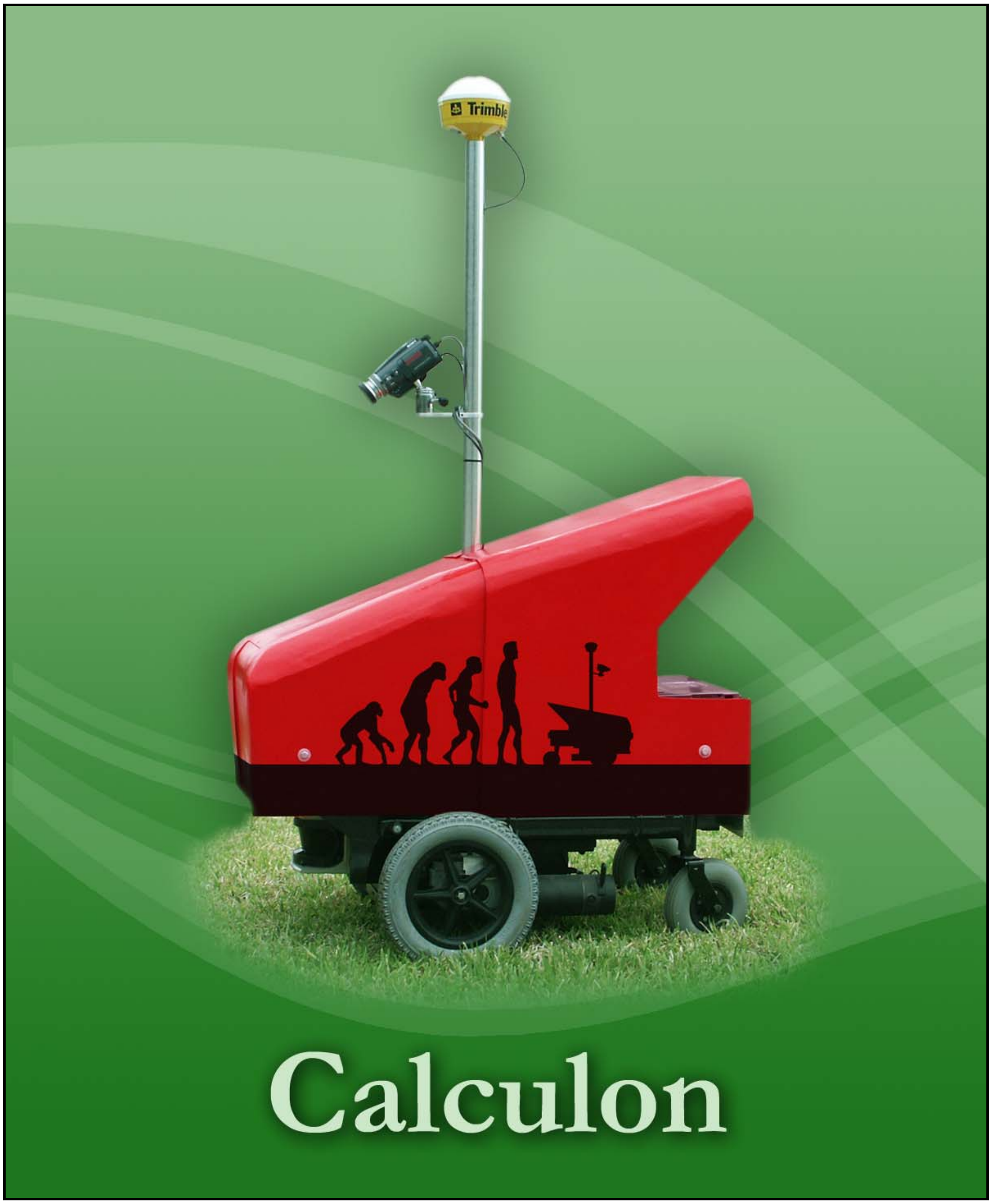


Faculty Statement:
I certify that the work done by all students on this project is consistent with a senior design course and that the vehicle has been significantly modified for this year's competition.

Dr. Fernando Gonzalez
fgonzale@pegasus.cc.ucf.edu

1. Introduction

This paper proudly presents the latest effort in ground-based robots by the University of Central Florida (UCF) called Calculon. Calculon is competing in the 14th annual Intelligent Ground Vehicle Competition (IGVC) in June, 2006. This vehicle is the culmination of many hours of work by dedicated team members using their experience from UCF's previous entries in the IGVC. Calculon has been designed and constructed to excel at all competitive events at the IGVC and addresses all problems encountered by previous entries. The goal of Calculon is an intelligent, flexible, and capable robotic platform which leverages an incremental design process, modular software design, innovative concepts, and the use of commercial off-the-shelf products (COTS).

2. Design Paradigm

2.1 System Design

Through experience and research in robotic systems design and implementation, it was found that a strict and unyielding design process, such as The Design Life Cycle, was not well suited to the team, lab structure or approach to this competition. Instead, an incremental design and development process was followed, which allows for more flexibility when needed as well as parallel development from various sub-groups, which was essential to the team.

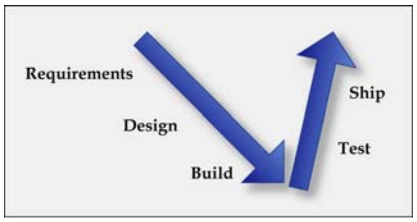


Figure 1: Illustration of the Validation 'V'

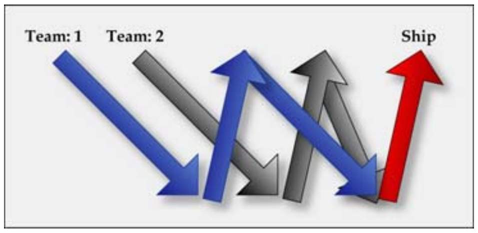


Figure 2: Parallel team development with the 'V-W' paradigm

The incremental design process used follows the 'Validation V' [1]. This method extends the Spiral Model [1], following a 'V-shaped' flow, which includes requirements stage, system design, construction, testing and completion. Initially it is difficult to identify how this is an incremental model; however, by connecting the 'Validation Vs' together you can see that a 'W' is formed representing the synchronization of multiple teams working together. With the use of COTS components during construction along with the use of standardized hardware, software interfaces, and protocols, the teams were able to attain modularity throughout the entire project that allowed many of the components to be built and worked on simultaneously. This 'V-W' paradigm works exceedingly well with this level of modularity. During project pre-planning, the team found this design paradigm to be most realistic for the group in terms of actual implementation. Many other design strategies seem feasible, but for a small group the 'Validation V-W' concept is more appropriate.

2.2 Implementing the Design Process

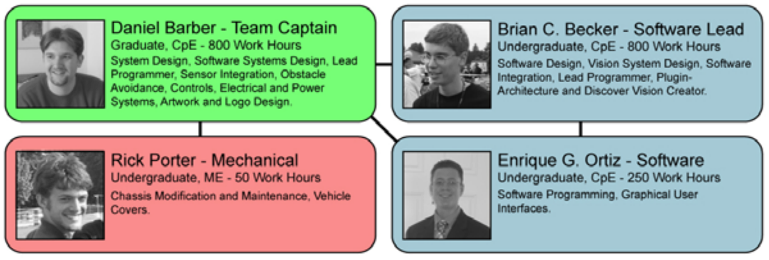
This methodology was executed by conducting weekly meetings where the layout of individual components was discussed and designed as a team, but completed individually. In the end each component was brought together to form a complete package. The weekly meetings ensured compatibility between modules, reducing risk associated with software development, and sharing new ideas with inexperienced team members.

2.2.1 Software Documentation

To reduce risk and distribute information between software developers, a strict code documentation process was implemented. Using the auto-documentation program Doxygen [2], creating electronic documents describing the relationships between modules and their functions was a simple task, saving time for developers to write software. Using the commenting standards required by Doxygen, algorithms and interfaces were described consistently, reducing confusion when integrating different modules.

2.3 Team Organization

The Calculon team consists of dedicated volunteer students from various disciplines at UCF including Computer, Electrical, and Mechanical Engineering.


Daniel Barber - Team Captain
Graduate, CpE - 800 Work Hours
System Design, Software Systems Design, Lead Programmer, Sensor Integration, Obstacle Avoidance, Controls, Electrical and Power Systems, Artwork and Logo Design.
Brian C. Becker - Software Lead
Undergraduate, CpE - 800 Work Hours
Software Design, Vision System Design, Software Integration, Lead Programmer, Plugin-Architecture and Discover Vision Creator.
Rick Porter - Mechanical
Undergraduate, ME - 50 Work Hours
Chassis Modification and Maintenance, Vehicle Covers.
Enrique G. Ortiz - Software
Undergraduate, CpE - 250 Work Hours
Software Programming, Graphical User Interfaces.

3. Design Innovations

When planning the systems for Calculon, importance was placed upon effectively developing solutions that solve the goals of the IGVC. The team felt that high flexibility and use of COTS components was a major factor in rapidly developing hardware and software systems for both known and unforeseen problems. A flexible system is one where solutions are not hard-coded or fixed to perform in only a single set of conditions and must be easily adaptable to new objectives or situations. As such, innovations were introduced through a set of novel tools and approaches that Calculon uses to achieve the IGVC objectives: COTS components, machine learning techniques, Discover Vision, and non-pixel based mapping.

3.1 Hardware Innovations

To create a winning vehicle over previous UCF efforts, the Calculon team made use of a COTS platform and high quality sensors for obstacle detection and vehicle control. Using an electronic wheelchair as the base reduces construction time and guarantees chassis reliability. Use of a more accurate DGPS and digital compass also improves vehicle control and stability over previous designs. The primary sensor suite includes a SICK LMS LIDAR combined with a high-quality 3CCD digital video camera. 3CCD creates significantly higher quality data for vision processing over 1CCD. The high color and resolution images produced by a 3CCD camera make accurate vision processing a reality in varied lighting conditions.

3.2 Machine Vision

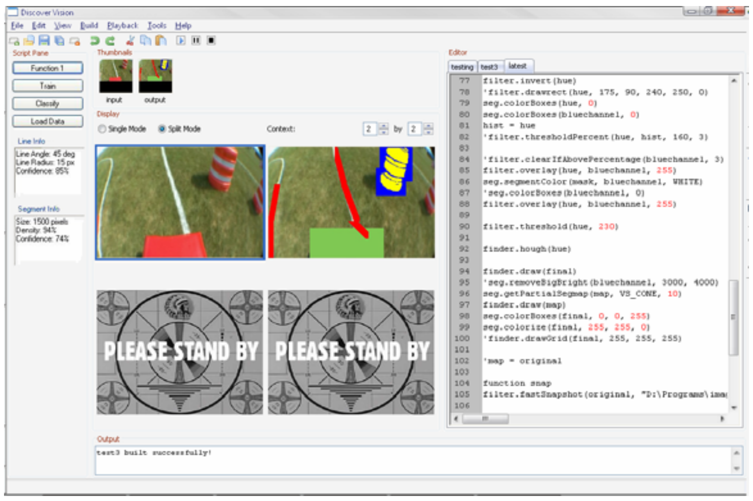
In the area of machine vision, two major innovations eased the tedious process of developing and combining machine vision algorithms to best identify objects in the environment: the use of machine learning algorithms and the creation of a rapid prototyping tool called Discover Vision. When identifying obstacles, variations in obstacle types, sizes, and poses can make classification difficult. To deal with these problems, a machine learning ARTMAP (an extension of the Adaptive Resonance Theory) based Neural Network [3] was investigated. The ARTMAP trains on a representative set of obstacle features (size, aspect ratio, color, density, etc.) to distinguish noise (such as a bright patch of grass) from a true obstacle. When compared to a simple thresholding approach that eliminates noise based on a fixed size or density threshold, the ARTMAP enhances flexibility. If new obstacles or variations are encountered, it only takes a short while to prepare and train the ARTMAP to recognize the new obstacle type, improving the rapid adaptability of the system.

Discovering the correct sequence of image processing filters to clean up and enhance an image for obstacle or line detection is a time consuming process. The solution space is extremely large and processing time can have a major impact on system performance. After researching other efforts to automate the process of evolving filter sequences [4], a genetic algorithm (GA) was used to create filter sequences for isolating lane-lines and removing background noise from images for line detection. The GA generated filter sequences of different lengths, combinations, and parameters. A filter sequence was evaluated based on how much time was required to process the image and how accurate line detection results were. The GA found solutions superior to previous methods used in both accuracy of line detection and processing time in different lighting conditions. Compared to the best human-determined solution used, the best-evolved solution performed roughly three times faster with double the accuracy. Using this system it is possible to find the best method for lane-line detection within hours based on new training images for different environments. The best solution found for line-detection is described in more detail in section 7.3.4 of this paper.

3.2.1 Discover Vision

The final machine vision innovation focuses on the creation of a rapid application development tool to better facilitate the vision system's development time. The need for such a tool arose from C++'s notoriously long compile times. To change the order or tweak the parameters of the machine vision algorithms used to identify lines and obstacles, a developer would have to close the program, make the changes to the code, wait for a long recompile & linking process, and finally rerun the program. To combat this time-consuming method of testing and experimenting, the team developed a rapid prototyping tool called Discover Vision 2.0. Based off the original Discover Vision written by the Calculon 2005 team, the aim of the tool is to enable developers to dynamically add, modify, or remove vision related algorithms with minimum latency and interruption between edits.

Discover Vision 2.0 is modeled after existing tools, such as MatLab and the original Discover Vision [5], presenting an interface capable of easily manipulating images or video. Discover Vision is composed of two parts, a Graphical User Interface (GUI) and backend module called the Discover Vision Engine. The GUI of Discover Vision sports two main components: a code editor and image display. The code editor incorporates syntax highlighting and line numbers to boost productivity. The image display uses two modes: a single pane for showing the full details of an image, or split N x N panes, each pane containing a separate scaled down image.



Context: 2 by 2

Line Angle: 45 deg
Line Radius: 15 px
Confidence: 85%

Size: 1500 pixels
Density: 94%
Confidence: 74%

```
filter.invert(hue)
'filter.drawrect(hue, 175, 90, 240, 250, 0)
seg.colorBoxes(hue, 0)
seg.colorBoxes(bluechannel, 0)
hist = hue
'filter.thresholdPercent(hue, hist, 160, 3)

'filter.clearIfAbovePercentage(bluechannel, 3)
filter.overlay(hue, bluechannel, 255)
seg.segmentColor(mask, bluechannel, WHITE)
'seg.colorBoxes(bluechannel, 0)
filter.overlay(hue, bluechannel, 255)

filter.threshold(hue, 230)

finder.hough(hue)

finder.draw(final)
'seg.removeBigBright(bluechannel, 3000, 4000)
seg.getPartialSegmap(map, VS_CONE, 10)
finder.draw(map)
seg.colorBoxes(final, 0, 0, 255)
seg.colorize(final, 255, 255, 0)
'finder.drawGrid(final, 255, 255, 255)

'map = original

function snap
filter.fastSnapshot(original, "D:\Programs\ima
```

```
test3 built successfully!
```

Figure 3: Discover Vision GUI

The split panes allow multiple filtered images to be viewed simultaneously in real time. This is extremely useful to determine the effectiveness of a specific filter. To maximize the usefulness of Discover Vision 2.0 (forthwith referred to as just Discover Vision), it can process images from a number of different sources, including a series of pictures, video cameras, and saved videos.

A simple scripting language was developed for Discover Vision in order to interface the GUI to the Discover Vision Engine. The scripting language currently supports over 22 classes and nearly 400 member functions, and it is easily extensible. The scripting language supports overloaded functions as C++ does and offers the ability to create user-defined functions that can be called from the GUI. To increase speed, the script is compiled into byte-code, which is run by the Discover Vision Engine. Whenever a 'recompile' occurs, the byte code is reloaded without any interruption to the program's execution resulting in instant changes in the video stream as the new set of filters is running. Using scripts, a completely new vision system can be created without changing any underlying C++ code of an application using the Discover Vision Engine.

Two unique features have been added to Discover Vision, increasing its value: pipelined filters and plug-ins. Pipelined filters are multiple filters that do not depend on each other and execute at the same time in different threads. Due to the current implementation of most CPU architectures, executing several filters simultaneously allows the processor to balance processing load better than executing filters serially. The primary reason for this is that if the CPU needs to pause the execution of one filter to wait for data, the CPU can execute one of the other pipelined filters. This reclaims CPU processing power that would otherwise be wasted and experiments have shown a four-fold performance improvement. In the team's attempt to ensure flexibility in all areas of the system, a plug-in architecture was added to Discover Vision. Without plug-ins, all the filters developed in C++ must be hard-coded into the application. If a new filter is created, the developer needs to add additional lines of code to the Discover Vision source and then recompile the program. With a plug-in architecture, all of the filters can be created separately and placed into a special "plug-ins" folder. Discover Vision will then load all these plug-ins dynamically at start up. This represents significant flexibility, as any member of the team can create a filter and use it in Discover Vision immediately without knowledge of the Discover Vision source code. Use of the Discover Vision program is further described in section 7.3 of this paper.

3.3 Geometric Based 3D Mapping (Non-Pixel Based Mapping)

In previous years at IGVC, UCF teams used pixel/image based representations for building maps of robot surroundings. This method had sensors draw obstacles into an image used by path-planning and obstacle avoidance software. Although easy to understand and implement, this method requires intensive processing time when performing map analysis because it becomes an image-processing problem. For example, if obstacle avoidance software needed to scan for obstacles within a radius of a point in the map, every single pixel covering the area of this circle would need to be examined. In summary, after a sensor has identified an obstacle and drawn it in the map, it must be rediscovered by obstacle avoidance software in the map image. This can be extremely computationally expensive and redundant, slowing down system performance. A new representation designed to solve this problem abandons the pixel/image based method in favor of geometric representations.

In this method, all obstacles, lane-lines, and other features are represented by data structures containing a feature's position, height, width, and length in the map. The map is broken down into smaller grids and obstacles are sorted. Each grid section of the map keeps track of how many features are located inside of it, and features can belong to multiple grid sections based on their dimensions. Using this representation, if a process needs to perform the same radial scan mentioned previously, a simple two-step method is used. First, a check to see what map grids the scan encompasses and only if those grids contain features is the second step performed. If features are found, they are checked using rules of geometry to determine if they are within the scan area. This drastically reduces the amount of processing time required because grids without obstacles are ignored, unlike the previous image based method requiring every pixel to be checked. Experiments performing a depth-first search with a depth of five and ten branches at each node on a local map showed a performance increase of 1000% over previous efforts. Additional benefits to this technique not found or difficult to do with the previous method include: separation of left and right lines of a lane, calculating lane direction, and reducing the amount of data represented in the map by grouping obstacles together into a single structure. This design innovation allows path-planning and obstacle avoidance, described later in this paper, to perform at a much higher frequency.

4. Mechanical System

In the continuing effort to use COTS components, a Ranger II Storm Series^{tm} motorized wheel chair was chosen as the base chassis. The wheel chair fits all of the project's needs providing a stable mobile platform with powerful motors, integrated motor controller, and large rugged wheels. The wheel chair uses a differential drive system making it very maneuverable and easy to control. In previous projects, an Ackerman drive system was used which made path-planning and control more difficult in tight areas. The zero-degree turning radius provided through differential steering lets the vehicle navigate around all obstacles easily.

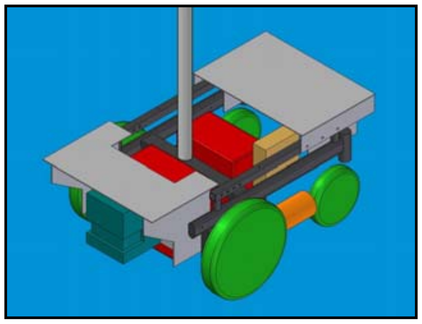


Figure 4: CAD Model of Chassis Modifications

Two large motors control the 12.5-inch diameter front wheels. Two small caster wheels allow for smooth motion and in-place turning. In addition to the wheel chair frame, additional mechanical components were added to facilitate the robotic platform. This includes a center pole supporting the digital video camera and DGPS antenna. Also added was a front mounting bracket to attach the LIDAR with protective hood to shield it from the environment and any ambient infrared light that could interfere with its operation. Finally, a drawer added to the back of the platform houses electronics necessary for operation as well as power distribution units. The front panel of the drawer contains power switches, battery charging port and electronics reset. On top of the drawer is a cork-padded resting place for the main computer described in more detail in the electrical systems section of this paper. A fiberglass cover provides moderate environmental protection from light rain allowing continuous outdoor operation. This cover also provides shade, making the screen for the main computer easier to see in all outdoor lighting conditions.

5. Electrical System

5.1 Power Distribution

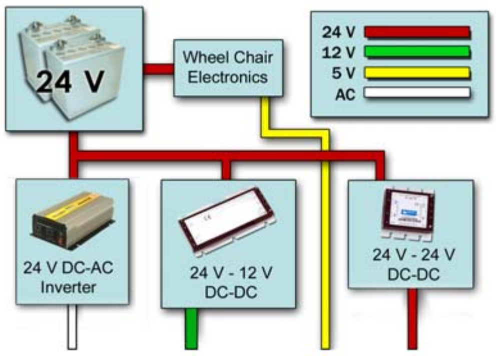


Figure 5: Power Distribution

Calculon's power is provided from two 55Ah 12-volt batteries connected in series to provide a single 24-volt power supply. The 24-volt source powers the wheelchair platform's motors and motor controller. The platform's motor controller also provides a 5-volt power source for custom electronics needed to interface the motor controller to the computer system. This 24-volt power source is also connected to a DC-to-AC inverter that supplies power for the main computer and the digital video camera. The 24-volt source also connects to a 24V-to-12V and a 24V-to-24V converter. The 24V-to-24V converter functions as a regulator to guarantee a 24-volt source for the SICK LIDAR and Trimble DGPS. The 24V-to-12V converter provides a regulated 12-volt source to power the magnetic encoders, digital compass, and wireless E-Stop. Two LED's on the front of the electronics drawer provide instant verification that electrical systems are functioning properly.

5.2 Electronics

A PIC 16F876A MCU and a D/A converter are used to interface with the onboard motor controller of the wheel chair. Through this interface, the MCU can vary the speed and steering of the wheel chair. A second PIC MCU is used to capture signals from the magnetic encoders mounted on each wheel. An RS-232 connection from each PIC MCU to the main computer allows it to get encoder information and control the speed and steering of the vehicle. Two individual connections were used to prevent communication saturation. This lets the MCU connected to the encoders continuously transmit information while not blocking drive commands to the MCU interfaced to the motor controller. Encoder information is used for both dead reckoning and vehicle control.

If one were to assign a 'brain' to Calculon, it would no doubt be the Alienware Area-51m 7700 desktop replacement notebook. With an Intel Pentium 4, 3.4 GHz Processor with Hyper-Threading Technology and 1.0 Gigabyte of RAM, the computer controls driving, path-planning, vision systems and interacts with the rest of the robot's systems via multiple RS-232 serial interfaces, RS-422 serial interface and FireWire. The computer runs the Microsoft Windows XP operating system.

When not in autonomous mode a Logitech wireless joystick is used to drive Calculon. This joystick's receiver is connected to the main computer over USB, and the server program uses information from it to send desired speed and steering commands to the onboard electronics. The server program is further described in the software section of this paper. The wireless joystick lets a user drive Calculon manually and switch between autonomous drive modes within an operational range of 50 feet.

6. Sensors

Highly accurate and efficient sensors are used by Calculon. A SICK brand LMS-291 LIDAR provides two-dimensional information up to 30 meters in front of the vehicle in a full 180° sweep. This information is transmitted back to the main computer via RS-422 at 500kbps giving exact placement of obstacles relative to the vehicle in real-time.

A Sony DCR-HC1000 3-CCD Mini-DV camera is used for machine vision. This camera provides a clear 720x480-resolution image. On previous vehicles 1CCD web-cams were used which do not handle varied outdoor lighting conditions representative of the IGVC. The 3-CCD chip provides high quality color and white balance reduction in various lighting conditions.

For location of GPS waypoints, a Trimble DSM 132 DGPS is used. This provides sub-meter accuracy with an update rate of 1Hz. The Trimble is connected to the main computer over an RS-232 connection at 19200bps. The Trimble was chosen because of its high accuracy and reasonable price. Latitude and Longitude information from the Trimble is converted to Universal Transverse Mercator (UTM) System. UTM provides a cylindrical projection in meters, which is highly accurate for use in waypoint finding.

A TCM2-50 digital compass provides heading, pitch, and roll information. Internal filtering keeps the compass accurate within two degrees. Heading information updates at a rate of 10Hz with filtering, providing almost instantaneous response to physical vehicle movement. Like other sensors in the system, it connects to the main computer via RS-232 at 19200bps. The TCM-250 was chosen over others because of its fast update time and accuracy, reducing oscillation when controlling vehicle heading.

Two Baumer Electric MDFK 08G2101 magnetic sensors are placed on each drive wheel. Each reader uses a 256 pulses ring magnet with 16 pole-pairs. The pulses from these sensors are used to calculate vehicle speed and position. This information is used for speed and steering control as well as dead reckoning. The output is connected to both PIC MCUs with one transmitting to the main computer via RS232 and the other controlling speed and turning rates. The dead reckoning position calculated using the magnetic encoders is used by a Kalman filter to reduce position error. In the event of lost GPS signal, position updates are replaced by the dead reckoning position. This allows the vehicle to continue operation while waiting for return of GPS signal. These magnetic sensors are easy to integrate with the COTS chassis and give fast and accurate information.

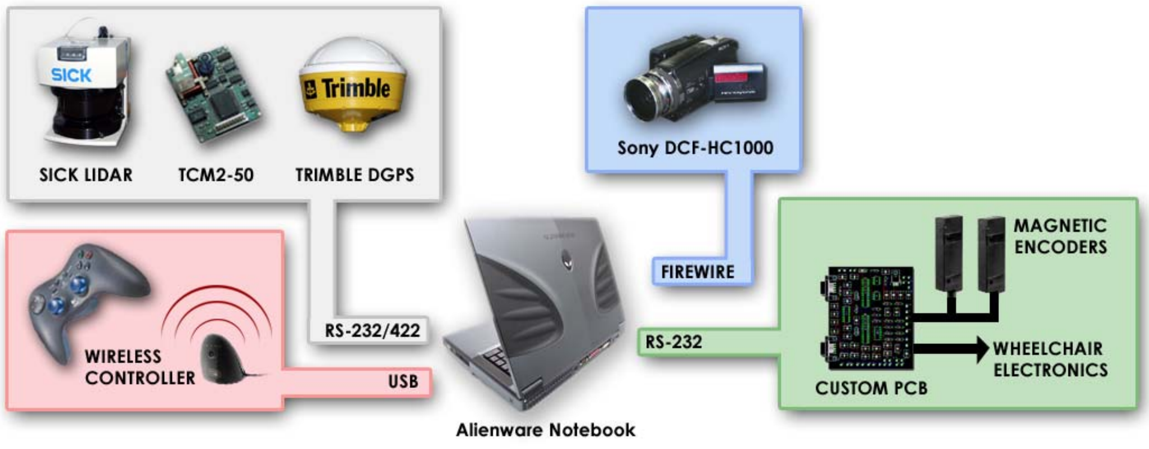


Figure 6: Systems Integration for Calculon

7. Software System Design

The overall goal for Calculon is to leverage COTS components in order to build a functional, easy to use, easy to maintain, and modular robotic platform. On top of this platform is the intelligence of the system, the software. After exploring various alternative programming methods such as pre-built robotic control libraries and visual based programming tools such as LabVIEW, it was decided none of these options afforded the programming team both the modularity and the low-level control they wanted. Instead, the software was implemented in C++. Since the software team was already knowledgeable in C++, an additional training phase was not necessary to learn a new language and provided the team with the flexibility and low-level control desired. In addition, the use of C++ provides the execution speed needed for real-time applications. To avoid some of the typical complexity problems associated with C++, a strict object-oriented development guideline was instituted resulting in high quality and reusable software.

In keeping with a modular software system, two main components make up the software of Calculon: the server and the client. The server program acts to maintain vehicle speed and steering while continuously retrieving positional sensor information and wireless joystick input. This server is a low-level system upon which intelligent systems can be built by providing a standard interface to drive the platform and capture sensor information for clients. The client and the server communicate over a TCP/IP connection. The client program acts as the intelligence of Calculon. This client performs obstacle detection, path-planning, and obstacle avoidance. It is able to drive the vehicle by sending speed and steering commands to the server.

7.1 Server

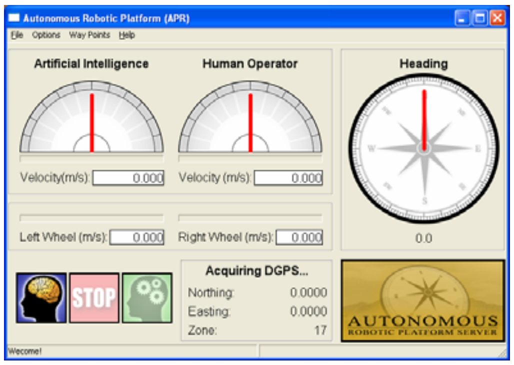


Figure 7: Server program user interface

The server program performs two main functions. The first is to create a base system for platform control and aid development of higher-level programs. The second is for manual driving of the vehicle by wireless joystick and initiation of autonomous drive mode. The server program interfaces to the DGPS, compass, encoders MCU and vehicle control MCU. It continuously processes the output from the sensors and makes it available to the client over TCP/IP. This acts as a level of abstraction so that sensors can be replaced or modified without affecting the client program. A Kalman filter uses information from the wheel encoders to reduce error in position from the DGPS, and provides updates in position through dead reckoning during loss of DGPS signal. Sensor information is displayed on a GUI for quick debugging, (Figure 7). The GUI displays the current heading, position, and speed of the vehicle, as well as the mode of operation (Autonomous, Manual, or E-Stop), so that the user can quickly determine if a sensor is functioning. Settings for these sensors can be modified through built-in menus. This program also provides an interface for the client to set the speed and steering rate of the vehicle. All of this functionality frees the client from having to interface directly to sensors and vehicle actuators using a standard interface.

7.2 Client

The client designed to connect to the server is the intelligent behavior engine of Calculon. Within this program machine vision, path-planning, and obstacle avoidance is performed. While connected to the server, position and heading information is retrieved, and the desired speed and steering rates are requested. Two different path-planning algorithms are used by the client, one for the Autonomous Challenge, and the other for the Navigation Challenge. Specific path-planning algorithms are chosen based on requirements of the challenge. The differences in the path-planning are further described in the Autonomous Challenge and Navigation Challenge sections of this paper. Both versions use the same machine vision engine, driver program, lookout, LIDAR interface, local map, and server interface. These systems are further described later in this paper. A GUI displays the local map, video from the camera, classified image data, and output from the path-planner. This interface is extremely useful for quickly determining what the vehicle detects and where it plans to go. Buttons and settings tabs on the interface switch between Autonomous/Navigation Challenge modes and other configuration settings.

7.3 Vision System

The vision system is created using Discover Vision. The Discover Vision Engine is incorporated into the client program, and runs scripts compiled using the Discover Vision GUI. The key modules used by the scripts include color classification, color segmentation, obstacle recognition, and line detection. Through this process, the surrounding environment is mapped from the incoming video of the Sony camcorder. Because the LIDAR can identify most solid 3D obstacles such as cones and buckets, special emphasis is placed on the vision system to identify lines and non-solid 3D obstacles such as barricades.

7.3.1 Color Classification Algorithm

One of the most important filters used is an adaptation of a Gaussian-based skin color-classification system [6]. It is trained on images found to be typical of the course. Training consists of using images from the camera system along with hand made custom classification masks indicating which colors to train on. The training algorithm then places a small Gaussian sphere of either positive or negative values depending on the mask into a large RGB cube. Once the training has been completed, color recognition is performed on new images by comparing each pixel in the input image to its corresponding location in the RGB cube. If the value at that location is over a preset threshold (meaning that color has been previously trained on), then the color is said to be detected. One of the major advantages to this method is that the colors are classified using a fast lookup during runtime, and color classification can be re-trained to handle different environments.

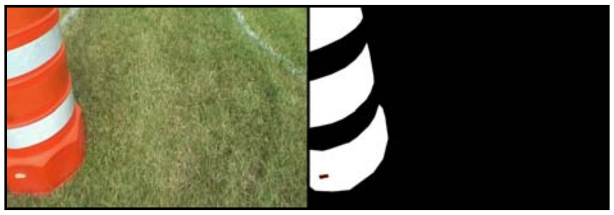


Figure 8: Color classification Training Image and Mask (Training on the Color Orange)

7.3.2 Image Segmentation

Image segmentation is the process of logically grouping pixels representing some object in the image into a cluster or segment. The basis for the image segmentation performed is the color classification algorithm. Because all obstacles at the IGVC have colors distinct from the surrounding environment, color classification is used on each image to detect which pixels are colors of competition obstacles. A fast line-scanning algorithm [7] is used to group pixels into segments and calculate a number of useful features about them. These features include size, density, aspect ratio, centroid, contour length, compactness, contour bending energy, and average color [8].

7.3.3 Obstacle Recognition

Once the image segments have been extracted from the image, the noise must be separated from the real obstacles through an obstacle recognition (or filtering) process. This process involves analyzing the different image segment features and attempting to remove those segments that most likely represent noise. Two different methods were implemented: thresholding and a machine-learning algorithm. Often times, a custom threshold on selected features such as size can best eliminate noise because it often has well defined characteristics (small size, high randomness, etc). However, such a model takes time to develop, and in cases where quick adaptability is required (for instance, the addition of a new obstacle type), the machine learning algorithm mentioned earlier, ARTMAP Neural Networks [3], [9], have proven to be quite successful in achieving nearly the same accuracy as a hand-tweaked model with significantly more automation. The ARTMAP can be re-trained for a new obstacle type in a matter of minutes, resulting in rapid adaptability when new obstacle types are introduced.

7.3.4 Line Detection

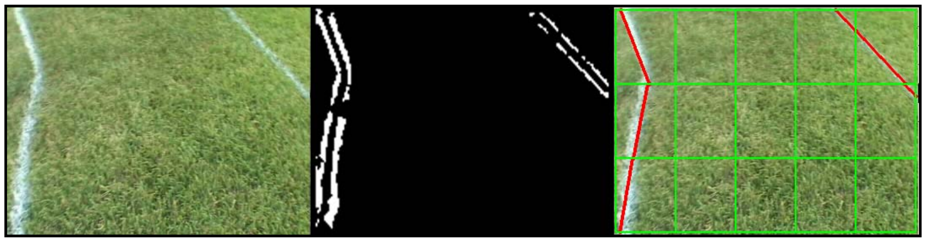


Figure 9: Original image (left), filtered image (middle), classified lines (right).

Line detection is performed by applying a Hough Transform [10] across a segmented filtered image. First, several filters are used to preprocess the image before applying line detection. These filters serve to remove noise and other obstacles that do not represent lines. The Genetic Algorithm mentioned in the Innovations section is responsible for the evolved filter sequence used which includes the following filters: sub-sampling, blurring, edge detection, thresholding, and erode [11]. The resulting image is decomposed into a matrix of 5x3 cells and the Hough Transform is applied to each one. Once line segments are identified, an additional step is taken to join adjacent lines together. The benefit of segmenting the image into a matrix before applying the Hough Transform is the ability to find left and right lane-lines and approximate curves in the course.

7.4 Cartographer

The cartographer program implements the non-pixel based mapping method described in the Innovations section of this paper. This system combines data from the SICK LIDAR and vision system and includes built-in functions for analyzing the map. These built-in functions make it easy to scan regions or paths within the map, reducing the complexity of path-planning software. Cartographer is the main juncture, which connects obstacle detection (sensors) to obstacle avoidance software (path-planning).

7.4.1 Reducing SICK LIDAR Data

An additional feature added to the SICK LIDAR interface is a technique for reducing the amount of incoming data. During a typical 180° scan, the LIDAR can return up to 360 points where obstacles are located. Most of these points are clustered together, representing wall or obstacle outlines, and can be combined. A split-merge method [12] fits points from the LIDAR along line segments. This method reduces the amount of data added to the map from the SICK LIDAR by a factor of 3-50, improving overall system performance.

7.5 High-Level Control

Good sensor data is useless unless the vehicle is intelligent enough to make correct decisions and drive the course. The team's design called for breaking the job of moving the vehicle through the competition courses into three major parts: driver, lookout, and path-planner. Breaking the high-level control into multiple parts made it easier to reuse as much code as possible. The driver program actually moves the vehicle by controlling the heading and vehicle speed. The lookout software ensures the vehicle does not drive into obstacles in the event of a previously unforeseen danger. Finally, the path-planner software chooses what direction and velocity the driver should use to navigate the two different courses of the IGVC. A different path-planning algorithm is used for each course and is described below.

7.5.1 Driver

Since the software systems are built in a hierarchical way to make use of parallel development, the 'driver' program receives navigation commands from the path-planner. Once the path-planner decides where to go, it is the responsibility of the driver software to make it happen. The driver uses a PID controller to choose steering/turning rates to maintain a compass heading chosen by the path-planner. The driver also makes sure there is nothing directly in front of the vehicle when driving forward as an additional safety precaution. Finally, the driver is capable of determining if the vehicle is stuck. The vehicle is defined as stuck if it cannot do one or more of the following: rotate in place, drive forward, or reverse. Several strategies have been added to remedy these situations autonomously. In the event the vehicle cannot turn in place, the driver program moves the vehicle forward or backwards at low speeds, attempting to get out of whatever terrain is causing problems. This is extremely useful when turning in thick grass or loose dirt. If the vehicle cannot drive forward, it reverses along the previous trajectory. After reversing, the vehicle drives forward at a higher speed, if the path in front of the vehicle is clear. This is useful when the vehicle has trouble getting over any bumps.

7.5.2 Lookout

The combination of the path-planner along with the driver software allows the vehicle to complete both competitions in an efficient and timely matter, in theory. However, good design as well as experience has proven that theory and reality do not always coincide, especially in competition. Because of this, a 'lookout' program whose responsibility is to keep track of nearby objects is used. In the unlikely event the path-planner routes the vehicle too close to an obstacle or a previously unforeseen, possibly moving obstacle appears in the vehicle path, the lookout program takes over control from driver program to stop the vehicle. The lookout continuously analyzes the vehicle's surroundings and predicts if it will hit an obstacle based on the current speed, heading, stopping time, and the desired vector the path-planner wishes to take. If it determines a hit is imminent, the driver and vehicle are stopped. When the vehicle comes to a complete stop, the path-planner attempts to plot a new course. If no solution is found because the vehicle is directly in front of an obstacle, the vehicle reverses along the previous trajectory taken until a new vector to travel is found.

7.5.3 Path-Planning for Autonomous Challenge

For the Autonomous Challenge, the path-planning system relies entirely on a local map. The robot is able to make decisions, plan a path and traverse the course entirely with only local obstacle and history information. During operation, Calculon records the vehicle pose on a fixed interval. The vehicle pose includes global position, compass, pitch, roll, velocity, and a time stamp. Using the history, statistical information is calculated, including trajectory traveled and a global position map. This data is extremely useful to maintain an overall direction along a course, avoiding the same dead-end/trap repeatedly, and determining if the vehicle has accidentally turned around.

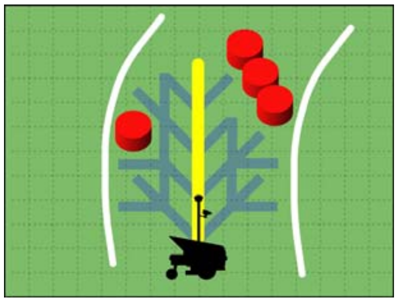


Figure 10: Visual Representation of Autonomous Challenge Algorithm

The main algorithm used during the autonomous challenge to choose a velocity and heading for the driver is a beam-search on the local map. This method finds all possible ‘safe’ paths the vehicle can take. A path is considered safe if it avoids obstacles and stays within the lane. Path solutions found from the beam-search are then evaluated to find the best possible path for the vehicle to take. Paths are scored based on their distance, changes to past trajectory, global history, and distance from lane center. A path covering a greater distance is less likely to lead into traps or dead-ends. Maintaining the current vehicle trajectory is important because it prevents the vehicle from turning around. Considering the global history prevents Calculon from repeatedly re-entering traps or dead-ends. Therefore, paths that revisit global positions are ranked lower than those that do not. Finally, a path that leads to the center of the lane or follows the direction of the lane is considered best because it will keep the vehicle on course.

7.5.3.1 Solid and Dashed Lane-Lines

Lane-lines are treated differently from obstacles in the local map. When identified, lane-lines are sorted based on orientation to the vehicle, namely left or right lane-lines. Using this information, the direction and center of the lane is calculated. As long as a lane-line (solid or dashed) is detected, Calculon knows what direction to travel to stay within the lane.

7.5.4 Path-Planning for Navigation Challenge

Because the Autonomous Challenge requires the vehicle to stay within a lane, more information is required than in the Navigation Challenge, and therefore a different algorithm is used. A queue of GPS points for the course is loaded from a text file and traversed in the order they are provided. Calculon then scans the area in the front of the vehicle and chooses a compass heading that leads towards the desired waypoint while avoiding obstacles. The direct path chosen must be wide enough for the vehicle to travel, with room for error to minimize the chance of hitting any obstacles. Similar to the Autonomous Challenge, additional criteria from the history is used to evaluate the final compass heading taken. Compass headings that lead to areas already visited are ranked lower than those that do not, preventing Calculon from going into a loop in any area of the map. Finally, an additional precaution is taken when evaluating the final heading chosen. To make sure that the vehicle is capable of following a wall, and not turning around and going back and forth trying to get to a waypoint, more extreme changes to the heading are ranked as less desirable. This helps to ensure Calculon will travel around large walls or groups of obstacles to reach a waypoint or escape an enclosure while still finding the most direct route to the waypoint.

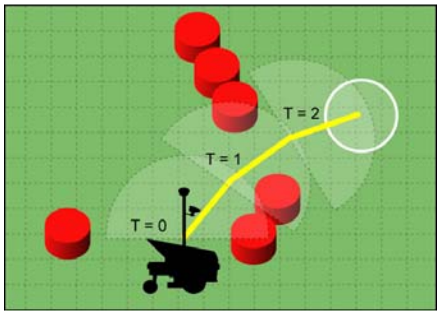


Figure 11: Visual Representation of Navigation Challenge Algorithm

7.5.5 Continuous Planning

In the previous sections of this paper, many innovations have been introduced which drastically improve the performance of the software systems over previous efforts by UCF. With the reduced computation time needed for map building and analysis, it is possible to make path-planning updates with high frequency. By continuously analyzing the current path for environmental changes, it is possible to drive autonomously at the best advisable speed because of better response time. Benchmark testing for both path-planning methods shows an average path-update time of 50ms, 20Hz, more than fast enough for driving at full speed. Additionally, Calculon determines the best advisable speed based on the density of obstacles near the path selected. Calculon runs at full speed in areas devoid of obstacles, but reduces speed when navigating in tight quarters.

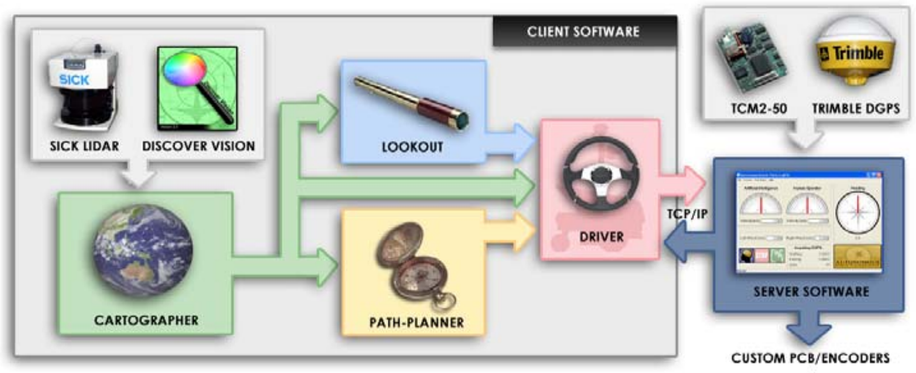


Figure 12: Software Systems Integration

8. Vehicle Performance

8.1 Speed

The COTS wheel chair used as a mechanical platform was built to transport an average sized person at low speeds. Because of this, speeds are limited to approximately 1.2 m/s, which is within the maximum allowed speed of the IGVC. The software system is able to maintain speed within 0.05 m/s.

8.2 Ramp Climbing

Calculon's wheelchair based mechanical platform was designed to carry humans weighing up to 300 lbs. With this fact in mind, Calculon was tested at a variety of different ramp gradients up to 10°. Thus, the team does not expect any ramp encountered during the competitions to pose any problem for Calculon.

8.3 Reaction Times

Calculations based upon software loop times and sensor refresh rates show that Calculon is able to detect new obstacles and react in approximately 0.05 seconds. During testing, Calculon was able to react and stop to new obstacles and unforeseen objects in its path within less than two feet when at full speed.

8.4 Battery Life

As described previously, power is supplied by two 55Ah 12-volt batteries. Testing has shown that these batteries can run Calculon for up to four hours on a single charge. For the competition, interchanging the batteries with a second set allows for a total run time of eight hours without recharging.

8.5 Obstacle Detection Distance

Calculon's sensors are able to detect solid objects in front of it as far as thirty meters using its LIDAR. Depending on the angle of the camera, Calculon's vision system is able to detect and identify objects as close as directly in front of it and as far as four meters. Through testing, it has been found that the vehicle can reliably detect obstacles up to nine meters away through a combination of sensor data from the LIDAR and the digital video camera. This distance is more than adequate for the completion of both competitions.

8.6 Dead Ends, Traps, and Potholes

Calculon is designed to deal with dead ends, traps, and potholes throughout both the Autonomous Challenge and the Navigation Challenge. Calculon's path-planning algorithms keep a complete record of the path previously taken by the vehicle. This means that in the event that Calculon inadvertently plans a path into a dead end, it can reverse along its previous trajectory to a safe point and re-plot a new course. Both path-planning algorithms make use of the path history to prevent plotting a course to the same dead-end or trap. Potholes are treated the same as obstacles and avoided by the path-planner when detected.

8.7 Waypoint Accuracy

Through the driver program's high level of control and the use of the Trimble DGPS, Calculon is able to reach GPS waypoints within one meter. This meets and exceeds the requirements of the competition to reach waypoints within a two-meter radius. In the event of lost DGPS signal, dead reckoning software can continue updating the current position until signal is reacquired. The accuracy of the dead reckoning software is within 0.2 meters along a path up to 15 meters.

9. Analysis and Conclusion

9.1 Budget

Item	**Company**	**Cost**	**Qty.**	**Total Cost**	**Our Cost**
Ranger II Power Wheel Chair	Inva-Care	$3,200.00	1	$3,200.00	$0.00
SICK LMS LIDAR	SICK-USA	$4,449.00	1	$4,449.00	$4,449.00
Sony DCRHC1000 3 CCD Digital Damera	Sony	$1,183.00	1	$1,183.00	$1,183.00
Desktop Replacement Computer	Alienware	$3,065.00	1	$3,065.00	$3,065.00
Trimble DSM132 DGPS Receiver and Antenna	Trimble	$4,050.00	1	$4,050.00	$1,000.00
Encoder Readers and Rings	Baumer Electric	$104.00	2	$208.00	$208.00

TCM2-50 compass	PNI Corporation	$800.00	1	$800.00	$800.00
Power Converters and Inverter	Vicor	$395.00	1	$395.00	$395.00
Wheels	Inva-Care	$124	2	$248	$496.00
Fiberglass Material	US Composites	$294.00	1	$294.00	$294.00
Machine Work	Geo Systems	$1,800.00	1	$1,800.00	$1,800.00
Circuit Board	4pcb.com	$48.00	1	$48.00	$48.00
Miscellaneous	Various	$1,000.00	various	$1,000.00	$1,000.00
			Total	$20,885.00	$15,061.00

9.2 Safety Analysis

No design is complete without a proper analysis of the safety measures taken to guarantee that no person can or will be hurt during the operation of the vehicle. The IGVC vehicles, while sometimes small, can be over 100lbs in weight and are often driven by powerful, high-torque motors. For Calculon, safety has been a primary consideration beginning in the pre-planning stages of design. Electrical connections are kept safely within the electronics drawer. Batteries are charged only via the charging port on the outside of the vehicle and all connections for sensors and vehicle control are done using standard port interfaces such as DB9, as opposed to direct solders. This allows for easy and safe disassembly of the vehicle without worrying about electrical problems.

In accordance with the IGVC rules, Calculon includes multiple forms of emergency stopping. On the outer casing of the vehicle is a large red emergency stop button that is wired directly to the power control system of the vehicle. When triggered, power is cut to motor controllers, enabling a mechanical brake, stopping the vehicle in place without rolling even while on an incline. An electronic wireless switch is also connected to the power system identically to the emergency stop button. The range of the wireless emergency stop is up to 500 feet and acts as another means to prevent any dangers to observers and surroundings.

9.3 Conclusion

Calculon is the result of a concerted effort among the hardworking members of the Robotics Laboratory at the University of Central Florida. The team is confident that Calculon's hardware and software design, especially its new innovative systems, is well suited to accomplishing the goals of the 2006 Intelligent Ground Vehicle Competition. Calculon's reliable chassis and sensor suite enable it to perform consistently at peak condition to achieve the IGVC goals. Its graceful handling of sensor failure, such as DGPS loss, while continuing to operate normally demonstrates the design's ability to function in adverse conditions. The incremental 'Validation V' design process involves all members in designing software modules and reduces risk between independent, parallel development. The flexible vision system solutions utilizing machine learning algorithms underscore Calculon's capability to adapt to new situations and quickly assimilate changes in the environment. Fast mapping combined with continuous path-planning enables not only quick reaction times, but also higher speeds and more detailed analysis. The Calculon team believes that these innovative design features will make Calculon stand out against the dozens of other competing vehicles at this year's competition.

Appendix A: References

1. Cockburn, A., “Using “V-W” Staging to Clarify Spiral Development,” *Human and Technology*, 1995.
2. Doxygen documentation system. http://www.stack.nl/~dimitri/doxygen/index.html.
3. Williamson, J. R., “A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps,” *Neural Networks*, Vol. 9, No. 5, 1996, pp. 881-897.
4. Brumby, S.P., Harvey, N.R., Theiler, J., Perkins, S., Szymanski, J.J., Block, J.J., Porter, R.B., Glassi, M., Young, C.A., “Comparison of GENIE and Conventional Supervised Classifiers for Multispectral Image Feature Extraction,” *IEEE Transaction on Geoscience and Remote Sensing*, vol. 40, no. 2, 2002.
5. Barber, D.J., Kiriwas, A., Gonzalez, F.G., Roberts, T., Becker, B.C. “Calculon.” In the *Proceedings of The 13th Annual Intelligent Ground Vehicle Competition*. AUVSI, http://www.igvc.org, June 2005.
6. Yan, N.A., “Gaussian Mixture Modeling of Human Skin Color and Its Applications in Image and Video Databases”, *SPIE/EI&T Storage and Retrieval for Image and Video Databases*, 1999
7. Bruce, J., Balch, T., Veloso, M. “Fast and Inexpensive Image Segmentation for Interactive Robots,” In the *Proceedings of IROS-2000*, Vol. 3, Takamatsu Japan, October 31, pp 2061-2066.
8. Nixon, M and Aguado, A. *Feature Extraction & Image Processing*. Newnes, Woburn, 2002 pp 247-290.
9. Anagnostopoulos, G. C. “Exemplar-based Pattern Recognition via Semi-supervised Learning” *IJCNN*, 2003, Portland Oregon, July 20-24, pp. 2782-2787.
10. Duda, R.O., Hart, P.E., “Use of the Hough Transform to Detect Lines and Curves in Pictures”, *Communications of ACM*, 1972.
11. Seul, M., O’Gorman, L., Sammon, M.J., *Practical Algorithms for Image Analysis: Description, Examples, and Code*. Cambridge University Press, New York, 2000 pp 118-176.
12. Nguyen, V., Martinelli, A., Tomatis, N., Siegwart, R., “A Comparison of Line Extraction Algorithms using 2D Laser Rangefinder for Indoor Mobile Robotics.” In the *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS’2005*, August 2006, Edmonton, Canada, pp 1929 – 1934.