WAVE: Warriors Autonomous Vehicle

2009 IGVC Entry

Wayne State University

Advising Faculty:

Dr. R. Darin Ellis

Team Members:

Yusuf Anderson

Richard Chase

Shawn Hunt

Sam Li

Shankar Manickam

Ajay Mudunuri

Prem Sivakumar

Ali Syed

1. INTRODUCTION

This technical report details Wayne State University's submission for the 2009 Intelligent Ground Vehicle Competition. This is Wayne State's first submission to the IGVC, which we all hope will only be the first of many entries in future years. Our team draws on students mainly from Computer Engineering but we have also had some assistance from students from Mechanical Engineering.

Section 2.0 describes the hardware architecture that we used including the electronics, electrical system, actuators, sensors, and computers. Section 3.0 describes our approach to signal processing, path following, and the control loop for moving the platform. Section 4.0 describes our system integration plan.

2. HARDWARE OVERVIEW

2.1 Introduction

This section describes the hardware used in our entry. Section 2.2 describes the platform. Section 2.3 describes the electronics and sensors we chose to use. Section 2.4 describes the electrical system. Section 2.5 describes the actuator that we are using.

2.2 Platform

The platform that we used for our entry started off as a wheelchair. Figure 1 is a picture of the platform equipped with some electronics. It was donated to WSU by Turing Associates, Inc. It is a skid-steer design with a rear passive wheel.

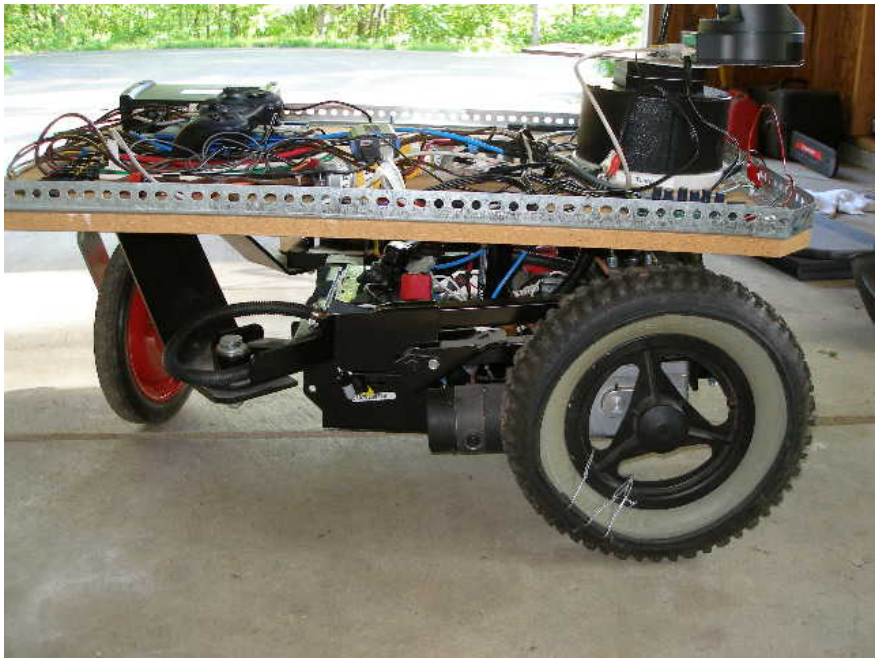

Figure 1: Our modified wheelchair base during the build process.

2.3 Electronics and Sensors

We are using one laptop, a Gateway tablet PC, that has been loaded with Ubuntu 8.04. This laptop does not contain any serial ports. A device made by Quatech accepts a USB connection in and provides eight RS232 ports out, each capable of data rates up to 921 Kbps. The devices that we have plugged into the RS232 box are an inertial measurement unit (IMU), a global positioning system (GPS), an RS232 connection for the Sony pan/tilt/zoom camera, and for communicating with the Roboteq motor controller.

We are using passive optics for our obstacle detection, which is described in more detail in our software overview. The Sony camera outputs an NTSC signal which is fed into a PCMCIA frame grabber from ImperX to the laptop. The overall view of our electronics is shown in Figure 2.

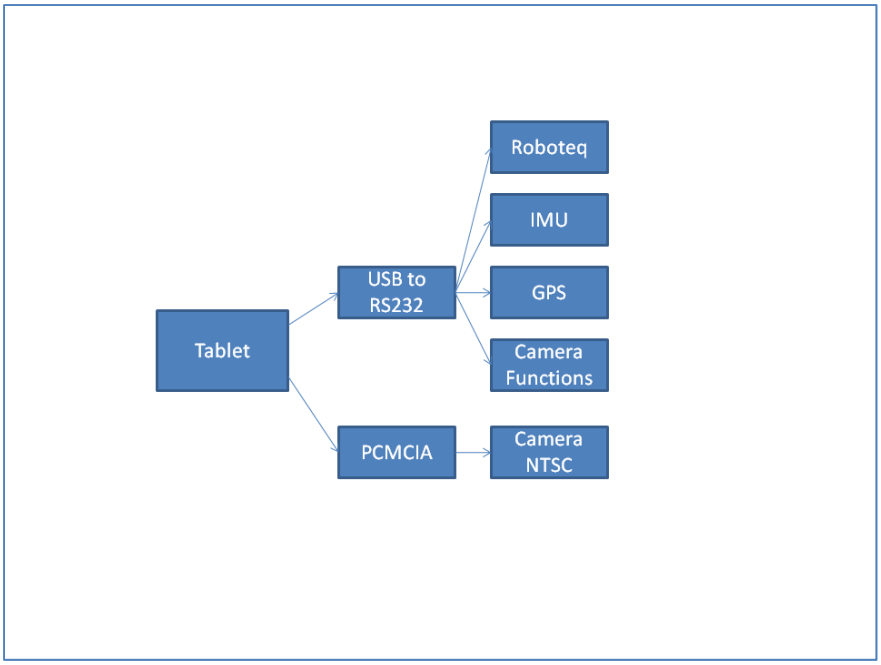

Figure 2: Architecture of our computer and sensors

2.4 Electrical System

There are two 12V batteries in the base of the wheelchair. They are connected in parallel to double the amp hours that are available. There is a connection coming off of one battery to power the equipment on top of the platform.

To avoid excessive power draw by the two motors, a 30A fuse is installed in series with the emergency stop. If the power consumption by any of the devices exceeds that limit, the fuse will disconnect the power supply to the platform. The hardware emergency stop button can also be used to quickly disconnect power to the platform and all devices, in case of abnormal operation.

The sensors being used can be divided into two groups based on their power consumption. The GPS receiver and the IMU both require 9VDC and the Sony camera requires 12 VDC. In order to supply this range, a DC-DC converter is used that accepts 6-30 VDC in and is able to step that voltage down to three separate outputs.

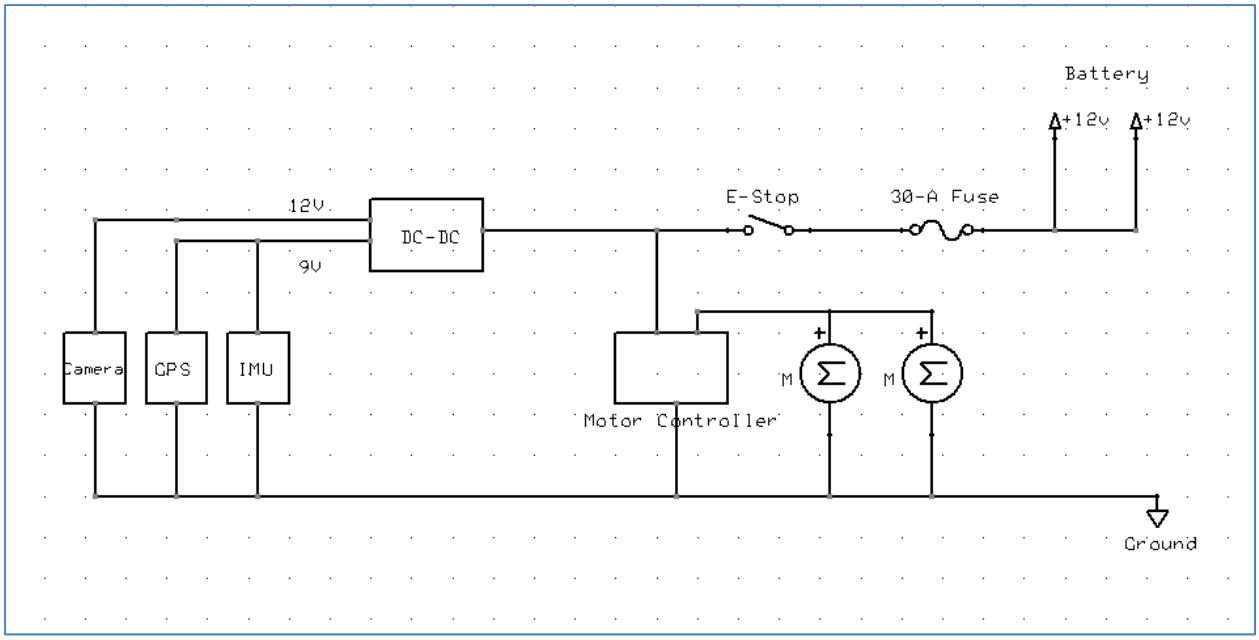

Figure 3: Electronic wiring diagram of our IGVC platform

2.5 Actuators

We are using a Situational Awareness Mast from GeoSystems, Inc. to mount our camera and GPS receiver on. This is a zipper mast that extends up to a little over three meters in the air. Figure 4 shows the mast slightly extended. The three sides join together when driven by the motor and each side coils down when collapsed. We are not able to drive the robot when the mast is extended over half a meter. If the platform drops into a hole, the mast can collapse.

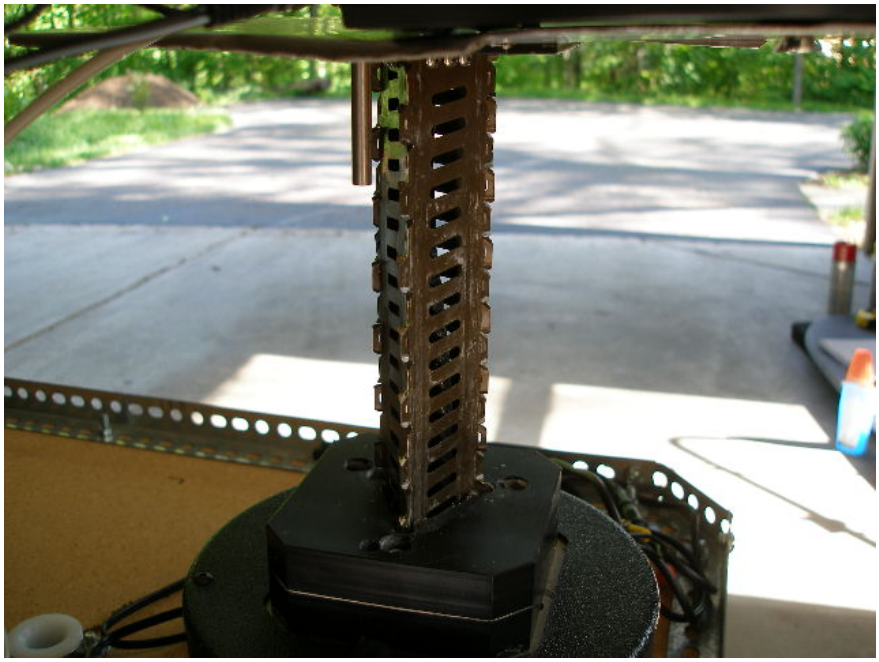

Figure 4: The SAM, Situational Awareness Mast, which we are using to mount our camera and GPS receiver on.

3. SOFTWARE OVERVIEW

3.1 Introduction

This section describes the software architecture that we used for our entry. Section 3.2 describes the signals we are processing. Section 3.3 details the approach we used to handle JAUS communications. Section 3.4 describes our algorithm for lane detection and path following. Section 3.5 summarizes our control loop for autonomous navigation.

3.2 Signal Processing

Our approach to reading data from each of the sensors was to use shared memory on the laptop. The program that reads from each device spawns a thread for each device and writes the current reading to its area in shared memory. There is a location in shared memory for image data, GPS data, IMU data, and feedback from the motor controller. The module that does the path planning reads this data in and then acts upon it. There is a mutex that has to be acquired by either the process doing the reading or the writing of the data in shared memory. This protects the data from becoming corrupted.

The images from the camera come in at 640x480 resolution. The pan, tilt, and zoom functions of the camera are available through the RS232 interface but right now, we are not using them in our control loop.

The Trimble GPS outputs NMEA data. The NMEA sentences that we are processing are GGA and RMC. The GGA sentence gives latitude and longitude. The RMC sentence gives us fix information. The data rate of the GPS is 1 Hz.

The data that we are reading from the IMU are the acceleration, angular rate, and Euler angles (pitch, roll, and yaw). The data rate of the IMU is around 15 Hz.

The data that we are reading back from the Roboteq motor controller are amps consumed by each motor, current battery voltage, and the relative encoder feedback from each motor.

3.3 JAUS

In accordance with IGVC's JAUS Challenge requirements, the platform will respond to a periodic status and position requests from the Judge's COP. This requires the updated JAUS standards published by SAE, i.e. the JAUS Transport Specification (AS5669A), the JAUS Core Service Set (AS5710) and the Mobility Service Set (AS6009) compliant. We originally planned to use OpenJAUS that is a free source robotics SDK for our JAUS implementation, but the available version 3.3.0 of OpenJAUS is not SAE AS5669A or AS5710 compliant. After more research, we decided to use Jr Middleware that is a SAE AS-4 (JAUS) AS5669 compliant software development tool for the platform. The platform uses Ethernet Wireless UDP communication, and the IP address 192.168.1.200 is used to the platform for testing. The JAUS ID is 200-1-1, only one component is used for the system. Figure 5 shows our JAUS system architecture and the system communication with JAUS Validation Tool.

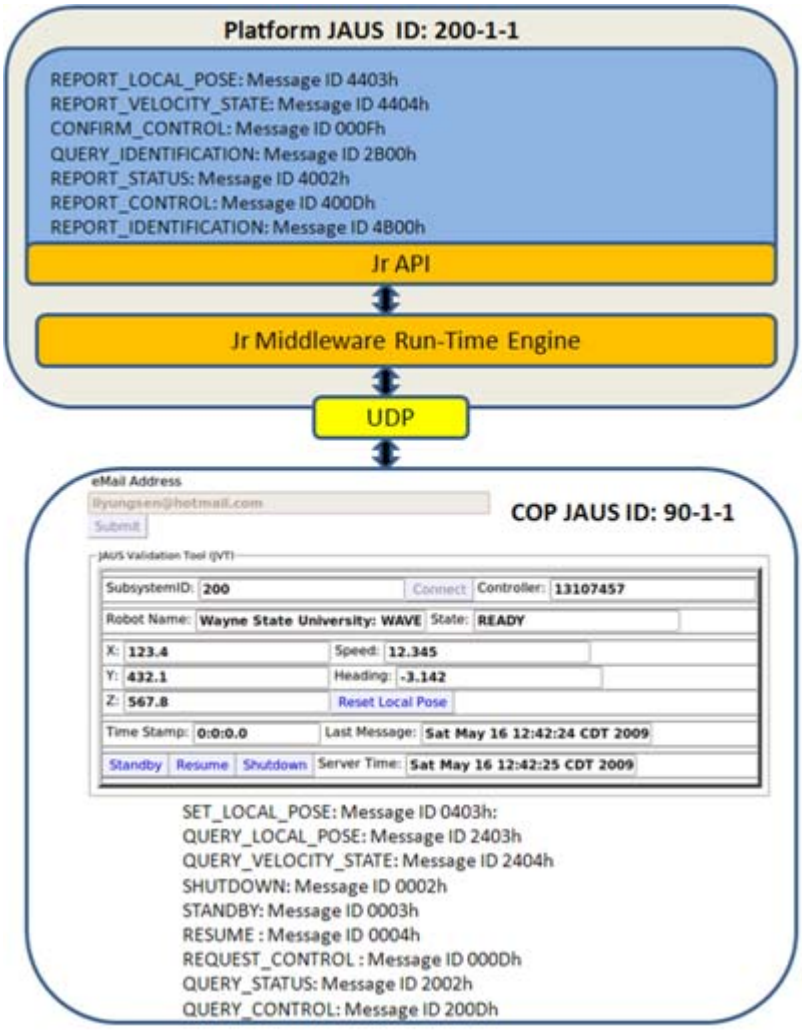

Figure 5: This shows the architecture of JAUS implementation and the system communication with the JAUS validation tool.

3.4 Path Following

We trained a supervised learning algorithm to recognize lanes in the video stream. Our approach uses a graph-based image segmentation algorithm [1] that segments the image into regions of similar color and texture. An example of the segmented image is shown in Figure 4. Once we have a segmented image, we run statistics on each region. These statistics include area, perimeter, length of the major axis, and the length of the minor axis. We collected data of images with lanes, then hand-labeled the regions as a dashed lane, a solid lane, or neither. These images were then trained using a naïve Bayes classifier to recognize the presence of a lane in the video stream.

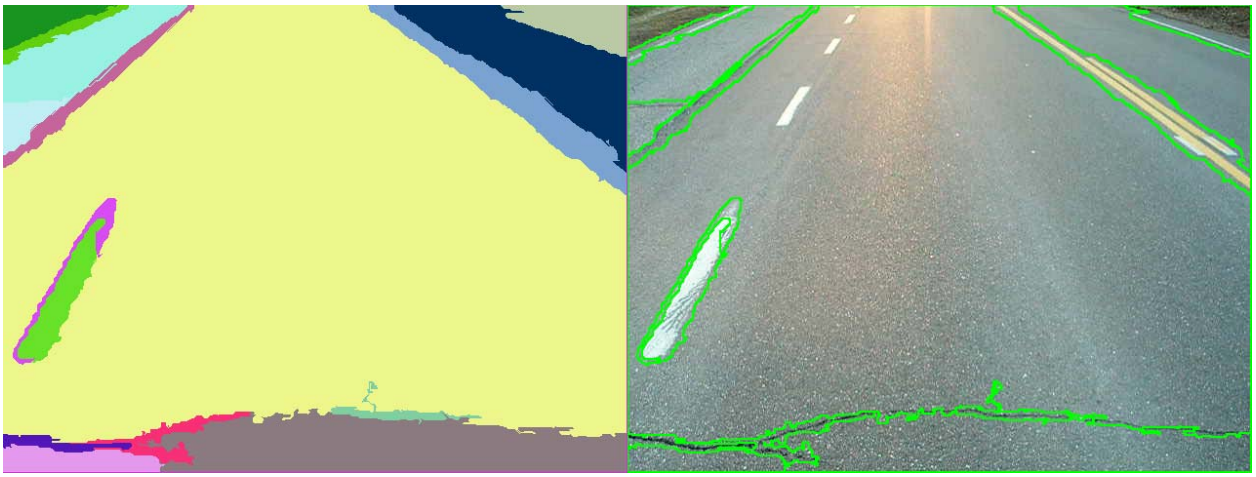

Figure 6: The image on the left is the result of doing a graph-based image segmentation of the image on the right. The green overlay on the input image is derived from the regions found on the left.

3.5 Control Loops

The control loop for the navigation challenge first requires the input of the waypoints. Our software grabs an image from the camera, segments it, checking for obstacles. It next grabs the current latitude and longitude from the GPS. If the path is clear, it moves toward the waypoint. If an obstacle is present, the platform attempts to navigate around the obstacle. Once the obstacle has been cleared, the current GPS location is obtained and a new path is calculated. This is repeated until the goal location has been reached.

The control loop for the autonomous challenge is similar to the control loop for the navigation challenge except here, once an image has been segmented; we pass it through our naive Bayes classifier for lane detection. The control module will align the robot so that a lane is to the left and to the right. Once the lanes have been found, the platform will move forward, checking for obstacles.

In addition to the hardware emergency stop described earlier, there is also a wireless emergency stop. There is a separate control box that houses a microcontroller and a ZigBee module. Once power has been applied to the circuit, the microcontroller starts transmitting a message that is received by a microcontroller that we have connected to the motor controller. If this message

stream is not received by the microcontroller on the platform, the microcontroller will send a command to the motor controller ceasing all movement.

4. SYSTEM INTEGRATION

4.1 Introduction

This section describes how our system operates as an entire entity. We describe how our platform is able to handle inclines and obstacles. We also touch briefly on our algorithm for GPS waypoint navigation.

4.2 System Integration.

We have set the current limit of our motor controller to 30A. This setting does not permit the platform to travel faster than the 5 MPH limit as set by the competition rules. Our platform is able to handle ramp climbing. We have tested this out on thirty-degree ramps without any trouble. This was problematic prior to setting the current limit setting on our motor controller. We would pop fuses if the incline was too steep or the platform became stuck in mud. Our platform is also able to handle potholes without any trouble. We tested on potholes that were three feet in diameter and four inches deep.

Our platform is able to detect obstacles from ten meters away. At the time of this writing, our code is still in development but we have been experimenting with using the segmentation code to segment out the scene, then on items which look like obstacles. Figure 7 shows a construction scene that has been segmented out. Our goal when we started this project was use only monocular vision for the detection of obstacles. Although we feel this is still possible, we might end up adding a SICK laser scanner to the front of our platform and try doing monocular vision only in next year's entry.

Figure 7: A construction scene with obstacles that has been segmented out using the graph-based image segmentation code.

Our platform is able to react instantaneously to the hardware emergency stop being pressed. We are able to detect an incline within one second by reading the pitch value from the IMU and detecting that the current draw from the motors has increased by reading the data from the motor controller.

We recently replaced the batteries in our platform. The platform is around ten years old and the batteries had never been replaced. We have been testing our code each day for several weeks, at least several hours each day, and we have not had to recharge the batteries yet. Initially, we had thought to use a separate power supply on top of the platform to power the electronics but the lithium-ion batteries would last for approximately one hour before needing to be recharged. We were concerned that using the power supply from the base would introduce power spikes that would cause unpredictable results with our sensors. This has turned out to not be the case.

As of the time of this writing, our waypoint navigation code has not been finalized yet. The preliminary results of our code are that we are able to navigate to waypoints with approximately two meter accuracy. We also have not yet addressed the more complex obstacles that we will encounter such as switchbacks and dead ends.

5. REFERENCES

1. P. Felzenszwalb, D. Huttenlocher, “Efficient graph-based image segmentation”, International Journal of Computer Vision, Vol. 59, Issue 2, pp. 167-181.

6. APPENDIX

A. Cost Breakdown

Item	Cost
Wheelchair Platform (donated)	
Roboteq AX3500 Controller	$ 395.00
Sony EVI-D70 PTZ Camera	$ 875.00
ImperX VCE-Pro Framegrabber	$ 599.00
Microstrain 3DM-GX1 IMU	$ 1,495.00
Trimble AgGPS	$ 5,000.00
Robotics Group Inc. DC-DC Converter	$ 295.00
Quatech USB to RS232	$ 800.00
Total	**$ 9,459.00**

B. Man Hours

Group	Person	Man Hours
JAUS	Sam	200
	Yusuf	40
	Shankar	80
Hardware	Ali	200
	Ajay	40
Computer Vision	Prem	40
	Shawn	200
Total		**800**