The University of Detroit Mercy Presents

Megatron

IGVC 2011 Design Report

Team Members

-Michael Dupuis
-Jonathan Wells
-Ryan Duprey
-Amy Edwards
-Matthew Westermann
-Thomas Lorenzo
-Darrien Bean
-Bryan Zink
-Phong Nguyen
-Sinan Alijony
-Eyad Zenio
-Matthew Parrish
-Alex Szmatula

Faculty Advisors

-Prof. Utayba Mohammad
-Dr. Chaomin Luo
-Dr. Mohan Krishnan
-Dr. Mark Paulik

Consultants

-Cheng-Lung Lee
-Maen Hammod

We certify that the engineering design in this vehicle undertaken by the student team, consisting of undergraduate and graduate students, is significant and qualifies for course credits in senior design and in the Master's program respectively.

_____

Prof. Mohammad, Dr. Luo, Dr. Krishnan, Dr. Paulik

1. INTRODUCTION

The University of Detroit Mercy (UDM) 2011 IGVC team is pleased to reintroduce *Megatron*, an enhanced version of *Cerberus,* the differential drive vehicle first introduced in the 2010 IGVC. *Megatron*, based on a proven platform, introduces significant innovations, ranging from extensive software enhancements to an entirely new power management system. The goal for this year's IGVC team was to enhance our earlier robot design, and develop and implement solutions to the challenges posed by new competition requirements. The resulting vehicle not only meets the additional requirements of the competition, but is even more capable and robust.

2. DESIGN INNOVATIONS

The innovations incorporated in *Megatron* are listed below and they are further explained in detail in the sections that follow:

- A new self-contained charging station complete with locating algorithm software for the vehicle to find it, using predetermined GPS points (details in Section 5.1).
- A new spline-based speed modulation strategy for the Autonomous Challenge that incorporates a completely new path planning algorithm based on breadcrumbs, to appropriately establish a contour between current speed and heading and future desired speed and heading (details in Section 6.3.4).
- A new spline-based speed modulation strategy for the Navigation Challenge based on our D*Lite path planning algorithm, with the incorporation of path contouring between the current speed and heading and the desired speed and heading at the next breadcrumb (details in Section 6.4.4).
- Use of Wireshark, a JAUS sniffer, in the JAUS Challenge, which captures and analyzes JAUS messages on the network (details in Section 8.1).

3. PROJECT MANAGEMENT

3.1 Design Process

The team wanted to approach the project in an organized manner that facilitated continuous communication between all team members, allowing for sharing of key information for the various sub-tasks to be completed and identifying and resolving project bottlenecks in a timely manner. In order to accomplish this, an adaptive project framework was used that could accommodate variable scope for the individual project tasks. This is an iterative project management framework that seeks to optimize performance for each task in the design-implementation cycle.

Early in the Winter Term, a literature search was performed to find potential solutions for modifications and/or improvements necessitated by competition rule changes as well as perceived shortcomings of our vehicle from last year. A detailed feasibility study of these possible solutions and an initial design process was undertaken. Weekly deadlines were setup as part of the design process for various individual requirements. This included biweekly meetings for reporting and biweekly individual written reports, as well as individual weekly progress reports. At the biweekly meetings, the team assessed individual difficulties in order to be able to direct resources when necessary to facilitate the removal of bottlenecks. The iterative project management strategy that was used optimized performance in every task undertaken by members of the team and ensured continuous progress.

3.2 Team Composition

The team comprises 10 senior electrical engineering undergraduate students and 3 robotics engineering graduate students. Individual assignment of tasks was based upon personal expertise and interest in topic areas. Figure 1 indicates the existing team structure and the assumed responsibilities by each team member. The total time dedicated to the development and implementation of this project was approximately 3000 hours.

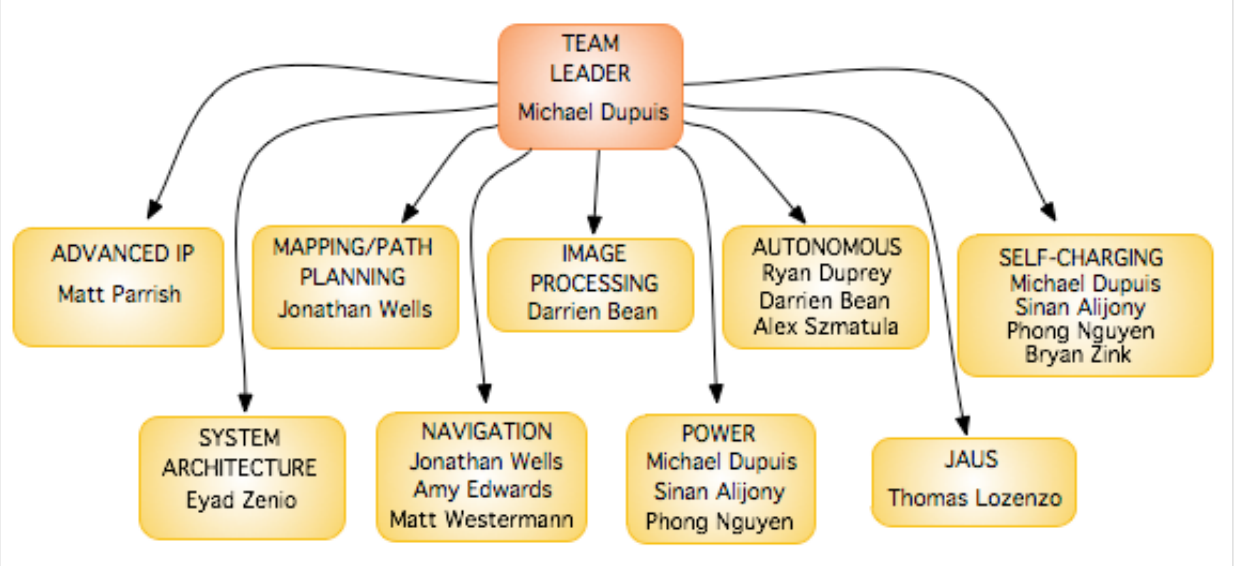

Figure 1: Team Organization

3.3 Design Objectives

In accordance with the principle of continuous improvement, the enhancements incorporated into *Megatron* were based on IGVC performance in the previous year. A few weeks after the 2010 IGVC ended, both faculty advisors and team members met to review our vehicle's performance and that of the competition. The document generated at that meeting served as the basis for the development of the design objectives for *Megatron*. These are:

1. Meet all IGVC requirements.
2. Enhance the vision strategy to deal with the newly incorporated flag detection requirements.
3. Improve and enhance speed modulation techniques for the Navigation Challenge so as to optimize completion time.
4. Improve and enhance speed modulation techniques for the Autonomous Challenge, in particular to handle the new course configuration that includes GPS coordinates.
5. Extend battery life of vehicle in comparison to last year by improving power efficiency through intelligent switching.
6. Improve upon performance in the JAUS Challenge

3.4 Cost Summary

Despite the fact that *Megatron* was a developmental vehicle, it was constructed while keeping cost-effectiveness in mind. The systems incorporated were either designed and built in-house or specifically purchased to best suit the project objectives in terms of functionality, performance, cost, and development time. Table 1 shows the approximate retail cost of the various sub-systems in *Megatron* as well as the team cost (some items were donated, reduced in price, or used from an earlier project).

Table 1: Cost Summary

Description	Retail Cost	Team Cost	Comments
Frame/Body	$746	$746	Volunteer work was involved
Drive Train	$2471	$2471	Purchased new
Rear Wheels(2) & Front Caster	$600	$0	
Batteries(4) & Charger	$430	$430	Purchased
Power PCB	$1100	$1100	Designed in-house
Remote PCB	$304	$104	Tranceiver donated by Aerocomm
Camera, Lens, Adapter	$937	$898	
LIDAR	$5500	$5000	Purchased
DGPS & Antenna	$6000	$3500	
Digital Compass	$1096	$0	Donated by PNI Corporation
MacBook Computers(2)	$5000	$5000	
OmniSTAR HP	$1250	$0	Donated
Charging Station	$600	$600	
TOTAL	**$25,219.96**	**$19,016.96**	**Savings=$6,203**

4. MECHANICAL SYSTEM DESIGN

Megatron has the same mechanical design of last year's vehicle *Cerberus*. It is a 3-wheel vehicle with two rear driving wheels and a leading caster. *Megatron* has a width of 0.8 meters, a length of 1.04 meters, and a height of 1.8 meters. The total weight of the vehicle, not including the payload, is 110 kilograms.

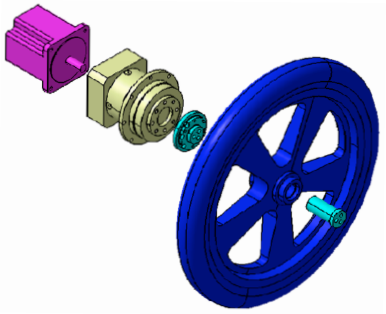

Figure 2: Drive Train

The frame is made out of 20 mm square steel tube stock and pre-painted 6.35 mm Alumilite sheets. This material has the same weight as 1 mm thick aluminum but is 50 times stronger, adding robustness while minimizing the vehicle's weight.

Megatron has a simple, yet solid drive train configuration (see Figure 2). It features a front caster and two rear driven wheels. Our two side-by-side 24V motors provide a continuous stall torque of 4.77 N-m. The motor's output is coupled with a 12:1 planetary servo gearhead, which has built-in bearings rated at 1400 N for radial loading and 3000 N for axial loading. It also features an output flange instead of a shaft, which takes away the need for an alignment coupling, along with two tapered roller bearings that would be required to allow for radial and axial loads. Torque is transmitted from the gearhead to the wheel through a two-part connector; it consists of a custom-connecting hub, which is bolted onto the gearhead. Four press-fitted pins transfer torque from the hub to the wheel, and a custom shaft securely clamps the wheel between the hub and itself with three bolts. The use of three bolts instead of one adds strength to the design; a single bolt could easily become detached, but since the load is symmetrically distributed between three bolts, this is less likely.

5. ELECTRICAL & ELECTRONIC SYSTEMS

The electrical systems design includes significant innovation as well as incremental improvements. Based on a review of previous performance, upgrades and enhancements were added that specifically improve safety, efficiency, and functionality. Innovations include a modular power-box design, a microprocessor-controlled power-

sequence switching system which improves efficiency, and an automated charging station. Improvements include an operational-status light and a low-battery alarm circuit.

5.1 Innovations and Improvements

For this year's power box, many functions have been added. In order to accommodate the new competition requirements, a safety light was added as an improved safety feature. This feature is used to alert those around the vehicle whether or not the vehicle is being controlled manually. This light is solid when the robot is in remote control mode and blinks when the robot is in autonomous mode.

One innovation that has been incorporated is a switching sequence for the power box. This switching sequence is controlled by the microprocessor which then controls the enable pin of switching regulators and the relay coils. The power for the LIDAR, motors, and motor controllers is controlled by the microcontroller in a specific order. The specific order is used to improve the efficiency of the system on startup. The LIDAR will be turned on first, followed by the motors, and then finally the motor controllers.

An improvement to this year's power supply system is a low power indicator. This low power indicator is a buzzer that is turned on with a specific acoustic pattern for different ranges of low power (i.e. low power and critically low power). This feature is intended to improve reliability of the robot when out in the field because the LCD screen present on the vehicle will not be visible when navigating autonomously.

The most significant hardware innovation for *Megatron* is the self-charging feature of the robot (See Figure 3). The low power signal is sent to the laptop computer and the USB interface from the power box. Once the low power signal is received, the robot initiates the docking/charging procedure. This procedure uses the GPS and LIDAR to find the docking station and will then turn 360 degrees to locate the opening of the docking station. The robot then slowly approaches the docking station making any necessary adjustments. Once the connection is made with the docking station, the batteries will begin to charge and the microprocessor will shut systems down that are not required.

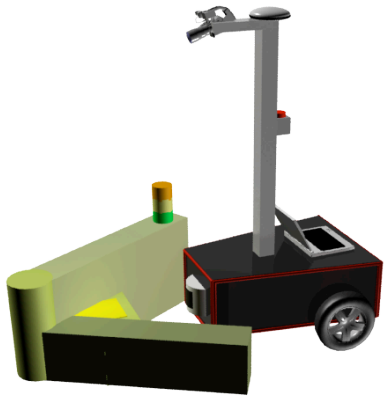

Figure 3: Docking Station

This docking station contains many components used in charging the robot. The docking station has an AC to DC converter to charge batteries located in the docking station. The batteries are used as storage devices to hold the energy needed to charge *Megatron's* on board batteries. Using batteries in the docking station allows it to be portable and therefore it can be used in a multitude of environments. When the robot is charging, a reverse switching sequence will be used to turn off power to subsystems that are not critical during the charging process. For example, the LIDAR can be shutoff to allow for less power to be consumed while charging, making charging faster and more efficient. The power supplies for the laptop computers, motors, and motor controller will also be shutoff. The remaining power supplies will be left on so that the microprocessor will continue to be able to monitor the status of the batteries.

5.2 Power Distribution

To provide adequate power to all sub-systems of *Megatron*, the requirements of each were first assessed under normal and worst-case conditions. The power for *Megatron* comes from two 12 V 55 Ah Powersonic gel-sealed batteries connected in series, and is distributed via a custom-designed PCB to the vehicle sensors, wireless router, motors, and two Macbook Pro computers. The overall power distribution scheme is shown in Figure 4.

5.3 Electronic System Protection and Safety

To help ensure that *Megatron* is safe, reliable, durable, and easily serviceable, several special features have been incorporated into the power distribution system. The PCB is designed such that high power components are isolated from lower power components. Fuses are strategically positioned on the PCB to prevent electrical damage due to unexpected current surges. The incorporation of high efficiency switching regulators provides stable outputs with low ripple. In addition, these regulators have been designed to protect the PCB from low battery voltage levels, short circuiting, and overheating, thereby extending the life cycle of the circuitry. A clamper circuit is connected to the motor power supply to absorb the motor's back EMF. The status of the power box is conveyed via a series of panel-mounted light emitting diodes (LEDs). Finally, vehicle-wide systems integration is addressed by the use of a real-time current and voltage monitoring system that sends status information from the power box to the main laptop through a USB connection. Thus, if a problem occurs, its source can be located quickly and diagnosed.

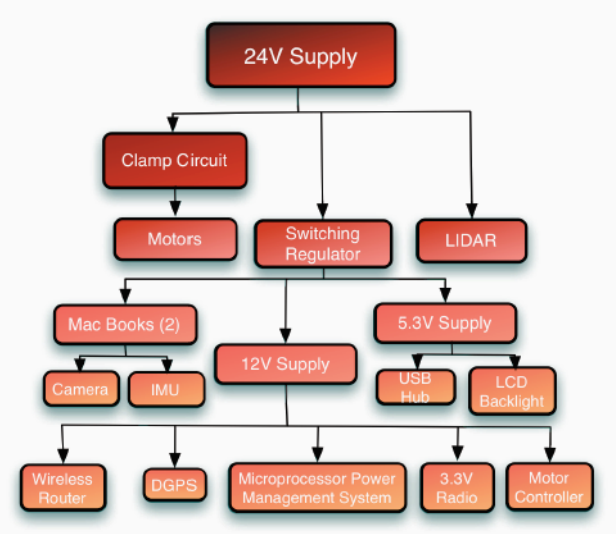

Figure 4: Power Distribution Scheme

Beyond protecting *Megatron* from possible internal problems, the electronics are protected from both water and dust. The vehicle is weather proofed such that light rain will not cause electrical short circuits. This involves the incorporation of NEMA enclosures for the power distribution system, as well as a shell that surrounds the vehicle chassis and the various components.

Lastly, for user safety, *Megatron* is equipped with hard, soft and remote E-stops controlled by a mechanical button, the microcontroller, and the remote control respectively. The remote control, which can operate in one of two modes-Computer Controlled (PC) or Remote Controlled (RC)-is made up of a custom designed PCB housed within a durable Futaba remote control shell. When the remote control is set to operate in PC mode, it transfers control of the motors to the computer (retaining E-stop control). If placed in RC mode, the operator can manually drive the vehicle.

The transceivers that are used in *Megatron's* design are Aerocomm AC4490-200A transceivers. Although the vehicle only needs to be controlled from a maximum distance of 50ft, with the implementation of the aforementioned transceivers and full antenna extension, the vehicle is capable of being controlled from nearly a mile

away. A twist-to-release remote E-Stop button is integrated into the remote control unit. As an added measure of security and immunity to interference, we transmit encrypted data over a spread spectrum wireless link for two-way communication between the vehicle and remote.

5.4 Sensor System

Megatron incorporates five sensors into its compact design: a camera, a LIDAR, a DGPS, a digital compass, and an IMU. Each sensor is enclosed in a waterproof case and firmly mounted to the vehicle while, at the same time, permitting relatively easy removal for servicing. The following is a brief description of the sensors that are used by *Megatron* as shown in Figure 5.

Camera: The AVT Stingray F-080C 1/3" CCD camera was selected as the vision sensor for this vehicle. This camera uses the IIDC IEEE 1394B protocol to relay images, which is ideal for machine vision applications, because the frames are uncompressed and various options such as region of interest and lookup tables can be set and executed in hardware. Also, the camera's progressive scanning and high frame rates minimize motion blurring. The CS-Mount lens design enables the camera to accept a very wide-angle low-distortion lens, which provides a 125^0 field-of-view, which makes navigation heuristics easier to implement.

LIDAR: A 270^0 SICK LMS111 LIDAR unit was employed for the purposes of obstacle detection. The unit is capable of collecting data over a 270° field-of-view with 0.25° resolution, a maximum range of 20 m, and a 25 Hz scanning rate.

DGPS: To obtain positioning data in the Navigation Challenge, Novatel's ProPak-LB Plus DGPS system was selected. The DGPS antenna is mounted to the top of the vehicle's mast while the receiver is securely positioned inside the chassis. Using Omnistar HP's DGPS system, the signal is corrected to provide up to +0.1m accuracy. This system provides data at a rate of 20 Hz, which is adequate for *Megatron's* expected speed and desired performance.

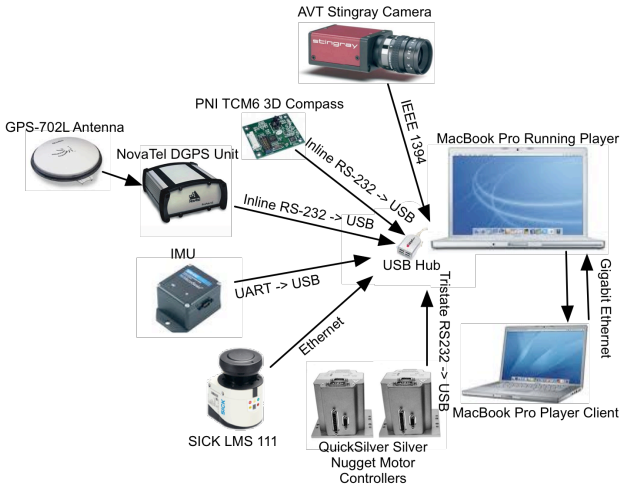

Figure 5: System Communication

Digital compass: A PNI TCM6 digital compass was integrated into the vehicle to help determine vehicle heading. This compass provides a heading accuracy of 0.5° and updates at 20 Hz, which again is sufficient for the vehicle's speed and desired performance.

IMU: MicroStrain's 3DM-GX2 is a high-performance gyro enhanced orientation sensor which utilizes miniature MEMS sensor technology. It contains a 3-axis accelerometer, 3-axis gyro, 3-axis magnetometer, temperature sensors, and an on-board processor running a sophisticated sensor fusion algorithm. The 3DM-GX2 outputs include Euler angles, rotation matrix, deltaAngle & deltaVelocity, acceleration and angular rate vectors with a maximum data refresh rate up to 256 Hz.

5.5 Data Communication Interfaces

The various electrical and electronics systems were interfaced in the manner illustrated in Figure 6. The AVT Stingray camera is connected via Firewire (1394B) to the MacBook Pro. The DGPS system is connected to the computer through a USB interface using an inline RS-232-to-USB adapter. The SICK LMS111 LIDAR is connected to the computer via an Ethernet link. The PNI TCM6 digital compass also uses RS-232. Finally, all the computers are networked via gigabit Ethernet.

6. SOFTWARE DESIGN

The primary goal concerning software this year was to improve upon last year's software structure, integration, and performance. Accordingly, more failure-tolerant algorithms were developed, and user-friendly debugging tools were created. This was accomplished through identifying problematic vehicle behavior, reverse engineering existing approaches, developing and validating the new approaches through simulation, and finally deploying the successfully validated algorithms and testing them on the new vehicle.

6.1 Development Environment

This year the *Megatron* team continued the use of a multi-layer environment for software development shown in Figure 6. These layers make communication between hardware and software possible. The first is the server layer which is occupied by Player™, an open-source, Unix-based (Linux or Mac OS X) robotic software system that serves as an interface between robotic algorithm implementations and the vehicle systems. Specifically, it provides standard interfaces for a typical set of robotic peripherals (LIDAR, cameras, motors, etc.) that can be accessed from the local computer or any other computer over a TCP/IP network. The second layer is the client layer occupied by user programs written in MATLAB®©. These layers communicate with Player™ over a TCP socket to acquire data from sensors and send actuator commands, which Player™ then passes on to the vehicle. The writing of efficient wrappers to enable communication between the two layers is what has enabled our program developers to utilize MATLAB®, with its wealth of proven and powerful resources, for algorithm development.

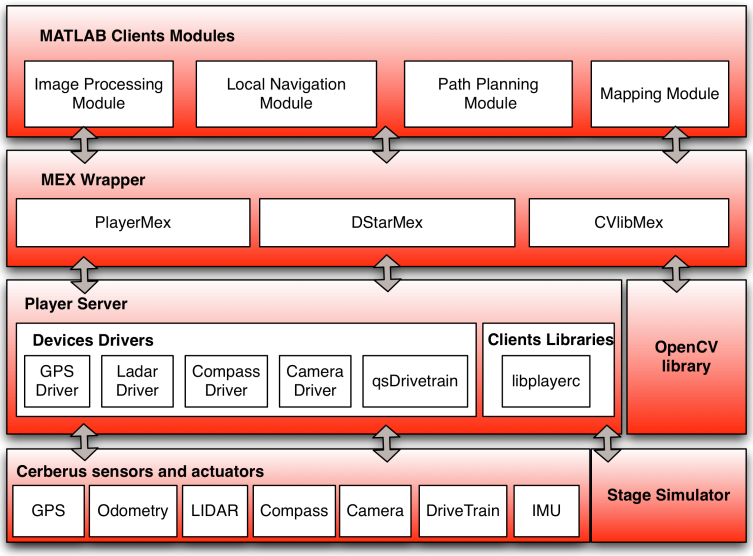

Figure 6: Software Development Architecture

6.2 Data Acquisition and Processing

With abundant data from numerous sensors being processed at once, there are multiple computers used to ensure that all needed data is acquired and processed as quickly as possible. *Megatron* distributes the processing tasks among these computers as explained below.

Sophisticated vision algorithms are central to a successful strategy for the Autonomous Challenge due to the complex obstacle, lane, and terrain features present. If, when implementing these algorithms, the associated computational complexity causes the overall image-frame processing rate to drop too far, the vehicle may not be capable of operating effectively at higher speeds.

In order to favorably address this tradeoff, a multi-pronged strategy to increase the frame rate was adopted. *Megatron* distributes its computational tasks over two laptop computers (with a total of 4 processor cores). Using Player™ as a server facilitates setting up this distributed computing architecture and provides a wealth of existing open-source code to draw from. The top computer, a MacBook Pro, is entirely devoted to running components of the overall vision algorithm. This computer utilizes the MATLAB® parallel processing toolbox to spawn multiple workers which exploit parallelism in several of the IP routines (e.g., color segmentation and adaptive thresholding). This enables effective utilization of the multiple processor cores. When running the Navigation challenge, vision is not utilized; and the top computer is assigned to perform the heavy computations of path planning. A second MacBook Pro, on the swivel, is responsible for accepting the data from the vehicle's sensors, implementing heuristics on the vision results, generating a map, selecting a goal, and navigating the vehicle towards that goal.

6.3 Software Design: Autonomous Challenge

6.3.1 Image Processing

In order to address the increasing complexity of the vision task in the Autonomous Challenge course, *Megatron* has adopted the proven solution from Cerberus's image processing methodology. In the past, images were captured in RGB format, which represents the image color components as red (R), green (G), and blue (B). Lane lines were then identified based on a gray scale image that promotes the lane line color combinations and demotes those of other objects in the scene. This grayscale image was derived with simple scaling and additions of the R, G, and B planes. Recently, colored barrels have been added to the autonomous course, and have made the gray-scale image formula and thresholding very sensitive to lighting conditions and scene content. Therefore, *Megatron* has moved to a heavily color-dependent recognition algorithm that is more stable and robust.

HSV Thresholding: *Megatron* processes images using hue (H), saturation (S), and value (V) planes. One aspect that makes the HSV representation appealing is that the lighting information is contained in the value plane, while the color information can be found using the hue and saturation planes. Each pixel in the image can be defined by these three values. The hue specifies the color, the saturation specifies the amount of color, and value specifies the intensity information. The image processing algorithm identifies lanes and obstacles. This is particularly useful since the LIDAR is unable to identify lanes as well as saw horses with portions outside its vision plane. The output of the image processing algorithm is passed on to a data fusion process that fuses the LIDAR readings and lane-obstacle information into one obstacle data set.

The image processing algorithm works as follows: first, the captured RGB image is converted into an HSV image, and then thresholding is applied to the hue and saturation planes to remove grass from the image. This leaves behind anything that is an obstacle or is a lane. Finally, thresholding is performed again to pull out anything that is white. In the HSV representation of an image, anything that is white will have a fairly low saturation value and can have almost any hue or intensity value. The algorithm then checks for connected components and removes

anything that is small and thus considered to be noise. Figure 7 shows a captured image and the corresponding output of the image processing algorithm.

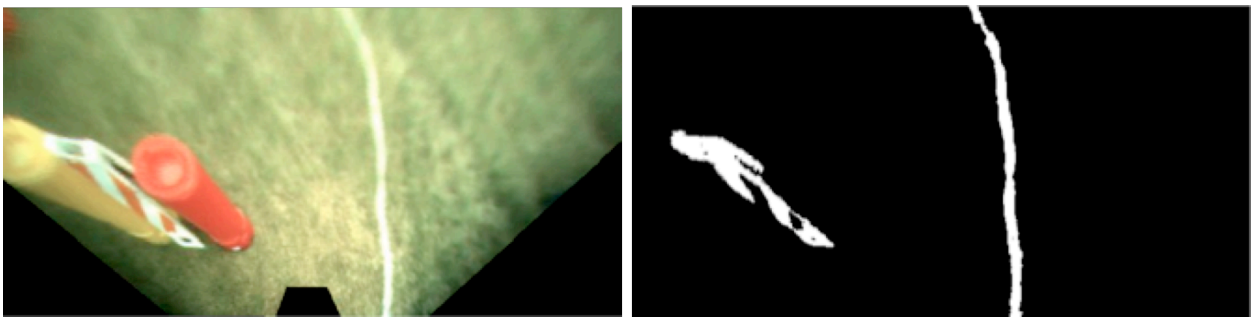

Figure 7: Captured Image on Left/Processed Results on Right

Ramp Detection: because of its glossy paint and bluish color component, the ramp confuses normal color segmentation algorithms and often shows up as an obstacle under bright lighting conditions. This calls for a ramp identification strategy and a corresponding lane detection algorithm. By studying the histogram of the saturation plane in the HSV image, the ramp can be successfully identified based on number of pixels in a particular range of intensities. Figure 8 shows two histograms, one for a regular section of last year's autonomous trial course, and the other for a ramp section of the same course. After identifying the ramp a series of transform operations is performed so that only the ramp vertical edges are retained. The final output is an image that only contains the ramp edges.

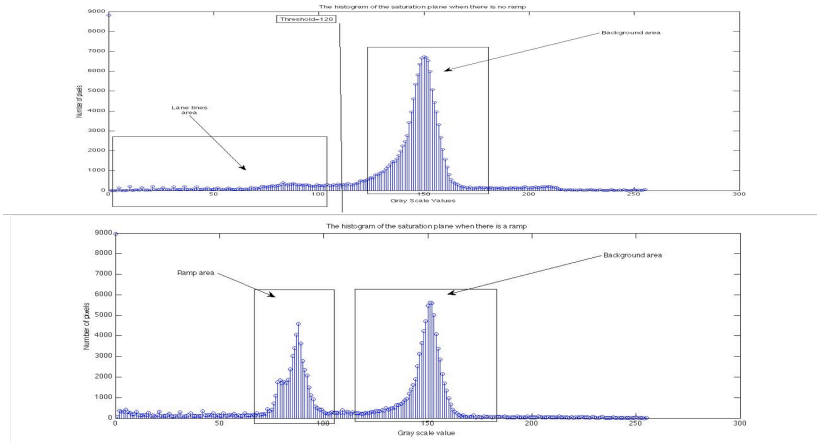

Figure 8: The top histogram: Autonomous course without ramp. Bottom histogram: with ramp

Flag Detection: New to the autonomous challenge are the red and green flags. Once identified, the flags will be used to guide the robot in a specific manner. To accomplish this task, color segmentation using Hue, Saturation, and Value (HSV) was used. By knowing the specific hue of the flags over a threshold saturation, the flags can be identified. The hue will relate to the color of the flag and the saturation will relate to how much of the color is present. As the saturation increases, the color becomes less white and closer to the color of interest. Segmenting the images independently for each color will result in a binary image that can be used to describe the locations of the flags present in the image. These locations can then be used in conjunction with an enhanced navigation strategy to maneuver between the colored flags.

6.3.2 Goal Selection

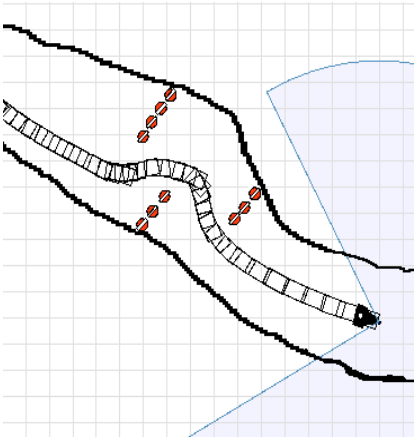

Figure 9: Switchback Navigation Scenario

The goal selection algorithm is concerned with determining the "forward" direction. For the Navigation Challenge this is relatively easy, as forward is towards the next waypoint. However, in the autonomous challenge, the forward direction is less easy to determine, since it requires the vehicle to "go around the course". The ability to successfully navigate the autonomous course requires the use of image processing, obstacle avoidance and goal selection. These three facets allow *Megatron* to remain in the lane lines and avoid obstacles, while continuing to move in a forward direction through the course. Keeping the robot moving in the forward direction is the biggest challenge, which is handled by goal selection. The forward direction then has to be established from the results of the Image Processing algorithms, which are not 100% reliable. Further complicating the situation is the presence of course features such as switchbacks and ramps, which can create apparent traps.

To deal with this situation, a heuristic layer is created. This layer combines two pieces of information to set the goal direction. The first is a "GPS history" direction established by GPS coordinates 4 meters apart from the immediate travel history of the robot. The second is the result of the Image Processing algorithm with its built-in level of confidence measure. When the IP results are less reliable, greater reliance is placed on the tail in setting the goal direction and vice versa. This approach contributes to improved navigation of switchbacks and traps. The effectiveness of this algorithm can be seen from the Stage™ simulation of Figure 9 which shows the robot navigating a switchback smoothly.

6.3.3 Navigation for Autonomous Challenge

The LIDAR data is read as a set of polar distances to obstacles. While this data is processed in a similar manner using the VFH+ algorithm, there are differences between LIDAR data processing for autonomous navigation and waypoint navigation. The LIDAR data is converted into a polar histogram. The polar histogram is divided into 54 sectors (5° per sector) to cover 270°. The obstacles are enlarged to further reduce the chance of hitting any obstacles and to provide smoother motion for the vehicle. After the obstacles are enlarged, the autonomous navigation algorithm then incorporates an obstacle grouping technique that prevents the choice of paths leading into a concave trap. Each sector is determined to be either blocked or available through use of a threshold. The best sector to go through is then picked based on a weighted formula that combines deviation from desired direction and associated obstacle densities.

Flag Avoidance: In order to drive on the desired side of red and green flags, the image-processing algorithm passes flag information to the navigation algorithm in two polar flag-arrays, one for green flags and the other for red flags. The flag arrays identify the incident angle and distance of flags with respect to the vehicle. This polar information is used to assign high cost for open polar sectors that don't comply with the flag convention forcing the vehicle to comply with the flags' guidance, and allowing it to tolerate noisy images with false flag detections.

6.3.4 Speed Modulation

One of the most important innovations in *Megatron*'s software design is the implementation of a new spline-based speed modulation algorithm. In previous years, our vehicles drove with an average speed of 1-2 mph in the autonomous challenge. The new speed modulation algorithm in *Megatron* allows it to drive along pre-determined smooth curves while avoiding obstacles thereby maintaining its momentum and enabling an average speed of 3-4.5 mph, depending on the path curvature.

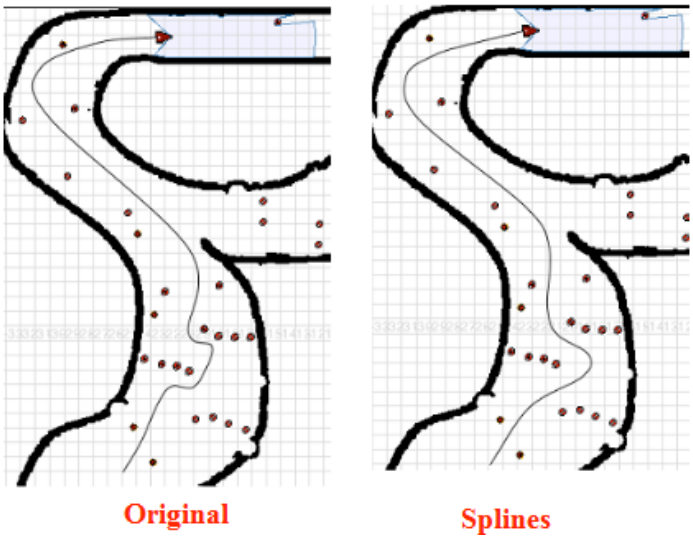

Figure 10: Speed Modulation Curve

To enable the vehicle to maintain maximum speed, an optimal path contour has to be planned and adopted. This contour is defined using time-dependent parametric equations that take into account the current pose of the vehicle and a virtual target pose. The virtual target pose is defined based on the desired target heading as provided by the goal selection algorithm and a pre-set distance along that heading.

The time derivatives of the path equations along with consideration of the initial speed and heading of the vehicle generate velocity equations along the pre-planned path. The speed modulation works in compliance with the VFH+ algorithm, providing the speed and heading information as an input to the VFH+ without interfering with its inherent decision-making logic. Figure 10 illustrates the smoothed paths that are generated when using the new speed modulation technique.

6.4 Software Design: Navigation Challenge

Given a set of waypoints and possible starting positions, a matching lookup table for waypoint sequencing is created. The D*Lite algorithm then provides an initial path (breadcrumbs) between pairs of waypoints, and the VFH+ navigation algorithm drives the robot according to those breadcrumbs. The following sections elaborate on the main software modules used in the Navigation Challenge.

6.4.1 Mapping

Mapping is an important capability for autonomous robots that facilitates good decision making. It is

Figure 11: Path Planning Debugging Tool

particularly beneficial in the Navigation Challenge, since it enables the use of path planning algorithms to determine the optimal route between waypoints. However, even in the Autonomous Challenge, the ability to maintain a global map could be used to enable the robot to partially retrace its path when it "thinks" it is in a trap. Mapping requires an accurate estimate of the robot pose (X, Y, Yaw) so that precise registration of the local map on the global map can be carried out. A Kalman Filter was implemented to estimate vehicle pose reliably by fusing data from the motor encoders, the DGPS, and the digital compass. This also allows GPS outages to be managed if they occur. The creation of a global map enables path planning to be used in the Navigation Challenge to optimize travel.

6.4.2 Path Planning

Performance in the Navigation Challenge is considerably enhanced if path planning is utilized. Path planning is carried out using the D*Lite algorithm, which provides the best route between waypoints. D*Lite can work off a partially complete map of the field, and progressively re-plans the optimal route when the map is augmented with new information as the robot explores. This is where the innovative D*Lite debugging tool proved its worth. Although D*Lite is a very effective algorithm to use for unknown terrain, its integration with a specific navigation task is difficult without a built-in debugging tool. Therefore, we designed a debugging tool which shows the breadcrumbs path from one waypoint to another as D*Lite plans (see Figure 11). Our program prints out these breadcrumbs, which update continuously according to the movement of the robot. Because the planned path can now be visualized by the programmers, debugging can be done by means of simulation.

6.4.3 Navigation

The robot navigation is carried out using a modified version of VFH+ algorithm. In this challenge obstacle grouping is not incorporated, as the challenge does not necessitate tight maneuvers, rather it requires smooth driving and careful path planning. The algorithm parameters (thresholds, swelling factor, and sector selection weights) were optimized to best serve the navigation challenge environment.

6.4.4 Speed Modulation

Similar to the speed modulation in the autonomous challenge (6.3.4), the speed modulation technique for the Navigation Challenge uses spline fitted paths to modulate the vehicle's speed. The only difference here is that the virtual target position that is used to generate a path is provided by the next breadcrumb from the D* Lite algorithm instead of the Goal Selection algorithm in the autonomous challenge.

6.5 Simulation

As stated previously, the software development environment was based on Player- Stage™; the features and merits of this choice have been discussed in Section 6.1. Stage™ provides a powerful simulation environment that can be used to develop and test algorithms in environments similar to those expected in actual competition. A substantial benefit accrues from the use of such a simulation system - the team can construct highly complex course scenarios, test and assess the performance of algorithms, and make necessary corrections

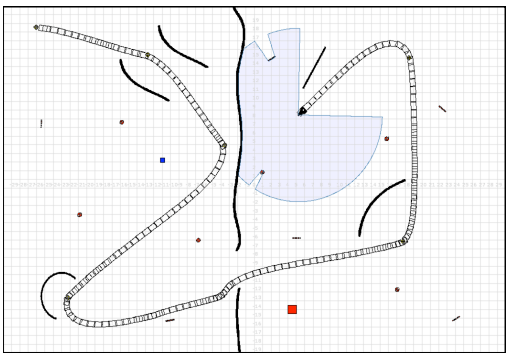

Figure 12: Navigation Simulation

much faster than with the actual vehicle. Figures 9 and 12 show sample Autonomous and Navigation Challenge simulation runs.

7. SYSTEM INTEGRATION

This project was divided into subtasks to facilitate development and assignment of tasks to individuals. However, this then requires a process to integrate all the parts into a single, working product. The team utilized the Player/Stage™ platform from the beginning. All hardware interaction was done through Player's common interface. This meant that all algorithmic code, being a Player client, could be developed and tested using the Stage™ driver set. Software system integration for *Megatron* was tested on a test course built on campus to represent possible scenarios that might be encountered at the IGVC. The physical integration of the subsystems, to fit properly and make efficient use of space, was given careful consideration during the chassis design process, by taking their dimensions and locations into account. This allowed the designers to allow for the expected footprint of each subsystem as well as the space needed for their interconnection.

8. JAUS

Our goal for the JAUS challenge was to implement a system that would meet the SAE JAUS requirements, while at the same time be compatible with each of the research robots used in our Advanced Mobility Lab (see Figure 13). We chose to implement JAUS using Jr Middleware. It is SAE AS-4 AS5669A compliant software that handles routing JAUS messages on a network. It provides an API that allows a program to create and send out a message through the use of a function call.

Our JAUS implementation interfaces with Player to obtain information, and perform JAUS-related tasks. The advantage is that our code could be compiled and run on any system that uses Player. To verify the functionalities of the system a COP program was written that would send out the messages expected at the competition and display the incoming messages from our system. This allowed us to verify the functionality of messages that are not supported this year by the JAUS validation tool.

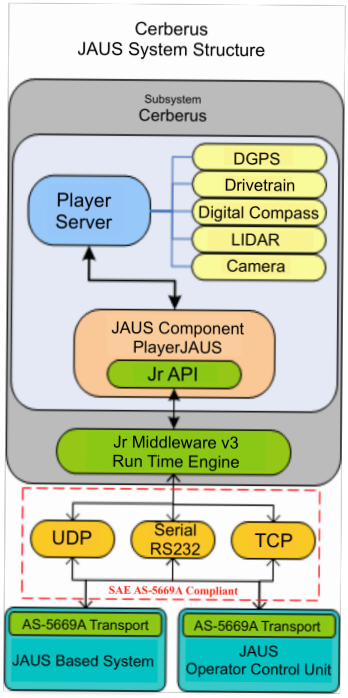

Figure 13: Megatron JAUS Architecture

8.1 Wireshark

In addition to implementing the COP software, Wireshark was used to capture and analyze JAUS messages on the network. Wireshark is an open-source network protocol analyzer, which interfaces to the computer's Ethernet or wireless card, captures all messages that are received or transmitted by those interfaces, and decodes messages based on the predefined field description of the different network stack protocol messages. Many standard protocols are supported in Wireshark such as HTTP, ICMP, TCP, UDP…etc. however, JAUS is not among those protocols.

In order to capture and analyze JAUS messages, the LUA scripting language was used to implement a JAUS message filter and dissector. JAUS messages are first filtered by the script, and then their different fields are

dissected in compliance with the JAUS documentations. At this point the filter is designed to capture and interpret only a subset of JAUS messages as defined per the IGVC rules. This functionality proved to be of a great advantage during troubleshooting and debugging. It allows the user to inspect exchanged JAUS messages as they are received and transmitted by Jr Middleware. The combined environment of COP, Wireshark, and JAUS-enabled vehicle, provided a very elaborate test-bed and validation platform to ensure the correctness of our JAUS implementation and its compliance with JAUS documentations.

9. PERFORMANCE

9.1 Speed

The theoretical maximum speed of *Megatron* is approximately 5.67 meters/second given its 40.6 cm wheel, 12:1 gear ratio, and 1600 rpm speed at optimal torque. However, the maximum speed that *Megatron* will reach during competition is capped at 2.24 meters/second (5 mph).

9.2 Ramp Climbing Ability

Based upon the rated torque output of the motors, the size of the vehicle's wheels and the selected gearing, calculations and testing have revealed that *Megatron* has ample torque to ascend an incline with a gradient of up to 30% (16.7°) without stalling. According to the IGVC rules, the vehicle needs only to be capable of climbing a 15% (8.5^0) incline.

9.3 Reaction Time

For the Autonomous Challenge, it takes approximately 100 ms (10 frames per second) to run the system algorithms (based on software timing estimates). At 5 mph, which is the maximum speed for Megatron, this cycle time translates to a decision being made approximately every 22 cm of travel. In the Navigation Challenge, the algorithms take approximately 40 ms to complete. At the 5 mph speed limit, this cycle time corresponds to a decision being made approximately every 9 cm.

9.4 Battery Life

Table 2 lists the power consumed by the vehicle components under normal as well as worst-case operating conditions. Using these values, it is expected that the vehicle will be able to run for approximately 5.7 hours under normal operating conditions and slightly over 2 hours under the worst-case conditions. These estimates have been exceeded in actual runs. A 480 W DC battery charger is positioned inside the vehicle to enable the batteries to be recharged.

Table 2: Power Consumption Estimates

Power Distribution						
	Normal Conditions			Worst-Case Conditions		
Device	Volts	Amps	Watts	Volts	Amps	Watts
LIDAR	24	0.4	9.6	24	0.5	12
Laptop	16.5	2	33	16.5	3.6	59.4
Laptop	16.5	2	33	16.5	3.6	59.4
DGPS	12	0.2	2.4	12	0.2	2.4
Compass	5	0.02	0.1	5	0.02	0.1
Camera	12	0.17	2.04	12	0.17	2.04
Motor/Controllers	24	6	144	24	20	480
Wireless Router	12	0.5	6	12	0.5	6
TOTAL POWER			**230**			**621**

9.5 Obstacle Detection Distance

The LIDAR unit on the vehicle is capable of detecting objects at a distance of 20 meters; for *Megatron*, the LIDAR is configured for a range of 10 meters. The camera is set up to view a shorter range to reduce glare and horizon effects (approximately 5 meters).

9.6 Accuracy of Arrival at Waypoints

The waypoints at the competition will be designed as concentric 2m and 1m radius circles centered on the GPS coordinates of the waypoints. *Megatron's* DGPS system provides an accuracy of + 0.1 meters in DGPS mode, and + 0.01 meters in real-time kinematic (RTK) mode. It can be seen that this accuracy is more than sufficient. This has also been demonstrated both via simulation and actual experimentation.

10. SAFETY, RELIABILITY, DURABILITY

Even though *Megatron* is a developmental vehicle, it is important for it to operate in a safe and reliable manner as well as be durable, just like any other product. The durability of its mechanical and electrical/electronic systems can be counted on because the design builds upon what worked well in our previous vehicles, while at the same time upgrading and/or redesigning what needed to be improved. *Megatron* includes several features that not only contribute to its performance, but also increase its safety, reliability, and durability. Three E-Stop systems are implemented to ensure that the vehicle can be stopped safely, quickly, and reliably. These are the soft, hard, and remote E-Stops, which are controlled by the microcontroller, the manual mechanical button on the rear of the vehicle, and the remote control, respectively. The vehicle is weatherproofed such that light rain will not cause electrical short circuits. This involves the incorporation of NEMA enclosures for the power distribution system, as well as a shell that surrounds the vehicle chassis and the various components. All electrical circuits are carefully fused to prevent electrical damage. Furthermore, individual currents and voltages are monitored in all circuits. Diagnostic software and LED indicator systems were developed so faults could be quickly identified and repaired.

Megatron implements three levels of "watchdogs" on the motor controllers to prevent unintended vehicle operation. The first watchdog is a hardware watchdog, which prevents vehicle operation in the event of a hardware failure. Every 500 ms the computer must send a specific message to the motor controller. If the message is not sent, an E-Stop is triggered. In the event of a hardware failure or computer crash, the message will not be received by the controllers and the vehicle will stop. The second watchdog is a software watchdog, to prevent vehicle operation in the event of a software failure. The motor driver will expect a new velocity command from the software algorithm at least every 2 seconds. If such a command is not received, the driver will halt the motors until a new command is received. The third watchdog monitors smooth wireless data transmission between the remote control and the vehicle. Any failure of the remote control or jamming of the wireless signal will trigger the E-Stop, acting as a hardware watchdog.

11. CONCLUSION

The UDM team is excited at the prospect of defending our IGVC title yet again with *Megatron*. There have been significant changes in the competition rules this year and we have responded by incorporating the necessary changes in our vehicle design. We will use the remaining days before competition working on improving system integration and optimizing performance.