GPS/IMU, encoder and vision) is isolated both physically and electrically. A physical mount is made for each sensor system which can be quickly unbolted from the frame as needed for rapid debugging, and lexan plates isolate each subsystem from through-chassis electrical faults. The only electric connection between each module and the rest of the robot is for power and a single Ethernet. Power for each system is regulated from 24V within the module. The sensors are also powered by a different source than the drivetrain. To avoid human error, there are different sized connectors are each end of the motors and motor amplifier, and between different power architectures.

Careful attention was paid during the design stage to make sure that all systems of the robot provided for safe and reliable functionality. One of the most important safety features is the ability to stop the robot immediately, either remotely or locally. The robot is equipped with a large emergency stop button placed prominently on the back. It is also equipped with a remote that has a line of sight range of 300 feet. These features provide enough flexibility to stop the robot in case of any emergency.

Reliability was a key factor in choosing hardware for the robot. Commercially available components were chosen to perform the required functions on the robot, where possible, to increase expected reliability and durability.

Employing such equipment, also provides recourse in the event of an equipment failure: most hardware components on the robot can be replaced easily through retail channels.

9. Performance

Ramp Climbing Capability: The ability of a robot platform to climb up a ramp is a function of the robot's available torque, center of gravity, and the friction between the robot's wheels and the contact surface. All three factors were measured. A model was developed to predict the robot's climbing capabilities based upon a quasi-static force balance analysis of a very slowly moving robot. The model predicts the robot can traverse a ramp with a 46.8^{O} while the actual incline the platform can traverse is 30^{O}. The robot's ability to traverse a steeper slope is limited by friction in both the model and during testing.

Maximum Speed: The maximum velocity of a robot is computed directly from the motor speed curve, gearbox ratio and wheel diameter. Using a National Power Chair (NPC) motors with a 20:1 gear ratio and 13 inch wheels, the model accurately predicts that the robot can achieve a maximum velocity of 8.55 mph. Testing confirmed this speed this speed to within 1 mph.

Cruising Distance: Cruising distance is defined as the distance a robot can travel on flat terrain without recharging its batteries. This distance (Δx) is a function of the energy available ($E_{Available}$), platform's velocity (V), and the power available ($P_{Available}$) from its batteries as seen in the first equation below.

$$\Delta x = \frac{E_{Available} V}{P_{Available}} \qquad E_{Available} = V' It = V' I \frac{C}{I^{n}} \qquad P_{Available} = \left(F - \sum R\right) V$$

The actual battery energy available, denoted as $E_{Available}$, in the middle equation describes the energy available in the battery when current is being drawn more quickly than the specified rate. Batteries are computed as a function of motor amplifier voltage (V'), the theoretical capacity of the battery (C), and the current draw rate (I). The Peukert Number, denoted as *n*, is an experimental constant derived for each battery type to describe the effects of the battery discharge rate. $P_{Available}$ is defined as the power required to overcome all of the resistive forces (ΣR) at a specified velocity. $P_{Available}$ is a value which is computed using both experimental and theoretical data. This model predicts a cruising distance of 6.6km and battery life of 1.8 hours. We tested the robot experimentally and found a maximum cruising distance of 5.3km and battery life of 1.6 hours, making the model accurate within a percentage error of 25%.

Goal-point accuracy: The accuracy of reaching a specified goal point is dependent on the accuracy of the DGPS system, the tracking accuracy of motion controller, and the accuracy of the path-planning algorithm, errors which are 0.5 meters, 0.1 meters, and 0.1 meters respectively. The measured accuracy of the robot is always measured within 1 meter, which agrees with the cumulative errors noted above.