ATHENA

2008 Intelligent Ground Vehicle Competition

Required Faculty Advisor Statement

I certify that the engineering design of the vehicle described in this report has been significant, and that each undergraduate team member has earned the equivalent of six semester hours of senior design credit.

[illegible]

Puneet Singla, Assistant Professor
Mechanical and Aerospace Engineering, University at Buffalo

University at Buffalo
The State University of New York

1. Introduction

UB Robotics' Unmanned Vehicle Team (UVT) at the University at Buffalo has designed and fabricated an autonomous unmanned ground vehicle to compete in the 16th annual *Intelligent Ground Vehicle Competition* (IGVC). Several undergraduate students in the fields of mechanical, electrical, and computer engineering collaborated to produce a rugged and robust vehicle platform that can meet or exceed customer requirements. Modularity in both the software and hardware allow seamless system expansion and effortless integration of additional components.

2. Innovations

The heavy use of TCP/UDP sockets for interfacing the various software modules such as GPS and motor control eased integration of the JAUS requirements since the capability of sockets was already built into the system. Only a module that decodes messages was needed to integrate the JAUS into the platform. The use of multithreading to enhance performance on multi-core processors allowed for very intensive image processing which would have caused slowdowns and increased data latency on single core processors. The diagnostics utility allows for data logging and remote control from any computer connected to the network that the robot is connected to using its Wi-Fi adapter. In addition, a terminal emulator, such as PuTTY or telnet, found on most computers can be used to connect to the diagnostics socket and retrieve data.

3. System Design

3.1 Team Structure

There were 11 active members working on this project from different engineering disciplines. These members focused on specific areas of the project which best matched their individual skill sets. Table 2.1 lists the team roster.

Name	Major	Year	Team Role
Mike DiSanto	EE	4th Year Undergrad	Team Leader
Tim Montgomery	EE	4th Year Undergrad	Hardware Team
Dan Muffoletto	EE	3rd Year Undergrad	Hardware Team
John Amend	ME	4th Year Undergrad	Mechanical Team Lead
Kurt Cavalieri	MAE	4th Year Undergrad	Mechanical Team
Brett Cotton	MTH	4th Year Undergrad	Mechanical Team
Mark Tjersland	CEN/EE	4th Year Undergrad	Software Team Lead
Jake Joyce	CSE	2nd Year Undergrad	Software Team
Shajan Thomas	MAE	4th Year Undergrad	Software Team
Brain O'Conner	CSE	3rd Year Undergrad	Software Team
Jesse Evers	EE	4th Year Undergrad	Software Team

Table 3.1: Team Roster

The team was broken down into a mechanical team, software team, and hardware team. Each sub-team focused on specific tasks that were distributed during team meetings. Figure 2.1 illustrates the sub-team hierarchy.

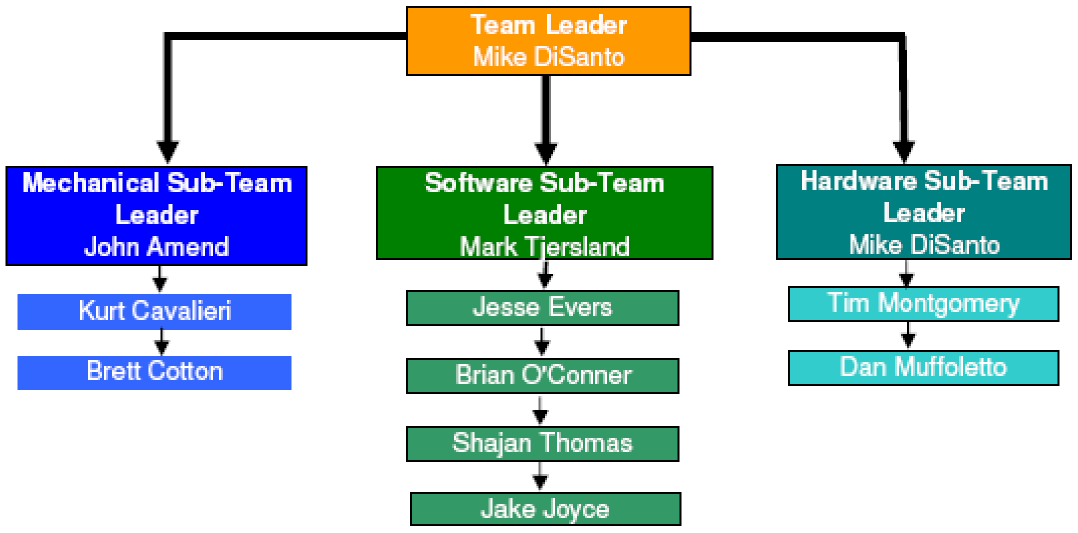


Figure 3.1: Team Member Hierarchy

3.2 Design Process

UB Robotics' UVT used different design techniques and project management tools to aid in system development. A private Google® group account was created specifically for this project. This allowed seamless, forum style communication between members and sub-teams as well as a place to store design files and meeting minutes. The first step in the design was to review the rules and system requirements. We then created a problem statement and system requirements document. Throughout the entire design process these documents were referenced to ensure that we were including the necessary customer requirements and not wasting time on features that may or may not be desirable. Since this was our first year competing in this competition, a lot of time was spent researching what sensing hardware would be needed in order for the vehicle to be able to navigate autonomously. With a high level architecture mapped out, each sub-team designed their respective components to satisfy the requirements of the system as a whole. This method proved to work well as team's worked in parallel and was able to communicate and track each team's progress through the Google® group.

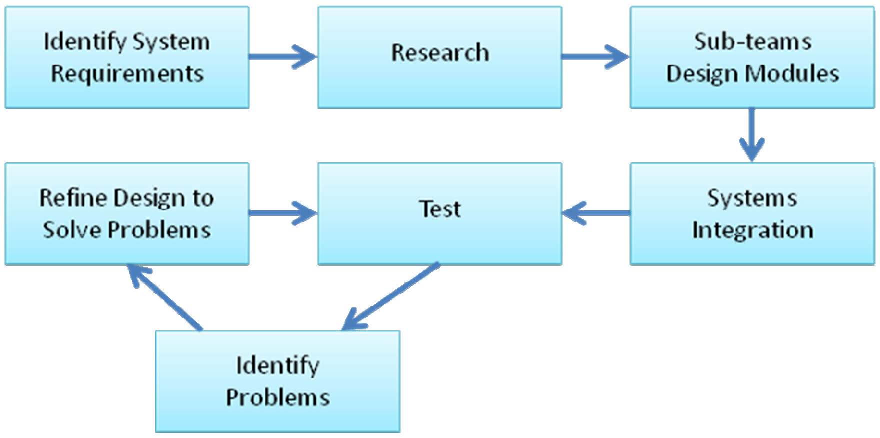


Figure 3.2: System Design Flow

4. Mechanical Design

4.1 Design Strategy

The mechanical design was driven by two main factors: cost and performance. We knew from the beginning that the average expenditures for IGVC teams were beyond our means, so we made every effort to reduce cost or obtain sponsorship where possible. Our primary attempt to reduce cost was by purchasing two electric ATVs and then cannibalizing them for parts. The ATVs were chosen over other possible vehicles because one of our goals was to make a vehicle that could realistically be used in an off-road setting. We felt that the competition had been lacking in that regard, with vehicles having difficulty traversing the fairly mild terrain in years past. Therefore these ATVs, with approximately 12in diameter knobby tires, were a logical purchase. **Error! Reference source not found.** is a photo of the ATVs with the plastic body removed from the unit on the right. The ATVs (model XA-750) were purchased from X-treme Electric Scooters, a sub-company of Alpha Products International, Inc. Their size was somewhat unusual, being

too powerful for a child, but too small for an adult to ride comfortably. Each ATV cost us $599, and we were able to use the motors, batteries, frame, wheels, axles, bearings, sprockets, roller chain, and horn. Pro/Engineer Wildfire was used to model the ATVs and design the final vehicle. The main performance specifications for the final design were that it was within the allowable size requirements, was mechanically governed to less than 5mph, and was able to accommodate everything it would need to contain (sensors, computer, payload, etc.). Our electrical team also specified that the vehicle be skid steer with as close to a square wheel pattern as possible. They felt that skid steering would be easier to control, especially for our inaugural entry in this competition.

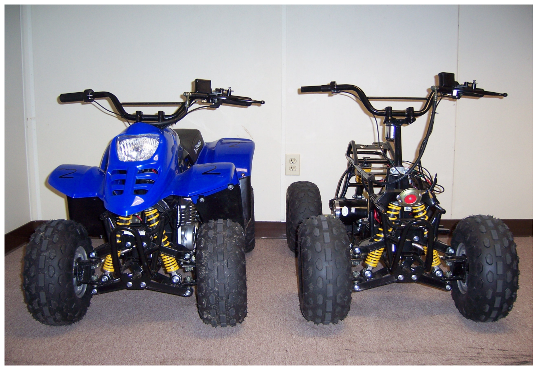


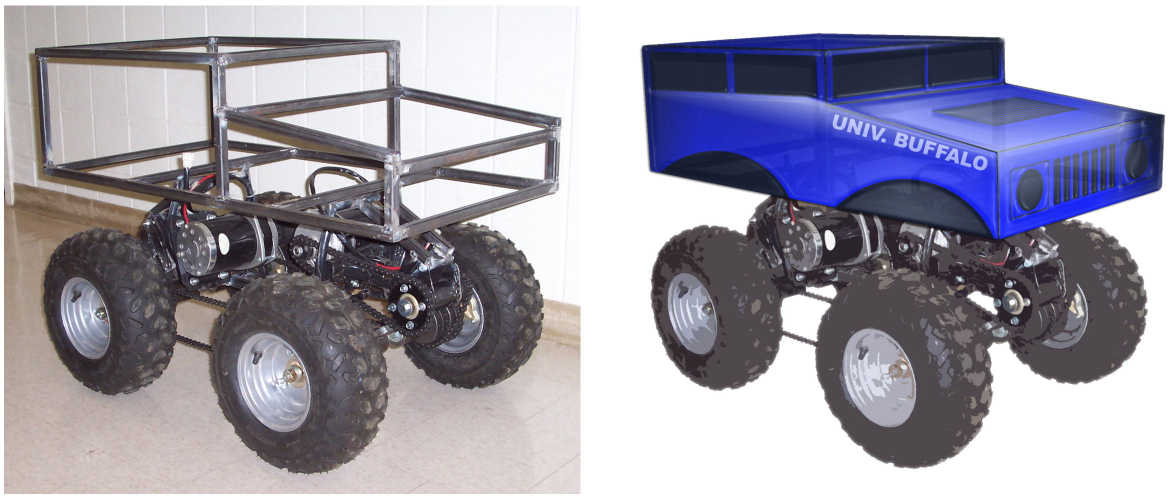
Figure 4.1: Electric ATV's

4.2 Performance

Initial testing of the ATVs yielded promising (and fun) results. The 750W (~1hp) brushed DC motor driving each ATV was powered by three 12V, 12Ah batteries, wired in series. Although uncomfortable for an adult to ride, each ATV was easily able to transport our club members. Testing was conducted on campus green space that was not particularly well manicured. We measured top speed of the ATVs at just below 10mph; the batteries lasted approximately 30 minutes under continuous testing; and the vehicles had no trouble carrying our 150lb team members up inclines greater than 30°.

In the final vehicle shown in Figure (a), the two ATVs have been cut in half and the rear ends have been spliced together. New, secondary axles were fabricated that allowed for chain-driven skid steering. The motor output, already geared down by ½ in the original

ATVs, was geared down by an additional ½ in the final design to set the top speed to less than 5mph. These modifications give Athena incredible performance statistics, including roughly four times the torque of either ATV alone. Completed, Athena has a top speed less than 5mph, an available 2 hp, and although torque was never specifically measured, it is much more than enough to power through anything it might encounter in the IGVC.



(a) (b)

Figure 4.2: Athena's frame and drive train (a) and conceptual body art (b)

5. Hardware

Athena uses a suite of sensors to perceive the environment. Vision is accomplished by way of a single 3CCD Panasonic digital color camera. Additionally, a Hokuyo laser range finder is on-board performing vision duties in the form of obstacle detection. Localization is achieved using a Novatel Propak V3 DGPS system, a PNI 3-axis digital compass with pitch/roll compensation, and wheel odometers. All of the sensors are interfaced to a Dell Latitude D830 laptop with an Intel Core2 Duo® processor running Windows XP®. The sensor data is fed into an Extended Kalman Filter (EKF) which performs the sensor fusion and determines the current position of the vehicle.

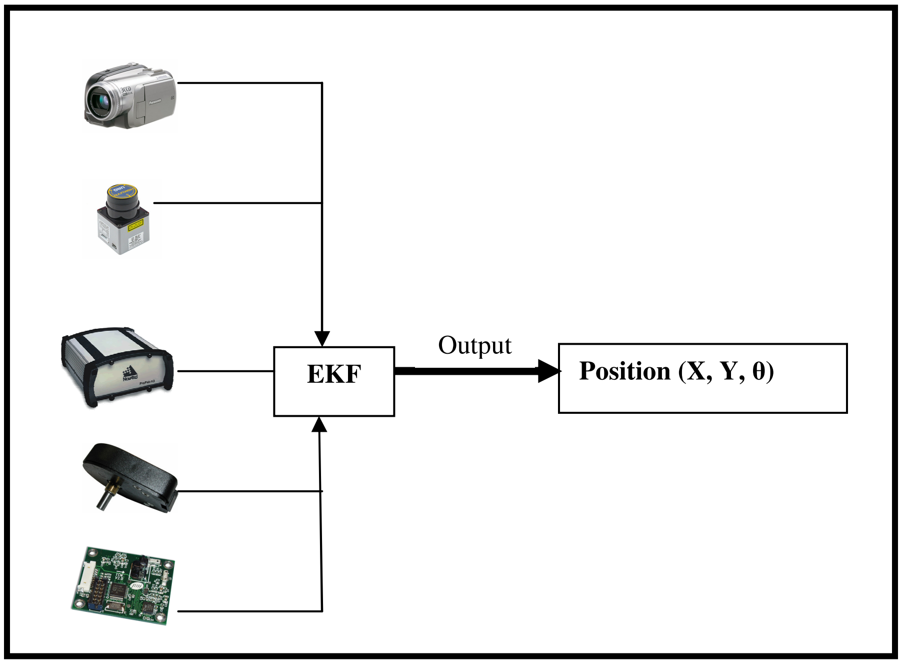


Figure 5.1: Hardware Data Flow

Component	Accuracy/Resolution	Refresh Rate
GPS	10cm	20 Hz
Camera	720 x 480	10 FPS
Laser Range Finder	4m Range, 240° FOV, 0.1°	10 Hz
Digital Compass	Heading: 0.1°, Pitch/Roll: 0.2°	8 Hz
Wheel Odometers	512 CPR	10 Hz

Table 5.1: Sensor Performance Characteristic

6. Electrical System

The main electrical system is derived from 3, 12Ah 12V batteries configured in series for a positive 36V rail. The motor controllers are connected directly to the 36V unregulated rail. A custom designed switch mode power supply (SMPS) is used to switch the 36V raw DC voltage to a more usable 12V rail. The custom power supply also is equipped with a 5V linear regulator to supply a positive 5V rail. The power supply has two independent 12V/5V regulator circuits. Currently only one of the circuits is being used, while the other one acts as an emergency backup or for future

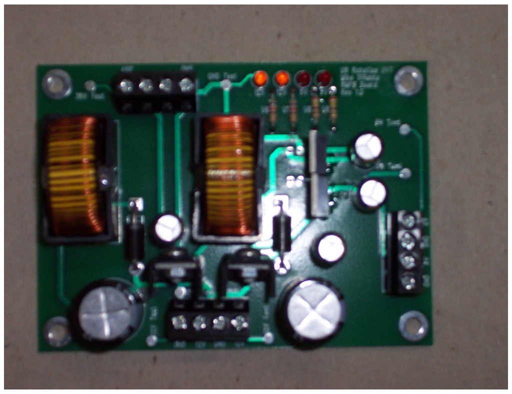


Figure 6.1: Switch Mode Power Supply

hardware expansion. The board is capable of delivering up to 6A of 12V power and 2A of 5V power. The small footprint (3” x 5”) allows the vehicle to have multiple power supply modules if needed for further expansion without sacrificing too much space.

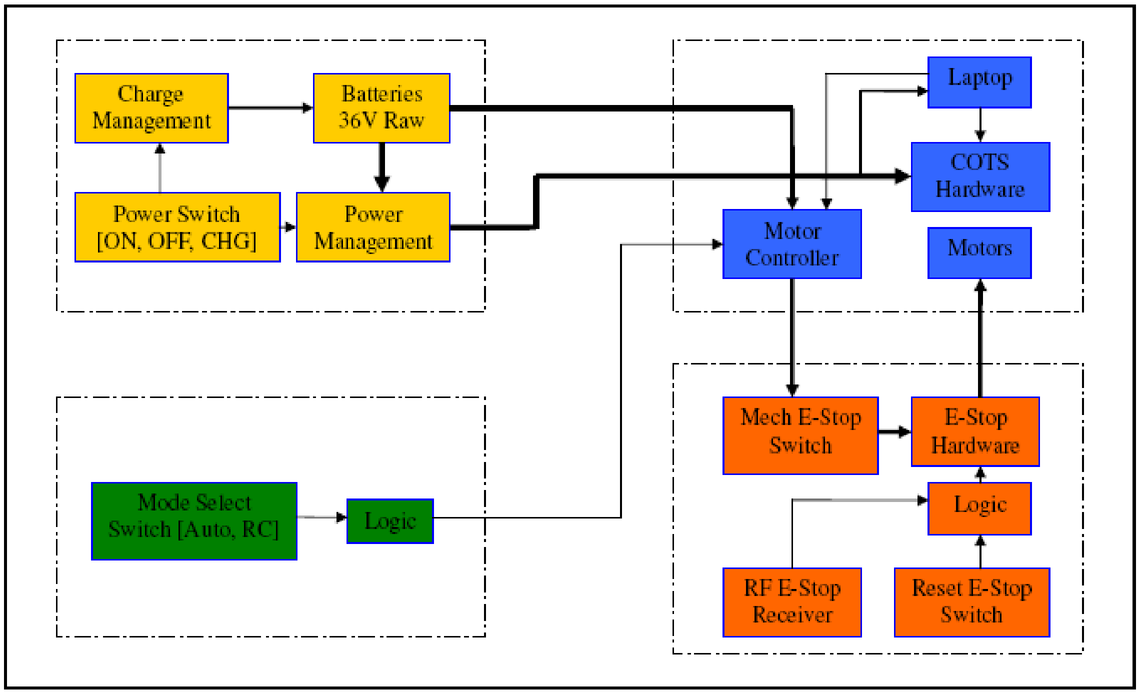


Figure 6.2: Electrical System Block Diagram

The GPS system is powered off of the 12VDC rail. The digital compass and LIDAR units are powered off of the 5V rail. The wheel odometers are powered directly off of the motor controllers which has its own power regulation circuitry.

6.1 System Monitoring

The Roboteq motor controllers that are used are capable of monitoring the electrical system parameters of the system. The controller can report the heatsink temperature of the output MOSFETs as well as the current. It can also monitor the battery voltage and the motor voltage. The software can take this data and take the necessary actions to prevent overheating due to excessive ambient temperature environments or excessive current draw. Additionally, battery voltages can be monitored and sent wirelessly over the diagnostics client to an operator.

6.2 E-Stop System

The vehicle is equipped with both a mechanical push button style E-stop as well as a wireless E-Stop. Both E-Stop systems are wired to the E-stop port on the motor controller module. Pulling the line low brakes the motors immediately and brings the vehicle to a stand still until a command is issued to the motor controllers to reset. The wireless E-stop is a custom made PCB with an embedded LinxRF® module. The module operates at 430MHz and has a max range of 1000m. The vehicle can also be E-stopped over the 802.11g wireless diagnostics client.

6.3 Power Consumption

Component	Operating Voltage	Power Consumption
GPS	12V	2.5W
LIDAR	5V	2.5W
Camera	7.9V	5.5W
Digital Compass	5V	110mW

Table 6.1: Hardware Power Consumption

Both the camera and the laptop run off of isolated battery supplies and are separate from the electrical system. They will run continually for an average of 4 hours. The rest of the vehicle also has an estimated average run time of 4 hours.

7. Software Design

7. 1 Software Environment

The entire software platform was designed around the Java Standard Edition 6.0 Application Programming Interface (API). For serial communications support, the Java Communications 2.0 API was used. The Java Media FrameWork 2.1.1e API was used to enable camera connectivity. Code is executed using the Java HotSpot Virtual Machine.

7.2 Version Control

The Subversion version control system was used to maintain current and past versions of code. Use of version control ensures synchronization between versions stored on individual software team developments computers and the current version stored on the repository. The histories maintained by the repository are useful for reverting code if unexpected bugs crop up or retrieving sections of code that may have been deleted in newer versions.

7.3 Integrated Development Environment (IDE)

For software development, the open source Eclipse IDE was used. The debugging features, refactoring support, and plug-in integration made it an ideal tool for rapid software production and testing. The open source Subclipse plug-in was used to manage the Subversion repository from within the IDE, seamlessly merging development and version control.

7.4 Software Architecture

The software architecture is designed around a central server that mediates between low level hardware interface software and high level processing and decision making software. The hardware modules read raw data from the physical sensors and report it to the server. Processing modules such as vision and the Kalman filter query the data from the server and perform their operations. The processed data is fed into the path planning and motor commands corresponding to its decisions are sent to the motor controllers to move the robot. A diagnostics utility is available to view raw and process data to aid in debugging and remote operation.

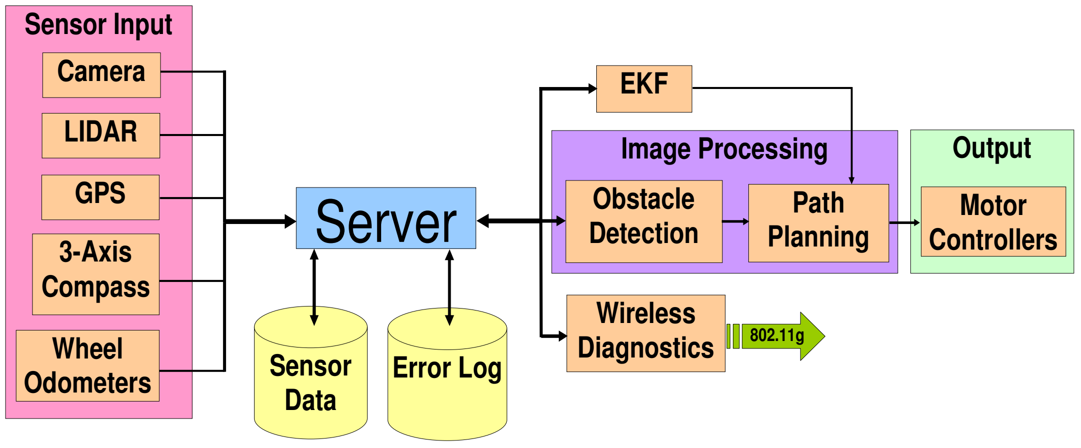


Figure 7.1: Software Architecture Diagram

7.5 TCP/UDP Sockets

Sockets were used to provide communication between the hardware interface software and the high level logic and control software. The use of sockets allows the low-level hardware code to be written in any language since Java is not ideal for platform dependent hardware interface and a language such as C is better suited for these applications. Another benefit of using sockets is that wireless networking such as Wi-Fi can be used to run diagnostic tools remotely during testing of the robot.

7.6 Concurrency and Parallelism

As the size reduction and its corresponding performance gains in processor cores becomes increasingly difficult, most new processors are designed with multiple cores to execute in parallel to increase speed. In order to exploit the increased performance of multi-core or multi-processor systems, multiple discrete units of execution, or threads, are created. In Java, the threads are explicitly created by the programmer but are then managed by the virtual machine which then will decide on which core/processor that thread should be executed on.

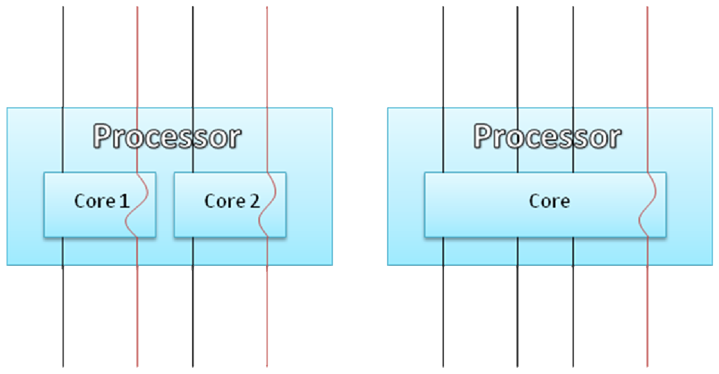


Figure 7.2: Multi-core (left) and single core (right) processor executing multiple threads (active threads in red and sleeping threads in black)

7.7 Path Planning

Path planning received data from the Kalman filter to determine the robot's current position and vision/LIDAR to find obstacles. The longest path that avoided obstacles and had the most forward movement was chosen. Mapping was used to store locations of known past obstacles in order to prevent problems if the robot needed to back up and to prevent unnecessary backtracking away from the goal. For planning of the Navigation Challenge path, an algorithm was used to solve the problem of finding the most optimal path from the given waypoints, a problem known as the Traveling Salesman Problem. The algorithm used was able to optimize a ten waypoint path in roughly 440 milliseconds.

7.8 Vision

The Java Media Framework was used to grab and decode frames from the 3CCD camera and export them as Java 2D images. These were then processed with custom software that performed operations such as edge detection, density analysis, and color filtering. Short circuit evaluation was used to increase performance by using computationally cheap operations such as color filtering on images to try to identify objects and then using more intensive operations such as edge detection to find objects if those fail.

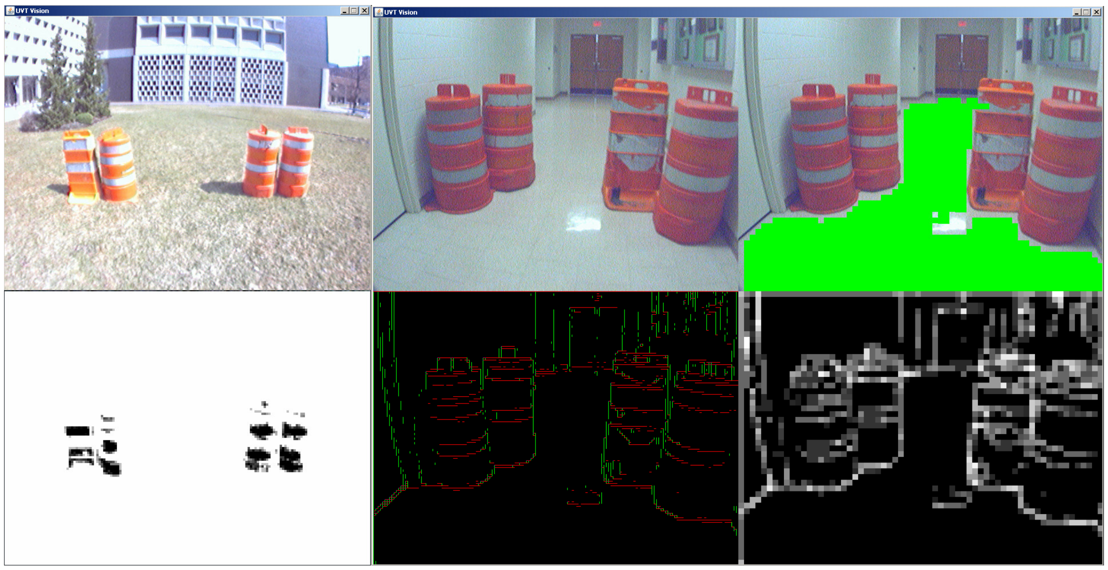


Figure 7.3: Examples of Color Filter (left) and Edge Detection (right)

7.9 Kalman Filter

It became apparent early on in the design phase of the project that we required a way to get an estimate of the vehicle's current position based on multiple sensor inputs. A Kalman filter provides a standard way of integrating our sensor data. It also provides a way of assessing the validity of sensor data based on a computer model of the physical system. The first part of the Kalman filter is the model of the vehicle. The **Error!**

Reference source not found. shows how our vehicle was modeled. The robot makes use of skid steering and is simplified as having two wheels. The forward velocity is simplified as the average velocity contributed by both wheels.

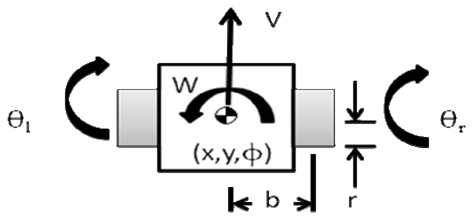


Figure 7.4: Vehicle Model

$$V = \frac{r}{2}(\dot{\theta}_l + \dot{\theta}_r)$$

The rate at which the vehicle turns is described with the following equation.

$$W = \frac{r}{b}(\dot{\theta}_l - \dot{\theta}_r)$$

The motion of the robot can be described with the following equations.

$$\dot{x} = V\cos\varphi$$

$$\dot{y} = V\sin\varphi$$

$$\dot{\varphi} = W$$

From the above equations we can derive the state space of the system that will be used in the main Kalman filter equations.

In the initial phases of the project, using the least squares method for processing sensor data was considered. This proved useful if the sensor data was reliable. The Kalman filter would be less affected by situation where there is high noise. The solid line represents a line that is defined. The dots are points randomly generated around the line. The dotted line is an attempt to reconstruct the solid line from the dots using a batch least squares method.

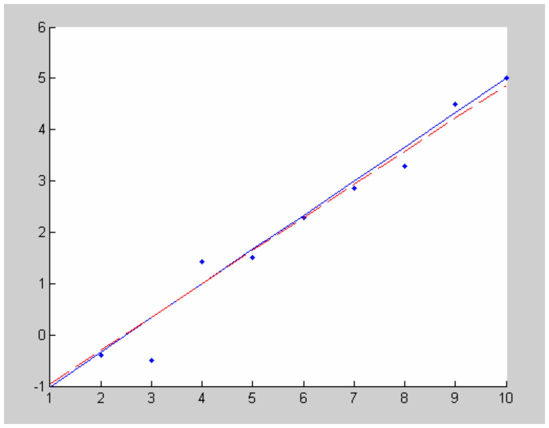


Figure 7.5: Least Squares Simulation

8. Joint Architecture for Unmanned Systems (JAUS)

8.1 Learning Process

The team learned the JAUS design and specifications from the material provided at www.jauswg.org under the “Current Documents” section. The communication protocol definitions specifically were studied in order to decode the JAUS messages into meaningful data.

8.2 JAUS Integration

Integrating the JAUS instruction messages into the software was a relatively simple process. Using the existing TCP/UDP socket capability already built into the system, a JAUS message processing module was created that listened for JAUS message packets on a specified port. The messages received and decoded from the datagram packet would then be translated into native commands that would place the robot in autonomous mode, turn on or off an accessory, and report the robot’s current position as required in the level 2 of the JAUS challenge.

8.3 Problems

One major problem for the JAUS challenge was that there was no way to verify that the testing done with a dummy OCU would accurately reflect what was to be expected at the competition. Since there were no sample JAUS UDP packets provided, the team had to create its own sample packets from the specifications provided and hope they were well formed. Another problem was determining whether to use a network device such as a switch or a simple crossover cable to connect the JAUS unit to the control computer. In the end, a switch was chosen to give the option for support of multiple payload packages.

9. Vehicle Costs

Component	Retail Cost	Team Cost
Dell Latitude D830 Laptop	$1,200	$0
Novatel Propak V3 DGPS	$8,000	$3,900
PNI TCM-2.6 Digital Compass	$850	$0
Panasonic 3CCD color camera	$800	$0
US Digital optical encoders	$150	$150
Roboteq AX1500 Motor Controller	$250	$250
Roboteq Encoder Module for AX1500	$150	$150
Extreme Scooter ATV parts (wheels, batteries, motors)	$1,250	$1,250
Mechanical parts	$500	$500
Electrical parts (PCBs, parts, wiring)	$600	$600
Total	$13,750	$6,800

Table 9.1: Component cost breakdown.

10. Conclusion

Athena is equipped with a robust sensor suite on a rugged and dependable vehicle platform. Combined with its innovative modular software architecture, Athena is capable of handling any task that the customer desires. Future software and hardware modules can be seamlessly integrated into the system to perform duties not currently supported. The wireless diagnostics capabilities give customers full control and real time insight into the vehicle's performance characteristics. We think that these key properties give Athena the edge is unmanned robotic vehicles.

11. Acknowledgments

We would like to thank all of our sponsors, Novatel, Omnistar, PNI, and the Energy Systems Institute (ESI) at the University at Buffalo for their product donations. We would like to think our IGVC project adviser, Dr. Puneet Singla, for his guidance in the development of the software. In addition we would like to thank Chin Pei Tang for his help with the Kalman filter. A special thanks go out to our club adviser Dr. Jennifer Zirnheld for her continued support, in addition to Kevin Burke, Jon McMahon and the rest of the ESI staff for their help in making this project possible