Lawrence Technological University

Viper

IGVC 2008 Autonomous Vehicle

Team Members

Gary Givental, Philip Munie, Shawn Ellison, Brandon Bell, Doug Stevens, Angelica Sierra, Aaron Richards, Jeremy Gray

Faculty Advisor Statement[1]

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Viper has been significant and each team member has earned credit hours for their work.

_____ _____

Dr. CJ Chung (chung@ltu.edu, 248-204-3504) Date

[1] Co-advisors: Dr. Lisa Anneberg, Dr. Patricia Shamamy

Table of Contents

1. Introduction 3
2. Innovations 3
2.1 Hardware Platform 3
2.1.1 Mechanical System 3
2.1.2 Electrical System 3
2.1.3 Software 3
3. Design Process 3
3.1 Collaborative Team Organization 3
3.2 Project Planning Process 4
3.3 Development Process 4
3.4 Testing Process 5
3.4.1 Hardware Testing 5
3.4.2 Software Testing 5
3.4.3 System Integration 5
4. Hardware Design 5
4.1 Robot Structure 5
4.1.1 Chassis 5
4.1.2 Suspension 5
4.1.3 Drive Train 6
4.1.4 Body 6
4.2 Motors and Electronics 6
4.2.1 Motor Control 6
4.2.2 Sensors 6
4.2.3 E-stop 7
4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware) 7
4.3 Electrical System 7
4.3.1 Power System 7
4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS 7
4.3.3 Power Source 8
4.3.3 Power Distribution 8
5. Software Design 9
5.1 Software Strategy 9
5.2 Sensor Fusion 9
5.3 World Map and Path Finding 10
5.4 Motor Control Module 10
5.5 Autonomous Challenge Details 10
5.5.1 Image Processing 10
5.5.2 Camera Calibration 10
5.5.3 Hough Transform 11
5.5.4 Blob Detection 11
5.5.5 SICK Data and Vision integration 12
5.5.6 Goal Node Selection 12
5.5.7 Lane Following and Obstacle Avoidance 13
5.5 Navigation Challenge Details 13
5.5.1 Path Planning 13
5.5.2 Path Execution 14
5.6 JAUS Challenge Details 14
5.6.1 Process for Learning JAUS 14
5.6.2 JAUS Integration into the Design 14
5.6.3 Challenges Encountered 14
6. Performance Analysis and Estimate 14
6.1 Safety 14
6.2 Robot Performance 14
6.3 Reliability 15
6.4 Vehicle Cost Summary 15
7. Conclusion 15
8. References 16

1. Introduction

For the 2008 Intelligent Ground Vehicle Competition (IGVC), Lawrence Technological University (LTU) presents Viper, the product of a collaborative effort among a diverse and multidisciplinary group of LTU engineering and computer science students. Viper is a brand new robot, with cutting-edge hardware and software designed for 2008. It represents LTU's latest venture into developing a safe, reliable, and cost-effective autonomous vehicles that are rich in progressive technologies.

2. Innovations

2.1 Hardware Platform

2.1.1 Mechanical System

Viper is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design provides the most stability and maneuverability, and it allows for a zero-degree turn radius. Furthermore, the vehicle's front suspension system helps stabilize the drive-train and minimizes the shaking of the camera pole. The Viper body is a custom fiberglass mold, and features a welded aluminum chassis.

2.1.2 Electrical System

Viper makes use of a several new components in the electrical system design. It uses a Programmable Logic Controller (PLC) coupled with a Human Machine Interface (HMI) display to create the Electronic Diagnostic Fail-Safe System (EDFSS). The EDFSS allows the team to quickly determine the status of all electrical systems and to control all components. Viper also has a manual backup electrical system that is controlled via a single electric box.

2.1.3 Software

The software for Viper was written entirely in the C# programming language using the Visual Studio 2005 Integrated Development Environment (IDE). It includes a Sensor Fusion module to bring sensor information together into one component and an easily pluggable motor control interface for movement. Additionally, custom Local and World map software help Viper keep track of lane, obstacle, and GPS waypoint information, and they allow it to be less reactive to the immediate environment. Furthermore, Viper utilizes a GPS Breadcrumb system to keep track of recently visited locations, and to detect when it is heading in an incorrect direction on the course. Also, Viper software is JAUS level 3 compliant.

3. Design Process

3.1 Collaborative Team Organization

This year the team for Viper included members from Computer Science, Mechanical Engineering and Electrical Engineering departments. The team members are:

Gary Givental, MSCS – Team Captain	Jeremy Gray, BSEE – Hardware Team Captain
Philip Munie, MSCS	Angelica Sierra, BSEE
Shawn Ellison, MSCS	Doug Stevens, BSME
Brandon Bell, MSCS	Aaron Richards, BSME

Since having a larger team required more communication, team members used an online discussion forum, Subversion code repository, and email to communicate effectively. There were also weekly meetings scheduled to allow for status reports and to help resolve emerging issues.

3.2 Project Planning Process

The development for this year's IGVC effectively started in September 2007. **Figure 1** below describes the project plan followed by the team. For this year's competition, the team met every week to go through the design plan and to perform integration testing. Since an agile, iterative development process was used, the design emerged over time as different options for the software were explored. This project plan acted as a guideline to keep the team on track, and to make sure core timelines were met.

Task	Due Date
Initial Brainstorming	9/1/2007
Internal Discussion Forum Setup	10/1/2007
File Repository Setup	10/1/2007
High Level Hardware Design	12/14/2007
Mechanical Design	12/14/2007
Electrical Design	12/14/2007
High Level Software Design	2/1/2007
Robot Platform Buildout	3/1/2007
Electrical Wiring	3/1/2007
Core Software Component Development	4/1/2007
Software and Hardware Integration	4/15/2007
System Testing	5/15/2007
Written Presentation and Oral Report	5/30/2007

Figure 1. Project Tasks

3.3 Development Process

The team used agile development and set-based, concurrent methodologies for the software design in this year's competition. The set-based, concurrent development allowed the team to investigate several options for software modules and algorithms, and to find the best solution to design issues. Correspondingly, the Agile development approach allowed the team to avoid the well-known pitfalls of the linear "waterfall" style approach. These techniques enabled an emergent, iterative methodology that proved to be the perfect fit and allowed for the functionality and features of the software and hardware to evolve through continuous testing and integration. Moreover, the team found that discovering new requirements was an on-going process and that being able to adapt the software and hardware easily was pivotal to the team's success.

The team also used the L2Bot robot platform to test various software approaches quickly during the development process. These L2Bots are very simple, small laptop robots developed by LTU which consist of a webcam and two low-powered wheels that are all controlled by a laptop. Additionally, the team had access to the previous year's H2Bot robot for testing various software projects. These two testing platforms allowed for many feedback cycles and iterations to obtain a near-optimal design.

Finally, the team scheduled regular weekly meetings to present status reports, to discuss progress, and to indicate any road-blocks encountered by team members during development. Frequently, students teamed up to work on a particular problem in hardware or software to get it resolved quicker, and to leverage the "pair-programming" concept.

3.4 Testing Process

A test track was setup on LTU campus, complete with various kinds of obstacles and curves in the lanes to approximate the challenges of the real competition. This allowed team members to collect real world data and to discover failures in software logic.

3.4.1 Hardware Testing

Viper's electrical and mechanical systems were tested throughout the build process, and extensive field testing was performed once the robot was operational. While performing this field testing, the team discovered several issues that needed to be addressed. The biggest issue found was that the robot's center of gravity was shifted towards the front, so it tended to tilt forward when stopping sharply. This was addressed by adding additional weight in the rear of the vehicle.

3.4.2 Software Testing

The team performed extensive unit testing, systems testing, and integration testing of the code throughout development. As new functionality was developed, it was demonstrated to the rest of the team, and, when possible, tested on the robot. During Viper's build process, software components were tested on the L2Bot laptop robot platform, and on the H2Bot robot from previous year. As soon as Viper was operational, test runs were performed regularly on the test track at LTU.

3.4.3 System Integration

System integration testing was executed as quickly as possible once the development of the core components was complete. When new modules were available for general testing, they were immediately integrated into the overall system and used by all team members for testing. This modular design of the software components made it easy for new functionality to be plugged into the overall architecture. The team extensively tested the integration between software and all the sensor hardware, as well as motor control.

4. Hardware Design

4.1 Robot Structure

4.1.1 Chassis

Viper is a three wheel vehicle with front differential drive steering and a rear caster wheel. This design allows for zero-turn radius maneuverability and simplified motion control. The chassis has a minimum of 4 inches of ground clearance, and this allows the robot to climb hills and ramps with an approach angle of 23 degrees. The 3-D CAD model shown in the follow **Figure 2** illustrates the design of the chassis using the ¾ inch 6063 aluminum tubing, which gives the robot the capability of carrying all of the necessary components, as well as the designated payload required for the competition. The chassis design enables Viper to maintain its structural integrity throughout the rugged off-road course.

4.1.2 Suspension

Viper is designed with an independent suspension that absorbs impact in the wheels from the off-road terrain. This design provides ample stability to the camera, thus allowing for a smooth video image feed. The suspension model in **Figure 3** shows the coil spring over a tube style shock that is mounted to independent control arms. This suspension style

gives the robot 1 inch of travel in the shocks, and it greatly reduces the twist and roll of the chassis and components. Furthermore, the shocks are rated for 350 lbs per inch and have an adjustable rebound feature.

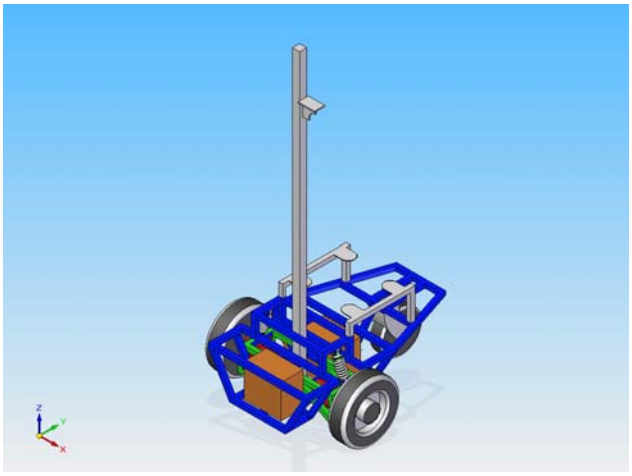


Figure 2: Chassis Model

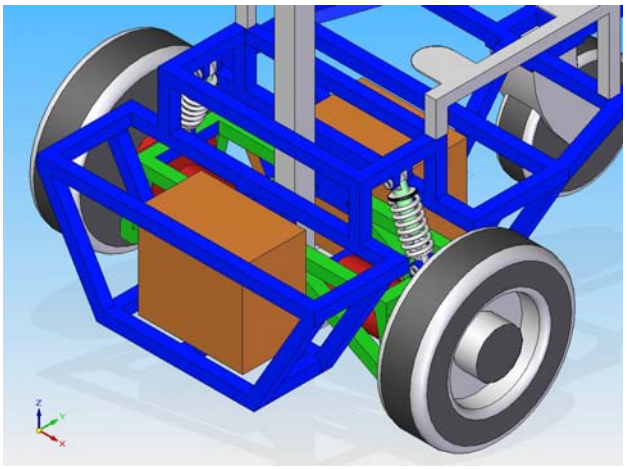


Figure 3: Front Suspension

4.1.3 Drive Train

Viper's design uses two 24 volt, permanent magnet type model NPC-T74 high torque motors with a 20:1 gear ratio that gives the robot the ability to generate up to 1480 in-lbs (123.3 ft-lbs) of torque and allows for a maximum speed of 10.4 MPH (at 248 RPMs). Thus, these motors give Viper plenty of power, and they eliminate the fear of premature motor failure. They ensure that Viper can easily navigate through rough terrain, and go up a 15 degree incline.

4.1.4 Body

The body that is covering and protecting Viper is fabricated from fiberglass, and its sleek design with smooth flowing curves and lines gives the feel of its namesake. It is designed to give easy access to the internal components through an easily removable top section, and it is fitted with electric fans to ensure that heat is removed from inside the robot. Furthermore, a tray that slides out of the right side of the fiberglass body allows access to Viper's internal laptop computer, and it can be used without removing the top section.

4.2 Motors and Electronics

4.2.1 Motor Control

Motor control is facilitated by a Roboteq AX3500 60A/channel controller with a C# serial port interface. The AX3500 provides velocity feedback measurements through optical encoders that are mounted on the motor shafts. Additionally, the E-Stop is wired to the controller's main drive power lines via a mechanical relay that can be triggered by the top-mounted kill switch or by the wireless control system.

4.2.2 Sensors

In keeping with the theme of modularity, all of Viper sensors (and controls) utilize RS-232 or USB 2.0 standard interfaces. **Table 1** summarizes the sensors used in Viper.

Sensor	Function
Optical Encoders	Motor shaft position feedback for servo control
NovaTel ProPack-LB DGPS Unit	Global positioning
Digital Compass/Inclinometer	Heading, roll, and pitch information
High Dynamic Range DV Camera	Capture field image stream
Sick Laser Measurement Sensor	Provides a 180 degree polar ranging array

Table 1. Vehicle Sensor Summary

4.2.3 E-stop

Viper is equipped with both manual and wireless (RF) remote emergency stop (E-Stop) capabilities. The ZX382164Y Remote Keyless Entry System manufactured by VILIANT is used as the wireless E-Stop component, and its door lock function is integrated into Viper's E-stop system. When this E-Stop is activated, it removes power from the drive train and engages both of the front wheel failsafe electromagnetic brakes.

4.2.4 Vehicle Alert System (JAUS Horn and Lights Hardware)

Viper includes an electrical module that can switch relatively large electrical loads (using relays) to the horn and light systems via commands sent over a USB interface. It is utilized by the vehicle's alert system, and it is activated by Viper's JAUS software.

4.3 Electrical System

4.3.1 Power System

For 2008, the IGVC team decided to use a new approach to the electrical system design. As a result, Viper has two parallel electrical systems: a logic control system and a manual control system. The logic control system consists of a Programmable Logic Controller (PLC) and a Human Machine Interface (HMI), and these components make up the Electronic Diagnostic Fail-Safe System (EDFSS). The manual system consists of toggle switches and fuses that allow the team to physically turn on and off each electrical component. The operating mode can each be selected by a keyed selector switch found on the main system control board.

4.3.2 Electronic Diagnostic Fail-Safe System - EDFSS

The EDFSS provides Viper with a nearly endless range of abilities for its current design and future design modifications. It makes the use of an external touch screen display that can be programmed to control each component's power and to display status information.

4.3.2.1 Status Monitoring and HMI Display

The Human Machine Interface (HMI) displays current battery life, and it displays warnings when system voltage is below a specified value. The EDFSS monitors the electrical system's voltage and battery life, and it has temperature sensors located within the power distribution control box of the vehicle. Thus, if a high temperature is detected, the EDFSS automatically turns on ventilation fans to help maintain a safe operating temperature. It also displays any high temperature alerts on the HMI and it completely shuts down the system if the temperature reaches a dangerous level.

4.3.2.2 External status indicator lights

During previous competitions, the team realized that easily visible power indicators for key component were needed during testing. The EDFSS solves this problem by controlling three indicator lights: the green light specifies all systems are ready, the yellow light signifies that Viper is operational, but not all systems are responding, and the red light indicates

that the system is not operational. Additional indicator lights are also available next to each manual switch on the electrical box. This functionality allows the IGVC team to quickly identify power issues during testing.

4.3.2.3 Security

Viper has two forms of security: a keyed selector switch that is used to control vehicle power and a vehicle security system that consists of onboard shock sensors, a siren, warning lights, and the "unlock door" function on the wireless E-Stop. Viper can be armed and disarmed by remote, and it emits a warning sound if unauthorized individuals attempt to tamper with it.

4.3.2.4 Manual Control

The manual system is a basic on/off component control. It acts as a backup electrical system if there are problems with the EDFSS system.

4.3.3 Power Source

Viper is powered by two 12V 50 Ah AGM batteries connected in series to provide a 24V electrical system. It uses a 24/12V DC/DC converter for the onboard 12V systems and an onboard charger with 110VAC interface for restoring battery power. **Figure 4** illustrates how power for the left and right motors is fed directly into the motor controller to maximize the motor power. However, the control power for the motor controller is regulated by the E-Stop and electrical control box.

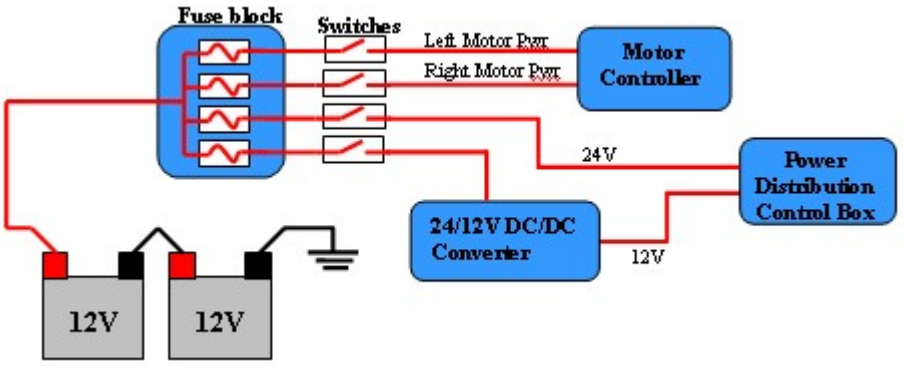
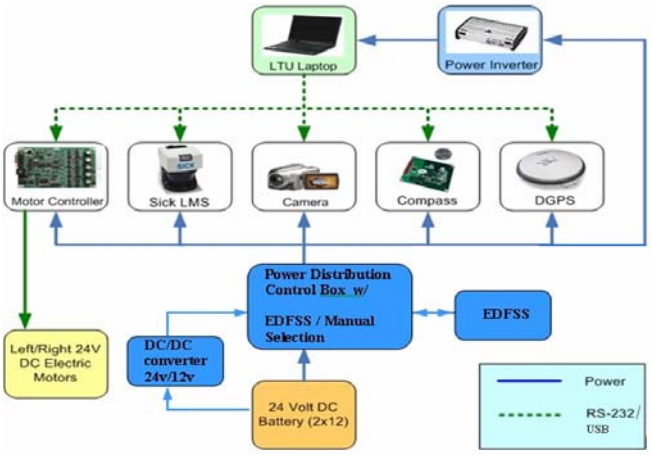


Figure 4. Power Source Schematic

4.3.3 Power Distribution

Viper is equipped with an all-in-one electrical control box. This power distribution control box contains the systems power distribution and emergency stop printed-circuit board (PCB), JAUS PCB hardware, wireless emergency stop PCB hardware, power control switches for each component, and the main system selector switch that controls the manual and EDFSS operation modes. It is designed to be self contained and removable from Viper, and it also offers additional testing and diagnostic abilities for system voltage and current ampere measurements.



The two main power connectors are automotive grade EDAC panel connectors, and they allow for an easy connection point for two wire harnesses, which route power to and from each hardware item and the EDFSS block. The power and communications control system schematic for Viper vehicle is shown in **Figure 5**.

Figure 5. Power and Communications Control System Schematic

5. Software Design

5.1 Software Strategy

C# and the Visual Studio 2005 IDE were chosen for all software development on the robot this year. C# is a powerful object oriented language, and the Visual Studio IDE allowed for rapid application development with easy to build visual interfaces. **Figure 6** shows the high level software architecture design.

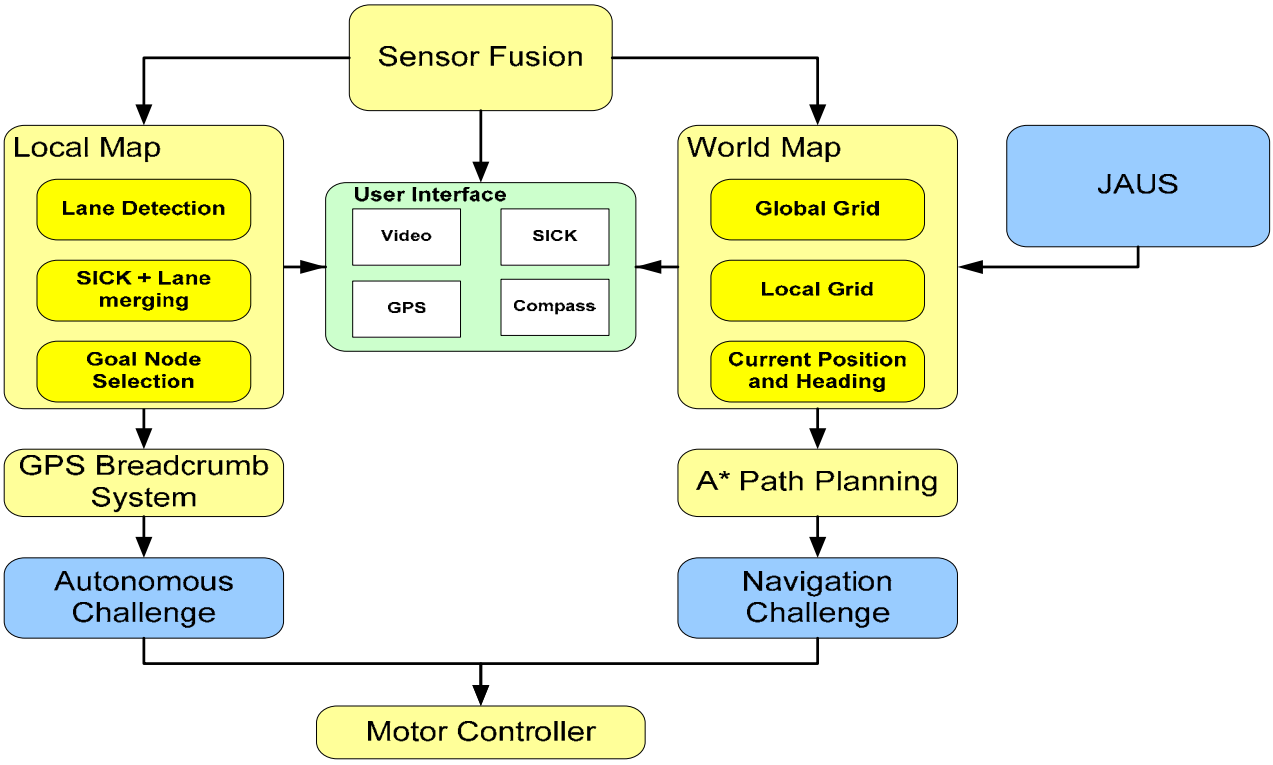


Figure 6. High Level Software Design

5.2 Sensor Fusion

For this year's competition, a Sensor Fusion Module was developed to collect data from all the hardware sensors, and to present this data to all subscribers in a consistent manner. This allows subscribers to access sensor data without knowing the details of hardware layer implementation.

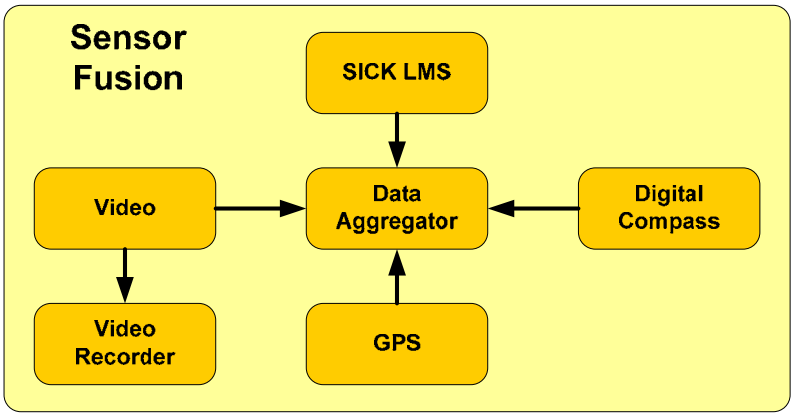


Figure 7. Sensor Fusion

5.3 World Map and Path Finding

The world map plots a geographic region to a Cartesian coordinate plane. The total size of the Cartesian map is determined by the size of the area to be represented and the desired resolution. Generally, a resolution of ½ meter has proven to work well during testing. Each point in the geographic region can be mapped to a cell in the Cartesian map, and vice versa, through the use of a conversion object. In this way, a simple A* path finding algorithm can be used to find the shortest path between any two given points.

5.4 Motor Control Module

The motor controller module was designed to be easily plugged into any of the team's projects. Once initialized, this module can accept simple operation commands for speed and direction, which it then translates into the ASCII communication strings that are sent to the motor controller via a serial connection. This module allows team members to have simple access to the hardware without knowledge of the motor controller's underlying communication protocols.

5.5 Autonomous Challenge Details

The main algorithm for the autonomous challenge consists of using computer vision to identify lane information and SICK data output to detect obstacles. To accomplish this, the camera is first calibrated so that the vision processing can map lanes onto a grid alongside the SICK data, and then a goal node selector algorithm is used to find the appropriate heading. Additionally, a GPS Breadcrumb utility, which uses GPS location information and compass headings to store a list of recently visited locations, is used to detect if the robot has become turned around. This helps the robot avoid going backwards on the course.

5.5.1 Image Processing

For this year's competition, several different image processing approaches were evaluated for handling lane detection and virtual pothole detection. Eventually, several filtering techniques were found to be ideal for this detection. To do this processing, the image is first scaled down from the original 160x120 pixel camera image because this reduces the pixel count and speeds up processing. Next, the image is converted to black and white and this filter can be adjusted by a threshold variable until the best conversion is found. Next, blob detection is used to filter out noise pixels that are not considered to be lanes. Once this noise has been removed, a Hough Transform is applied to find any partial lanes in the image. Next, after all of the image processing is performed, the lane pixels are mapped onto a local grid, which represents both the lane information and the SICK obstacle data. Finally, a local goal node is found between the lanes and the obstacles and this is used to set as an immediate heading for the robot.

5.5.2 Camera Calibration

The camera calibration is responsible for synchronizing the lane data with the SICK obstacle data. It works by calculating the distance of each pixel in the vision's captured image, and it uses the SICK as the reference point. It is accomplished by measuring the distance of predefined positions on the captured image and interpolating the remaining positions on the image.

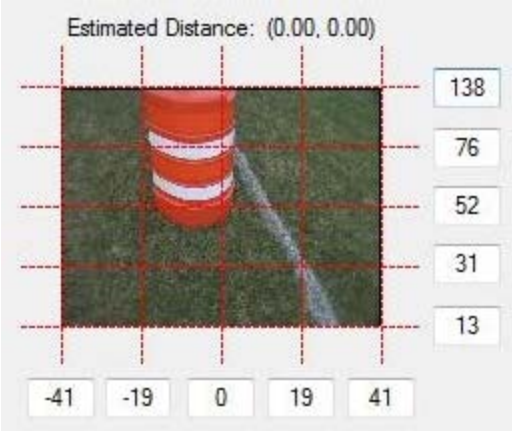


Figure 8. Camera Calibration Tool

5.5.3 Hough Transform

A Hough Transform was used to identify lanes in the captured video. It is primarily used for filling in a lane when only a partial (dashed) one exists, which can occur during image capture because of the obstacle course design or because of improper image processing. See **Figure 9** for an example of the Hough Transform filling in missing data. This transform is applied to two separate images that are created when the captured image is split vertically. This image splitting forces the detection of a lane on both sides of the image, and this becomes important in cases where both of the lanes exist on the image at once. This is because each lane needs to be evaluated for missing data, and the use of a single image does not guarantee the Hough Transform will recognize both lanes. This lack of recognition becomes extremely apparent when the second lane's intensity is less dominant.

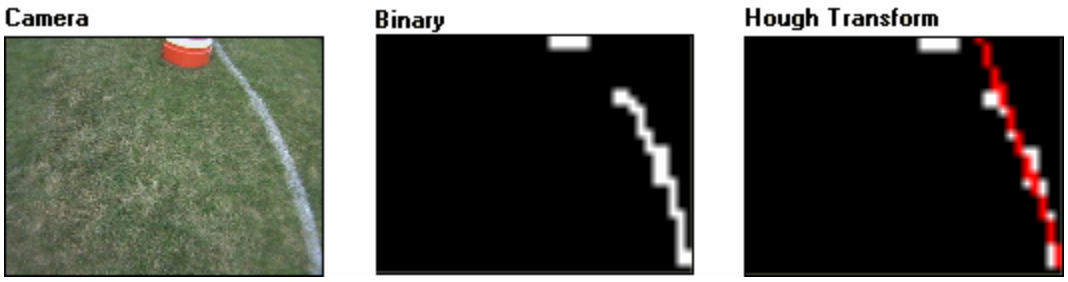


Figure 9. Hough Transform Filling In Missing Data

5.5.4 Blob Detection

As the image is scanned, each pixel is examined to determine its color. If the pixel is not already mapped to a blob, a new blob is created to hold it. Each surrounding pixel is then examined to determine if it is the same color. If it is, it is also added to the blob. If the pixel already belongs to a different blob, the two blobs are merged into a single blob. Once this process is finished, each contiguous region of color in the image will have a corresponding blob object, and these blobs are useful for several reasons. First, they allow identification of all regions of color and this can help the detection algorithm filter out blobs that are too small (and thus are just noise). Secondly, blobs of "non-obstacle" space can be identified, and this ensures that the detection algorithm selects the goal node that is in the same region of the robot. Thus, the robot will not cross a lane to get to the local goal location.

5.5.5 SICK Data and Vision integration

The SICK LMS software module handles initializing the sensor to the appropriate detection mode, starting the continuous scanning mode, and interpreting the sensor's data into an array of Cartesian coordinate data points. It was designed to be easily plugged into various team projects, and it is fully integrated with the vision sensing software. Additionally, a sensor data point map interface was also developed to ensure that the hardware was functioning properly. **Figure 10** shows a screenshot of this hardware testing interface.

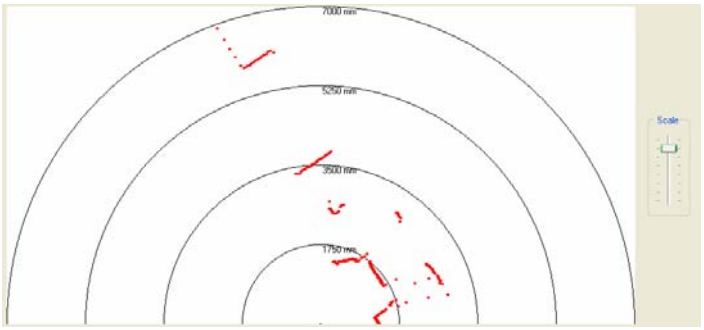


Figure 10. SICK LMS Data Point Map

As the SICK data is collected, it is mapped onto a local grid that already contains the visual lane information from Viper's camera. **Figure 11** demonstrates this integration.

Figure 11. SICK Data Points Mapped onto the Local Grid

5.5.6 Goal Node Selection

Goal node selection is an important aspect of the autonomous navigation challenge, and it relies on the local grid that is built by combining the vision module's lane detection with the SICK obstacle detection data. Once the grid is built, it is then transformed into an image, and a goal location for the robot's heading must be found. This is handled by finding the largest gap between obstacles and lanes. Ideally, this gap is at the top of the image and as far away as possible from the robot. However, if a lane bisects the image, then the gap is calculated from either the left or right side of the image. The images below demonstrate the goal node selection algorithm. The goal location is the orange dot in these images.

Figure 12. Local Goal Node Selection Test interface

5.5.7 Lane Following and Obstacle Avoidance

Once both visual lane information and SICK data is mapped onto a local grid, the goal node selection algorithm is then used to find the largest gap in the image, where it plots a goal node in the center of it. If the largest gap on the map is not larger than the robots size, the robot moves backward and searches for a path that it can fit through. When a gap is found to be big enough for the robot, the goal node is selected, and the robot then moves in the direction of the goal node.

5.5 Navigation Challenge Details

The primary algorithm of the navigation challenge uses the GPS, compass, and SICK laser range finder sensors in conjunction with dead-reckoning information available from the motor controller (derived from optical encoders on the motor shafts) to move Viper through the course to the various GPS checkpoints.

5.5.1 Path Planning

The navigation challenge module relies heavily on the World Map to plan its paths. A navigation object sits on top of the map and handles the generation and updating of the current path as information about the world, such as obstacle locations, becomes available. The navigation object relies on a waypoint scheduler object to determine the order in which the waypoints are to be visited. **Figure 13** shows a test World Map interface with obstacles and a path from the current position to the first waypoint.

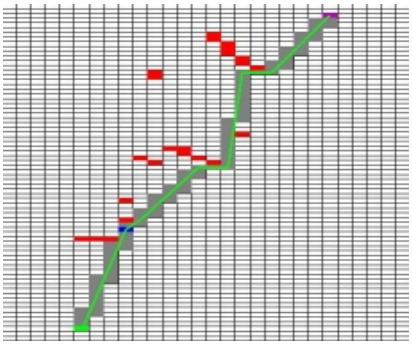


Figure 13. Navigation World Map

5.5.2 Path Execution

The navigation object supplies the current sub-destination, which is the furthest point along the current path that can be reached by traveling in a straight line. The heading to the sub-destination is determined from the World Map and the information from the compass is used to orient the robot along this heading. As new obstacle information is discovered, this sub-destination is updated.

5.6 JAUS Challenge Details

For this year's competition, the JAUS handling code has been adapted from the previous year's entry. The new JAUS module handles Level 2 and Level 3 JAUS challenges. The Level 3 challenge is handled by simply swapping out the standard waypoint scheduler for the JAUS waypoint scheduler. This scheduler uses the existing JAUS module to receive the waypoint information and the existing navigation challenge module to navigate the waypoints. The JAUS waypoint scheduler is the only additional piece of code required.

5.6.1 Process for Learning JAUS

To learn about JAUS, the team employed resources available on-line and the JAUS reference architecture specification provided by Jauswg.org. Additionally, the team was provided with the JAUS Compliance Tool Suite (JCTS), which allowed team members to send JAUS messages and helped them understand how these messages were composed. This tool was essential for testing the team's JAUS module.

5.6.2 JAUS Integration into the Design

To integrate JAUS into Viper, a module was created to handle listening for JAUS messages sent via the UDP protocol. The received messages are checked for validity and if the JAUS ID matches that of the vehicle, then the commands are parsed and passed back to Viper's main controller modules. However, only the commands described in 2008 IGVC rules are supported at this time. To support Level 2 of the JAUS challenge, the JAUS module can also fetch information from the waypoint navigation module and send out a properly formed JAUS message containing the waypoint information.

5.6.3 Challenges Encountered

The main challenge in implementing this module was in the understanding of how JAUS messages are composed and designing the code to parse out the needed information. The provided JCTS tool was helpful, but it had a significant learning curve in order to use it properly. Additionally, the integration of this system with the rest of the Viper was also a non-trivial task.

6. Performance Analysis and Estimate

6.1 Safety

Viper safety features include a manual and remote E-Stop, failsafe brakes, and a capped maximum speed of 5 mph.

6.2 Robot Performance

Vehicle performance was estimated based on the design parameters. These same attributes were measured after integration as shown in **Table 2**.

Attribute	Design Prediction
Maximum Speed	4.2 mph
Climbing Ability	2.5 inch curb
Nominal Power Consumption	500 watts
Battery Operating Time	4 hours
Waypoint accuracy (with Omnistar)	.1 to .8 meters

Table 2. Performance Analysis

6.3 Reliability

Viper utilizes dual modular power sources and a thoroughly tested drive train design that performed well in the 2007 competition. Additionally, the robot uses a rigid aluminum frame with extensive weather-proofing. Furthermore, extensive testing has been done for both the hardware and the software components. Specifically, regression and system testing was done on each module to ensure proper functionality and integration with the hardware components.

6.4 Vehicle Cost Summary

Table 3 summarizes the total material cost for the Viper vehicle.

Component	Retail Total Cost	Team Cost
MPC Laptop	$2,215	$0
Sick LMS 291-S05 LMS	$7,000	$0
NovaTel ProPak-LB DGPS & GPS-600-LB Antenna	$4,567	$2,700
24 V NPC-T74 Motors (2)	$648	$648
Panasonic PV-GS500 digital camcorder	$800	$0
Miscellaneous Electrical Hardware	$820	$820
PNI TCM2-20 digital compass/inclinometer	$769	$0
Roboteq AX3500 Dual Channel Motor Controller	$395	$0
Main Battery 12 Volt 50 Ah AGM	$323	$323
Promariner Promite 5/5 Dual 12V Battery Charger	$110	$110
DC-to-DC Converter	$118	$118
EDFSS Hardware Components	$2,308	$780
Hollow Shaft Optical Encoder Kit (2)	$130	$130
Chassis Materials	$320	$320
Miscellaneous Hardware (nuts, bolts, etc…)	$200	$200
14" Tire & Wheel Assembly (2)	$186	$186
Rear 300 Lb Capacity Caster Wheel	$22	$22
Total	$20,931	$6,357

Table 3. Vehicle Cost Summary

7. Conclusion

Viper continues the LTU tradition of innovation and continuous improvement. Viper features a distinctive EDFSS system with HMI interface, as well as a backup electrical system, stable camera mount and shock absorbers, and a modular, innovative software design. All software and hardware components are easily swappable and can be upgraded in future designs. Viper's unique design and state-of-the-art software technology set it apart from other competitors in the 16th annual Intelligent Ground Vehicle Competition.

8. References

[Ward 1994] Ward, A.C., Sobek, III, D.K., "Toyota and Set-Based Concurrent Engineering," 1994 ASME Design Theory and Methodology Conference Proceedings, Minneapolis, MN

[**http://www.jauswg.org**] The Joint Architecture for Unmanned Systems. "Reference Architecture Specification. Volume 2," 2004.

Poppendieck, M., Poppendieck, T. "Lean Software Development: An Agile Toolkit". Addison-Wesley Professional (May 18, 2003)

[**http://en.wikipedia.org/wiki/Hough_transform**] Hough Transform. March 2008.

Nilsson, N. J. , "Artificial Intelligence: A New Synthesis". Morgan Kaufmann Publishers, 1998, San Fransisco, CA.