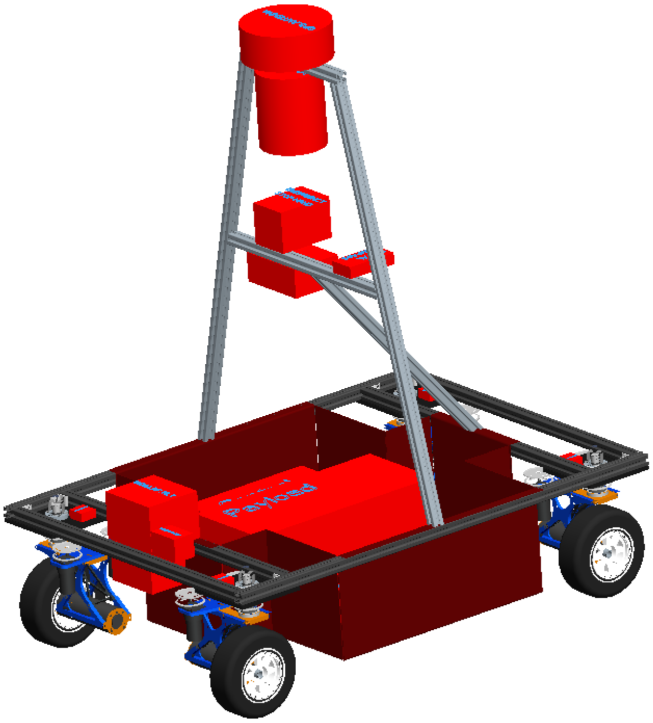
The UTA Crazy Horse Platform



University of Texas at Arlington

Required Faculty Advisor Statement (*)

I hereby certify that the engineering design and development on Crazy Horse has been significant and equivalent to what is required to award credit in a senior design course.

(*) Co-advisors: Dr. Bob Woods, Dr. Arthur Reyes, Dr. Kamesh Subbarao, Dr. Atilla Dogan

Professor Dr. Brian Huff

Industrial and Manufacturing Systems Engineering Department

Section 1.0- Introduction

The University of Texas at Arlington is proud to present Crazy Horse, a new vehicle platform designed to compete in the 17th Annual Intelligent Ground Vehicle Competition (IGVC). Our design originated from last year's platform, which was based on the Israel Military Industries MK-5 Hornet bomb disposal robot. The new Crazy Horse platform's biggest differentiation from the MK-5 includes the omni-direction motion behavior system and easily re-configurable design. Opposed to the previous vehicle, if caught in between obstacles, the Crazy Horse can translate in any appropriate direction, on any angle freeing itself from any situation. We have made a great deal of headway for the University's team in terms of our coding efforts and the mechanical design. We will further discuss the work of our teams' individual work on the mechanical design, electrical subsystems, and algorithm development teams.

Section 2.0- Team Organization

The Crazy Horse team traveling to the 17th IGVC is a multi-disciplinary cohort of students from the University's College of Engineering and integrated the technical efforts of three two-semester design course sequences: Mechanical Engineering Senior Design (MESD)- Platform Work, Electrical Engineering Senior Design (EESD)- Connection/Coding of electronic components (Plug-A-Pods, h-bridge, motor connections), Autonomous Vehicle Systems Design two-course sequence (AVSD I and AVSD II)- coding of electronic components, sensor integration, algorithm development.

The team also consists of students who completed the AVSD course sequence and participated in the IGVC Competition last year as well as new volunteers who wanted to contribute to the project. We included a table summarizing the hours spent on project.

Table 1- Team Organization Content Hours

	Dept	**HRS**
Mechanical Team	ME & AE	1500
Electrical Team	EE	600
Algorithm Development	EE	600
AVSD I / AVSD II	IE & ME	450
Volunteers	ENGR	720
	Total Hrs	**3870**

Section 3.0- Conduct of design process

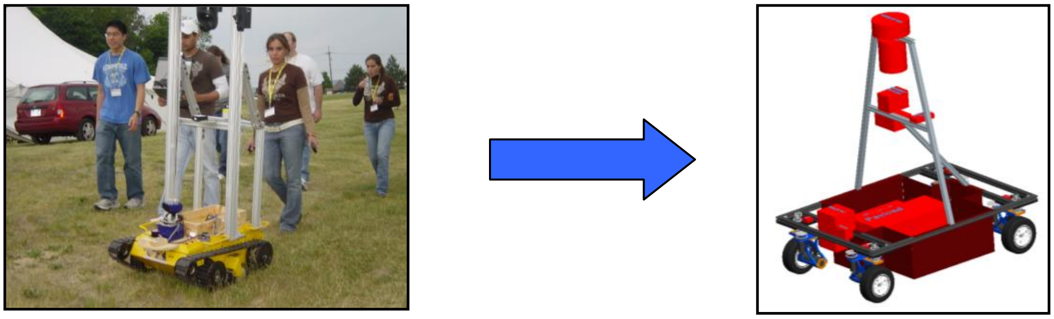
The system design process used to create our 2009 IGVC entry has followed a traditional system development. What is unique, is that this was conducted in parallel by three different groups of students and three different two semester design experiments. In the first semester each group studied the IGVC competition and developed a set of system requirements. At the compeletion of the first semester each team was expected to create a conceptual design for a system capable of meeting these requirements. Each of the three design experiences focused on a different aspect of the system's design. The mechanical design team focused on the development of the physical omni-directional vehicle platform. The electrical design team focused on the development of a distributive motion control architecture, using the CAN Bus architecture. The autonomous system design team focused on the system level integration of sensors and the motion control platform to successfully perform the IGVC Autonomous and Navigation Challenges. During the last semester, all three groups developed the sub-systems and integrated them into the vehicle system we call Crazy Horse.

In the detailed design phase, the 3D solid modeling tools Pro/Engineer and SolidWorks were used to model the overall structure of the vehicle and the Wheel Pod Assemblies. The SolidWorks models of the Wheel Pods were further analyzed in the COSMOS Finite Element Analysis package to evaluate the stresses and deflections that could be expect in this critical subsystem.

Microsoft Excel and the MATLAB analysis packages were used to cross validate the kinematic equations developed to translate the overall vehicle motion inputs into the four sets of turn angle and wheel speed commands sent to each of the Wheel Pods. Once validated, the MATLAB model was converted to a C# Dynamic Link Library (DLL) and used as part of our motion control software application. We have used Microsoft Robotics Developer Studio (MRDS) and Visual Simulation Environment (VSE) to predict the performance of our system. We have specifically investigated the use of simulation to evaluate our sensor selection and algorithm development efforts as they relate to obstacle avoidance.

Section 4.0- Innovations

For this year's entry we have elected to develop our own mobility platform through the Mechanical Engineering Senior Design (MESD) Capstone. UTA's Autonomous Vehicles Laboratory (AVL) was seeking a mobility platform that could support future research in real-time mobility control. The MESD Team was asked to design a four-wheeled platform that could support multiple motion control and steering strategies. In response to this key design requirement, the MESD team developed the Omni-Directional Mobility Platform with independent all-wheel drive and steering. The MESD team was required to design the platform to satisfy the physical and performance constraints specified within the 2009 IGVC Rules.



UTA’s 2008 IGVC Entry

UTA’s 2009 IGVC Entry

Figure 1- 2008 and 2009 Platforms

In this section highlights the major innovations that the UTA 2009 IGVC team has undertaken within the last year. The details of how these innovations have been designed, fabricated, procured, and integrated will be presented in the various sections to follow. Major areas of innovation for the 2009 IGVC Team include:

Mobility Platform Design Innovations -The 2009 competition mobility platform was designed and fabricated from scratch by the MESD team. Modularity and the use of commercially available components for the vehicle fame were key design requirements. How these design requirements were satisfied will be discussed when the design of the mobility platform is presented in greater detail below.

Innovations in Motion Control – The tracked platform used in 2008 had a relatively limited set of motion options that proved difficult to control with accuracy. Therefore, the 2009 platform was designed to support multiple modes of motion while minimizing the amount of skidding or dragging of the wheels to accomplish the turns. This vehicle is an omni-directional mobility platform consisting of four-wheel pods mounted on a simple aluminum extrusion frame. Each wheel pod contains an independent wheel steering mechanism and a motor-in-hub assembly, which directly drives the wheel. The axis of the wheel pivot is located directly over the center of the contact patch of each tire. In addition to the mechanical wheel pod design, the platform’s kinematic models have been developed to allow three vehicle centric motion parameters (turn radius, angular position of turn center, and tangential velocity of the center of the vehicle) to determine the appropriate turn angles and wheel speeds for each of the four wheel pods needed to execute a specific motion behavior. The platform can perform zero-turn-radius pirouettes, vector or crab moves in any direction, and radial turns.

Software Implementation Architecture Innovations – For the 2009 competition, we have adopted a component-based software implementation architecture that utilizes pre-existing software components and software/control development CASE tools to standardize and componentize our software development efforts. Over the last year we have investigated, and in several cases integrated, third-party software components, software development environments and architectures. Examples include: MATLAB/Simulink software component development tools, the Microsoft RDS software components including their Visual Simulation Environment, the RoboRealm machine vision toolset, and the Franson GPSTools software components.

Innovations in Distributed Controls - Our entry will be driven by eight brushed DC motors used to steer the four wheels and an additional four brushless DC motors used to drive the wheels. A Controller Area Network (CAN) will link five New Micros DSP56F803-based Plug-A-Pod controllers. Four of these controllers, with associated H-Bridges, will be used to control the two brushed steering motors on each Wheel Pod. The fifth Plug-A-Pod controller will serve as a CAN to Serial bridge with the Vehicle Driver application running on a Windows XP Pentium Processor. The vehicle will also host a seven-node TCP/IP-based local area network. Four of these nodes will be Stellaris brushless DC motor control boards, which will act as independent wheel speed control servers to the Vehicle Driver client application. The higher-level perception, obstacle avoidance, navigation and vehicle control functions will be distributed across three networked Windows XP Pentium multi-core laptop PCs. This networked multi-Laptop computer strategy supports the distribution of sensor/controller connections, facilitate run-time system monitoring, and simplify integration/maintenance tasks.

Section 5.0- Platform

Section 5.1- Structure of the Vehicle

The vehicle was designed to provide a versatile, omni-directional four wheeled platform, capable of being reconfigured to fit different control schemes. The structural pieces of the pods were created using water-jet cutting process, reducing the amount of conventional machining of the components. This reduces the time, resources, and money required to make the pods. The chassis and mast of the platform were constructed from 80/20 extruded aluminum members because it is easy to construct, sufficiently rigid, and allows components to be easily relocated. The 80/20 is also relatively inexpensive, and allows future teams to reconfigure the chassis and mast structures .

Section 5.2- Electrical Subsystems

Section 5.2.1- Inter-Laptop Communication

For the main supervisory control application, we have decided to distribute our navigation, obstacle avoidance, and motion control functions across three laptops. This architecture was chosen based on the following criteria and has the following advantages over a single processor approach: (1) Create a setup in which each team could code their individual pieces, rather than everyone waiting on one resource. (2) The whole application can be subdivided into major subsystems such as obstacle avoidance, navigation, and image processing. (3) If an application fails, it can be easily debugged and that failure would not halt the entire control system.

The laptops are connected to a local hub, and the inter-process communication is supported by TCP/IP communication protocols. This hub will also connect four motor controllers to the system level motion control software.

Section 5.2.2- Platform Electrical Systems (Actuators & Processors)

We have adopted and distributed a computing approach to drive the platform. The Omni-directional platform consists of four wheel pods. Each pod contains two motors that control the steering angle and one motor that controls the wheel velocity. This is a distributed computing approach where each wheel has its own controller and all the controllers communicate with each other over the CAN network. They will in turn communicate with the master controller over the network. We are using a distributed computing solution instead of a single controller solution because it means that we have less wires coming from each motor to the master controller and if we decide to replace any part of the system it could be done relatively easily without changing the control and programming architecture.

Processors Used: Plug-A-Pod (DSP 56F60x)- We have a processor devoted to each of the four wheels to communicate with the master processor. The master vehicle controller sends the desired turn angle to each wheel pod controller through the master Plug-A-Pod which convert steering command messages into CAN words which are sent over the CAN network. The wheel pod controller will then read the CAN message with its designated address and sets the desired angle. The wheel pod controller will also have its own internal feedback mechanism to control the desired turn angle.

The steering mechanism has two motors and both need to be controlled to make the wheel turn. The velocity of each wheel is controlled by its own microprocessor. These microprocessors communicate with the master vehicle controller over a TCP/IP network.

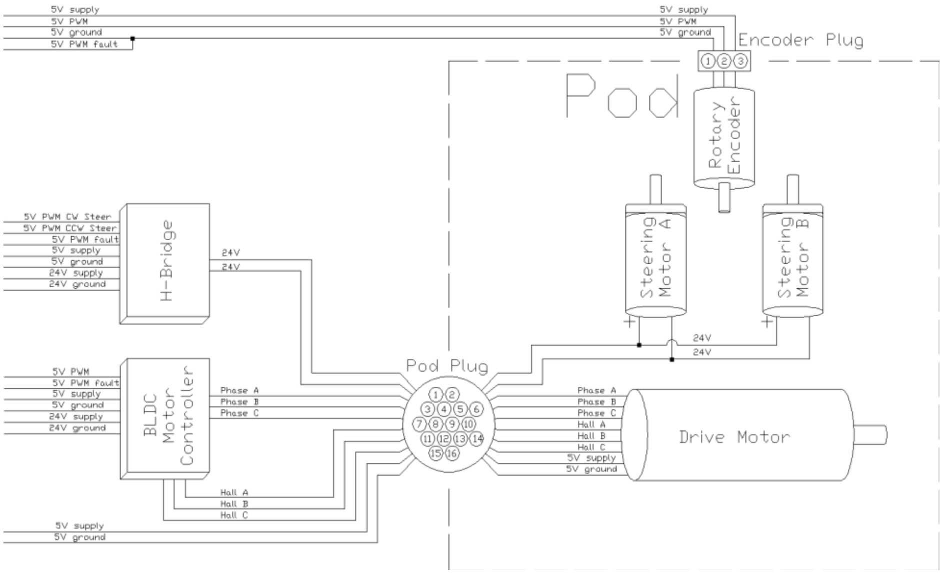


Figure 2- Plug-A-Pod (DSP 56F60x) Wiring Diagram

Section 5.2.3 Sensors- (Physical descriptions)

-The Sony EVI-D70 Color Pan/Tilt/Zoom camera supports the obstacle avoidance system. The resolution of the camera is approximately 380,000 effective picture elements. The camera accepts VISCA commands, which allows control of the pan, tilt, and zoom functions through an RS-232 connection, in order to get a wider range of vision. This gives the vehicle more response time and the ability to detect both boundary lines in the autonomous challenge.

-The Hokuyo URG-04LX Laser Range finder with a four-meter range. It has a 240-degree scanning range with 0.36 degree angular resolution. It has USB and Serial (RS-232) interfaces for maximum flexibility. Its accuracy is +10/-10 millimeters. The scan time is 100 milliseconds.

-The Opti-Logic RS100 Laser Range Finder that outputs readings to an RS-232 port. The RS100 has a range of 100 yards. Its main purpose is to detect the entrance to the Mesa during the Navigation Challenge. The laser range finder is mounted on a servo motor to provide a scanning capability for our potential fields algorithm.

-The navigation challenge sensor system is composed of a digital compass, differential GPS and four absolute encoders. The Revolution GS digital compass, a critical sensor in our system, consumes low power and renders more than 27 readings per second. The effective dynamic heading accuracy is within three degrees and the pitch and roll accuracy is within one degree. To increase the position accuracy the GPS system includes two independent receivers: a NovAtel Propak-G2plus and a NovAtel OEM4-G2 GPS Card that communicate with the CPU. These units provide the global location of the vehicle. Four USDATA Absolute Encoders, with 10,000 counts per revolution (CPR), will be used for odometry purposes in order to complement and increase the accuracy level of the information provided by the digital compass and GPS.

Section 5.4- Software

Section 5.4.1- Autonomous Challenge Strategy

In the autonomous challenge the Crazy Horse is expected to stay within the two white lines and avoid obstacles. In order to accomplish this challenge the methods described below are used.

Section 5.4.1.1- Robotic "Vision"

Robotic vision plays an essential part of the autonomous vehicle. It has been used to identify objects along a path and to provide information about these objects so navigation algorithms can be developed. This section will describe a method used for white line detection. This method provides essential inputs to the potential field's algorithm and is expected to accomplish the vision subsystem's purpose, to detect and stay within white lines that set the boundary of the path. The line detection process is illustrated in Figure 3.

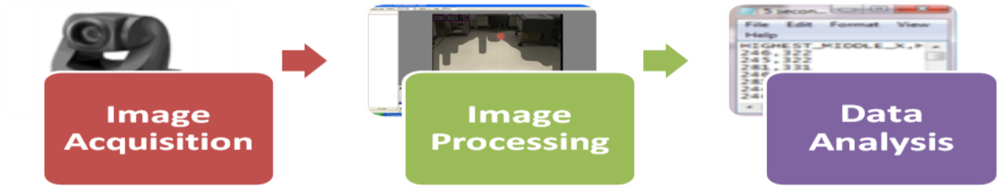


Figure 3- Vision Subsystem Process

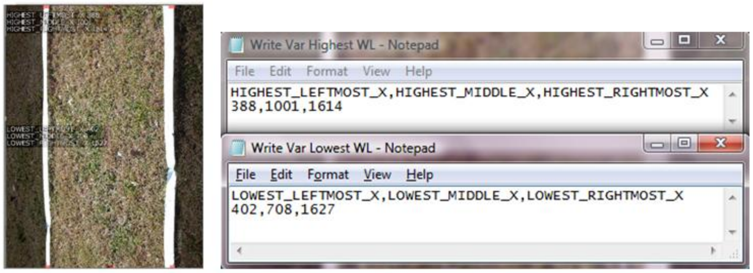
Section 5.4.1.2- RoboRealm

The RoboRealm vision software is an "application that provides a simple and quick interface for experimentation into Machine Vision using a simple WebCamera" (RoboReal.com). In the vehicle's vision subsystem RoboRealm will be used to perform image processing and the extraction of data from the processed image. The use of different modules within RoboRealm, such as the "Floor Finder" technique, will aid in achieving the subsystem's goal: white line detection.

Section 5.4.1.3- Line Detection

The white line detection technique used in RoboRealm consists of obtaining an image of what is ahead of the vehicle from the camera. A series of experiments were conducted to determine the camera height and tilt angle to obtain an adequate image of the boundary lines.

The image seen by the camera is also seen by RoboRealm. At this point, RoboRealm will perform the second step shown in Figure 3, which is image processing. One of the RoboRealm modules obtains the extreme coordinate pixels of the white lines seen by the camera. Once the white lines have been identified and the pixel points have been extracted, a calibration process is performed to convert pixels coordinates to Cartesian coordinates, which can then be processed by MATLAB. Figure 4 shows the pixel points obtained from RoboRealm and the extracted values that were sent to a text file.


Write Var Highest WL - Notepad
File Edit Format View Help
Write Var Lowest WL - Notepad
File Edit Format View Help

```
HIGHEST_LEFTMOST_X,HIGHEST_MIDDLE_X,HIGHEST_RIGHTMOST_X
388,1001,1614
```

```
LOWEST_LEFTMOST_X,LOWEST_MIDDLE_X,LOWEST_RIGHTMOST_X
402,708,1627
```

Figure 4- White Line Detection and Variables Extracted

In order to transform the pixels obtained from both methods used in RoboRealm, a set of tests were conducted to determine the best calibration process for converting pixels into feet. This calibration process will be discussed in further detail.

The calibration process allows you to accurately map what the camera sees to the real world. Once calibration is established the camera images can be used to map the white lines within the potential fields obstacle avoidance algorithms. With the white lines represented as obstacles they will be perceived as a barrior that will not be crossed.

Section 5.4.1.4- Calibration

The calibration process is performed to convert the pixel coordinates obtained from RoboRealm to Cartesian coordinates so they can be used in the Potential Fields Algorithm. Several steps were needed in order to perform the calibration of the camera. First, RoboRealm is setup using the Threshold module for the purpose of locating a light on the gridded floor. Next the camera is positioned upright, five feet above the floor and tilted to an angle of 30 degrees down from the horizontal. The VISCA commands provided in the camera control software are used to position the camera at zero degrees pan and 30 degrees tilt. Finally, the data points are taken by placing the flash light over one foot increments in the x and y real world coordinate system. Each point is located on the RoboRealm screen and the pixel x and y locations are recorded with the corresponding real world locations.

The following graphs depict the results that are determined from the testing. These show the pixel coordinates that are found in the test and the real world x and y coordinates that are used to locate the pixel coordinates. From Figure 5 we can observe that the points taken along a straight line trend toward a vanishing point. The next step was to determine how to transform the pixel data into real coordinates.

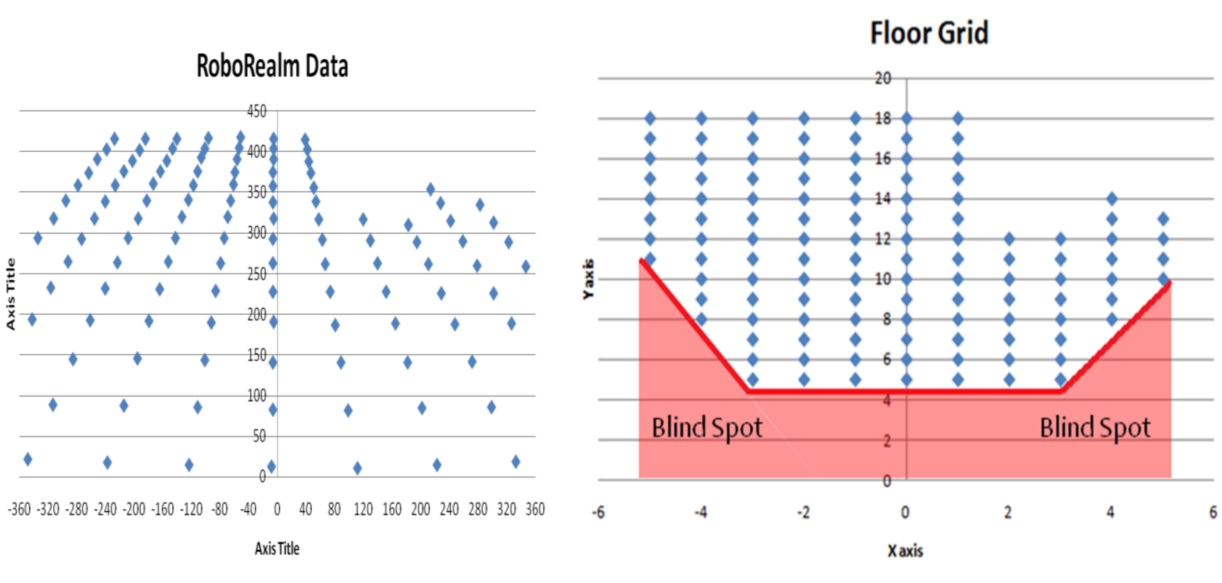


Figure 5- Pixel Locations and Real World (x, y) Coordinates in Feet

The most prominent method for calibrating camera images is the Direct Linear Transformation method. However, because of the trend in the data the decision was made to use an empirical multiple linear regression approach to derrive an equation to predict the real world coordinates. The regression of the y coordinates are first chosen, because they are horizontally in line based on the distance from the camera. Using the SAS System, a multiple regression stepwise analysis is performed using the y^2 and the y values. Both of these values are entered into this model with an R-Squared value of .9948, suggesting 99.48% of the variance within the process is accounted for by the model. The equation for predicting y real world values from y pixels is then obtained. Next, the x pixel values are plotted against the x feet values. It is noted that x values are dependent on the y values. Using the multiple regression stepwise analysis in SAS, the equation for predicting x real world values from x pixels values are obtained. It should be noticed that the equation for the prediction of x needs the y predicted value. The maximum amount of variance found in the testing was 4 inches, which is acceptable given the application. The equations obtained are as follows:

$\hat{y}$ = 0.00007431*y^2-0.00191*y+5.4004

$\hat{x}$= 0.0092*x-0.3612*$\hat{y}$+7.11E-08*y^2-1.34846

Section 5.4.2- Obstacle Avoidance:

We are going to take a Local and online approach to detecting obstacles. To support this strategy we plan to use the artificial potential fields approach. This approach is designed to support a reactive obstacle avoidance strategy. Either an obstacle or a target are the only two elements that are present in potential fields. We are attracted towards targets and repelled by obstacles. The sum of all the attractive forces and repulsive forces are used to determine the resultant vector. This determines the direction of motion to avoid the obstacles and move towards targets.

The Artificial Potential Field method is a real time control process, which is not process intensive. It is widely applied for the real time obstacle avoidance and smooth track control. Figure 6 shows two obstacles that are mapped by the artificial potential fields algorithm. The potential on each element, either obstacles or targets is given by, $V(x) = \sum_{i=1}^{N} k_i V_i(x)$

Where $V_i(x)$ is the individual potential fields from the *i*-th obstacle/target, *N* the number of obstacles and targets, and *k*$_i$ a relative weighing coefficient.

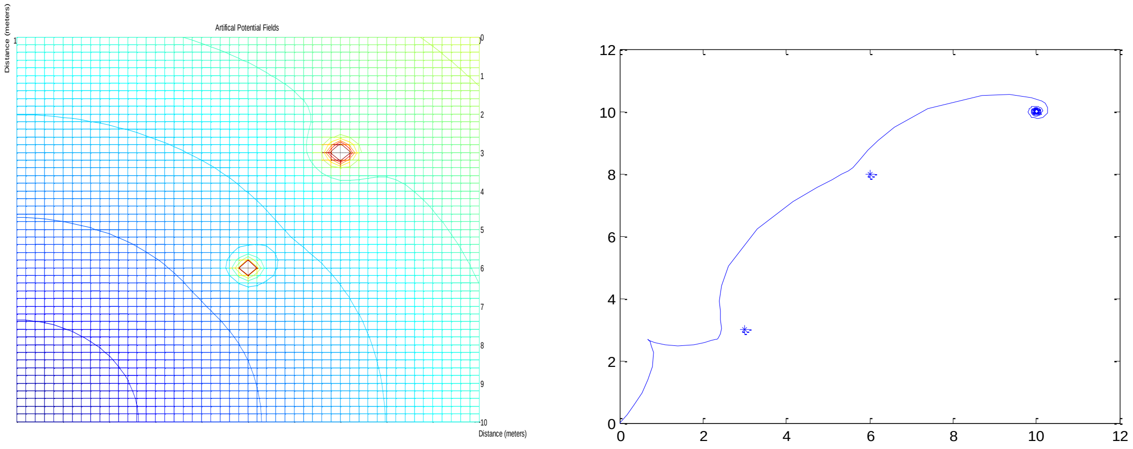


Figure 6- Magnitude Of Artificial Potential Fields With Two Obstacles In Sight

Obstacles are detected using the Hokuyo Laser Range finder. The laser range finder gives the output as a function of distance and angle of the obstacles detected. This input is given to the Potential Fields algorithm, and the outputs of the program are Magnitude and Angle. These are needed for the vehicle to avoid obstacles.

Section 5.4.3- Integration of Obstacle Avoidance and Line Detection

The strategy is to get the coordinates of the white lines using RoboRealm into MATLAB by using RoboRealm API. We parse the strings coming from LRF in C# and the array of obstacles detected is imported into MATLAB. The lines are treated as virtual obstacles and are applied different potentials compared to the actual obstacles. Then the potential field algorithm is executed to get the next angle and magnitude to avoid obstacles. Here, the whole algorithm of potential fields is implemented in MATLAB. By using a MATLAB to C# (.net builder) tool box to convert the .m files into .dll's, we can convert our obstacle avoidance algorithms into the C# environment.

This strategy gives us the flexibility to treat the actual and virtual obstacles and apply the same algorithm to both. By setting a proximity circle or by varying the constant of the algorithm, this flexible strategy allows for control for how close we get to an obstacle.

The output of this algorithm is an angle and magnitude. This is then given to two separate algorithms which calculate the speed and angle of each wheel to reach the resultant vector. As a result, two values are obtained for each wheel. The output of the potential fields algorithm is a resultant vector which is used by vehicle motion controller which are used to produce the steering and velocity inputs to each wheel.

Section 6.0- Navigation Challenge Strategy

The navigation challenge is performed by a combination of the vision , navigation and vehicle motion systems. Over the course of the year, a final strategy was developed which integrates the waypoint navigation system with the obstacle avoidance system through the use of the potential-fields algorithms previously discussed. The potential-fields algorithm output is then used to determine the motion behavior of the vehicle and the subsequent input values to each of the four motors.

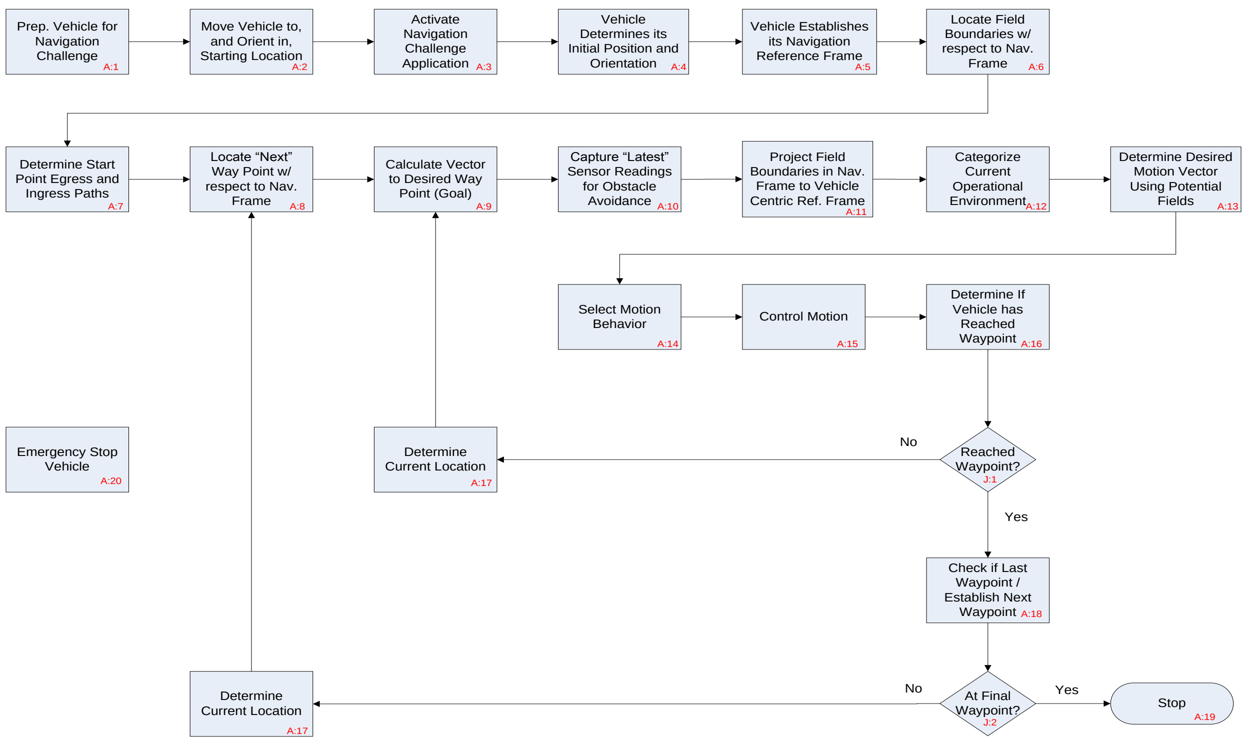

Prep. Vehicle for Navigation Challenge A:1
Move Vehicle to, and Orient in, Starting Location A:2
Activate Navigation Challenge Application A:3
Vehicle Determines its Initial Position and Orientation A:4
Vehicle Establishes its Navigation Reference Frame A:5
Locate Field Boundaries w/ respect to Nav. Frame A:6
Determine Start Point Egress and Ingress Paths A:7
Locate "Next" Way Point w/ respect to Nav. Frame A:8
Calculate Vector to Desired Way Point (Goal) A:9
Capture "Latest" Sensor Readings for Obstacle Avoidance A:10
Project Field Boundaries in Nav. Frame to Vehicle Centric Ref. Frame A:11
Categorize Current Operational Environment A:12
Determine Desired Motion Vector Using Potential Fields A:13
Select Motion Behavior A:14
Control Motion A:15
Determine If Vehicle has Reached Waypoint A:16
Emergency Stop Vehicle A:20
Determine Current Location A:17
No
Reached Waypoint? J:1
Yes
Check if Last Waypoint / Establish Next Waypoint A:18
Determine Current Location A:17
No
At Final Waypoint? J:2
Yes
Stop A:19

Figure 7- Logic Strategy For Straight Path Vector Calculation.

The navigation systems utilizes the existing logic strategy presented in Figure 7 to develop a straight-path vector calculation to a waypoint location based upon the latitude and longitude coordinates given. The output of the vector calculation, are vector magnitude (R) and the vector direction (θ). These vectors are then utilized by the potential-fields algorithm as the coordinates of the carrot target location methods.

The vector calculation application is developed in the Visual Studio environment, which utilizes C# code as per the overall software strategy previously mentioned. The application parses input data strings from the GPS receiver via the Franson GPSTools dynamic-link libraries in the NMEA GPGGA Long data structure, which outputs a degree. Decimal degree lat/long coordinate set and converted UTM northing/easting coordinate set. A subtraction method is used to dynamically calculate the tangential vector from the vehicle's current UTM location to the specified waypoint UTM location, which gives the radius of the desired turn, R-value, as input for the potential-fields algorithm and a vector heading relative to True North. The application reads the input data strings from the digital compass to provide the bearing of the vehicle relative to True North. The subtraction method is used to dynamically calculate the turning angle (θ) value input for the potential-fields algorithm.

Section 7.0- Safety, Reliability, Durability

Section 7.1- Safety

There are two emergency e-stops integrated into the system: a hardwired manual push-button positioned at the center of the vehicle and a wireless e-stop. Both are located on the vehicle mast to break the power connection between the motors and the rest of the vehicles power system necessary for a quick halt.

Additionally the maximum speed reached by the vehicle is no more than five miles per hour; which guarantees the maximum speed requirement. The electrical wiring connections are well insulated, identified and labeled to prevent errors in connecting the electrical subsystems and power supplies.

Section 7.2- Reliability

The Crazy Horse platform is designed for robustness. Engineering analysis performed on the wheel pod indicates that they can support a much higher load than we expect to experience during competition environment. Sound electrical wiring practices were used to avoid electrical faults due to lose connections

Section 7.3- Durability

System uses a rigid and 80/20 aluminum platform is capable of sustaining an impact. It is also easily reconfigurable with commercially available materials. The wheel system is also designed as discussed for an omni-directional motion behavior. This gives us less mechanical movement and more organic style of motion that will able us to prevent deadlocks between obstacles. Also the knobby tires will give us good traction in the terrain the vehicle will be operated on.

Section 8.0- Summary:

The UTA Team is proud of this year's entry in the 2009 IGVC. Crazy Horse is a totally new platform designed and fabricated at UTA. This system is considerably more complex than last year's entry. The motion platform is driven by a combination of both brushed and brushless DC motors. A total of twelve motors are used to drive the system. The vehicle is modular in both its physical design and its controls architecture. Due to its modular design, the size to the body of the vehicle can be easily modified with commercially available components. The Wheel Pods are self-contained motion modules that provide independent steering and velocity control on each of the vehicle's wheels. The Wheel Pod Motion Controllers are also self controlled units that are located directly over each of the Wheel Pods. This vehicle uses two different types of distributed controller networks: a CAN network to control the coordinated steering angles of all four wheels and a TCP/IP network linking the wheel velocity controllers and the system level control processors that handle vehicle motion control, obstacle avoidance, and vehicle navigation. A modular software architecture has been adopted that encourages the use of software components that support control development, machine vision, and waypoint navigation.

As innovative as the vehicle system itself is, the integration of three different two semester design experiences offered in the Mechanical and Electrical Engineering Departments as well a new inter-departmental two course sequence in autonomous systems design has been quite unique and challenging. This team of Mechanical, Electrical, and Industrial Engineering students at both the Undergraduate and Graduate levels have created a very dynamic and learning for all involved in this project