

Faculty Code: TP1
Project Sequence: IGV2
MQP Division: RBE

REALIZATION OF PERFORMANCE ADVANCEMENTS FOR WPI'S UGV-PROMETHEUS

A MAJOR QUALIFYING PROJECT REPORT
SUBMITTED TO THE FACULTY OF WORCESTER POLYTECHNIC INSTITUTE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF BACHELOR OF SCIENCE

Submitted by:

Mehmet Ali Akmanalp
COMPUTER SCIENCE

Ryan Doherty
ROBOTICS ENGINEERING

Jeffrey Gorges
MECHANICAL ENGINEERING

Peter Kalauskas
COMPUTER SCIENCE & ROBOTICS ENGINEERING

Ellen Peterson
ROBOTICS ENGINEERING

Felipe Polido
ELECTRICAL AND COMPUTER ENGINEERING & ROBOTICS ENGINEERING

Advisor:

Taskin Padir

Co-Advisors:

Michael J. Ciaraldi
William R. Michalson
Stephen S. Nestinger
Kenneth A. Stafford

April 27, 2011

Abstract

The objective of this project is to design and implement performance improvements for WPI's intelligent ground vehicle, Prometheus, leading to a more competitive entry at the Intelligent Ground Vehicle Competition. Performance enhancements implemented by the project team includes a new upper chassis design with a modular payload area, a reconfigurable camera mount, extended Kalman filter-based localization with a GPS receiver and a compass module, a lane detection algorithm, and a modular software framework. As a result, Prometheus has improved autonomy, accessibility, robustness, reliability, and usability.

Contents

List of Figures	v
List of Tables	vi
1 Introduction	1
1.1 Motivation	1
1.2 Intelligent Ground Vehicle Competition (IGVC)	2
1.3 Starting Condition of Prometheus	3
1.3.1 Sensors	3
1.3.2 Software Architecture	4
1.3.3 Intelligence	5
1.4 Other Competitive and Innovative IGVC Entries	6
1.5 Goals and Requirements	7
1.6 Specifications	7
1.7 Paper Overview	8
2 Robot Structure	9
2.1 Background	9
2.1.1 Chassis Mechanics	9
2.1.2 Usability Improvements	10
2.2 Methodology	12
2.2.1 Chassis Mechanics	13
2.2.2 Usability Improvements	24
2.3 Results	28
2.3.1 Chassis Mechanics	28
2.3.2 Usability Improvements	29
2.4 Conclusion	32
2.4.1 Chassis Mechanics	32
2.4.2 Usability Improvements	32
3 Sensors	33
3.1 Global Positioning System	33
3.1.1 Background	34
3.1.2 Methodology	34
3.1.3 GPS Receiver Results	35
3.1.4 Conclusion	38
3.2 Inertial Measurement Unit	39
3.2.1 Background	39
3.2.2 Methodology	40
3.2.3 IMU Results	41
3.2.4 Conclusion	43
3.3 Encoders	43
3.3.1 Background	43
3.3.2 Methodology	44

3.3.3	Results	44
3.3.4	Conclusion	45
4	Software Architecture	46
4.1	Background	46
4.1.1	Joint Architecture for Unmanned Systems	48
4.1.2	Robot Operating System	48
4.2	Methodology	48
4.3	Results	53
4.4	Conclusion	56
5	Intelligence	57
5.1	Sensor Fusion	57
5.1.1	Background	57
5.1.2	Methodology	59
5.1.3	Results	63
5.1.4	Conclusion	65
5.2	World Representation	66
5.2.1	Background	66
5.2.2	Methodology	67
5.2.3	Results	72
5.2.4	Conclusion	73
5.3	Path Planning	73
5.3.1	Background	74
5.3.2	Methodology	75
5.3.3	Results	78
5.3.4	Conclusion	81
5.4	Vision	81
5.4.1	Background	81
5.4.2	Methodology	87
5.4.3	Results	90
5.4.4	Conclusion	91
6	Conclusion	92
A	Authorship	94
B	Frequency Spectrum Analysis	95
C	Budget	98
D	Products	102
E	ION lawn mower competition	107
F	Team Links	109
	Works Cited	110
	The 19th Annual Intelligent Ground Vehicle Competition Rules	114
	The Eighth Annual Robotic Lawnmower Competition Rulebook	143

List of Figures

1.1	Prometheus 2010 and components	2
1.2	Example of an obstacle course at the IGVC with Prometheus 2010 in the foreground.	2
1.3	The major sensors and computing devices of Prometheus 2010. In this design, the cRIO plays an essential role in parsing data from the sensors.	4
1.4	City Alien, City University of New York's entry for the 2010 IGVC	6
1.5	Prometheus 2010 at IGVC with a typical obstacle course in the background	8
2.1	Commercial weatherproof RGB LED strip	13
2.2	Measurements for describing the position of the camera and calculating the distance of the field of view from the robot	15
2.3	Comparison of the location of the camera in the old and new Prometheus designs.	17
2.4	Side and top view of Prometheus with the field of view displayed in red	18
2.5	New camera cover design adjustable for yaw, pitch, height and baseline	19
2.6	Planar knurling used to keep rotational joints from slipping	19
2.7	The locations where satellite signals could not be received indicated in red at left, and the final location of the receiver at right	20
2.8	Prometheus with a Unmanned Ariel Vehicle (UAV) landing pad attached to the modular payload area	21
2.9	Corner joint of the frame	22
2.10	Extruded t-slot joint fixture viewed from the top	22
2.11	Extruded t-slot joint fixture viewed from the bottom	22
2.12	Fixture for one joint of an extruded t-slot frame construction	23
2.13	A two dimensional comparison of force distributions for setscrews on a shaft and a keyed shaft. The force distribution is depicted by the red arrows, and the direction of rotation is indicated by the grey arrow.	23
2.14	Keyed shaft	24
2.15	External interface concept design	25
2.16	Commercial weatherproof 3M Touchscreen Monitor	26
2.17	Futaba 2.4GHz 6-channel Transmitter	27
2.18	Futaba 2.4GHz 6-channel Receiver	27
2.19	5V PWM driving circuit for RGB LED strip	28
2.20	Computer compartment with wiring properly secured	30
2.21	Different visual cue colors representing different states	30
2.22	Prometheus external interface and visual cue	31
2.23	Prometheus touchscreen and remote control	31
3.1	Plot of Sokkia Axis3 DGPS data on September 30, 2010	35
3.2	Plot of Sokkia Axis3 DGPS data on October 7, 2010	36
3.3	Plot of MediaTek data on October 7, 2010	37
3.4	Plot of Trimble data on November 6, 2010	37
3.5	Plot of Trimble data on January 16, 2011	38
3.6	Indoor calibration results of PNI IMU	41
3.7	IMU EMF Error Compensation LabVIEW routine	42
3.8	IMU output with linear compensation	42

3.9	Quadrature encoder functionality, courtesy of National Instruments (enc, 2011)	43
3.10	Us Digital E8P Optical Encoder	44
3.11	Sample encoder output information for 3.5m linear run	45
4.1	The major sensors and computing devices of Prometheus 2011. In this design, the cRIO is primarily used for motor controls and odometry information.	49
4.2	Rviz, a tool that is integrated with ROS, being used to visualize Prometheus as it performs indoor mapping.	50
4.3	The handshake used to send configuration information between the cRIO and On-Board Computer.	52
4.4	A simulation of the robot performing navigation on a map loaded from an image file.	55
5.1	The frame of the robot relative to the world frame. The world frame's x-axis always points north, it's y-axis always points west, and its origin is always positioned at the starting position of the robot.	59
5.2	Operation of the extended Kalman filter. Image courtesy of Welch & Bishop (1995)	60
5.3	LabVIEW Logging of Raw and Filtered Path from Sensor Data	63
5.4	LabVIEW Logging of Raw and Filtered Path from Sensor Data	64
5.5	Map created using GMapping on Prometheus using LIDAR for range data and encoders for odometry information. This test was performed using teleoperation to navigate Prometheus on the first floor of Atwater Kent Laboratories at WPI.	68
5.6	Comparison of a picture of the environment and an estimation of the environment state using GMapping with LIDAR and encoders in an outdoor environment. This test was performed using the same techniques as in Figure 5.5 in a different environment.	68
5.7	Class diagram of different <code>Measurement</code> objects.	70
5.8	Class diagram of the core classes of the <code>local_map</code> package.	71
5.9	The progression of the techniques used for creating the local map from LIDAR data. All techniques used a maximum age of 10 seconds for the LIDAR measurements.	73
5.10	Paths generated with A* usually do not follow a straight line from the source to the destination even when all cells are marked as unoccupied.	74
5.11	The arced paths that are part of one of Prometheus' speed sets	75
5.12	The inflated grid cells of an arced path. These cells account for the width of the robot, and can be overlaid on the local map to evaluate the drivability of a path	76
5.13	Algorithm for determining the edges of the classification area of a path.	76
5.14	Tentacle planning and A* path planning being used together to drive to a goal. The drivable arced paths are shown in green and the undrivable ones are in red. The best path as planned by A* is drawn in blue. Prometheus uses A* to choose which drivable arced path to follow. The dark green grid cells show the cells that the robot will drive through as it follows its chosen arced path.	77
5.15	The inflated footprint being used to prevent Prometheus from continuing to follow a path that will cause it to strike an obstacle.	78
5.16	Class diagram of different <code>Path</code> objects with getters omitted for brevity. The <code>BasicPath</code> and <code>ArcedPath</code> objects are used for basic instantiations of the <code>Path</code> class. <code>BasicPath</code> and <code>ArcedPath</code> are used for the A* planner and tentacle planner respectively. The decorator pattern is used for both the <code>RatedPath</code> and <code>GridPath</code> objects. These objects append additional information to the <code>BasicPath</code> and <code>ArcedPath</code> objects	79
5.17	Each type of planning algorithm extends the <code>PathPlanner</code> objects. Only one type of planner can be used at a time. To combine A* with tentacles, the <code>TentacleChooserStrategy</code> object holds a reference to a <code>AStarPlanner</code> object.	79
5.18	UML for some of the objects used by the <code>TentaclePlanner</code> class. Two of the objects <code>TentaclePlanner</code> is composed of make use of the strategy pattern. The reason tentacle evaluation and tentacle choosing is handled by two separate objects is that it makes it easier to implement safe choosing functions that are guaranteed to avoid obstacles.	80

5.19	Prometheus using tentacles to traverse a curved hallway in simulation. Undrivable paths are shown in red, and drivable paths are shown in green and blue. Blue indicates the path the robot has chosen to drive on when using a tentacle choosing function that always chooses the center-most tentacle from the largest continuous grouping of drivable tentacles.	80
5.20	The effect of median filtering on an image. Take note of sharp edges in the control image on the left and the lack thereof on the right.	82
5.21	The effect of morphological erosion and dilation on an image. Notice how on the eroded image the smaller shapes are removed but the larger shapes have smaller borders. Similarly, on the dilated image, all objects are larger.	83
5.22	The effect of a morphological opening compared to the raw image and to a morphological erosion. While in both the erosion and opening the same objects are removed, the opening retains the original sizes of the objects while the erosion makes them smaller.	83
5.23	The effect of channel splitting. Adding the red, green and blue channels result in the original image.	84
5.24	The effect of thresholding. The binarization is apparent in the right image since the only remaining 2 colors are red and green.	85
5.25	The effect of Canny edge detection and a subsequent Hough Transform.	86
5.26	The effects of Inverse Perspective Mapping.	89
5.27	Demonstrating the three phases of detection.	90
6.1	Prometheus navigating during the April 8, 2011 demonstration in Institute Park	92
B.1	Frequency spectrum analysis	96
B.2	Frequency spectrum analysis	96
E.1	Lawn mower four bar linkage	107
E.2	Partially constructed lawnmower attachment	108
E.3	Weatherproof lawn mower electrical connection box	108

List of Tables

2.1	Comparison of the different options for using a manual controller with Prometheus	12
2.2	Description of the variables for camera position dimensions	15
2.3	Comparison of possible configurations for the camera mount on Prometheus	16
2.4	Visual cue states	27
3.1	Comparison of specifications of three of the DGPS receivers that were considered for use with Prometheus. A more complete table can be found in Table D.1.	34
3.2	GPS Error Measurements	38
3.3	TCM and HMR3400 comparison	40
4.1	Known difficulties in creating complex programs in LabVIEW for the cRIO	47
4.2	Comparison of the advantages and disadvantages of using a single process or multiple processes when creating software, assuming code in not intentionally obfuscated	47
4.3	Comparison of the advantages and disadvantages of different robotics software frameworks that were considered for use on Prometheus	51
4.4	Comparison of the advantages and disadvantages of using a single process or multiple processes when creating software, assuming code in not intentionally obfuscated	53
5.1	Distance tests for the encoder and IMU filter (Filter 1) based on R and starting covariance	65
5.2	Comparison of the advantages and disadvantages of using a small, local map positioned relative to the robot versus a large scale map positioned relative to the world frame.	69
C.1	Overall budget.	99
C.2	Chassis budget.	100
C.3	External interface budget.	101
D.1	Comparison of specifications of DGPS receivers that were considered for use with Prometheus	103
D.2	Available Motor Controllers Comparison	104
D.3	Lithium Ion Battery Comparison for a 24V 55Ah system	105
D.4	Available Touchscreen Monitors Comparison	106

Chapter 1

Introduction

1.1 Motivation

The Major Qualifying Project (MQP) acts as the capstone or culmination of studies to a Worcester Polytechnic Institute (WPI) undergraduate student's education at the university. This project is directly related to the "major field of study and should demonstrate application of the skills, methods, and knowledge of the discipline to the solution of a problem that would be representative of the type to be encountered in one's career" (WPI, 2010). The major fields of study of the students completing this project are computer science, electrical and computer engineering, mechanical engineering, and robotics engineering.

This MQP involves the study of autonomous robotic ground vehicles. There has been a need for intelligent ground vehicles over the past two decades. In 1990, the Department of Defense (DoD) consolidated several ground vehicle development projects under the Joint Robotics Program (JRP), which is directed by the Office of the Secretary of Defense (OSD) (Pike, 2011). The Defense Advanced Research Projects Agency (DARPA), which commissions advanced research for the DoD, sponsors the DARPA Grand Challenge, in which teams enter intelligent ground vehicles. The vehicles that are typically seen in this challenge are automobiles, which have been transformed to operate autonomously (DARPA, 2011).

The VisLab Intercontinental Autonomous Challenge involved two unmanned vans that were tasked with traveling about 8,000 miles from Milan, Italy to Shanghai, China. The unmanned vehicles follow two driven vans, from which the unmanned vehicles take visual cues. The unmanned vehicles are responsible for avoiding obstacles on their own (Vislab, 2010).

This MQP is based on the autonomous robotic ground vehicle named Prometheus. Prometheus, shown in Figure 1.1, is the creation of a 2010 WPI MQP group which built it from the ground up. Prometheus had a custom-welded frame, two driven wheels in back, and one steered wheel in the front. The vehicle used an array of sensors which included a differential GPS receiver, a digital compass, cameras, and a light detection and range (LiDAR) device (LIDAR). These sensors constantly collect and process information about the robot's environment. The robot was designed with specifications, outlined below, that would allow it to compete in the 2010 Intelligent Ground Vehicle Competition (IGVC). Next, an overview of the IGVC is provided.

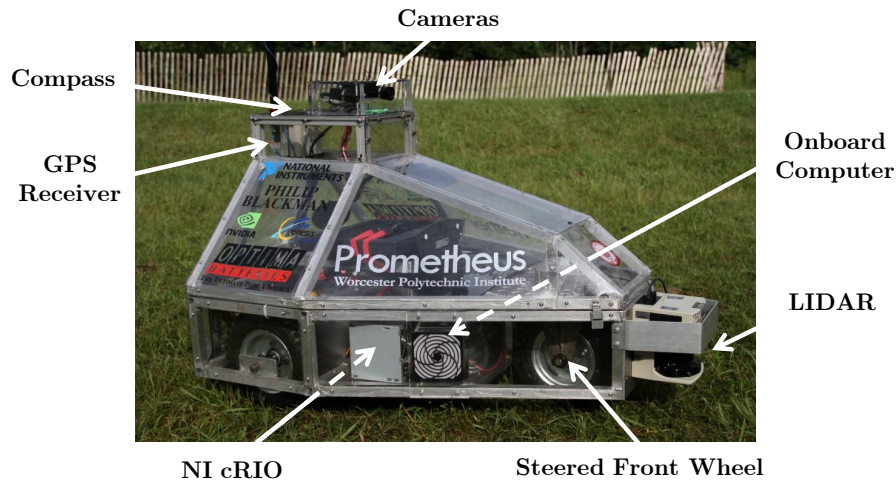


Figure 1.1: Prometheus 2010 and components

1.2 Intelligent Ground Vehicle Competition (IGVC)

The IGVC requires teams of undergraduate and graduate students to design and build an autonomous ground vehicle. There are several requirements of the vehicle which must be considered during the design phase. These requirements include minimum and maximum speed limits, size specifications, emergency shut offs, etc. The goal of the ground vehicle is to compete in a variety of challenges that change from year to year. The 2011 competition will include autonomous, design, navigation, and Joint Architecture for Unmanned Systems (JAUS) challenges. The IGVC rules can be found on page 114 at the end of the report.

The autonomous challenge requires the ground vehicle to navigate through an obstacle course. The robot must avoid the randomly placed obstacles while at the same time staying within the lane designated by spray painted lines. An example of an IGVC obstacle course is shown in Figure 1.2. The amount of the obstacle course that was completed, as well as time it took to do so are factors that affect scoring.



Figure 1.2: Example of an obstacle course at the IGVC with Prometheus 2010 in the foreground.

The design challenge requires each team to document and present the design strategy and process used to develop their intelligent ground vehicle. Documentation is in the form of a fifteen page or less report which includes specifications, innovation, the systems used, and the organization of the team. The presentation

is given to a panel of judges and allows the teams to make use of graphical aids to expand upon what they discussed in their paper. The judges then do an examination of the vehicle and score it based on the categories specified in the IGVC rules.

The navigation challenge requires the ground vehicle to travel from its starting point to an array of waypoints. These waypoints are provided to the teams in the form of latitude and longitude and may have obstacles in between them. The robot must go to these waypoints in any order it decides and then return to its starting point. Number of waypoints travelled to, as well as time it took to do so, are the factors that affect scoring.

The JAUS Challenge requires each team to interface with a judge's Common Operating Picture (COP) to communicate and provide information using the JAUS protocol as requested by the judges. This basically entails connecting to an external wireless router and relaying information to the judge's computer.

There are several benefits of competing in the Intelligent Ground Vehicle Competition. Participants are working with new technology and applying it to the design and creation of their vehicles. There is a wide range of student specialties that can be applied to the project tasks, which allow for hands-on education and experience. This not only improves the student's particular skills, but also provides the opportunity to work on a multidisciplinary team and contribute to the ideas of the group (IGVC, 2011).

1.3 Starting Condition of Prometheus

The 2010 MQP team returned from the IGVC with the Rookie of the Year Award. Being honored as the best first year team was a huge accomplishment and provided a solid foundation for future teams to build upon. Prometheus did well in the JAUS competition, had obstacle avoidance capabilities, and was innovative enough to impress the judges during the design competition. However, despite being able to avoid obstacles, Prometheus was not competitive in the autonomous challenge due to its lack of ability to detect lines effectively. Prometheus also could not compete in the navigation challenge. The navigation challenge relies heavily on the use of an accurate Global Positioning System (GPS). Although Prometheus was equipped with a differential GPS that had been tested and implemented in the past, it experienced accuracy problems at the competition, which rendered it useless for the challenge. The lack of a GPS in addition to non-functioning sensor error reduction made localization inaccurate as well.

1.3.1 Sensors

Each sensor was connected to either the NI cRIO or the On-Board Computer. All sensors except for the cameras were attached to the cRIO. An overview of this topology can be seen in Figure 1.3.

To input image data, a pair of Point Grey FL2C-08S2C cameras with CCD sensors were utilized, which are capable of streaming color images with a resolution of 1024x768 at 30 fps. The cameras were placed on top of the robot, both facing the forward direction, and separated by approximately 20 cm. An IEEE 1394b connection between the computer and each camera supplied data transfer capabilities as well as power.

Prometheus 2010 gathered a range of odometry data. Heading data was collected with a PNI Compass-Point V2Xe 2-axis digital compass. This provided Prometheus with its heading in degrees. Prometheus

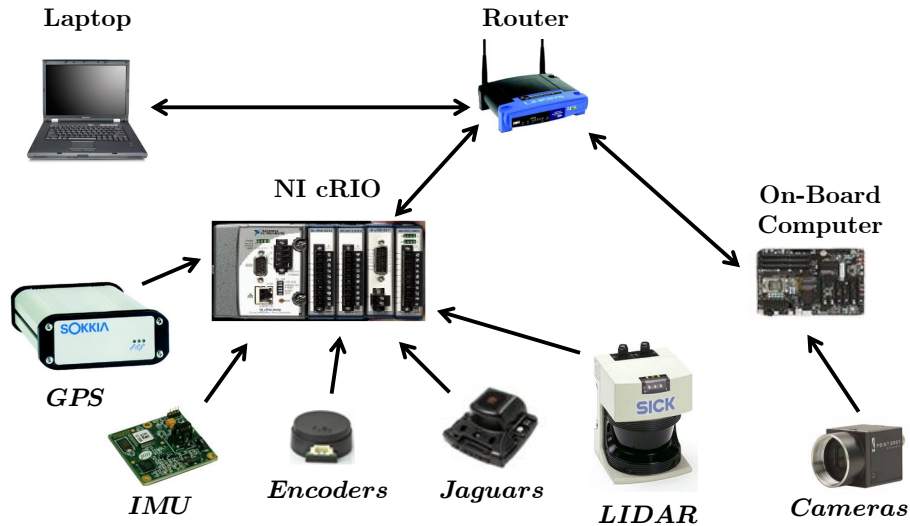


Figure 1.3: The major sensors and computing devices of Prometheus 2010. In this design, the cRIO plays an essential role in parsing data from the sensors.

2010 also gathered odometry data using quadrature encoders on both of the back, driven wheels. To aid in localization, the team used a Sokkia Axis 3 differential GPS receiver, which delivered promising results throughout the year. A SICK LMS291-S05 light detection and ranging (LIDAR) sensor was used for obstacle detection.

1.3.2 Software Architecture

The intelligence for Prometheus 2010 ran in a single process on the On-Board Computer. The process was responsible for performing localization, storing the world map, performing path-planning, communicating with the cRIO and Command Center Computer with the 2010 Prometheus Protocol, and competing in the IGVC JAUS challenge.

In an effort to make the code for the On-Board Computer more understandable, the 2010 MQP Team divided it into several libraries. Unfortunately, each library remained highly coupled with the other libraries, and the team's goal was not achieved. Regardless, they continued to use this model to compile their code, and they planned to refactor the source code at a later date.

The 2010 MQP Team also made some use of the OpenJAUS 3.3 framework to structure their code. In theory, this would have allowed their code to be more modular and extensible. Unfortunately, the team did not use the full potential of OpenJAUS, and merely used it for the main entry point of the On-Board Computer software. The team also defeated the purpose of using JAUS by defining their own messaging protocol and running everything in a single process.

The messaging protocol that the 2010 MQP Team designed and implemented was very simple. The protocol made use of UDP datagrams up to 4000 bytes in length that each used 9 bytes for header information and 3991 bytes for data. Messages were used to communicate between the cRIO, On-Board Computer, and

Command Center Computer.

Because of its simplicity, the protocol was easy to implement in a variety of environments. The 2010 MQP Team implemented programs in C++, Java, and LabVIEW that all communicated using this protocol (Wang et al., 2010). Unfortunately, this protocol has poor extensibility. While the protocol supports packages up to 4000 bytes in length, the format in which data is sent is very limited because there is no control over how this data is interpreted. Programs need to be hard-coded to interpret the package data a certain way (Wang et al., 2010). It would be difficult to change the format of the data for a certain message. For example, changing the number of bytes used to store a value could prove to be cumbersome as the programmer will need to change the length of the value in other areas of the code as well. Fortunately, frameworks like JAUS and Robot Operating System (ROS) make it easy to define new messages so that the format of a message only needs to be changed in one place.

The 2010 MQP Team ran into several problems with the software.

- Difficult compilation errors
- Unintuitive program flow
- Lack of testing

1.3.3 Intelligence

The 2010 MQP Team used a finite Cartesian map to store the state of the environment. The map was split into squares of size 10 cm by 10 cm, and the robot was always positioned in the center of the map, while other objects were positioned relative to the robot. The probability of an object occupying a grid space would decrease over time or disappear altogether when the robot moved out of range. The mapping collected data from the LIDAR to calculate the probability of an object occupying any of the nearby locations.

The image data from the two cameras was processed on an NVIDIA Tesla C1060 General Purpose GPU. The data was interpreted stereoscopically to enable the gathering of information regarding the distance of objects from the robot. To achieve this, the data went through two distinct processes: rectification and pixel matching. The rectification step takes the two images from the two cameras and transforms them onto a single plane. The pixel matching step takes the two transformed images and matches up pixels between the images that represent the same point in space. These two steps combined took around 53 ms of time. In addition to these two processes, an additional empirical process of camera calibration was used to obtain calibration matrices that describe the true parameters of the setup, such as the focal lengths of the cameras, the skew coefficient between the cameras and the radial and tangential distortions of each camera. These matrices are used to correct the depth data when calculating the 3D location of a pixel in the image. The calibration step is done manually, by displaying a known structure to the camera multiple times from different viewpoints. It must be repeated every time the orientation of the cameras is changed. Finally, using this information, the image was segmented using the self-organizing map and simulated annealing (SOM-SA) algorithm as described in (Dong & Xie, 2005), to be used in object detection and line detection (Wang et al., 2010).

The 2010 MQP team also performed navigation by creating a set of arced paths as described in von Hundelshausen et al. (2008). Each arced path originated from the robot's turning center and could be summarized by two motor commands, one for each the left and right wheels. During the navigation challenge,

the robot used a heuristic that took into account the drivable length of each arced path as well as the robot's distance to the goal after following the path to decide which path to drive on. A description of the chosen path was sent to the cRIO where it was converted to motor commands.

1.4 Other Competitive and Innovative IGVC Entries

The IGVC gathers over fifty teams from across the nation for a multi-day contest. The competition itself can be broken down into specific task requirements, including line detection, obstacle avoidance, sensor fusion and path planning. The teams aim to most effectively accomplish these tasks, ultimately cultivating a huge variety of hardware/software approaches and implementations. Listed here are some of the noteworthy approaches researched. They were chosen based upon effectiveness and innovation, both characteristics valued by the IGVC.

Path planning, in particular, consists of the problem of generating a drivable path between two points in a local map. Some path planning algorithms previously implemented in award winning IGVC robots include Anytime Dynamic A* algorithm (Newman et al., 2011), and a combination of D* Lite and Field D* algorithms (Abiola et al., 2010) by Princeton University, a Dynamic Delaunay Triangulation-based planning approach (Chen et al., 2010) by the University of Detroit Mercy, and a modified Vector Field Histogram (VFH) algorithm (Nepal et al., 2009) by Trinity College.

Outdoor line detection can be by itself a very challenging problem due to the dynamic light conditions, and varying glare sources. Line detection implementation approaches vary widely throughout IGVC teams. Some of the prominent approaches include a RANdom SAMple Consensus(RANSAC)-based algorithm (Newman et al., 2011) by Princeton University, which is a multi-step process consisting of plane extraction, morphological analysis, edge extraction and interpolation (Nepal et al., 2009) by Trinity College, and a row-adaptive statistical filter followed by a global threshold comparison (Chen et al., 2010) by Detroit Mercy.



Figure 1.4: City Alien, City University of New York's entry for the 2010 IGVC

For obstacle detection most teams rely on a LIDAR sensor due to its reliability. Nonetheless, teams have experimented with different approaches. Stereo vision, as implemented by Princeton University (Newman et al., 2011), is a financially cheaper solution, but it is also a more computationally challenging approach to the problem. Moreover, hybrid approaches that use both computer vision and LIDAR input, such as (McKeon et al., 2006) can improve the map generation. Noteworthy to this category is the City University of New York single camera 360 degree 3-D obstacle and line detection system (cit, 2010).

1.5 Goals and Requirements

The goal of this MQP is to expand upon Prometheus' capabilities, bring it to a more competitive state, and create a more efficient and robust chassis. Competitive state means that the robot will qualify to compete in all IGVC challenges (example shown in Figure 1.5) and be a contender to place in all of them. An efficient and robust chassis means inner components of the robot are easily accessible, the chassis can be expanded upon during future research projects, and the robot is durable in an outdoor environment. The following are our project requirements:

- Develop a competitive entry for the Intelligent Ground Vehicle Competition
- Autonomously navigate through a flat, outdoor, static environment given no prior state information
- Autonomously follow and stay within a path marked by two lines on a grassy field
- Plan a path and autonomously navigate to several GPS coordinates that are given as input while avoiding obstacles
- Create a rugged robot that can function in a variety of weather conditions
- Develop a manual control system that is intuitive and easy to use

1.6 Specifications

Based on these requirements, we have developed a set of design specifications:

Mechanical

- Autonomously navigate between 1 and 5 MPH
- Operate under sunny and rainy conditions
- Improve usability by simplifying peripheral connections
- Allow for direct interaction with the robot (no laptop)
- Make start-up manual mode take less than 30 seconds
- Have reliable wireless e-stop command from at least 50 feet away
- Create a safety light / visual cue relating state of the platform at 50 feet away
- Have a turning radius less than 5 feet at 1 MPH
- Ability to quickly load and carry a 18in.x8in.x8in. 20lbs payload
- Improve chassis to allow for future research possibilities (eg. Robotic arm and helipad)

Sensing

- GPS localization under 0.5m

- Detect 3-inch white lines on grass, outdoors, on a sunny or cloudy day
- Filter odometry sensor error to be lower than unfiltered error

Intelligence

- Maneuver through obstacles that are a minimum of 6 feet apart
- Generate the shortest path between several waypoints, and drive through them while avoiding obstacles
- Communicate with the judges' common operating picture using JAUS
- Account for up to 15% incline deviations on the field while driving autonomously



Figure 1.5: Prometheus 2010 at IGVC with a typical obstacle course in the background

1.7 Paper Overview

Prometheus has several different subsystems and each of these subsystems was analyzed and produced results. Because of this, the paper is broken down in a manner that addresses each subsystem individually. The four subsystems focused on, chosen at the highest possible level, are Robot Structure Design, Sensors, Software Architecture, and Intelligence. The smaller subsystems that fall under each of these chapters are addressed as well. Each chapter is broken down into sections, with a general introduction, background, methodology, results and analysis, and a conclusion.

Chapter 2

Robot Structure

Prometheus' main purpose is to be a functional autonomous ground vehicle. However, for the vast majority of time it operates under testing conditions. Most of the research and prototyping on Prometheus is done on one specific subsystems at a time, such as vision detection, mapping, or sensor fusion. Our team believes having an emphasis on the human-robot interaction portion of Prometheus can ultimately make the developing, debugging and testing phases happen significantly faster and smoother. In order to accommodate such a requirement we came up with a multifaceted solution. Prometheus' chassis is undergoing an update that will include, but is not limited to:

- Creation of a robot external interface panel.
- Top cover redesign for environment isolation and sensor reposition.
- Addition of a touch screen monitor to the chassis for information transmission and control.
- Addition of a visual cue informing the current status of the robot.
- Smoother control under manual mode.

2.1 Background

This section summarizes the analysis that was used to determine areas of improvement on the 2010 Prometheus design with regards to the physical structure of the robot. This is broken down into two main categories - the chassis mechanics and the user interface.

2.1.1 Chassis Mechanics

The 2010 Prometheus cover met the previous year's design specifications. The cover included mounting areas for cameras, GPS receiver, wireless e-stop receiver and compass. Also, the cover was fully weather proof. Meeting these design specifications worked well for this version of Prometheus, but there were a number of shortcomings that needed to be addressed. Some of the errors with the camera functionality were determined to be caused by the camera position. For example, the glare of low level light during sunrise and sunset made it very difficult to detect white lines on grass. Also, the minimum distance visible in front of the robot was more than two feet in front of the robot. This caused problems with line detection when

the robot made sharp turns near lines. Another problem with the 2010 design is that the entire cover must be opened to change the batteries. While the cover protects the electronic components from the elements during operation, all these components are exposed when the batteries must be exchanged.

The 2010 cover design was built with the intention of keeping all the sensors attached to specific locations on the cover. Making changes to the location of sensors is difficult and would require the restructuring of the cover. This type of single purpose design does not make additions or modifications to the robot very easy. As Prometheus transitions toward use as a research platform, it is important to be able to add other devices and sensors to the robot with minimal changes.

One of the problems with the current design is that the driven pulley is attached to the wheel yoke by two set screws on portions of the steering shaft. This has created two problems. One is that when forces are applied to the wheel when turning, the shaft had been worn down allowing the set screws to consistently loosen. This leaves play in the steering mechanism which means that the front wheel does not immediately react to the driving motor.

Another issue with the current system is that the motor is used to hold the wheel in a stationary position. This was a problem with previous driving motors. The motors were intended to be self-cooling when spinning. However, being stationary prevented them from cooling down and led to two motors burning out. The previous team replaced the smaller motor with a much larger motor that used much less of its capacity to hold the wheel in place and could more easily dissipate the heat. This solution has not shown any problems and therefore investigations of the improvements to the steering system have been focused on the shaft connection to the pulley rather than the drive system.

On further analysis of the front wheel steering mechanism, it was observed that the only thrust bearing on the assembly was the turntable bearing made with 0.029 inch galvanized steel. This particular bearing is rated for a load capacity of 500 lb. While the weight from the robot on the front wheel is much less than this, the forces are very dynamic. Rough terrain can cause large accelerations and jerks on the turntable. Excessive stress placed on the pieces of the turntable can be alleviated with a more effective use of a more robust thrust bearing.

2.1.2 Usability Improvements

External Interface

The need for an external interface occurs from the recurring interaction with the many different hardware components susceptible to weather damage. During debugging there is a need to directly interact with Prometheus' on-board computer, cRIO and power systems, but without an external interface the cover has to be open to reach the many different I/Os, gratuitously exposing the many electronic components to a non-friendly environment.

Our solution was to implement a central external interface. This panel will have a wide range of I/O interfaces, such as the on-board computer power button, the cRIO reset button, LED status, as well as USB and VGA inputs. This elegant approach will virtually eliminate the robot exposure because of communication interfacing. Furthermore, this improvement virtually stops the need to move and connect wires within the robot. This is beneficial because loose wires tend to disconnect while Prometheus is operating, causing a

wide range of unwanted behaviors.

Monitor

Our team believes that the addition of a monitor to Prometheus' chassis greatly improves usability. A monitor allows real-time feedback of the many systems of the robot in a concise and informative way.

Due to the outdoor nature of the competition, the monitor set up has to be visible in direct sunlight, weatherproof to match Prometheus' current capabilities, DC powered (preferably 24V), relatively small (10-15 inch), and have a simple input interface. A touchscreen in particular would be perfect to meet this final requirement.

The touchscreen allows for the implementation of a Graphical User Interface(GUI) that can quickly relate information from the many different systems of the vehicle. When the GUI is fully implemented a given user can select between multiple tabs, each corresponding to a different system of the robot. For example, the "position" tab can display the current input from the GPS signal, as well as IMU information, while the "vision" tab can display live video stream from the on-board cameras, or the Hough transform results if we are dealing with line detection.

The variety of scenarios this tool can be applied to is almost endless. A touchscreen monitor located directly on the chassis is a step towards a robust robotics platform.

Remote Control and Wireless Emergency Stop

Prometheus is made to drive autonomously, but having a manual remote control is essential during transportation, debugging, and developing phases. In order to implement a manual control our team had two options. The first solution would be connecting a laptop to Prometheus' existing Wi-Fi network, and using a joystick to send commands to drive the robot around.

The second option involved using an existing hobby-grade Radio Controller, which was originally intended for RC hobby cars, helicopters, and airplanes. More specifically we are researching one of the major brands in the market, Futaba, and their FASST line of products. The following table gives a succinct comparison between the two different approaches.

Having a portable system, effortless start-up, and prompt reconnectivity were the three major components that drove our decision. Because Prometheus is a research platform, having an intuitive and effortless start-up system is necessary. Beyond that, the reconnecting time is critical when viewed through a safety perspective.

Based on this comparison we felt opting for the Radio Controller was the correct choice for our application. Driving Prometheus only requires a controller with 2 channels (for direction and thrust), but our team is looking into a system with three or more channels in order to keep it expandable for future research projects.

The IGVC competition, as well as our own specifications, requires us to have a wireless emergency stop that can function 50 feet away from the robot. The 2010 set up used a wireless car key alarm system with a range greater than 100 feet, but because of radio interference caused by the many other components on the robot, the system is not reliable at distances greater than 30 feet. For a detailed radio interference study of Prometheus systems refer to Appendix B.

<i>Feature</i>	<i>Joystick(Laptop)</i>	<i>Radio Controller</i>
Wireless Frequency:	2.4GHz Orthogonal Frequency-division Multiplexing (802.11n)	2.4GHZ Frequency Hopping Spread Spectrum
Start-up sequence:	<ol style="list-style-type: none"> 1. Start-up laptop 2. Connected to network 3. Run specific software 	<ol style="list-style-type: none"> 1. Turn on
Portability:	Laptop and joytiscck and wire	Convenient remote control
Fine trimming:	No	Yes
Reconnection (after signal is lost):	> 5 seconds after manual start-up	< 1 second
Interface:	Wi-Fi	Receiver transmit servo commands to cRIO
Range (tested on an open field):	Approximately 100ft	Not measured
Future expandability	Yes	Yes

Table 2.1: Comparison of the different options for using a manual controller with Prometheus

The extra communication channels on the remote control allow us to bind a switch to the wireless emergency stop. This solution brings the emergency stop away from the interference frequencies, increasing range and reliability while keeping it portable.

Visual Cue

After extended hours working with Prometheus, it became apparent that there is not a simple way of detecting the robot’s state, which causes several issues and ultimately slows down the development process. For example, low battery levels cause some Prometheus systems to only partially work, but because there is not an easy display of such information the programmer believes it is his changes that are causing the faulty response, so he proceeds to spend a significant amount of time trying to fix the code before he realizes that it is not the main problem.

Our solution to this problem is to use a multi-color controllable source of light to display the robot’s status. The 2011 IGVC rules include a new regulation that demands every competitor have a safety light, which only reinforces our decision.

Background research led us to several off the shelf solutions, but instead we decided to use 12V weather-proof Red Green Blue (RGB) Light Emitting Diode (LED) strip. This system will be controlled by a circuit designed and built in house.

2.2 Methodology

A new cover was designed to meet the requirements as specified earlier. This was done to improve the position of sensors, improve the structure of the robot and improve the usability of the robot. Specific parts of the overall design were made to address each of these issues. The combination of each of these

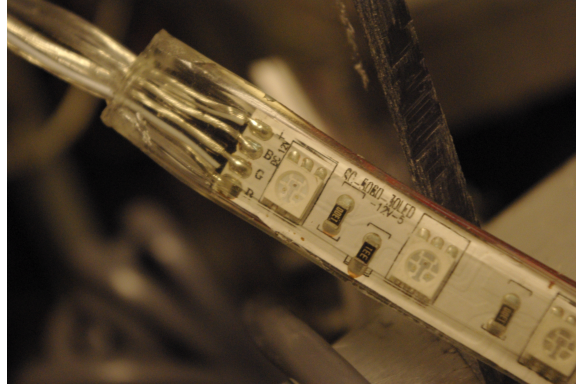


Figure 2.1: Commercial weatherproof RGB LED strip

improvements was checked for compatibility. After determining the best overall design, a new cover was built including all the new improvements.

2.2.1 Chassis Mechanics

Improvements were made to the chassis of the robot to address a couple major difficulties. To begin with, the cover was redesigned to improve sensor functionality and expansion capacity as well as providing the foundation for improved user interface. Also, portions of the chassis with questionable mechanical integrity were replaced such as the connection of the steered front wheel.

Design of Cover

The goal of redesigning the cover of Prometheus is to improve the functionality of the cameras, provide space for improved user interface and provide a better platform for future robotics research while protecting the internal electronics from the elements.

The cover needs to meet the following performance specifications. These specifications are based on what the cover needs to accomplish and some of the constraints of the system.

- Cameras must be able to see within at least one foot of the front of the vehicle.
- Modular devices such as robotic arms and rotor craft landing platforms must be attachable and interchangeable.
- Electronic components must be weatherproof from a reasonable amount of precipitation.
- Electronic components must be easily accessible by a user without turning off the robot and while a modular device is attached.
- Buttons, switches and connections for the user interface must be accessible while the robot is running.
- The center of gravity should not be any higher or significantly further from the center of the polygon of contact than in the current design in any configuration.
- Maintain an aesthetic shape.

If the cameras on Prometheus cannot see very close to the chassis, the ability for the robot to follow lines close to the chassis is limited. This also limits distance at which the cameras would be able to see both the

lines on either side of the lane spaced approximately ten feet apart. Decreasing the nearest visible distance will improve the line following functionality.

The angle of the cameras relative to the light source can also be a problem such as in the morning and evening. By improving the limit of the field of vision in front of the robot, the cameras will also be looking down on the ground at an angle closer to the preferred normal to the ground.

One of the objectives of Prometheus is to be able to use the robot as a research platform in the future. Having specific locations on the chassis that can accommodate a wide variety of sensors and devices will decrease the setup time for research experiments.

It is important that Prometheus is able to perform in a wide variety of weather conditions. To begin with, the IGVC continues regardless of rain. In order to be competitive, it is necessary to be able to run the robot in rainy conditions. Also, by having a weather proof cover, Prometheus can have an extended testing period. Ordinarily, outdoor testing and research might be limited to days with nice weather, but if the electronics are sufficiently protected, the robot can be tested more regularly.

The user should be able to perform a number of operations with the robot without having to open the cover. This includes turning the robot on and off, resetting the cRIO, switching between automated and controlled, connecting a mouse and keyboard to the On-Board Computer and observing real time data from the On-Board Computer.

Design Description

The following sections contain comparisons of particular solutions to the listed performance specifications and a synthesis of the optimal solutions. The final synthesis of solutions was then compared to the performance specifications to ensure that individual solutions to performance specifications do not contradict other specifications.

Camera Position

Two thoughts for improving the field of view would be to increase the viewing angle of the camera by using a different lens, by changing the position of the camera, or by a combination of the two. However, changing the lens would not actually improve the observable distance from the front of the robot. This is instead determined by the relation between the position of the camera and the furthest forward portion of the robot.

The following analysis is done based on the assumption that only one camera is used, since only one is required for line following. Stereo vision could be implemented with two cameras, but it is not immediately necessary for the success of the robot, as discussed later.

Figure 2.2 shows a silhouette of the chassis with annotations for some of the key measurements needed to calculate the field of view and compare camera locations.

Using simple trigonometry, the distance from the front of the robot to the bottom edge of the field of view can be described by equation 2.1

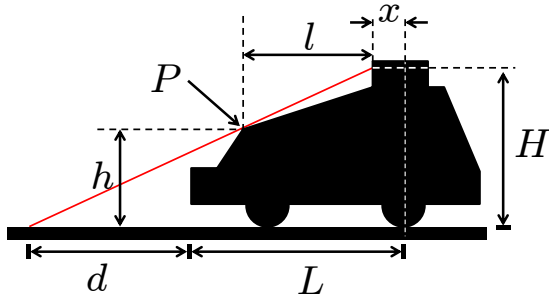


Figure 2.2: Measurements for describing the position of the camera and calculating the distance of the field of view from the robot

$P =$	Point of limitation - the highest obstacle limiting the camera field of view
$H =$	Height of camera from the ground
$L =$	Distance from the center of the back wheels to the furthest forward part of the robot
$h =$	Height of P from the ground
$l =$	Lateral distance from the camera to P
$x =$	Lateral distance from center of back wheel to the camera
$d =$	Distance from the furthest forward part of the robot to where the edge of the field of view meets the ground

Table 2.2: Description of the variables for camera position dimensions

$$d = \frac{hl}{H} + l - L \quad (2.1)$$

There are two independent variables in this equation, namely the horizontal position of the camera l and the vertical position H . The horizontal position has a linear relation to the distance while the vertical position has an inverse relation. This means that as H and l increase and decrease respectively, changes in l will eventually have a more significant effect on d than changes in H .

Table 2.3 shows different reorientations of the cameras, the advantages and the disadvantages.

Based on this basic analysis, it was determined that the cameras should be moved forward. It was also determined that the vertical position of the camera should be adjustable in order to experiment with different camera angles and the relation to video quality.

Figure 2.3 compares the location of the camera in the old design, the new design and a modified new design.

A couple other considerations for the location of the camera is how the angle and position of the camera affects the maximum viewable distance as well as the distance at which the two white lines ten feet apart can be seen. For example, if the camera is positioned higher above the ground, it will have an almost vertical angle to view just over the front structure of the robot. This would mean that the top edge of the field of view would be closer to the robot than a camera position with a more horizontal angle.

Based on the given field of view, the extent of the camera image can be calculated. The current camera and lens configuration gives a horizontal viewing angle (Figure 2.4(b)) of 53.13 deg and a vertical viewing angle (Figure 2.4(a)) of 41.11 deg. A representation of these angles is shown in Figure 2.4.

By adjusting the height and angle of the camera, the field of view can be adjusted as future design specifications may dictate for effective line following.

Camera Mount Design

The camera mounts were designed with a few specifications in mind.

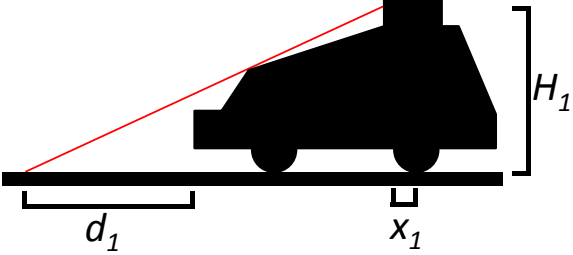
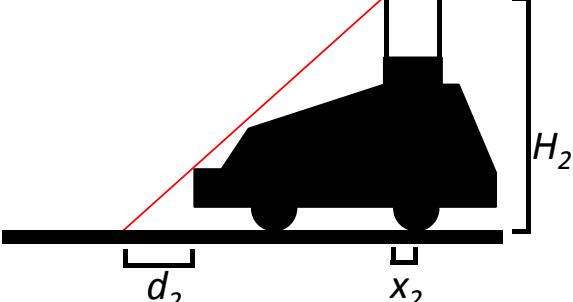
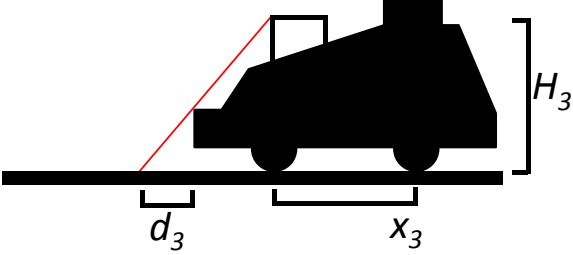
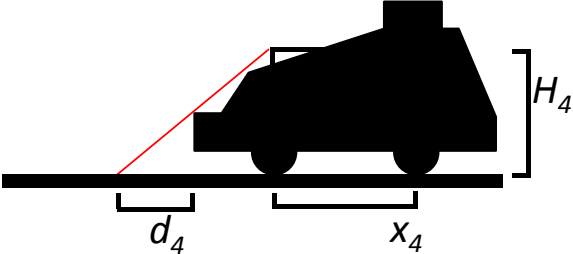
<i>Instance</i>	<i>Advantages</i>	<i>Disadvantages</i>
	<ul style="list-style-type: none"> • No changes needed • Aesthetic orientation 	<ul style="list-style-type: none"> • d_1 is too large
	<ul style="list-style-type: none"> • d_2 is shorter than d_1 • Can be installed on current structure 	<ul style="list-style-type: none"> • Raises the center of gravity (CG) • Takes away from low profile design
	<ul style="list-style-type: none"> • d_3 is much shorter than d_1 • Moves CG closer to center of polygon of contact 	<ul style="list-style-type: none"> • Requires new structure further forward • Takes away from low profile design
	<ul style="list-style-type: none"> • d_2 is shorter than d_1 • Moves CG closer to center of polygon of contact • Lowers CG 	<ul style="list-style-type: none"> • Requires new structure further forward

Table 2.3: Comparison of possible configurations for the camera mount on Prometheus

- The mounts must be able to locate the cameras in an optimal location as indicated previously.
- The mounts must be weatherproof.
- The mounts must be expandable for use with future stereo vision research.
- The two cameras must be mounted at the same height on the same plane.

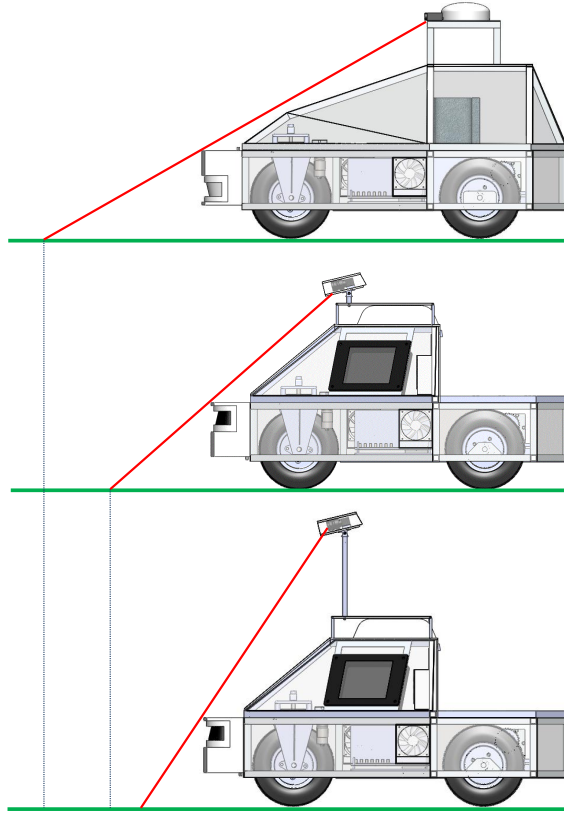


Figure 2.3: Comparison of the location of the camera in the old and new Prometheus designs.

- The two cameras must have independent yaw positioning.
- The height, yaw, pitch and baseline must be adjustable.
- Once cameras are set in a position, they must not slip.
- Incident light and reflections must be minimized.

With these specifications, a few different designs were sketched. To meet the first specification, the cameras would have to easily mount to the new cover design. As such, the idea was to have an aluminum plate on the top of the cover to which various devices could be affixed. This in itself was not particularly limiting in the design.

Weatherproofing the cameras was another issue. There could be no cover directly in front of the lens and the cable must be able to reach the inside of the computer compartment on the robot. Previously the two cameras were contained within one Lexan box with an open front, with the cameras set back far enough that they would be protected from all but practically horizontal rain. It was determined early that each camera should have a separate weatherproof housing so that individual yaw positioning might be adjusted more easily without potentially having to open an all-encompassing cover. This would also decrease the amount of materials necessary. In order to have an adjustable baseline, a single box would have to cover all configurations of the baseline. With individual covers, this could be adjusted more easily without having to initially cover the whole span.

Previously the cover was made from a rectangular box of Lexan. It was determined that this was not the optimal method. Lexan lets in too much light which can cause glares or other problems that would affect

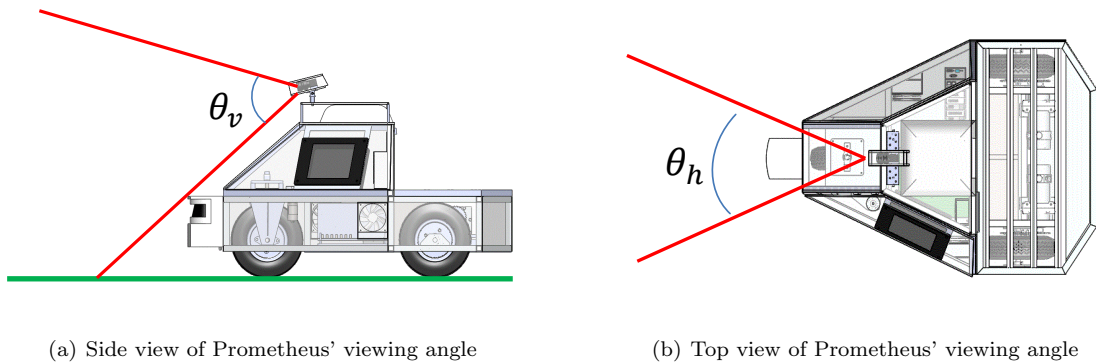


Figure 2.4: Side and top view of Prometheus with the field of view displayed in red

the view of the camera. Therefore an opaque cover was decided upon. Also, it was decided that the inside of this cover would be colored a matte black to further reduce any glare from incident light. It was also determined that a rectangular box was not the most optimal for manufacturing. Instead, a cylindrical tube was decided upon. This would decrease the number of joints that would need to be weather sealed. Also, this would decrease the total number of parts to be made.

The distance from the front of the camera mount to the lens is based upon the angle of view of the camera. This distance was determined to place the camera as far from the open face of the cover without any of the cover showing in the image from the camera.

The adjustable nature of the camera mounts was a design consideration with many possible options. The most complex design would involve driven actuators that would adjust the angles and height of the cameras as needed. This idea was quickly abolished because of not only the time and effort needed to implement such a system, but also because of the problems with repeatability in the actuators and the necessary mechanical devices to ensure the fixed location of the cameras during operation.

Ideally the cameras need only be adjusted on a very infrequent basis. For example, once there is working code for the line detection, the position of the cameras should not be changed, because the code will then have to be changed. Because of this, a much simpler design was acceptable. A manually adjustable device would be simpler to build and implement, have potentially better repeatability and would require no further power consumption.

For height adjustment it was determined that fixed height posts that could be exchanged would be more practical than telescoping poles. Stability in the camera mounts is important and every joint of a telescoping pole is just another point that may increase the flexure of the overall pole. Again, because of the infrequent nature of height adjustment, it was found that it was not necessary to have immediately adjustable height. One problem is that to have a wide variety of heights, much more material is needed. Because of this it was determined that a short set of posts and a long set of posts would be made initially and future posts could be made to length as the need might arise.

Adjustable yaw and pitch were a more difficult problem. In order to make these lock into repeatable positions it was determined to use geared teeth. The simplest approach was to have geared teeth such that when a bolt was loosened, the cameras could be rotated, but when the bolt was tightened, the mounts would no longer slip. The solution was the use of planar knurled plates as seen in Figure 2.6. These matched

knurled circles on a plane would allow for adjustability in discrete units. This also makes the position of the knurling repeatable for both the yaw and the pitch. The teeth of the planer knurling are flat on the top with the cut between each, sloping up toward the middle to ensure full contact between the two sides even as the radius decreases toward the middle.

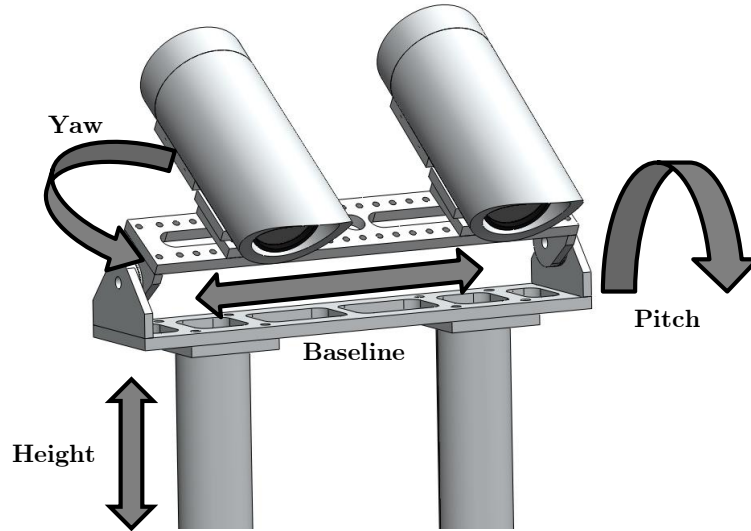


Figure 2.5: New camera cover design adjustable for yaw, pitch, height and baseline

The radius of the circle, the depth of the teeth at the outer edge, the length of the teeth and the angle between teeth are all interrelated variables that define the geometry of the knurling. Since the knurling in this case would not be subjected to extreme stresses, and a small angle between teeth would allow for a more precise range of motion, it was determined to use an outer radius of 0.5 inches with teeth six degrees per tooth. Based on this the geometry for the depth of the tooth at the outer edge was defined. Assuming 90 degree teeth for ease of manufacturing, the only dimension left was the length of each tooth. An initial prototype was made with 0.063 inch teeth, but it was determined that in order to distribute forces better, the teeth should be made longer. Each tooth was therefore made to be 0.188 inches long. This increased the surface area for contact between the two pieces of each knurling and decreased the maximum stress.

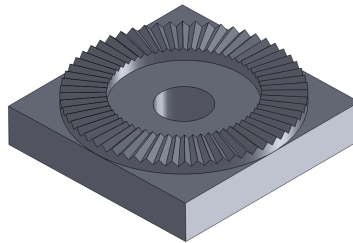


Figure 2.6: Planar knurling used to keep rotational joints from slipping

GPS Location

The location for the GPS of the 2010 Prometheus design included the main receiver inside the cover and an antenna attached to the back approximately 3 feet above the top of the robot. Originally, the thought was that this could be repeated. However, when it was determined to use a different GPS, as discussed in the Sensor section, the antenna and receiver were part of the same unit. The receiver unit was therefore larger than the previous GPS antenna and the thought was that it would be unwieldy if it was placed as high as the antenna on the old design.

The plan for the new cover design included the GPS on the outside of the robot, sitting on the top plate of the cover. This position was tested and it was found that the receiver did not receive signals from this position. By keeping the receiver plugged into the computer and slowly moving it around the robot, it was determined where the receiver had satellite signals and where it did not. It was found that within one foot of the top of the robot, the receiver had a signal, but on the top of the cover it did not. Moving the receiver behind or in front of the camera mounts or the camera mount posts also interrupted the signal as seen in Figure 2.7. This signal interruption was attributed to the metal frame, camera mounts and posts interrupting the satellite signal.

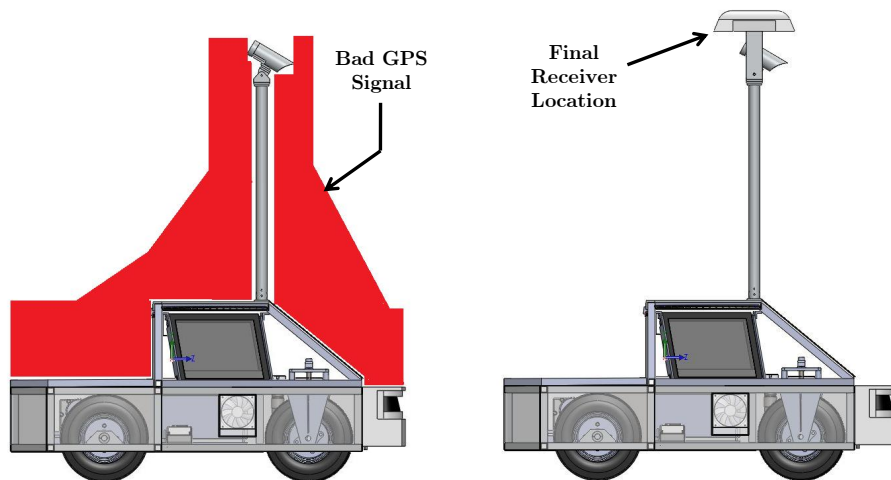


Figure 2.7: The locations where satellite signals could not be received indicated in red at left, and the final location of the receiver at right

The most reasonable location for the GPS was determined to be the top of the camera mounts as seen in Figure 2.7. An attachment was made to fit over the camera mounts and secure the GPS receiver.

Modularity

In order to accommodate future expansion and reconfiguration of Prometheus, it is important to have locations on the chassis where more sensors or devices can be added. By moving the cameras forward, it was determined that the back portion of the chassis would make an ideal place for added devices. Since most additions would be external, maximizing the open area on the back of the structure is an important factor. Also, creating a modular interface with which devices can be connected is also important. Lastly, any devices added to the structure would be considered an added payload. In order for the additional weight

to minimally affect the performance of the robot, the location of the center of gravity (CG) of the addition should closely relate to the location of the entire structure's CG.

Based on these criteria, it was determined that the most logical solution would be to create a flat-bed over the back half of the chassis. This allows for an open environment for moving or oversized components. This also places the additional weight over the batteries and the motors, two of the heaviest components, minimizing the effect of the change in the position of the CG. The flatbed would also have regularly spaced T-slots that would allow for easily attached and positioned components.

The flatbed has room for the attachment of different devices. This can include a UAV landing pad, a robotic arm or other structures. The current attachment for the robot is a lawnmower for the ION Robotic Lawnmower competition. This device uses a four-bar linkage system to keep a platform of five flail-head weed trimmers level with the back of the robot. Adjustable wheel heights allow for the adjustment of the height of the cut. The four bar linkage is bolted to t-nuts in the extruded aluminum of the back platform. Figure 2.8 shows a possible configuration with a UAV landing pad.

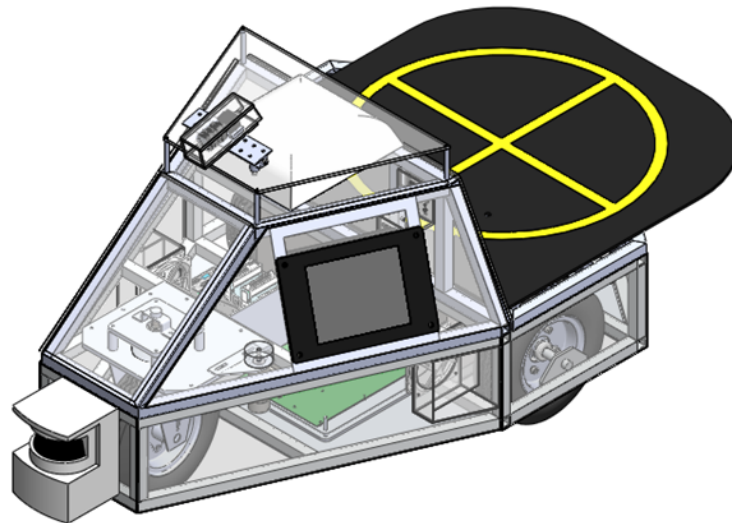


Figure 2.8: Prometheus with a Unmanned Ariel Vehicle (UAV) landing pad attached to the modular payload area

In order to keep the components protected, the frame of the cover will be covered with a waterproof material. Lexan sheets will be used to cover most surfaces. With the side panels and back opening, it will be important to create a weatherproof seal on these edges using a combination of weather stripping and tight latching mechanisms.

The new design will use the same principles, except that the side panels of the front cover and the flatbed will be able to open. Also, the external interface on the back will have a sealed sliding door. Because of this, most joints will require weather stripping or some other gasket material between the polycarbonate and the aluminum frame.

Materials and Manufacturing

The material for the cover was chosen to reflect the success of the materials used in Prometheus 2010. This includes an aluminum frame with a polycarbonate shell. One possible construction of the frame would be with extruded aluminum t-slot material. This would allow pieces of the cover frame to be interchanged more easily and would allow different items to be attached to the frame more easily. An investigation was conducted into the feasibility of using this type of material. Figure 2.9 shows a corner joint of the frame with the three different pieces highlighted with different colors.

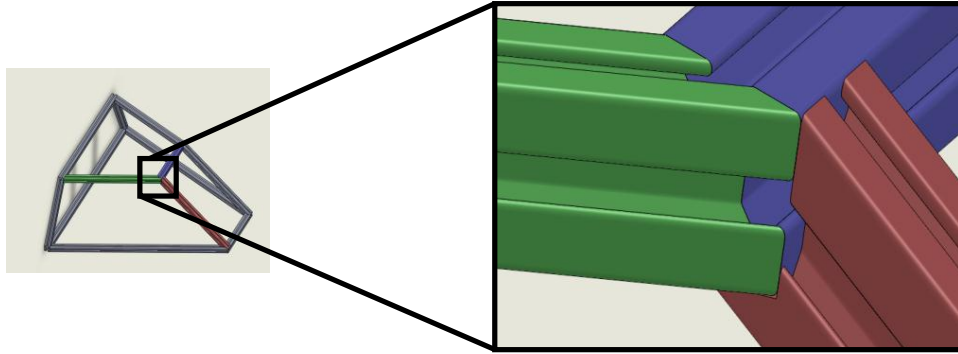


Figure 2.9: Corner joint of the frame

Figures 2.10 and 2.11 show a fixture that would use bolts in the t-slots to hold the joint together. This would create a fixture as seen in Figure 2.12.

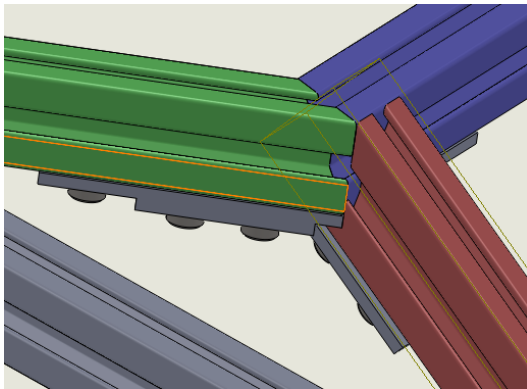


Figure 2.10: Extruded t-slot joint fixture viewed from the top

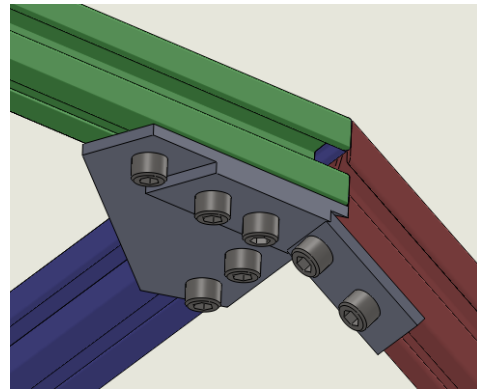


Figure 2.11: Extruded t-slot joint fixture viewed from the bottom

This type of a fixture, representing just one joint, in the frame would make the manufacturing very time consuming. Also the odd angles would need to be manufactured with a fairly high tolerance in order to create a firm joint with no gaps.

Because of the difficulty in manufacturing, it was decided to make the frame from welded one inch square aluminum tubing with a 1/8 inch wall thickness. This is the same construction method as was used for the chassis. This allows for a looser tolerance when cutting the lengths and angles of the pieces and will create a joint that is as strong if not stronger than the members.

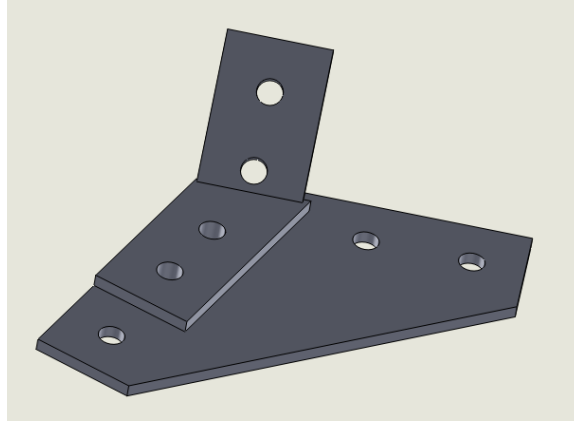


Figure 2.12: Fixture for one joint of an extruded t-slot frame construction

The polycarbonate shell will use 1/4 inch thick panels rather than the 1/16 inch thick pieces used previously. Many of the panels will have some sort of structural task including the side panels acting as doors, the back panel securing a sliding door and the bottom of the flatbed protecting the motor compartment from the payload. Any seams between the panels will be fused together using methylene chloride. These fused seams are waterproof and increase the strength of the joints much as a weld does on a metal joint.

Steered Front Wheel and Drive Train

Two potential solutions exist to fix the shaft more securely to the pulley. The first is to simply use a thread lock adhesive to keep the set screws from loosening. Another standard method of fixing axles to wheels is to use a keyway. This involves having a “key” that fits into a slot on both the pulley and the shaft as seen in Figure 2.14. One advantage to the keyway is that the force applied on the shaft by the pulley is distributed the length of the key as seen in Figure 2.13. This means that the weakest part of the connection is based on the shear strength of the three parts over the length of the key.

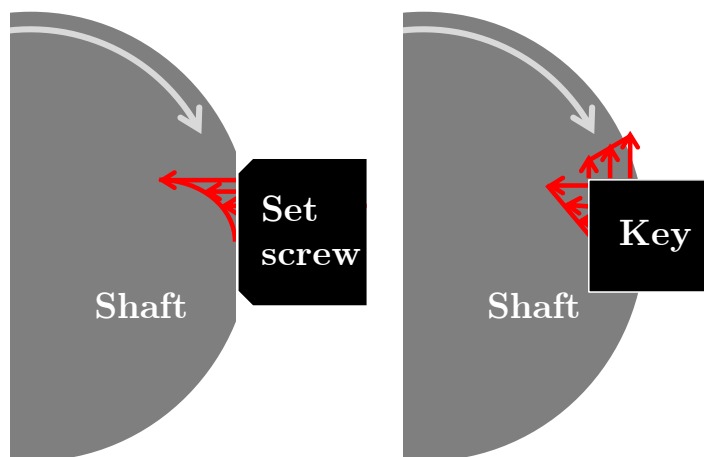


Figure 2.13: A two dimensional comparison of force distributions for setscrews on a shaft and a keyed shaft. The force distribution is depicted by the red arrows, and the direction of rotation is indicated by the grey arrow.

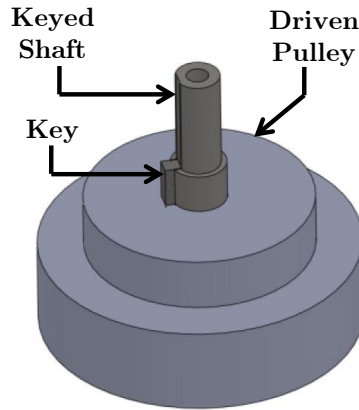


Figure 2.14: Keyed shaft

With a set screw, large torques can cause damage to either the screw or the shaft. More deformation leads to more play in the system which causes more shocks, which continue to compound the problem of deformation. The stresses applied to a set screw are distributed over the contact area of the end of the screw with a concentration at the apex of the curve. Stresses applied to a keyed shaft are distributed over two faces of the key which have significantly more surface area than the end of the set screw. Even though there are sharp corners in the key slot, the stress concentrations are less than for the apex of the set screw surface.

The new maximum speed limit on the course is 10 miles per hour, up from 5 mph last year. The drive train of Prometheus was made to limit the speed to approximately 5mph. If the team is to increase the speed to approach the new speed limit, the drive train must be reconfigured.

The maximum speed of the motors is 238 rpm. Using twelve inch diameter wheels, Prometheus must have a wheel speed of 280 rpm to travel 10 mph. If Prometheus is to travel at this maximum speed when the motors are spinning at the maximum speed, the conversion $\frac{drive}{driven} = \frac{238}{280} = 0.85$ can be used to determine the gear ratio. This is compared to the current gear ratio of 1.6.

The problem with using that particular case for determining the gear ratios is that the motors would then draw approximately twice as much current for the same speed. This could prove to be an unacceptable alternative.

2.2.2 Usability Improvements

External Interface

After extended interaction with Prometheus our team developed a detailed list of components that compose the external interface. The final design has all the following Input/Output connections:

USB Ports

- External connection to Mouse, Keyboard, Webcam and any other peripheral

Ethernet Connector

- For wired connection to the robot network, allowing higher speed connection.

- Potentially used to debug the wireless network

Computer power switch

- For starting/resetting the computer without exposing Prometheus

cRIO power switch

- For starting/resetting the cRIO without exposing Prometheus

LIDAR, Monitor and GPS power switches

- For starting/resetting individual components

Computer and cRIO power LED status

- Quick visual check

12V connector

- External power supply for a wide range of peripherals

Battery voltage display

- To check the battery life on the current version you need to physically open the robot and place a multimeter on the batteries.
- Either digital or analog

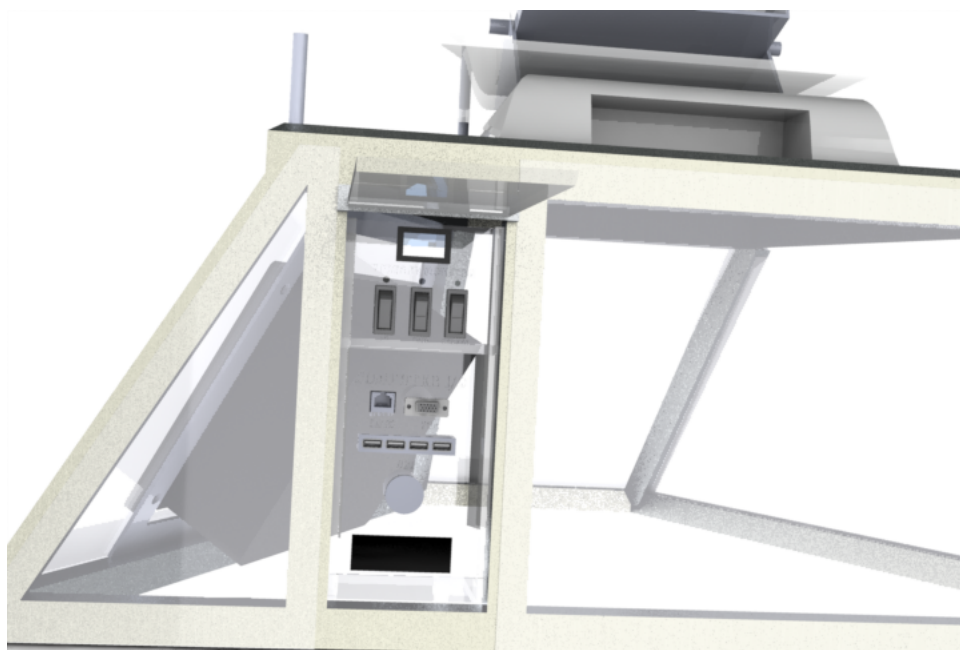


Figure 2.15: External interface concept design

This external interface panel will be easily accessible, as well as weatherproof. This will be achieved by isolating the connections, switches, and status display through a transparent window with cable slots at the bottom. The exterior robot interface will ultimately help Prometheus become the independent, user friendly research platform we envisioned.

Monitor

Based on the requirements listed previously we decided an outdoor commercial use touchscreen monitor would be the most ideal product for our application. After extensive research and comparison we decided on a 3M MicroTouch Display C1500SS (15") Serial. This decision was mostly made due to its comparably low price range versus the many desirable features that are incorporated in this product like 12V DC input, ruggedness and a simple serial interface. More information can be found in Appendix D.



Figure 2.16: Commercial weatherproof 3M Touchscreen Monitor

As of now the touchscreen capabilities have not been implemented, and the touchscreen is only being used as monitor. This setup in conjunction with a mouse and keyboard already allow us to program directly on the robot. The touchscreen facilitated tasks such as analyzing the line detection effectiveness, debugging waypoint navigation and evaluating the local map.

Remote Control and Wireless Emergency Stop

Ultimately we opted for a Futaba 6-channel 2.4GHz Transmitter and Receiver. The receiver has a very small footprint, weighing only a few grams. Because the communication works at 2.4 GHz, its position on the chassis has less effect on the signal strength than other lower frequency solutions. The control layout for Prometheus' new remote control is labeled on the image below:

We choose this controller because of several of the advantages it has, including:

- 2.4 GHz FHSS communication, avoiding problematic interference from other components (problem discussed in Appendix B)
- 6-channel permits future expandability
- Rechargeable battery
- Simple and reliable connection
- Fast start-up time

In a normal application this receiver would control servos on a RC helicopter using Pulse Width Modulation (PWM) signals at 50Hz, the duty cycle of each pulse directly correlates to the position of the controller. In order to capture this information with the cRIO we are inputting the signal into a digital I/O pin of the



Figure 2.17: Futaba 2.4GHz 6-channel Transmitter

FPGA. The length of each pulse is stored in a variable, which is then used to set the motor speeds, the turning angle, and the emergency stop signal, among other features.

The controller layout is as follows: Channel 5 switches Prometheus between autonomous and manual mode, channel 1 and 2 are for manual driving, and channel 6 is connected to the wireless emergency stop.



Figure 2.18: Futaba 2.4GHz 6-channel Receiver

Visual Cue

We added strips of weatherproof RGB LED to Prometheus' chassis. The robot will glow specific colors dependent on the cRIO state, and consequently, the robot state. This will be visible during daylight, and create a remarkable effect during the night.

The visual cue will instantly display the status of the robot. This system can also be very useful when dealing with known fault status, such as low battery and network communication errors. The current set up is seen in Table 3.

<i>Prometheus Status</i>	<i>Color</i>
Red	Wireless E-stop locked
Blinking Green	Autonomous mode
Blue	Test mode
Purple	Manual control
Yellow	Low battery

Table 2.4: Visual cue states

The driving circuitry for this set up will use a 3-channel 5V Pulse Width Modulation (PWM) signal from the cRIO to drive the LEDs using the 24 V available. The driving signal will vary the duty cycle of the PWM, directly controlling the light intensity of each color, this allows us to generate an infinite amount of colors and patterns with little coding necessary.

The LED strip has a common positive, so the design will use three fast switching N Channel MOSFETS with the supporting circuitry to take the cRIO input and drive the lights. The RGB LED strip consumes 3 watts per feet, so assuming 4 feet of lighting, this set up does not consume more than 12 Watts.

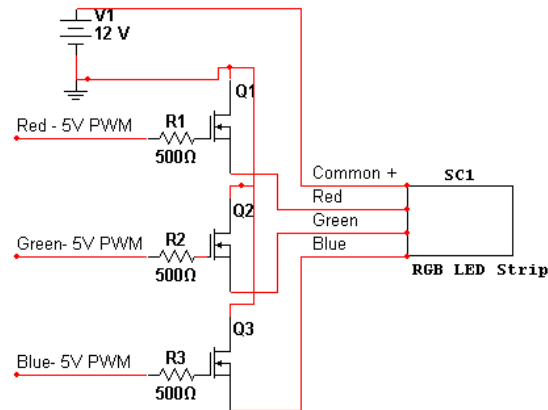


Figure 2.19: 5V PWM driving circuit for RGB LED strip

The ease of use of the visual cue will save us time in the testing and final phases, hopefully giving us the edge to succeed at the competition.

2.3 Results

The following is a discussion of the results of the design work for the robot structure improvements. Based on the previously discussed designs, these are the resulting implementations.

2.3.1 Chassis Mechanics

The cover was designed and constructed based on the analysis performed. A model of the entire new cover was created using SolidWorks. The frame components were cut from one inch square 6061 aluminum tubing with 1/8 inch walls. This was welded together using a TIG welder. The back cover frame was welded from the same material with anodized 6061 extruded T-slot bars welded at 4 inch intervals across the back. The top plate was cut from a 1/4 inch thick aluminum plate. The frame for the external interface was machined from aluminum and welded to the cover frame. The frame was bolted to the main chassis of the robot and the top plate was bolted to the frame. The polycarbonate panels were cut from 1/4 inch material to fit to the welded frame. These panels were then bolted to the frame. The panel covering the back electrical components was hinged and weather proofed to provide access to these components. The external interface was made from polycarbonate for the sides and laser cut acrylic for the faces. A number of faces were laser cut as the definition of needs for the external interface evolved. The external interface was then bolted to

the welded frame. Pneumatic lifts were installed on the back cover of the new design to control the lifting and lowering of the cover and to keep it in position when the batteries are being changed. Latches were also installed to keep the cover down. The camera mounts were machined using CNC mills with the code generated directly from the models created in SolidWorks. The cameras were installed in the mounts and the mounts were attached to the robot. By simply viewing the output from the cameras, the positions were adjusted and marked for two different positions, 4 inch posts and 30 inch posts. The LIDAR was flipped over and the bumper lowered. This helps protect the LIDAR better from curbs and eliminates the detection of thick grass as an obstacle. A frame was built for the touchscreen monitor to hang the monitor just within the side of the robot. The screen was attached to the frame and the frame was bolted to the main cover frame. A panel was made between the computer compartment and the front wheel to separate the two. On this panel a fan was mounted to aid with the cooling of the computer compartment. With the two side doors of the cover closed, this will be the main channel for airflow to the outside world, an important aspect of keeping the computer components cool. The GPS was originally installed on the top of the plate on the cover. Unfortunately, this was a bad position for receiving satellite signals and therefore a frame was built over the top of the camera mounts to hold the GPS. An acrylic box was made for the IMU to attach to the back panel of the robot. The device was originally held in place with metal screws, but these proved to interfere with the IMU signal. So, they were replaced with nylon hardware. The front wheel of the robot was disassembled and the old shaft was cut off. The new shaft was welded on to the wheel yoke and the driven pulley was broached to receive the key. The supporting blocks for the shaft were re-machined to use new bearings including the heavier thrust bearing. With the replacement of thinner pieces of material with thicker, more rigid pieces, a new structure for the front wheel encoder was machined.

2.3.2 Usability Improvements

The usability improvements made on Prometheus proved to be very effective during field tests. The interaction between the many different systems of the platform is substantially more symbiotic. The enhancements allow quick testing, swapping, and comparing of different algorithms. It also permits us to identify software bugs and hardware malfunctions much faster than before, ultimately decreasing development time.

The external interface allows a direct interaction with Prometheus, without exposing any components to the environment. The USB connections are used constantly by a variety of peripherals, including thumb drives, a mouse, a keyboard, and webcams. The switches are also extremely useful when it is necessary to reset or power on or off the computer, cRIO, and LIDAR. Lastly, checking the voltmeter became second nature for our team, which means the batteries never reach a critical level. Furthermore, there were little to no incidents of loose wires or bad connections during all the testing. A substantial amount of time was spent organizing and aligning the great amount of cables and connections inside Prometheus. In order to avoid disconnections due to vibrations, zip ties were used to secure any loose wiring to the chassis.

The visual cue also became an essential component of Prometheus. The different colors are clearly associated with Prometheus' respective states, and the user can glance at the vehicle and know what to expect from its behavior. In addition, the low battery emergency state is a great warning when the batteries are reaching critical levels.

The addition of the monitor to the chassis vastly improved field testing. The ability to quickly analyze the line detection, local map, and waypoint navigation proved to be crucial. There were many times during the

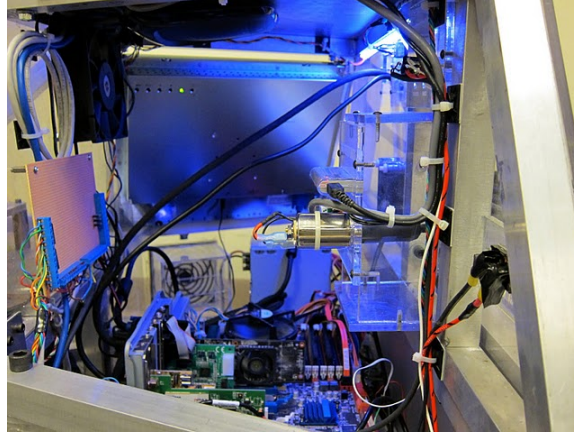


Figure 2.20: Computer compartment with wiring properly secured

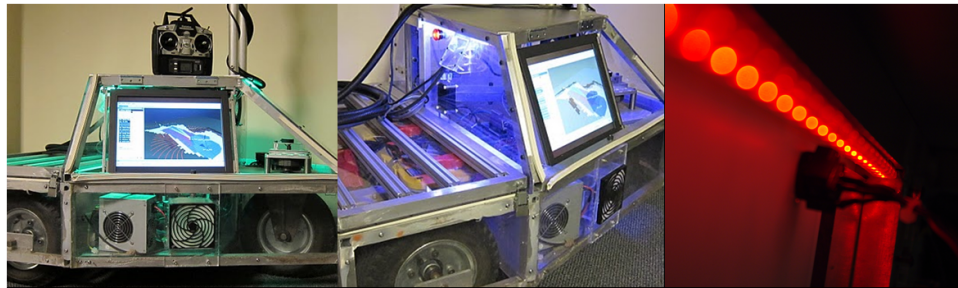


Figure 2.21: Different visual cue colors representing different states

test phase in which our team would walk alongside the robot and catch software bugs simply by looking at the screen. As mentioned earlier, due to time constraints, we have not been able to implement the touchscreen capabilities of the monitor. This could be considered a disadvantage because in order to interact with the computer a keyboard and mouse need to be plugged in.

Moreover, future efforts can be directed toward creating a graphical user interface for the monitor that can quickly display all the status information from a variety of different systems. A given user can select between multiple tabs, each corresponding to a different system of the robot. For example, the "position" tab can display the current input from the GPS signal, as well as IMU information, while the "vision" tab can display live video stream from the on board cameras, or the Hough transform results if we are dealing with line detection. The variety of scenarios this tool can be applied to is almost endless.

Lastly, the remote control worked flawlessly. We drove Prometheus on an open field beyond 100 meters without a single communication problem. The emergency wireless e-stop is very reliable. Moreover, the switch between manual and autonomous modes proved to be especially practical. That is because when testing we can switch from autonomous to manual mode, move Prometheus freely, and switch back to autonomous, all without touching the computer.

Current sensing is the one usability improvement we were not able to explore due to time constraints. In order to keep track of Prometheus' performance, supervising the power consumption of each individual system is beneficial. To do so it would be necessary to implement current sensors on the main distribution lines.

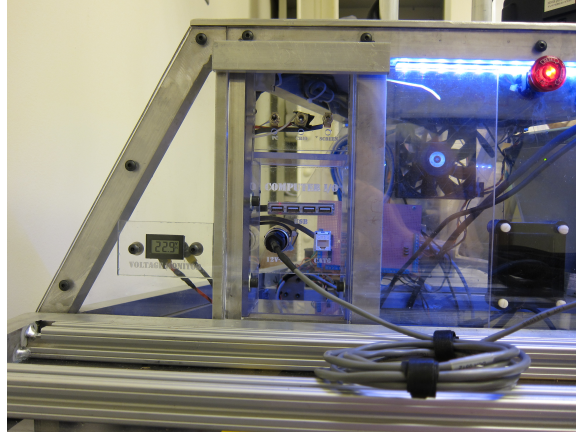


Figure 2.22: Prometheus external interface and visual cue

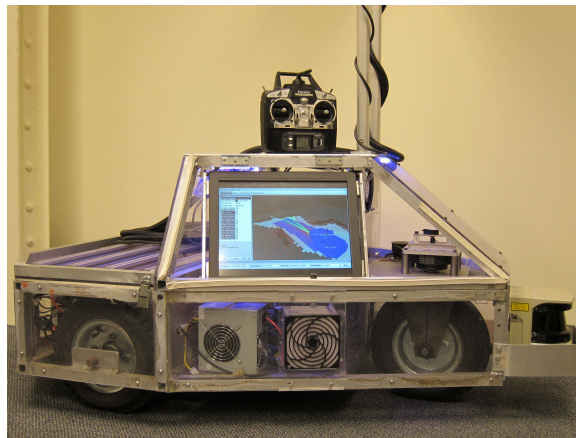


Figure 2.23: Prometheus touchscreen and remote control

For DC applications there are two major families of sensors, Hall Effect sensors and resistor sensors. Hall Effect sensors are advantageous because they are physically isolated from the circuit being tested. The charge flowing through a wire (current) creates a magnetic force in the form of concentric circles around the wire. The Hall Effect sensor measures this flow using a coil wrapped around the wire being measured.

The resistor sensor on the other hand puts a very high resistance (0.5M Ω , as an example) in shunt with the circuit being tested, and measures the voltage drop across this resistor. The voltage drop is directly proportional to the current, assuming the resistor is behaving independent of temperature.

Current sensing benefits the research platform aspect of Prometheus. The ability to calculate the currents of different components would allow us to graph the overall power consumption of the robot in different conditions. We would do so by placing one sensor in each major power consumer line: the motor drivers, the On-Board computer, the cRIO, the LIDAR and the router.

This information can be fed in real time into the cRIO while Prometheus undergoes a variety of conditions; the power consumption can then be plotted in real time through the touchscreen, and logged with a time stamp for later analysis.

2.4 Conclusion

The designs for the robot structure were successfully implemented. This section briefly describes how the design specifications were met.

2.4.1 Chassis Mechanics

The mechanical aspects of the cover design meet the specifications outlined by the team. The cover was successfully constructed based on the plan. The sensors including the LIDAR, cameras, IMU and GPS are all in positions for favorable use. The user interface is easily accessible. The modular payload area has been tested. Besides carrying more than 150 pounds of direct load, the payload area has been used as a hitch for towing three office chairs with developers and incorporating a lawnmower design for the ION Robotic Lawnmower competition.

2.4.2 Usability Improvements

In conclusion, the usability improvements substantially increased the development efficiency. Field testing now runs significantly smoother because of the many different ways the team has to interact and understand what is truly happening with Prometheus' many different systems. Moreover, when hardware or software problems come up, the development team can quickly pinpoint it, saving time and patience. One might say that a considerable amount of time was spent on usability improvements, but our team believes that the usability improvements gave Prometheus major advantages. These persistently decrease prototyping time, and therefore increase overall work throughout.

Chapter 3

Sensors

Prometheus uses various sensors to complete all the required tasks at IGVC, which include localization, obstacle avoidance, and line detection. In terms of localization, the robot combines data from a Global Positioning System (GPS) receiver, an Inertial Measurement Unit (IMU), and wheel encoders. In order to avoid the various obstacles in both the Autonomous Challenge and the Navigation Challenge at IGVC, Prometheus uses a light detection and ranging (LIDAR) sensor. Prometheus also uses two cameras to recognize and follow or avoid the white lines that outline the courses at IGVC. The LIDAR is used in its same capacity as it was in 2010, and therefore only obstacle detection methods are discussed later in this paper. The cameras on Prometheus are the same from 2010, but there was a change in how exactly they were used and where they were located. Because the hardware did not change, we have only focused on line detection methods detailed later in this paper. The GPS receiver and IMU are the newest additions to Prometheus, which are outlined in detail in this chapter.

3.1 Global Positioning System

Global positioning system “provides specially coded satellite signals that can be processed in a GPS receiver, enabling the receiver to compute position, velocity, and time” (Dana, 2000). A differential GPS receiver differs from a GPS receiver in that it is more accurate through error correction techniques. A DGPS receiver is a vital component of the project because of the navigation and localization assistance it provides. A DGPS receiver would allow Prometheus to measure its current position in terms of latitude and longitude. Knowing this helps greatly in the IGVC Navigation Challenge because the robot can now determine how far it is from the waypoints, as well as override incorrect measurements taken by other sensors in the system aiding in localization.

The IGVC Navigation Challenge waypoints are contained within circles with a minimum diameter of two meters. With the waypoint located at the center of this circle, one meter of leeway is located around the coordinate provided for that waypoint. This means that a DGPS receiver with sub one meter accuracy is ideal for implementation on Prometheus.

<i>Brand</i>	<i>Update Rate</i>	<i>Size</i>	<i>Position</i>
MediaTek MT3329 GPS	10 Hz	16 mm × 16 mm × 6 mm	<3 m
Sokkia Axis3 DGPS	5 Hz	19 cm × 12.5 cm × 5.1 cm	<1 m
Trimble AG252 DGPS	1, 5, or 10 Hz	29.72 cm × 6.93 cm × 30.61 cm	<1 m

Table 3.1: Comparison of specifications of three of the DGPS receivers that were considered for use with Prometheus. A more complete table can be found in Table D.1.

3.1.1 Background

Since a functioning and accurate DGPS receiver is critical for Prometheus in terms of localization, we have researched several DGPS receivers in order to compare the specifications to that of the formerly used Sokkia Axis3. Table D.1 lists all of the DGPS receivers that appear to be similar or better when compared with the Sokkia Axis3.

All of the DGPS receivers listed here have very similar specifications. On paper, the Sokkia Axis3 and Trimble AG252 appear to have the best accuracy. Our desire is to use the GPS receiver with the highest accuracy and update rate, specifically 5Hz or more.

3.1.2 Methodology

In order to determine which GPS receiver would provide us with the best results, we performed testing on all of the units we had in our possession: Sokkia Axis3, MediaTek MT3329, and Trimble AG252.

The desired results to be obtained from testing were data logs recorded by the GPS receiver, for use in later analysis. The testing procedure was to take each GPS receiver (one at a time) out to Institute Park and place it in a manner such that it would remain stationary throughout the data logging session. The GPS receiver was plugged into a laptop and configured to log its data in Putty.

After all data was logged, each set was analyzed using Matlab code given to us by the 2010 MQP team. The code produced a plot that was a visual representation of how accurate the results were and several different error measurements.

Once data was acquired from each, analysis could be performed on the results. With each set of data, three different types of error values were calculated. Root Mean Square (RMS) error value represents the amount of accuracy one can expect to receive 67% of the time, or in other words, on average. The 50% Circular Error Probable (CEP) represents a more desirable amount of accuracy, while the 95% CEP is often regarded as the worst case amount of accuracy. These values are calculated using the respective percentage of the returned values. Since we were looking to use the GPS receiver with the lowest error, these values were important to consider when comparing the three GPS receivers.

In order to calculate these errors, the first step was to find the average x and y positions. These were calculated as seen in Equations 3.1 and 3.2. For reference, Easting is x and Northing is y of a given latitude, longitude coordinate.

$$X_{average} = \frac{Easting_{total}}{n} \quad (3.1)$$

$$Y_{average} = \frac{Northing_{total}}{n} \quad (3.2)$$

The next step was to calculate the general error, which was done using Equation 3.3.

$$error = \sqrt{((Easting - Easting0) - X_{average})^2 + ((Northing - Northing0) - Y_{average})^2} \quad (3.3)$$

This error was then simply multiplied by one of three fractions to get the respective errors. To calculate the 50% CEP Error, we multiplied by .5. To calculate the 95% CEP Error, we multiplied by .95. To calculate the RMS Error, we multiplied by .67.

3.1.3 GPS Receiver Results

All data logged had values for, or which could be used to calculate, the latitude and longitude at which the GPS was located. The files containing the data were read by the previously mentioned Matlab code and used to calculate the errors, as well as plot the recorded positions.

Sokkia Axis3

The Sokkia Axis3 was the first GPS receiver taken out for testing. These first set of measurements were obtained differently than all other GPS logging we did because the GPS had known accuracy problems. We visited a Department of Public Works GPS benchmark on the corner of Highland Street and Park Avenue in Worcester, Massachusetts. This provided us a known latitude and longitude that we could compare our results to. Figure 3.1 is a plot of the positions measured by the Sokkia at the benchmark on September 30, 2010, over a period of fifteen minutes.

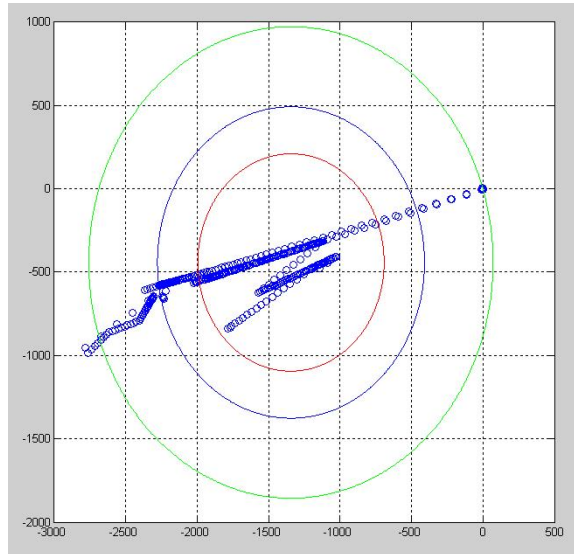


Figure 3.1: Plot of Sokkia Axis3 DGPS data on September 30, 2010

The data fit the following errors: $\text{RMS} = 934.433 \text{ m}$, $50\% \text{ CEP} = 652.045 \text{ m}$, and $95\% \text{ CEP} = 1414.717 \text{ m}$. Based on these poor results, we decided that we should obtain a new set of data from the Sokkia Axis3 in a more open environment.

Figure 3.2 is a plot of the data obtained on October 7, 2010. This set of data was retrieved in Institute Park, in a more open area than where we previously visited, over a period of ten minutes.

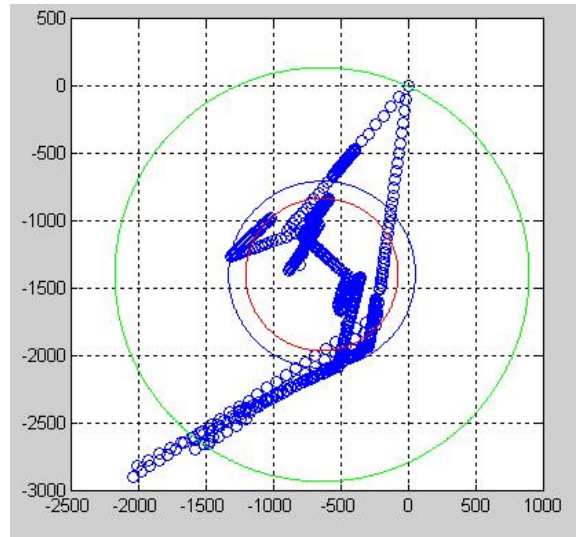


Figure 3.2: Plot of Sokkia Axis3 DGPS data on October 7, 2010

The data fit the following errors: $\text{RMS} = 693.951 \text{ m}$, $50\% \text{ CEP} = 566.339 \text{ m}$, and $95\% \text{ CEP} = 1538.323 \text{ m}$.

MediaTek MT3329

The MediaTek MT3329 data was logged next. Figure 3.3 is a plot of the data obtained on October 7, 2010. It was measured over a period of ten minutes, at approximately the same location in Institute Park where data from the Sokkia was logged.

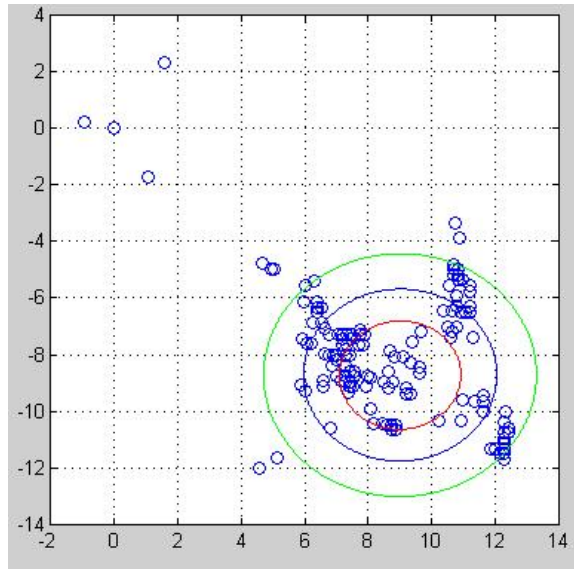


Figure 3.3: Plot of MediaTek data on October 7, 2010

This data fit the following errors: RMS = 3.049 m, 50% CEP = 1.918 m, and 95% CEP = 4.301 m.

Trimble AG252

Finally, the Trimble AG252 receiver was taken out for measurements. Figure 3.4 is a plot of the data obtained on November 6, 2010. It was measured over a period of fifteen minutes, at approximately the same location in Insitute Park where data from Sokkia and MediaTek was logged.

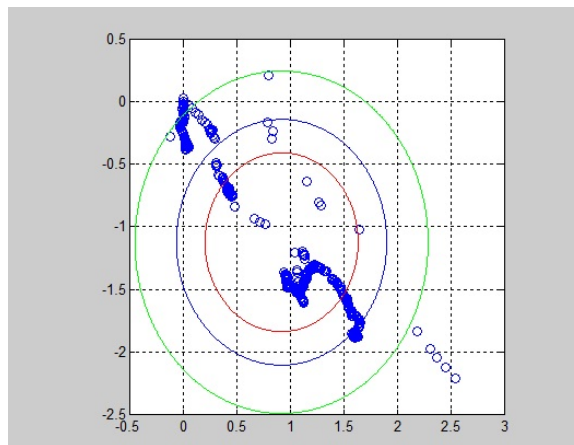


Figure 3.4: Plot of Trimble data on November 6, 2010

This data fit the following errors: RMS = 0.985 m, 50% CEP = 0.717 m, and 95% CEP = 1.373 m.

We also tested the Trimble DGPS receiver at the WPI Football Field in the winter, once we downloaded the OmniSTAR HP subscription. According to OmniSTAR's website, the HP correction service has about 10 centimeter accuracy. We obtained approximately thirty minutes of data at the field on January 16, 2011.

Figure 3.5 is a plot of the data we retrieved.

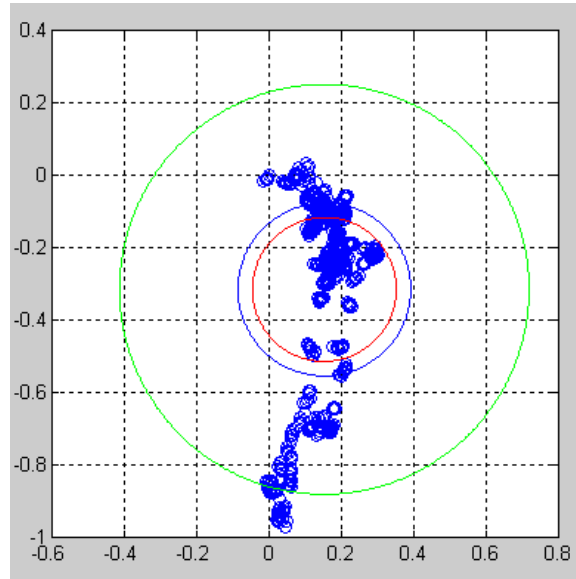


Figure 3.5: Plot of Trimble data on January 16, 2011

This data fit the following errors: $RMS = 0.240$ m, $50\% CEP = 0.199$ m, and $95\% CEP = 0.567$ m.

The results obtained from all three GPS receiver are depicted in Table 3.2.

	<i>RMS</i>	<i>50% CEP</i>	<i>95% CEP</i>
<i>Sokkia Axis 3</i>	693.951 m	566.339 m	1538.323 m
<i>MediaTek MT3329</i>	3.0489 m	1.918 m	4.301 m
<i>Trimble AG252</i>	0.239 m	0.199 m	0.567 m

Table 3.2: GPS Error Measurements

3.1.4 Conclusion

It can be seen from our results that the Sokkia has about 690 less meters of average accuracy than the MediaTek and Trimble. We hoped to see better results from the Sokkia DGPS data than the 2010 MQP team saw at the IGVC, but there appeared to be an unknown error with the device. The measurement range was nearly two miles and the accuracy was nowhere near acceptable. Based on this data, we came to the conclusion that the Sokkia Axis3 would provide no improvements to the performance of Prometheus.

Further comparisons between the MediaTek and Trimble show that the MediaTek has 2.063941 less meters of average accuracy. Based on consistent performance by the Trimble, and a subscription that has further improved this performance, we have selected the Trimble AG252 GPS for implementation on Prometheus. We believe that this GPS receiver will allow Prometheus to be a strong competitor in the Navigation Challenge at IGVC.

3.2 Inertial Measurement Unit

An IMU is a sensor that measures velocity, acceleration, and orientation in the x, y, and z directions using a combination of accelerometers and gyroscopes. In 2010, Prometheus had a magnetic compass on board which provided heading information. An IMU would provide Prometheus with accelerations in every direction, including rotational acceleration, and would also provide the robot's heading. An IMU would appear to be a great addition to Prometheus, due to the possible high accuracy the device, not to mention that it would make Prometheus a much better competitor at IGVC. With a high accuracy IMU in conjunction with the wheels encoders, Prometheus would essentially know its absolute position and velocity without the help of GPS data. We have researched several different IMUs offered at PNI Sensor Corporation and Honeywell. We also were able to obtain a sensor from the Mechanical Engineering department at WPI to test, in order to determine if that particular system would be helpful in the development of Prometheus.

3.2.1 Background

PNI Sensor Corporation manufactures high end sensors for many applications, including robotics. In order to determine if a PNI sensor can fill the void left by not having an IMU, we have compared several of their sensors. There are some important considerations to keep in mind when installing any of the PNI compasses or for that matter, any electric compass. The compass should be kept away from any large masses of ferrous metals, large electric currents and permanent magnets. It should also be kept away from the payload area, especially if there is the chance that the payload would be a large ferrous object, a permanent magnet or have a large electric current. It is also important to try and isolate the devices from excessive shock, oscillation or vibration.

The first, most basic model offered is the CompassPoint V2Xe. This was the device used by the 2010 MQP team. It is a 2-axis compass module with an onboard microprocessor. The calibration is stored in non-volatile memory which allows the calibration to be saved even after shutdown. The compass also has both hard and soft iron correction which allows the compass to account for local static effects on the magnetic field. This particular compass also has very low power consumption.

The CompassPoint Prime is a 3-axis compass using magneto-inductive sensors and a 3-axis MEMS accelerometer. The calibration includes hard and soft iron correction along with a quality of calibration score which allows the user to determine if the calibration is reasonable. This device also has several features that can be programmed by the user including output damping, reporting units and sampling configuration.

The FieldForce TCM is a high precision compass that again uses the magneto-inductive sensors and a 3-axis accelerometer. Four different types of field calibration can be used to help tailor the compass to the intended use. Full range calibration is used when more than a 45 degree tilt is possible, two-dimension calibration for measurements constrained to the horizontal, limited tilt calibration for five to forty-five degrees of tilt and hard iron calibration only which just takes into account the effect of nearby magnetic distorting components. This model also allows for sixteen different mounting orientations.

Lastly, the FieldForce TRAX is a high precision compass that has more features that allow for greater dynamic capacity. A standard unit does not have the same algorithms to mesh the data from the accelerometers with that from the compasses to develop a complete picture of the 3D orientation of the device. Appendix

C has a table containing the various characteristics that are provided by the manufacturer.

In order to better understand the advantages and disadvantages of a particular device, it is important to compare it to a similar device from another manufacturer. In this case, the PNI FieldForce TCM can be compared to the Honeywell HMC6343 Three-Axis Compass. This device uses the HMR3400 digital 3-axis tilt compensated compass which is compared to the TCM in the following table. Based on the specifications listed in the table, it would appear the FieldForce TCM is a better device. The accuracy is slightly better and the range of pitch and roll is significantly higher.

The details of our comparison can be seen in Table3.3.

		<i>TCM</i>	<i>HMR3400</i>
<i>Heading</i>	Accuracy on level		1.0deg
	Accuracy ± 60 deg	0.3 deg	4.0deg
	Resolution	0.1 deg	0.1deg
	Repeatability	0.05 deg	0.2deg
<i>Pitch and Roll</i>	Pitch and roll range	± 90 deg pitch, ± 180 deg roll	± 60 deg
	Accuracy 0 to ± 60 deg		0.5deg
	Accuracy ± 30 deg to ± 60 deg	0.2deg	1.2 deg
	Resolution	0.01 deg	0.1 deg
	Repeatability	0.05 deg	0.2 deg
<i>Magnetic Field</i>	Range	± 1.25 Gauss	± 2 Gauss
	Resolution	0.5 milli-Gauss	0.1 milli-Gauss
<i>Electrical</i>	Input voltage	3.8-5V unregulated DC	4.8-5.2V regulated DC
	Typical current	16mA	15mA
<i>Input/Output</i>	Sample rate	Max 25-32Hz	8Hz
	Communication	Binary RS232 UART	Edge connector
<i>Physical</i>	Dimensions	35×43 mm	15×38 mm
	Weight	6.8 g	3.75 g
	Operating temperature	-40 °C to 85 °C	-20 °C to 70 °C
	Storage temperature	-40 °C to 85 °C	-55 °C to 125 °C

Table 3.3: TCM and HMR3400 comparison

3.2.2 Methodology

We decided that the specifications listed for the PNI FieldForce TCM were what the team was looking for, and we contacted PNI about a possible sponsorship. They agreed to sponsor the team and sent us a FieldForce TCM XB Evaluation kit. Since receiving the device, we have downloaded an IMU plug in on the National Instruments (NI) website specifically for the TCM. After incorporating the code into our existing LabVIEW project, we were able to connect the IMU to the cRIO and gather raw data at about 30 samples per second. We specifically recorded the yaw, pitch, and roll of the device, or more specifically, the angle of the device about the x, y, and z axis. The data from the IMU would appear to be of very high accuracy and we are in the process of analyzing the data for further use.

3.2.3 IMU Results

The results we have obtained for IMU testing are from the PNI FieldForce TCM XB, since we received a sponsorship from PNI. A major task for the IMU that needed to be completed was the calibration. We did a 2D calibration once the IMU was mounted on Prometheus. This required that entire system be rotated 360 degrees at a minimum of 10 intervals, taking a reading at each step. Once this was completed, we received a calibration score representing how well the calibration was. An acceptable score, representing the error, is below two. It is also possible to have a rather high calibration score in situations where there is severe interference from outside magnetic sources.

The first time we calibrated the IMU, we were inside the lab, where the magnetic field is different and much more present than if we were outside. In order to calibrate the system, we rotated Prometheus through a full circle, taking readings at every 22.5 degrees, 16 total steps. We did this as accurately as possible by marking the floor of the lab with duct tape and labeling each step. Once the full 360 degree rotation was complete, the IMU software computes the calibration score. We received a decently low calibration score as seen in Figure 3.6, indicating that calibration was somewhat successful. As you will notice, the calibration score was rather high, but this was due to the indoor environment. We used this calibration throughout our indoor testing in the winter months.

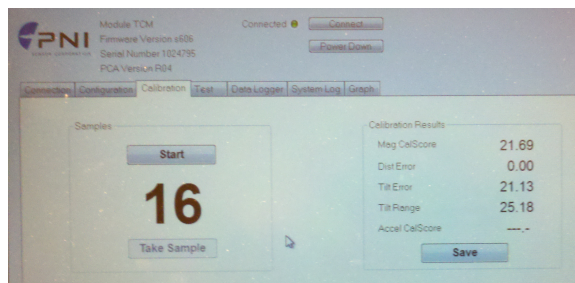


Figure 3.6: Indoor calibration results of PNI IMU

Once we were able to bring Prometheus outside, we needed to calibrate the IMU once again. Setting up the 16 different steps again outside was not practical, so we chose to use less steps. We took readings at every 30 degrees, 12 total steps. We marked the center of the circle with a small stick. We tied string from the center to each of the points at every 30 degrees to use as a marker. We rotated Prometheus through each of these steps, and once again received a low calibration score, around five, a little above the target, most likely due to interference from the On-Board computer. Overall, calibration of the IMU inside and outside was successful.

The IMU outputs heading based on an internal 3-axis magnetometer. Because the output is based on the magnetic fields around the IMU, like a compass, it can be easily deceived by local magnetic fields. During development there was two instances in which the IMU output was clearly wrong due to external influences.

The first time it was caused by the metal screws that held the IMU casing together; after placing the IMU inside Prometheus we noticed an absurd amount of distortion on the output, after experimenting with the IMU position for a while we were able to determine that the metal screws that held the custom acrylic casing was causing most of the error. Those were promptly replaced by plastic screws.

The second source of signal disturbance is caused by the ElectroMagnetic Field(EMF) generated from

the two driving wheel motors. After extensive analysis we discovered the error to be linear to the torque from the wheels. The heading variation is independent of motor direction, and increases linearly to the torque of the motors. For the purpose of mitigating the error we assume the torque is directly proportional to the known wheel speed, therefore we can subtract the predicted discrepancy in software and substantially decrease the error.

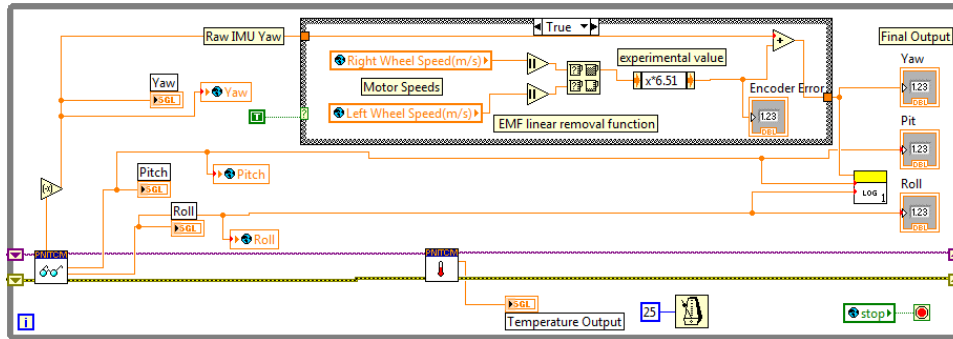


Figure 3.7: IMU EMF Error Compensation LabVIEW routine

This new approach substantially decreases the error generated by the magnetic field from the spinning motor:

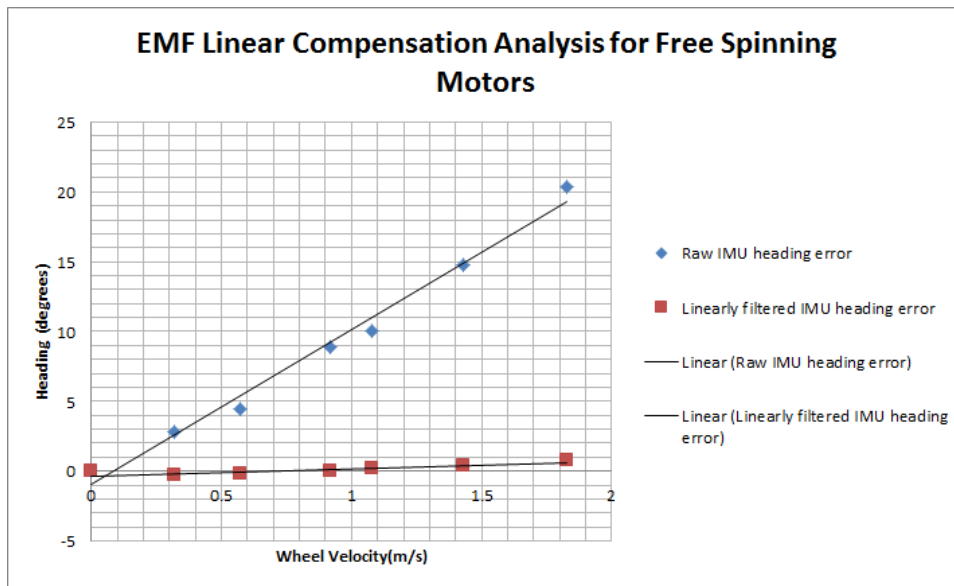


Figure 3.8: IMU output with linear compensation

This filtering process can be improved by actually calculating the torque of the motors. The torque of a dc motor is directly proportional to the current it draws. Moreover, current can be easily calculated using a current sensor in series with the motors. Due to higher priorities implementing a current measuring system was not possible, but this would be a great possibility for future work.

3.2.4 Conclusion

The PNI FieldForce TCM XB has been a great addition to Prometheus thus far. It is primarily used to collect heading information, but due to the extensive capabilities of the unit, its is much more expandable. It has successfully be calibrated for use both indoors and outside. It has also been used extensively in testing robot localization methods. The FieldForce TCM is a more accurate device than compass used on Prometheus in 2010. It will allow the robot to perform more accurately and competitively in the IGVC in 2011.

3.3 Encoders

Prometheus also relies on optical wheel encoders as sensory information. Rotary encoders are electro-mechanical sensors that give position information. Moreover, encoders are simple and reliable sensors used in a wide range of industries and manufacturing. The resolution of a encoder is based on the amount of slits around the axle. The most commonly known rotary encoders are optical encoders, and there are three main distinct types: simple, quadrature, and absolute encoders. The former two are able to give rotational direction and absolute axle position respectively.

3.3.1 Background

Optical encoders operate by attaching a disc with a series of slits cut into it onto a rotating axle. Inside the encoder housing there is a light source and a light receiver located on opposite sides of the disc. Each time the motor shaft rotates it causes the light receiver to observe a blink for each slit, which is then translated as an electrical pulse. By knowing the amount of slits on a given disc one is able to calculate the amount of degrees the shaft rotated.

Moreover, quadrature encoders have two sets of offset slits per disc, with a distinct light receiver and source for each set; the phase shift between the two slit sets supply information on the direction of the rotation, information that is nonexistent on a simple encoder.

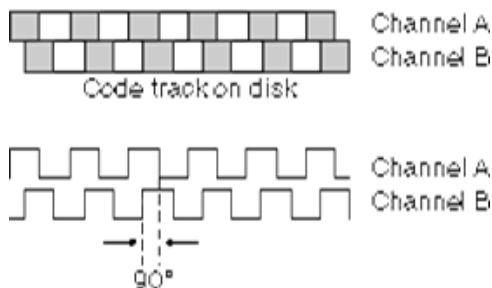


Figure 3.9: Quadrature encoder functionality, courtesy of National Instruments (enc, 2011)

3.3.2 Methodology

The encoders being used on the vehicle are two US Digital E8P Optical Quadrature Encoders with 512 counts per revolution (slits per encoder disc). The encoders are mounted inside directly to the motor shaft, before the reduction box, this allows for higher resolution measurement of the driving wheels. These specific encoders have a maximum rating of 60 kHz, which is much greater than the fastest signal generated from the motors of 15 kHz.



Figure 3.10: Us Digital E8P Optical Encoder

Encoders are by nature position sensors, but because of the relatively high speed of the signal, Prometheus processes it in time intervals. This means that the processed information comes in as distance over time units, or velocity.

Based on simple odometry experiments it was determined that some major error was occurring with the encoder output. The experiments consisted of driving Prometheus a specific distance and comparing the encoder counts, but these would show a great discrepancy between analogous runs. A test code that would drive the wheels at specific speeds while monitoring the wheel velocity was developed to test this discrepancy. Due to the nature of the system there are three possibilities for the error source: the physical encoder limitations, the signal acquisition limitations of the cRIO or a software issue.

Lastly, it is also worth mentioning that the vehicle's steered front wheel has a quadrature encoder as its only form of position feedback. The corresponding control loop for this sensor on the other hand worked seamlessly since the start of the project, and no changes beyond fine tuning were made.

3.3.3 Results

There were three possible error sources that were causing the encoder output to vary so dramatically. The first step was to check each of these possibilities to determine the true source. First, the physical encoder output was tested using an oscilloscope. Even with the motors running at the highest RPM the encoder output was a clean square wave. This narrowed the problem to either the cRIO signal acquisition process or a software bug.

Second, because the specific cRIO module used has a sampling rate of 9us per channel (110 KHz) it was assumed it would be fast enough to handle the 15 kHz signal from the encoder. From here extensive tests in conjunction with a National Instruments employee were done to try solving the issue in software. Unfortunately, after several software iterations this approach was mostly fruitless. Therefore, the only possible source of the problem left was some sort of overloading on the cRIO acquisition system.

After the problem was pinpointed to be the cRIO acquisition rate the simplest solution was to down sample the encoder signal to a lower frequency by passing it through a logic counter chip. The specific counter chip used can divide the signal by a factor of 2,4,8 or 16. For the purposes of bringing the signal to a frequency in which the cRIO could handle, it was divided by a factor of 8. One drawback of this approach is the inability to determine direction from the phase shift of the quadrature encoder output. This happens because the counter chip is asynchronous, so a given channel divided pulse has a random phase shift compared to the other channels signal.

This new approach demonstrated consistent results and allowed a solid base to build higher level abstraction on. Encoders by nature accumulate error, thus the raw output data will show discrepancies if not used in conjunction with other sensors and filtering algorithms.

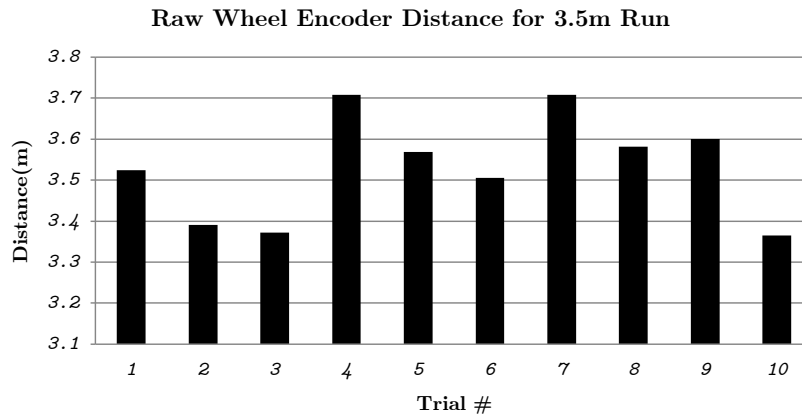


Figure 3.11: Sample encoder output information for 3.5m linear run

3.3.4 Conclusion

In conclusion, the new hardware solution proved to solve the high speed sampling problem, ultimately allowing a dependable output from the encoders. Optical encoders accumulate error over time, and therefore without proper filtering their data will diverge eventually. Although the encoder output is adequate for our current state, future work could be beneficial in developing a faster encoder data acquisition system.

Chapter 4

Software Architecture

Our robot will need a suitable software architecture to help us develop the intelligence required for the competition. A suitable architecture for Prometheus will have a high level of extensibility and modularity. Extensibility is a measure of how easily new features can be added to the system, while modularity is the amount which specific pieces of the software can be swapped out for others.

Modularity will be important when developing and testing different algorithms for the intelligence on Prometheus. In order to be most productive when testing the robot, it will need to be relatively easy to swap one algorithm in favor of another. An extensible software architecture will also ensure that it is easy to develop new programs for Prometheus. In the future, this means that Prometheus will be able to serve as a robust and modular research platform.

4.1 Background

Prometheus has two computers for programming: the National Instruments cRIO and the On-Board Computer. The NI cRIO has very few options available for programming; there are virtually no alternatives to using the VxWorks operating system that it ships with and programming the device in LabVIEW. While NI does provide support for C, it was decided to continue using LabVIEW since much of the Prometheus software requiring the cRIO was already written (National Instruments Corporation, 2010b). However, because of the known difficulties in creating large, complex programs LabVIEW for the cRIO (see Table 4.1), it was decided that the best approach to using the cRIO was to minimize the extent to which it was used for intelligence. One the approach is to this is to program and test the filtering algorithms in MATLAB and then port the scripts to LabVIEW's MathScript syntax, which is similar to MATLAB (National Instruments Corporation, 2010a).

One of the advantages of using the On-Board Computer for a majority of the programming is that is a typical desktop computer made from standard parts, so there are many options available for choosing an operating system and software architecture to run the intelligence algorithms on. The remainder of this section will explore some of the different approaches to writing software for the On-Board Computer on Prometheus.

- Compiling large FPGA programs takes upwards of 40 minutes, depending on the speed of the computer. This makes testing FPGA code increasingly more difficult as the size and complexity of the program increases.
- LabVIEW graphical interface becomes slow and occasionally freezes when working on large projects.
- Source control cannot be performed in a meaningful way since LabVIEW files are stored in a binary format.

Table 4.1: Known difficulties in creating complex programs in LabVIEW for the cRIO

One approach to designing software is to place all routines in a single process. However, giving a single process so many responsibilities makes it prone to failure. Massive processes are not modular and difficult to extend. It is difficult to follow the program’s logic and identify discrete pieces of code responsible for simple tasks. Also, it is difficult to make massive processes use standard streams in a coherent manner that a helper-process could use to extend the program’s functionality (Raymond, 2004). This also means it is difficult to write tests for massive processes since it is difficult to isolate an individual responsibilities of the program.

	<i>Advantages</i>	<i>Disadvantages</i>
<i>Single Process</i>	<ul style="list-style-type: none"> • Easy to write initially • Easier to share data between tasks 	<ul style="list-style-type: none"> • Difficult to find memory leaks • Difficult to use standard streams in a coherent manner • Must restart entire system if a single task fails
<i>Multiple Processes</i>	<ul style="list-style-type: none"> • Easy to find memory leaks • Easy to use standard streams in a coherent manner • Easy to restart a single task if it fails • Easy to debug 	<ul style="list-style-type: none"> • Difficult to write initially

Table 4.2: Comparison of the advantages and disadvantages of using a single process or multiple processes when creating software, assuming code in not intentionally obfuscated

An alternative to the massive-process model is to break the program into many small, communicating processes, each with a single responsibility. However, to fully benefit from structuring a program in this manner, the operating system must support inexpensive process-spawning and simple inter-process communications (Raymond, 2004). Fortunately, these are both supported by Linux, and there are also many software architectures available that aid in the process of designing and implementing such systems. For our project, we researched several frameworks specifically designed for unmanned, robotic platforms that would make it easy for us to implement many small, communicating processes.

4.1.1 Joint Architecture for Unmanned Systems

The Joint Architecture for Unmanned Systems (JAUS) is an open architecture originally developed as an initiative by the US Department of Defense and now maintained by the Society of Automotive Engineers (SAE) (Joint Ground Robotics Enterprise, 2007). JAUS is message-based architecture, meaning that discrete computing systems share information using an application-layer protocol. The specification is intended to be highly configurable and interoperable between differing unmanned systems and components (Rowe & Wagner, 2008). Unfortunately, since its transition to SAE, the new specification documents are no longer freely available (SAE International, 2010). Older specifications remain publicly available; however, these are no longer maintained (Joint Ground Robotics Enterprise, 2007).

The Reference Architecture Specification is one of the JAUS documents no longer being maintained (Joint Ground Robotics Enterprise, 2007). Developed by the JAUS Working Group in the later 1990s, the reference architecture provides a base upon which software frameworks can be written.

4.1.2 Robot Operating System

Robot Operating System (ROS) is an open source framework originally developed to meet the needs of large-scale domestic robots developed as a part of the STAIR project at Stanford University and the Personal Robots Program at Willow Garage (Quigley et al., 2009).

A program using ROS consists of a set of intercommunicating processes, or *nodes*, that each have a single responsibility. The framework is distributed and runs using multiple processes and is designed to support being distributed among multiple hosts. Rather than providing a single monolith process for managing the system as a whole, ROS provides many small tools for working with the framework. The framework also works independently of the operating system or language chosen to run it on.

In practice, ROS is implemented as a set of packages for Linux. The packages provide a set of dependencies for compiling nodes as well as tools for debugging and diagnosing problems. At present, ROS only supports the Ubuntu Linux distribution and the C++ and Python programming languages in its stable release (ROS, 2010).

Communications Between the On-Board Computer and the cRIO

Because there is no ROS support for LabVIEW, the cRIO and On-Board Computer will need to communicate using a very simple messaging protocol. There are two options available for this: use a single stream for passing all types of messages, or use multiple streams running on different ports, each of which passes a different message type. After tabulating the advantages of each method in Table 4.2 it was decided that using a separate stream for each service was clearly the best option for network communications.

4.2 Methodology

The computer each sensor was connected to changed as a result of the decision to make the cRIO less responsible for Prometheus' intelligence. In the current setup, the LIDAR is connected directly to the On-

Board Computer because it was not used by the Kalman filter. In addition, the GPS is connected to the On-Board Computer for two reasons. First, it is easier to communicate with the sensor stream from the On-Board Computer. Second, Prometheus' path planner needs a method of converting GPS coordinates to Cartesian coordinates relative to the starting position of the robot. This is easiest when the path planner, which runs on the On-Board Computer, can send messages to another process requesting the conversion rather than requesting the conversion over the network from the cRIO. Figure 4.1 shows an overview of the major computing devices and sensors are connected.

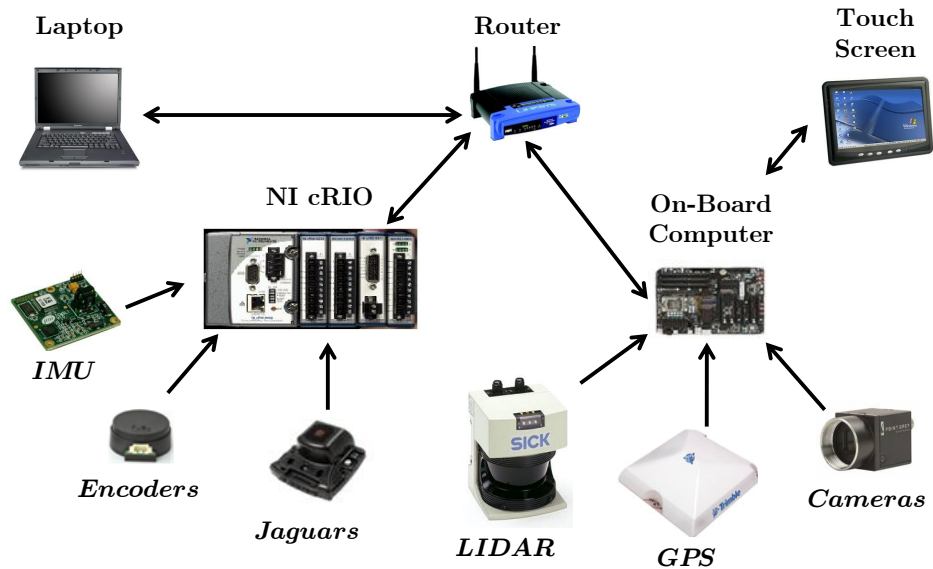


Figure 4.1: The major sensors and computing devices of Prometheus 2011. In this design, the cRIO is primarily used for motor controls and odometry information.

After weighing the advantages and disadvantages of using JAUS or ROS (see Table 4.3), it was decided to use ROS as the framework for our robot. The primary reason for this decision is that ROS is extensively documented, comes packaged with a number of libraries that will be useful for Prometheus, and has a growing community of users. Because ROS does not already have ROS packages to support JAUS, our team will create a ROS wrapper for the Jr Middleware JAUS libraries. Jr Middleware is small API that supports the JAUS specifications required by the IGVC JAUS challenge (Devivo AST, 2008).

One of the most useful tools that ROS provides is named “rviz.” Rviz is a robotics visualization software that listens to ROS’ standardized messages and is capable of creating a 3D model of the robot’s perceived environment. The tool is extremely useful for debugging robotics software because it makes it possible to see what the robot perceives, and because ROS supports multiple hosts, a laptop can connect to the Prometheus Wi-Fi to visualize the robot’s perceptions in real-time. Figure 4.2 shows rviz being used to visualize Prometheus as it performs mapping indoors. ROS also provides tools for logging ROS messages, and when used in conjunction with rviz, a history of the robot’s performance can be visualized. These tools will prove invaluable as we test Prometheus and work to improve its intelligence.

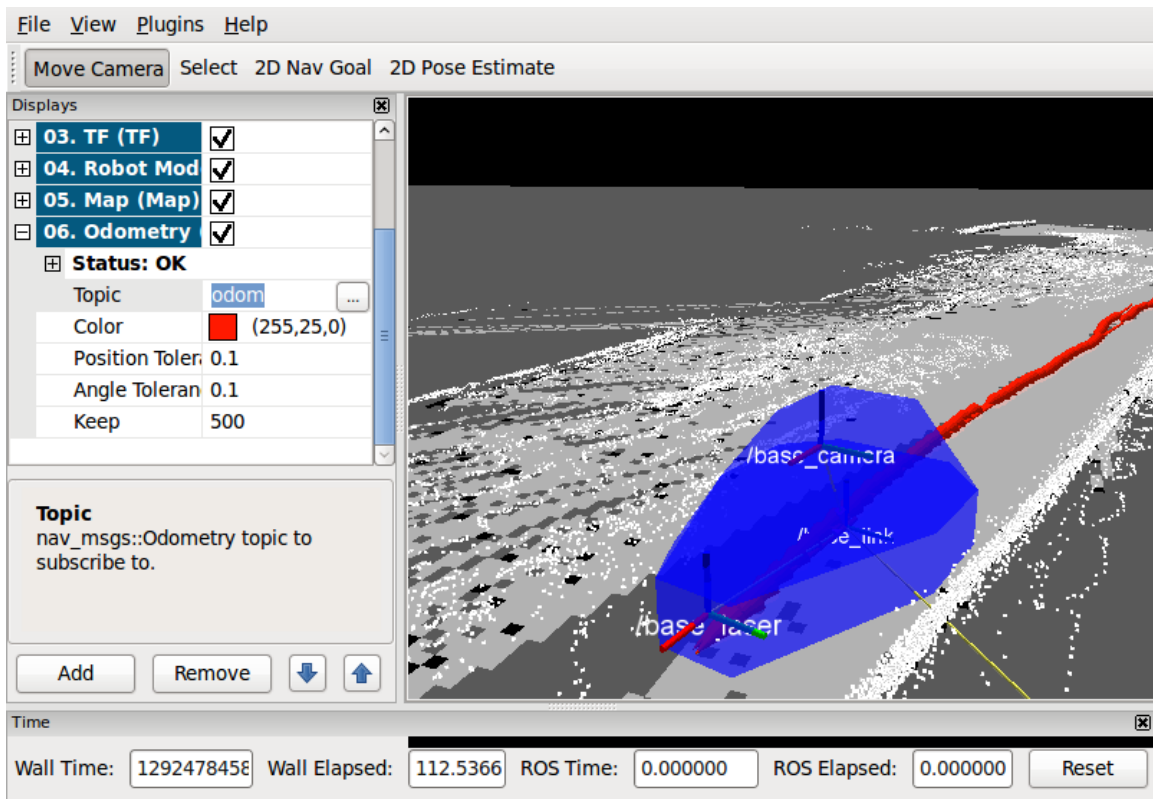


Figure 4.2: Rviz, a tool that is integrated with ROS, being used to visualize Prometheus as it performs indoor mapping.

	<i>Advantages</i>	<i>Disadvantages</i>
<i>OpenJAUS</i>	<ul style="list-style-type: none"> • Programs are split into <i>components</i> • Uses JAUS protocol for passing messages between components, a standard that will need to be used to compete in the IGVC JAUS Challenge 	<ul style="list-style-type: none"> • Small user-base
<i>Robot Operating System (ROS)</i>	<ul style="list-style-type: none"> • Programs are split into <i>nodes</i> • Uses Simple passing message • Integrates with many pre-existing open source software packages • Contains graphical tools for debugging • Large user-base, widely used in robotics community 	<ul style="list-style-type: none"> • Does not currently have JAUS support

Table 4.3: Comparison of the advantages and disadvantages of different robotics software frameworks that were considered for use on Prometheus

Communications Between the On-Board Computer and the cRIO

The cRIO and On-Board Computer perform a handshake using a TCP connection before any other actions are taken. During the handshake, the cRIO and On-Board Computer exchange information about what ports they are using for specific services. Examples of services include the cRIO listening for velocity commands and the On-Board Computer listening for odometry messages. After the handshake is performed, the cRIO continues to wait for more handshake messages, and if it receives another handshake, it restarts all of its processes to work with the new settings. The handshake was designed this way so that the robot can continue to operate in the event that the On-Board Computer crashes. The handshake is also implemented on both the cRIO and On-Board Computer in such a way that either device can start first, and the handshake will be completed.

An alternative to the current handshake would be to have the handshake not send any information about which ports are running which services. In this circumstance, each device would need to be hard-coded to know which port to use for each service. The problem with this design is that each time a port would need to be changed, both the software on the cRIO and On-Board Computer would need to be updated.

Another aspect of the handshake is that the cRIO determines the address of the On-Board Computer by performing the handshake. This allows us to use a dynamic IP address for the On-Board Computer and avoid hard-coding its IP address in the cRIO's software. The advantage of using a dynamic IP address is that less network configuration is necessary and tests of the cRIO's services can be easily run from a computer other than the On-Board Computer.

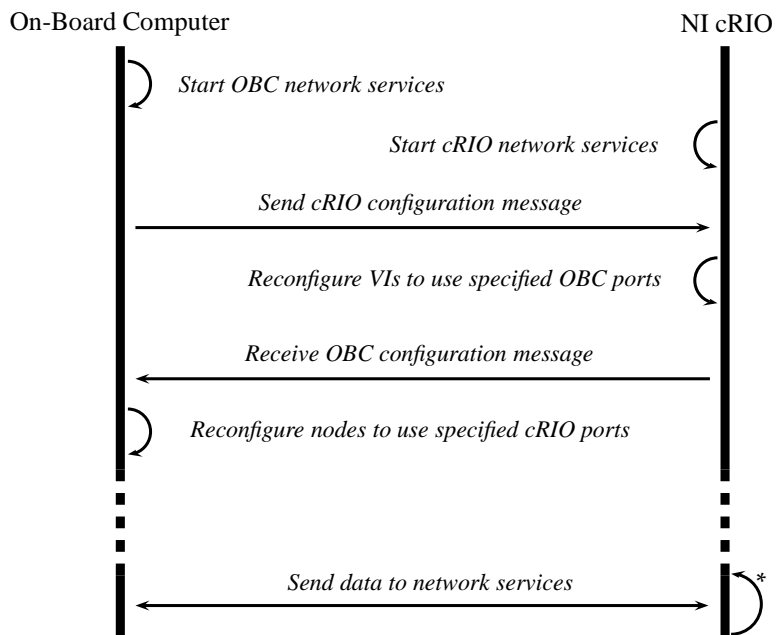


Figure 4.3: The handshake used to send configuration information between the cRIO and On-Board Computer. The cRIO’s IP address and handshake port are known beforehand to the On-Board Computer. The handshake begins when the On-Board Computer sends a configuration message to the cRIO’s handshake TCP server. The configuration message contains a series of 32-bit integers, each of which represents port running a service on the On-Board Computer. The cRIO then replies with a similar message where each integer represents a port running a service on the cRIO. The expected ordering of the ports in both messages is known beforehand to both the cRIO and On-Board Computer.

	<i>Advantages</i>	<i>Disadvantages</i>
<i>Single Stream</i>	<ul style="list-style-type: none"> • Easy to write initially 	<ul style="list-style-type: none"> • Requires headers for each message type • A router must be constructed in software to pass messages to their appropriate processes • Creates a bottleneck on the system that all network data must flow through • All processes fail if the stream is broken
<i>Multiple Streams</i>	<ul style="list-style-type: none"> • No headers are required, so data can be sent as a simple byte stream. • Easy to extend to more services once written • If a single stream breaks, the rest of the system won't necessarily halt 	<ul style="list-style-type: none"> • Difficult to write initially

Table 4.4: Comparison of the advantages and disadvantages of using a single process or multiple processes when creating software, assuming code is not intentionally obfuscated

4.3 Results

The programs on the On-Board Computer are organized into several ROS packages. Each package contains zero or more programs that can be run as individual processes, called nodes in ROS terminology. An outline of the functionality of each ROS package implemented for this project is as follows.

crio_comm The `crio_comm` package handles communications between the NI cRIO and the On-Board Computer. The package uses the `crio_init` node to initialize the handshake with the cRIO. Once communications is established, the package serves as a wrapper for other nodes that need to communicate with the cRIO. The package has two kinds of nodes for communication: sources and sinks. Sources listen to a port on the On-Board Computer for packages from the cRIO. When sources receive data, they build a ROS message that they immediately publish. Sinks subscribe to a ROS topic and forward the data they receive to the cRIO over the network.

The two sinks of the `crio_comm` package are the `cmd_vel_sink` and the `gps_sink`. These respectively handle velocity command messages and GPS messages. Velocity commands are specified as a list of two doubles and are forwarded to the cRIO using TCP. The first specifies the forward velocity along the x-axis of the robot's local frame (in other words, the speed the robot should drive), and the second indicates the angular velocity (the radians per second the robot should turn). Because of how the coordinate frame for the robot is defined, a positive angular rotation will turn the robot to the left.

The `gps_sink` uses TCP to send odometry information from the GPS to the cRIO. It receives Cartesian coordinates in meters relative to the starting position of the robot. The coordinates are specified as a

list of two doubles, x and y , where the x -axis points north, the y -axis points west.

The only source of the `crio_comm` package is the `odom_source` node. This package receives the position and rotation of the robot relative to the world frame that is output by the localization algorithm running on the cRIO. The `odom_source` node uses this information to broadcast a ROS odometry message as well as a ROS transform.

local_map This package listens to laser range data from the SICK LMS and detected lines sent from the `line_detection` package. It then stores the data as raw measurements and periodically builds an occupancy grid that it publishes over a ROS A more complete description of this package can be found in Section 5.2.

path_planner This package listens to navigation commands and occupancy grid messages. It then plans a path from the robot to the navigation goal. It then forwards this path to the `motion_planner`, which converts the path into velocity commands for the cRIO. A more complete description of this package can be found in Section 5.3.

motion_planner The `motion_planner` package is responsible for following paths it receives from the `path_planner`. It interprets the path into linear and angular velocity commands and then sends these commands to the `crio_comm` package. The `motion_planner` stops the robot if it veers from the path it was assigned to follow. A more complete description of this package can be found in Section 5.3.

line_detection The `line_detection` node processes the images from the cameras to detect line. A more complete description of this package can be found in Section 5.4.

prometheus_teleop This package contains various code for controlling the robot, both in autonomous and teleoperated modes. Most importantly, this package contains the `send_nav_command` node, which reads navigation goals from its standard input and forwards these to the path planner. This node operates in either GPS mode or Cartesian mode. In Cartesian mode, the commands are specified as a position in the world frame, x and y , in meters. In GPS mode, the commands are specified as longitude and latitude, and the node uses the `gps` package to lookup a conversion to the world frame.

The package also contains two teleoperation nodes that are useful for testing. The first is the `send_velocity_command` node, which allows the user to manually specify velocity commands at the command line. This node is provided as a convenience for testing whether velocity commands are being accurately handled by the cRIO. The second is the `joy_control` node, which allows the user to send navigation commands using a joystick or gamepad attached to the On-Board Computer. This node is useful for ensure the network connection between the cRIO and On-Board Computer.

In addition, the package contains `virtual_odometry_server` node that is used strictly for testing the robot. This node replaces the `odom_source` and `cmd_vel_sink` nodes of the `crio_comm` package, to virtually move the robot in ideal conditions. The node reads velocity commands and then periodically publishes the odometry information that would result from following the command.

prometheus_sensors This package contains the ROS messages used for communicating with the GPS as well as launch files for starting the LIDAR and cameras.

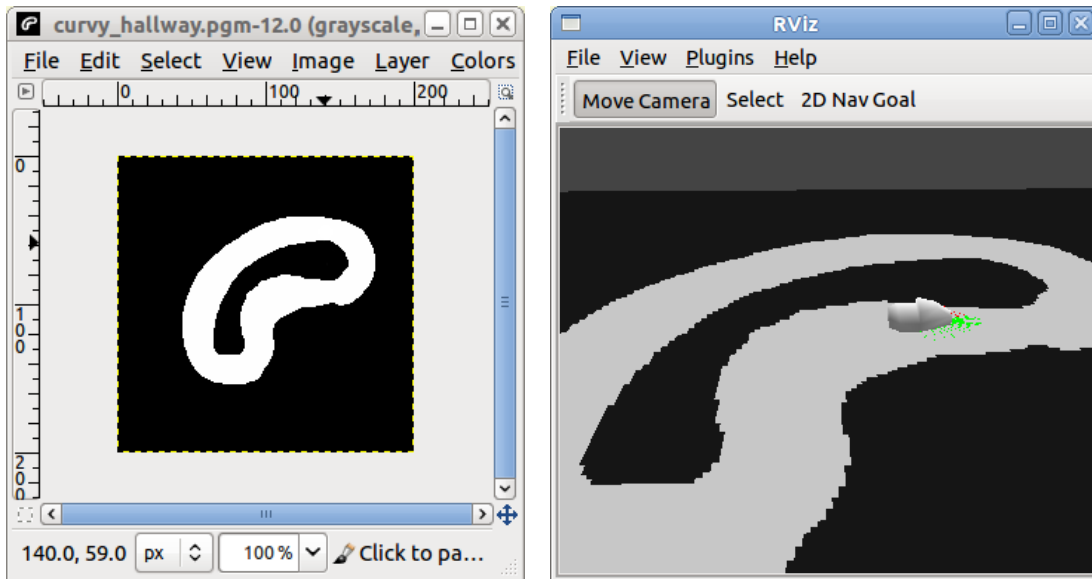
gps This package communicates with the GPS and reads and interprets longitude and latitude values. It performs two functions in ROS: it serves as a broadcaster for GPS data, and it runs a GPS conversion

service. The broadcast GPS messages are eventually sent to the cRIO by the `crio_comm` package's `gps_sink` node. The GPS conversion service is used by the `send_nav_command` node for converting GPS coordinates into Cartesian coordinates. A more complete description of this process can be found in Section 5.2.

prometheus This package contains configuration and launch files for rviz and a launch file that starts all nodes that are critical to perform path planning including nodes from the cRIO communication, mapping, path planning, motion planning packages.

prometheus_description This package contains configuration and launch files that describe the physical shape of the robot. This includes a 3D model of the robot that is useful for visualizing the robot's shape in rviz. In addition, the package contains a launch file that starts static transform publishers that periodically broadcast the transforms from the robot's base link to each of its sensors.

map_io This package handles input and output of ROS occupancy grids. The package contains nodes to load and save occupancy grids using image files. The package uses the Portable Gray Map (PGM) files, which use an image format that is designed to be easy to write programs for (Poskanzer, 2003). The format is ideal for testing ROS maps because it can be edited in GIMP, an image editor for Linux, and it supports a human-readable ASCII format as well.



(a) A map being edited as an image file in GIMP (b) Prometheus navigating on an occupancy grid loaded from an image file.

Figure 4.4: A simulation of the robot performing navigation on a map loaded from an image file.

The `map_io` package also has a `static_map_publisher` node that is useful for integration testing. This node loads a map from an image file and continuously broadcasts it at a specified rate.

integration_tests The `integration_tests` package contains various tests for ensuring the ROS packages are communicating properly, and that the robot's nodes are properly working together as a whole. Most notable of these tests are the simulations, which make use of the `map_io` package and the `virtual_odometry_server` node to perform navigation on maps loaded from image files.

4.4 Conclusion

We chose to use ROS because of its strong community support and integration with popular libraries. In addition, we are using a multi-process architecture because this is used by ROS by default. Since ROS does not have a library for LabVIEW communication, we created our own communication scheme. This scheme is service based and separates the communications into multiple streams.

Communications Between the On-Board Computer and the cRIO

It is important to note that there is a flaw in the handshake from a security standpoint. The only method the cRIO employs to check that the On-Board Computer is correct is that it checks the length of the configuration message it receives to the expected length; if the two are not equal, it drops the connection. While it is possible for an adversary to disable the robot by sending the correct number of bytes to the cRIO's handshake port, it is assumed for this project that the wireless router provides enough security with its WPA encryption.

Chapter 5

Intelligence

The project's primary research focus is to enhance Prometheus' autonomous capabilities. The intelligence of our vehicle correlates to its ability to fuse the many different sensors on-board, generate a local map of its surroundings, as well as plan and traverse this map to a specific goal. There are boundless ways of implementing the different aspects of Prometheus' intelligence. This section outlines the different approaches our team researched, developed, and implemented to ultimately bring Prometheus to a competitive state for the IGVC.

5.1 Sensor Fusion

Accurate localization, navigation, and mapping require the output of multiple types of sensors in order to provide as much information about the robot's whereabouts as possible. We are looking to implement the sensors for localization by fusing their useful data, then calculating the state of the robot in terms of its x position, y position, and heading. Unfortunately, each sensor has some amount of error in its returned data. This makes the exact state of the system difficult to determine as the error accumulates, and in effect, inaccurate. Kalman filters can help with this problem.

5.1.1 Background

There are several definitions available for the Kalman filter. A good way of summing them up is by saying that they recursively estimate an evolving state over time. This is done by performing calculations on measurements taken at known intervals of time during observation of that state. Kalman filters are applicable to our robot in that it is meant to be in motion and will therefore have a constantly changing position and heading.

The implementation of a Kalman filter has proven to be capable of minimizing error, and doing so in an efficient manner. The consistent use of statistical measures of uncertainty makes it possible to quantitatively evaluate the role each sensor plays in overall system performance. Further, the linear recursive nature of the algorithm ensures that its application is simple. For these reasons, the Kalman filter has found widespread application in many different data fusion problems. (Siciliano & Khatib, 2008). This makes the Kalman

filter well suited for our sensor error correction and fusion needs.

The Kalman filter can be broken down into two phases: time update and measurement update. The time update equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the estimates for the next time step (Welch & Bishop, 1995). In the time update phase, the filter uses the equations below:

$$x(k|k-1) = F(k)x(k-1|k-1) + B(k)u(k) \quad (5.1)$$

$$P(k|k-1) = F(k)P(k-1|k-1)F^T(k) + G(k)Q(k)G^T(k) \quad (5.2)$$

where $F(k)$ is the state transition matrix, $B(k)$ is the control input model, $u(k)$ is the odometry vector, $G(k)$ is the Gaussian noise, and $Q(k)$ is the process uncertainty. Equation 5.1 represents the predicted state of the system, while Equation 5.2 represents the error covariance of that measurement. Covariance is the average value obtained after calculating the standard deviations of the measurements taken. Calculating the covariance provides a good idea of how consistent the measurements affecting the system were (Siciliano & Khatib, 2008).

During the measurement update phase, the sensor measurements are collected, or in other words, observations are made. Based on the acquired data, an update of the prediction equations takes place. To do so, the Kalman gain (K) is computed, and then used to update the state estimate equation and error covariance matrix. The Kalman gain is a calculated value that helps to determine the certainty of the equations. A way of thinking about the weighting by K is that as the measurement error covariance approaches zero, the actual measurement is “trusted” more and more, while the predicted measurement is trusted less and less (Dyer, 2002). The equation for calculating K is:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (5.3)$$

where H is the observation matrix and R is the sensor noise matrix. An observation matrix is necessary when the information in the state matrix does not match that of the measurement matrix. The state matrix needs to be converted into the same units and format as the measurement matrix so that calculations can be carried out. The sensor noise matrix contains values that can be altered in order to produce better filter results.

When it comes to choosing a type of Kalman filter, there are two main options. The first is the standard Kalman filter. Although its equations are simpler, and it is good for dealing with sensor error, it only works well for linear systems.

The other option is an extended Kalman filter (EKF). This filter’s method is to apply the first-order Taylor expansion to the system, forcing a non-linear system to be linear. The Taylor Expansion allows a way of evaluating a function over an interval, but using only information about that function and its derivatives at one point. This is useful for using current information to predict the next state of the system. Unfortunately, the expansion is only capable of calculating an approximation. This can produce large prediction errors, especially when applied to highly non-linear functions. Another downside is the need to create Jacobian matrices, which serve to linearize the non-linear functions. Jacobian matrices can become large and in effect slow down the system, which is why the need to create them is a disadvantage (Dyer, 2002).

5.1.2 Methodology

We had to analyze the Kalman filter options before choosing which we would implement. Prometheus' movements cannot always be represented by linear equations. This means that the original Kalman filter would not be very effective in our system. After coming to this conclusion, we moved on to looking at our non-linear filter option, the extended Kalman filter. Based on the good number of examples, explanations, and reviews we had found of the EKF, we decided that the EKF would be an acceptable choice for the type of Kalman filter to use for reducing the error in the calculation of the robot's state. The first step was to analyze the EKF equations and adapt them to our implementation. This involved altering and combining equations found, and determining what roles our sensors would play in terms of filter inputs. Figure 5.2 depicts the general equations necessary to create this Extended Kalman Filter, as well as what phase of the filter they are used in.

Figure 5.1 aids in visualization of how we determine the robot's position in relation to its initial position at 0, 0. By adding the x component of the robot's new position to the x component of its former position, we get the overall x position. The same can be done for the y components.

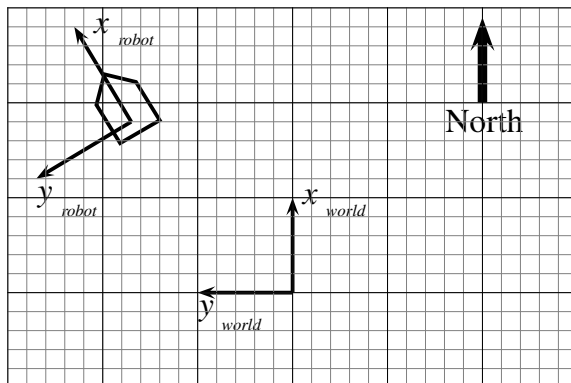


Figure 5.1: The frame of the robot relative to the world frame. The world frame's x-axis always points north, it's y-axis always points west, and its origin is always positioned at the starting position of the robot.

Since the position is determined based on past position, any error introduced by the sensors will be accumulated. This is where the Extended Kalman filter comes into play. We created two separate Kalman filters. One (referred to as filter 1) is applied to the raw encoder and IMU data, while the other (referred to as filter 2) is applied to the raw DGPS and IMU data. The encoders provide the velocities of the left and right wheels in meters per second, the IMU provides the heading of the robot in degrees, and the DGPS provides the location of the robot in Cartesian coordinates measured in meters. We chose to apply our sensor data to two separate filters because it allowed for modular testing. We could observe the results being obtained from one filter and alter it, and then do the same to the other. This also allowed for filter implementation during autonomous runs even when the GPS was experiencing interference or the encoders were being unreliable. Later in the paper we will discuss how the data from these two filters was fused to produce one overall state. These filters' odometry vectors (state of the robot) are of the form:

$$\vec{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T \quad (5.4)$$

This means we are looking to get the robot's x position, y position, and heading based on the data

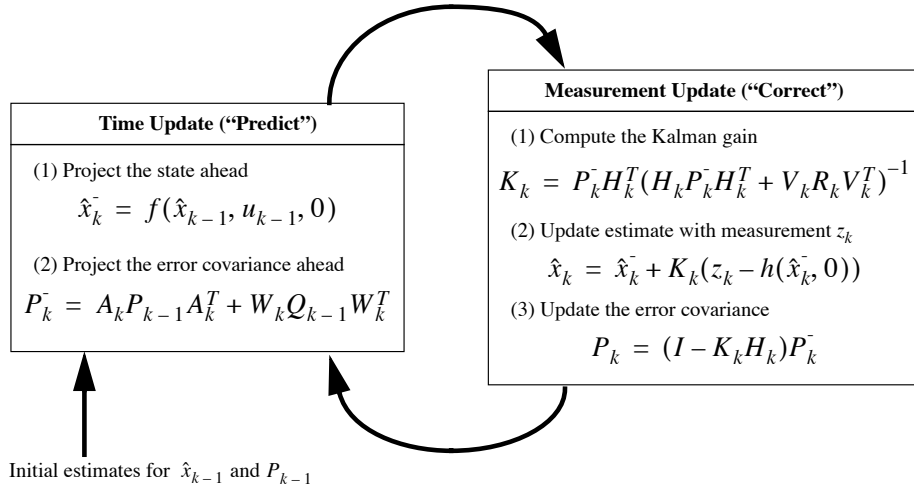


Figure 5.2: Operation of the extended Kalman filter. Image courtesy of Welch & Bishop (1995)

provided by the sensors. The following is a breakdown of the equations used in our extended Kalman filters.

1. Average Velocity. Filter 1: Use the values from the left (lv) and right (rv) wheel encoders to calculate the average velocity (av) of the robot. Filter 2: No need to calculate the average velocity.

$$av_1 = \frac{lv + rv}{2} \quad (5.5)$$

2. Translational Displacement. Filter 1: Use the average velocity and the amount of time between sensor readings (Δt) to calculate the translational displacement ($d.t$) of the robot (distance travelled). Filter 2: Perform the Pythagorean Theorem on the change in x and y position.

$$d.t_1 = av * \Delta time \quad (5.6)$$

$$d.t_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (5.7)$$

3. Time Update: Project the State Forward

$$x^- = \begin{bmatrix} x^+ \\ y^+ \\ \theta^+ \end{bmatrix} \quad (5.8)$$

Calculate the new values of x, y, and theta. This is done by adding the change in x and y to the former values of x and y. Since the IMU is significantly more accurate than the encoders it is better to obtain the new heading (θ) by taking the reading from the IMU, rather than calculating it with the encoders.

$$x^+ = x + d.t * \cos(\theta) \quad (5.9)$$

$$y^+ = y + d.t * \sin(\theta) \quad (5.10)$$

$$\theta^+ = \theta \quad (5.11)$$

4. Jacobians. The state equation then needs to be linearized, so we must compute the Jacobians for use in part 2 of the Time Update phase.

$$A = \begin{bmatrix} 1 & 0 & -d.t * \sin(\theta) \\ 0 & 1 & d.t * \cos(\theta) \\ 0 & 0 & 1 \end{bmatrix} \quad (5.12)$$

$$B = \begin{bmatrix} \cos(\theta) & -d.t * \sin(\theta) \\ \sin(\theta) & d.t * \cos(\theta) \\ 0 & 1 \end{bmatrix} \quad (5.13)$$

$$W = \begin{bmatrix} \cos(oldheading) * \Delta time & 0 \\ \sin(oldheading) * \Delta time & 0 \\ 0 & \Delta time \end{bmatrix} \quad (5.14)$$

5. Covariance. The covariance (P) during the time update phase is calculated as follows:

$$P^- = A * P * A' + Q \quad (5.15)$$

Covariance is the past covariance, and Q is a three by three process noise matrix with values in the 1,1; 2,2; 3,3 positions. These values are obtained through testing and can be altered for different performance from the filter.

6. State Update: Update the State with Measurements

7. Measurement Matrix (z). A measurement matrix must be created. Filter 1: We are measuring the left wheel velocity, right wheel velocity, and robot heading. Filter 2: We are measuring the robot's x position (*gps_x*), y position (*gps_y*), and heading.

$$z_1 = \begin{bmatrix} lv \\ rv \\ \theta \end{bmatrix} \quad (5.16)$$

$$z_2 = \begin{bmatrix} gps_x \\ gps_y \\ \theta \end{bmatrix} \quad (5.17)$$

8. Measurement Observation Matrix. A measurement observation matrix must be created. This matrix is used to convert the odometry vector values into the same form as the measurement matrix.

$$H_1 = \begin{bmatrix} \frac{1}{\Delta time} & 0 & 0 \\ 0 & \frac{1}{\Delta time} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.18)$$

$$H_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.19)$$

9. Sensor Noise. Sensor noise must be introduced. This is represented by a three by three R matrix with values in the same positions as the Q matrix. These can also be altered to change filter performance.

10. Kalman Gain.

$$K = \frac{P^- * H'}{H * P^- * H' + R} \quad (5.20)$$

11. Update the state and covariance.

$$x = x^- + K(z - H * x^-) \quad (5.21)$$

$$P = (eye(3) - K * H) * P^- \quad (5.22)$$

What is now present is an x matrix containing the x, y, and heading components of the filtered states. These values are used to represent the position of the robot, and are looped back into the filter for its continued implementation.

Each filter has varying levels of accuracy which can change at any given time. This makes it desirable to be able to change which filter output is being relied upon. Equation 5.23 can be used to weight each state based on its covariance. The lower the error involved in a state's estimation, the more that state is weighted and trusted in its contribution to determining the overall state (Drolet et al., 2000).

$$x_{1,2} = \frac{P_2}{P_1 + P_2} x_1 + \frac{P_1}{P_1 + P_2} x_2 \quad (5.23)$$

The filtered state of the robot is in relation to its starting position (0, 0). However, the waypoint navigation challenge provides us with waypoints in terms of latitude and longitude in the world frame. This means that in order for the filtered output to be useful, the latitudes and longitudes must be converted to Cartesian coordinates. This conversion is detailed in the World Representation chapter.

The GPS receiver was necessary to bridge the gap between Cartesian coordinates and latitude and longitude. The way we dealt with this challenge was by converting all of the given waypoints from latitude, longitude, to local x, y coordinates. We then use the GPS receiver to provide us with the latitude and longitude of the robot's initial position. This location is then converted to Cartesian coordinates, and subtracted from all of the waypoint coordinates to make all values in the same frame.

5.1.3 Results

Filter 1-Encoders and IMU

Our first positive results were obtained on February 28, 2011. A LabVIEW vi was made that allowed for the live plotting of the robot's position as it travelled. Figure 5.3 shows the data obtained while driving the robot forward and then beginning to take a ninety degree turn.

The yellow plot represents the path calculated with raw data from the encoders only. Since this path did not include the IMU, the initial heading is zero, whereas IMU inclusive data has an initial heading of whatever the IMU is giving at the start. The light blue plot represents the path calculated with the raw data from the encoders, as well as the IMU. Its Cartesian coordinates (in meters) at the end of data logging are contained in the light blue box. The green plot represents the path calculated by our EKF. This plot indicated progress in that it was producing a shape similar to the raw data, with the offset likely being attributed to error correction. Its Cartesian coordinates at the end of data logging are contained in the green box.

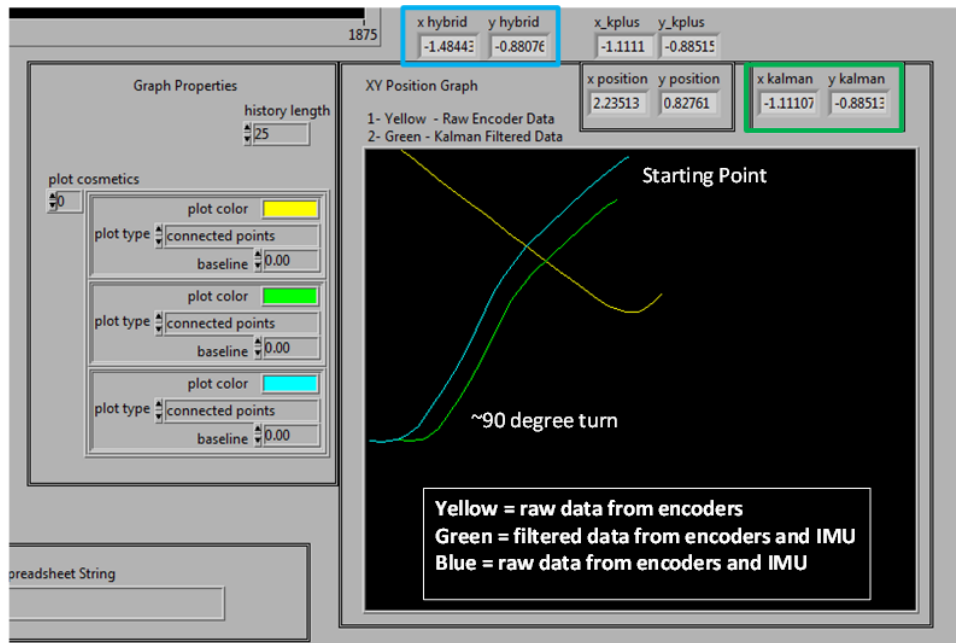


Figure 5.3: LabVIEW Logging of Raw and Filtered Path from Sensor Data

Additional testing was done in a way that allowed us to observe the paths being calculated over a longer range of time. LabVIEW code was written to perform live logging to a text file of the Cartesian coordinates being calculated by the raw encoder and IMU data, as well as the EKF. We then drove the robot for two laps around the main hallways of the first floor of Atwater Kent. (It is important to note that at the beginning of logging the laps, some raw data was still present from past testing, and therefore offset the starting coordinates of the raw data representation). Figure 5.4 is a Matlab plot of the Cartesian coordinates calculated with the raw encoder and IMU data, as well as those calculated by the EKF.

The red plot represents the raw data and the blue represents the filtered data. It can be observed in the raw data that the second lap was offset from the first, an indication of error accumulation. The filtered data

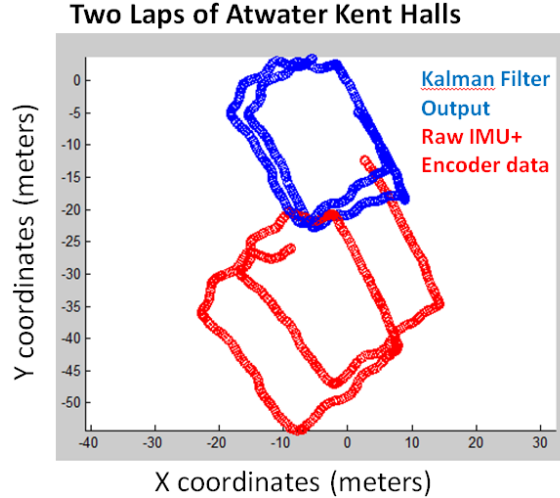


Figure 5.4: LabVIEW Logging of Raw and Filtered Path from Sensor Data

shows the second lap to be much more in line with the first lap, an indication that the filter was reducing the error accumulation. We obtained these results while using an R noise value matrix of $[\text{.0001 } 0 \ 0; 0 \ \text{.0001 } 0; 0 \ 0 \ \text{.000001}]$, and Q and starting covariance matrices of zeros.

Once we knew our filter was accomplishing error reduction we moved on to improving its accuracy in estimating distances traveled. We drove the robot along a generally straight path and marked its starting and ending points. The distance traveled was measured with a tape measure. We then plotted the logged raw and filtered data in Matlab to observe the path calculated. By using Matlab’s cursor feature we were able to determine the starting and ending coordinates of the plots. Equation 5.24 was used to calculate the distance travelled according to the raw data and filter plots.

$$distance = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (5.24)$$

By comparing the raw distance versus the actual distance and the filtered distance versus the actual distance, we could determine which was closer and alter the noise and starting covariance values to improve the filter’s performance. Table 5.1 summarizes the results.

Actual Distance (meters)	Mode	Filtered Difference (meters)	Raw Difference (meters)	R Noise Matrix	Starting Covariance
7.62	Controlled	1.0882	0.24	0	[0.0001, 0.0001, 0.000001]
7.62	Controlled	3.576	1.39	[0.0001, 0.000001]	[0.0001, 0.0001, 0.000001]
7.62	Controlled	0.085877	0.386059	[0.001, 0.001, 0.00000001]	[0.0001, 0.0001, 0.000001]
3.070225	Autonomous	-1.51178	-1.27178	[0.001, 0.001, 0.00000001]	[0.0001, 0.0001, 0.000001]
6.00075	Autonomous	-1.29025	-1.18597	[0.001, 0.001, 0.00000001]	[0.0001, 0.0001, 0.000001]
4.0767	Autonomous	-1.3793	N/A	[0.00001, 0.00001, 0.000000001]	[0.00001, 0.00001, 0.000000001]
3.54965	Autonomous	-1.30735	1.57135	[0.01, 0.01, 0.000000001]	[0.00001, 0.00001, 0.000000001]
3.76555	Autonomous	-1.47645	-1.33845	[0.015, 0.015, 0.000000001]	[0.00001, 0.00001, 0.000000001]
3.641725	Autonomous	-1.21928	-1.43128	[0.015, 0.015, 0.000000001]	[0.015, 0.015, 0.000000001]
5.27685	Autonomous	-1.52815	-2.32215	[0.015, 0.015, 0.000000001]	[0.015, 0.015, 0.000000001]
5.2959	Autonomous	-1.6131	-2.5351	[0.1, 0.1, 0.000000001]	[0.1, 0.1, 0.000000001]
4.81965	Autonomous	-0.83035	-1.19935	[1.01, 1.01, 0]	[1.01, 1.01, 0]
4.7371	Autonomous	-0.8839	-1.3769	[1.001, 1.001, 0]	[1.001, 1.001, 0]

Table 5.1: Distance tests for the encoder and IMU filter (Filter 1) based on R and starting covariance

5.1.4 Conclusion

Based on the testing of filter 1 (encoders and IMU) and observations made about filter 2 (GPS and IMU), conclusions were made about their performance, and future work has been suggested.

Filter 1-Encoders and IMU

The best results obtained while manually driving the robot were filtered values within 0.086 meters of the measured distance, approximately 0.3 meters better than the raw data. When programming the robot to autonomously drive a set distance, the best results were filtered values within 0.83 meters, approximately 0.35 meters better than the raw data. Referring to the table can give general guidelines for what range of values to stay within when altering the noise and covariance matrices.

Filter 2-GPS and IMU

We have hypothesized that filter 2 was not producing results because of the way LabVIEW interprets data types. When determining the translational displacement, a square root calculation is performed. Even

though the values inside of the square root are guaranteed to be positive (due to the squaring of the change in distance), LabVIEW converts the result to complex number format.

Future Work

Further work could be done to improve the overall performance of the filter. Now that a range of acceptable noise values has been established, testing within those ranges could produce more accurate results. If a solution to the GPS/IMU filter problem can be found, testing of good noise values can be determined in the same way as those of the encoder/IMU filter. Once both filters are producing acceptable results, equation 5.23 can be applied, and observations can be made as to whether or not the weighted overall state is more accurate than either individual filter.

Another area of research to pursue involves a different way of combining the data from the three sensors. This involves using only one filter, eliminating the need to have a sensor fusion algorithm. This type of implementation would require additional research.

5.2 World Representation

To effectively compete in the Intelligent Ground Vehicle Competition, Prometheus will need to relate its position to the positions of nearby obstacles. This is necessary for both detecting obstacles and navigating the robot around these obstacles. A common method of doing this is to create a map that estimates the actual state of obstacles in the world. In this chapter, the different techniques for creating probability maps as well as the final implementation used on Prometheus will be discussed.

5.2.1 Background

One approach to creating a probability map is position the map at the origin of the robot's coordinate frame. As the robot moves in the world, it updates the expected position of objects it previously observed in the world. Then, the robot reinforces the probability of objects in its map when it observes these objects again in its new position.

Another approach is to create a map that is positioned in its own frame that is relative to the world reference frame. Since the map will grow large, and the robot's sensors will always have some error, it will also be necessary to estimate the location of the robot within the map. This problem is known as the simultaneous localization and mapping (SLAM) problem. The SLAM problem has been the focus of much intensive research in the past twenty years, and it is often considered a solved problem in static environments (Frese, 2010).

Most SLAM implementations use Kalman filters to perform localization by observing landmarks. Unfortunately, the complexity of using a Kalman filter grows polynomially with the number of landmarks (Montemerlo et al., 2002). Because of this, many alternative methods of performing localization for SLAM problem have been subject to thorough investigation in recent years. One alternative is the particle filter, of which a Rao-Blackwellized particle filter is available for ROS. This implementation of SLAM, known as GMapping, is very easy to interface with, and provided us with a good reference point for the capabilities of robotic

mapping.

A related problem to world representation is determining the coordinate conversions between the world representation and the actual world. Whether the map is chosen to be fixed to the robot's reference frame or the map is placed at a fixed point in the world, the location of the robot must be constantly updated. The most commonly known method to refer to a specific location in the world is to use latitudes and longitudes, which is an angular measurement system that is relative to a constant location (usually the Royal Observatory in Greenwich, England). However, there are alternative methods of reference that lend themselves better to our purposes. One significant problem with the geographic latitude and longitude system is that it employs a single projection for the whole Earth, with poses a problem of accuracy. It is difficult to come up with a single simple and accurate projection because the shape of the Earth is shaped irregularly and thus if a projection is made more accurate in one portion of the Earth it is likely to become more inaccurate in other portions. Another problem with geographic latitude and longitude is that calculating the distance between two points can not be done through simple subtraction and instead must be performed trigonometrically. An alternative method of reference is the Universal Transverse Mercator (UTM) system that divides the world into 60 different zones, each with its own projection, and in which two point references may be subtracted to find their relative distance.

Finally, there are many different geodesic datums [sic] that describe a set of points that describe a surface that can be used as a reference to describe other unknown points. To account for shifting tectonic plates and other such geographical changes, these datums are sometimes updated, and different datums are used to describe different regions of the world. Common datums include the North American Datum (NAD) 27, NAD83, the World Geodetic System (WGS) 83 and the International Terrestrial Reference Frame (ITRF) 2000. In applications, the datums from different sources of data must match or must be converted to match to ensure that the results are correct and accurate.

5.2.2 Methodology

Table 5.2 shows a comparison between the two approaches that were researched for estimating the world state in a probability map. Because neither option appears to be clearly favored over the other, it was decided that each method should be further investigated. While there are many open-source implementations of SLAM available (see Stachniss et al. (2010)), it was decided to first experiment with GMapping because it already has wrappers provided by ROS. GMapping is an open source SLAM implementation using Rao-Blackwellized particle filters. By using the LIDAR and only the encoders for odometry information, it was possible to create a map of the floor plan of a building (see Figure 5.5). The robot encountered the most error during tight turns, and as can be seen in the figure, GMapping was unable to close the cycle and identify that the robot had returned to its starting location.

While GMapping performed quite well indoors given only encoder data, it was still necessary to test the algorithm outdoors where there are less landmarks. After running tests with Prometheus running GMapping in an outdoor environment (see Figure 5.6(b)), it appeared that either 1) the data was insufficient to build a complete map, or 2) the implementation of SLAM was not able to make use of this data properly.

After some deliberation, it was decided that SLAM was unnecessary for the competition. In most of the events, the robot will rarely need to return to locations it had explored earlier, and even if this were the case, the slight speed increase that may be gained does not seem to adequately compensate for the extra

	<i>Advantages</i>	<i>Disadvantages</i>
<i>Map fixed to robot's local frame (Local Map)</i>	<ul style="list-style-type: none"> • Because the map is smaller and landmarks are forgotten, calculations can be performed faster 	<ul style="list-style-type: none"> • Robot can only make short term path-planning decisions using the data
<i>Map relative to world (SLAM)</i>	<ul style="list-style-type: none"> • If performed properly, the robot will be able to navigate regions it has already visited at an increased speed. 	<ul style="list-style-type: none"> • Calculations take longer given that all landmarks are stored • More difficult to implement than the alternatives

Table 5.2: Comparison of the advantages and disadvantages of using a small, local map positioned relative to the robot versus a large scale map positioned relative to the world frame.

processing required to store a large map.

Prometheus creates a local map of its environment that is fixed to the origin of its local frame. Laser measurements and detected lines are stored in the program as a list of continuous points. The points are later used to build a grid map that is broadcast using a ROS `nav_msgs/OccupancyGrid` message. The reason for storing measurements in a continuous format rather than immediately converting them to discrete grid cells is that it preserves more accuracy from each measurement. In addition, the map moves relative to the robot, so there needs to be a mechanism for determining when a measurement has moved from one grid cell to another. This is easiest to do when using continuous points because we can transform the measurement from its old location to its new location and then convert the point to a grid cell.

The width, height, and resolution of the map are all configurable when starting the program. In addition, parameters are available for configuring when points expire. These include the maximum age of a measurement and the maximum distance a measurement can be from the robot. In addition, a parameter is available to restrict the total number of measurements so that the list does not grow too large.

Program design

The ROS package for building the local map this is named `local_map`, and it was implemented from scratch as part of this project. The main entry point of the map creation process is the `LocalMapRosWrapper` class. This class is mainly responsible for receiving measurements and publishing the map as a ROS message. All measurements are originally generated by other ROS packages and published as a ROS messages. The `LocalMapRosWrapper` uses different callbacks for each type of measurement, so when it receives a measurement, it knows what type it is and how it should be stored. Each type of measurement is a subclass of the `Measurement` class, which records the time and pose of the robot when a measurement was taken as well as the measurement's priority. `Measurements` are eventually used to build an `OccupancyGrid` object, which stores the two-dimensional map as a list of integers. The integers represent a cell's occupancy, which can range inclusively from 0 (unoccupied) to 100 (occupied). In addition, a value of -1 means the occupancy of the cell is unknown. This format for storing occupancy was chosen because it is compatible with ROS's `nav_msgs/OccupancyGrid` message, which is necessary to use for visualizing the map in rviz. For conve-

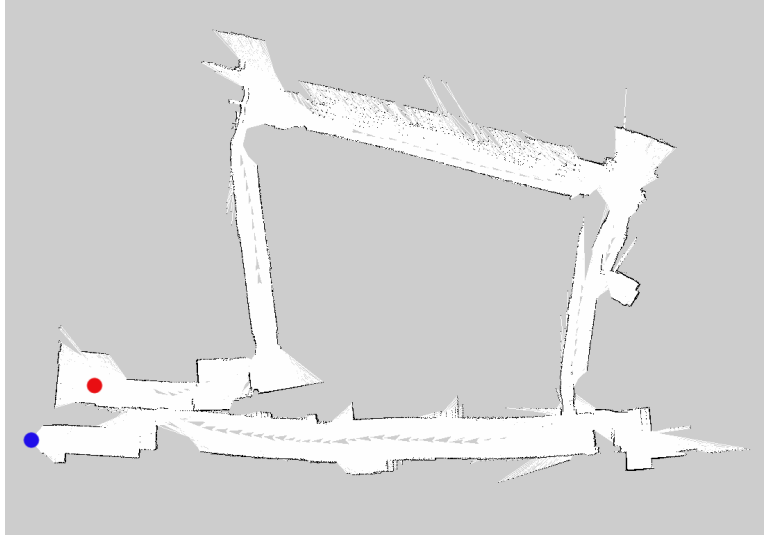
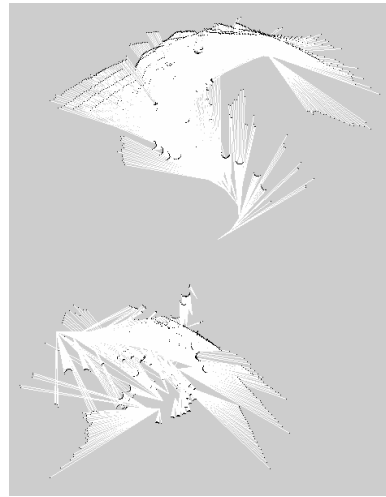


Figure 5.5: Map created using GMapping on Prometheus using LIDAR for range data and encoders for odometry information. This test was performed using teleoperation to navigate Prometheus on the first floor of Atwater Kent Laboratories at WPI.



(a) Actual configuration of obstacle course.



(b) Map created of outdoor obstacle course using GMapping with LIDAR and encoders.

Figure 5.6: Comparison of a picture of the environment and an estimation of the environment state using GMapping with LIDAR and encoders in an outdoor environment. This test was performed using the same techniques as in Figure 5.5 in a different environment.

nience, `GridCell` objects are used to access cells of the `OccupancyGrid`. These objects simply store the row and column as integers, but they are used extensively for referring to specific locations of the map. The `OccupancyGrid` also contains a `MapDescription` object, which stores the resolution of the map in meters per cell as well as It is important to note that the width and height of the map should always be odd numbers so that the robot's local frame is always positioned at the center of a grid cell.

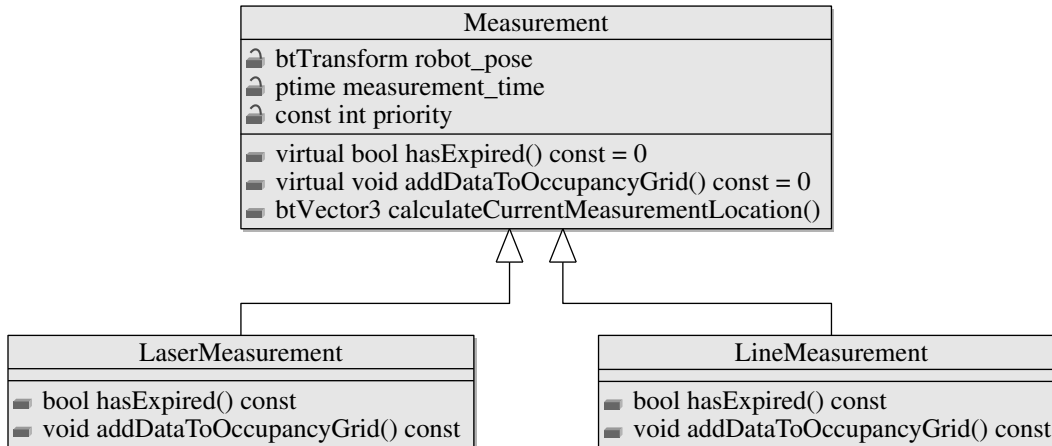


Figure 5.7: Class diagram of different `Measurement` objects.

The `TargetCollection` class is used to store the measurements for drawing on the `OccupancyGrid`. This class disposes of measurements when they have expired or there are too many, and it iterates through measurements in a specific order to create the map. `Measurements` with a higher priority are always drawn on the map last. In the current implementation, only two priorities are used, and laser measurements are always given a lower priority than line measurements. This is laser measurements draw a line of sight from the robot's position to the detected object. If laser measurements were to be drawn after line measurements, they would sometimes overwrite the line measurements, making the lines disappear on the local map. Details of how a measurement should be painted on the `OccupancyGrid` as well as when they have expired are delegated to subclasses. In addition, the `OccupancyGrid` is always built using a `MapBuilder` object. The `MapBuilder` contains functionality for incrementing and decrementing the probability of an object occupying a given `GridCell`.

The `LineMeasurement` is simply stored at two points, and is drawn on the map by using Bresenham's line algorithm to draw a line of occupancy between them. A `LineMeasurement` expires when it is older than the maximum age for a `Measurement` or when both of its points have gone out of range of the map. The `LaserMeasurement` raises the probability of the cell of the detected object and also draws a line of sight from the robot's position to the point. A margin is added between the last points of the line of sight and the obstacle so that future measurements of the obstacle, which are likely to be slightly different, do not erase the obstacle with their line of sights. When the `LocalMapRosWrapper` handles a measurement message, it stores it in the `TargetCollection` and then immediately constructs and publishes an updated `LocalMap`.

Local to World Frame Conversions

As mentioned in the Background section, the UTM system is more suitable for our needs thus it was required that our input from the Trimble AG252 GPS receiver be converted into UTM Grid Coordinates.

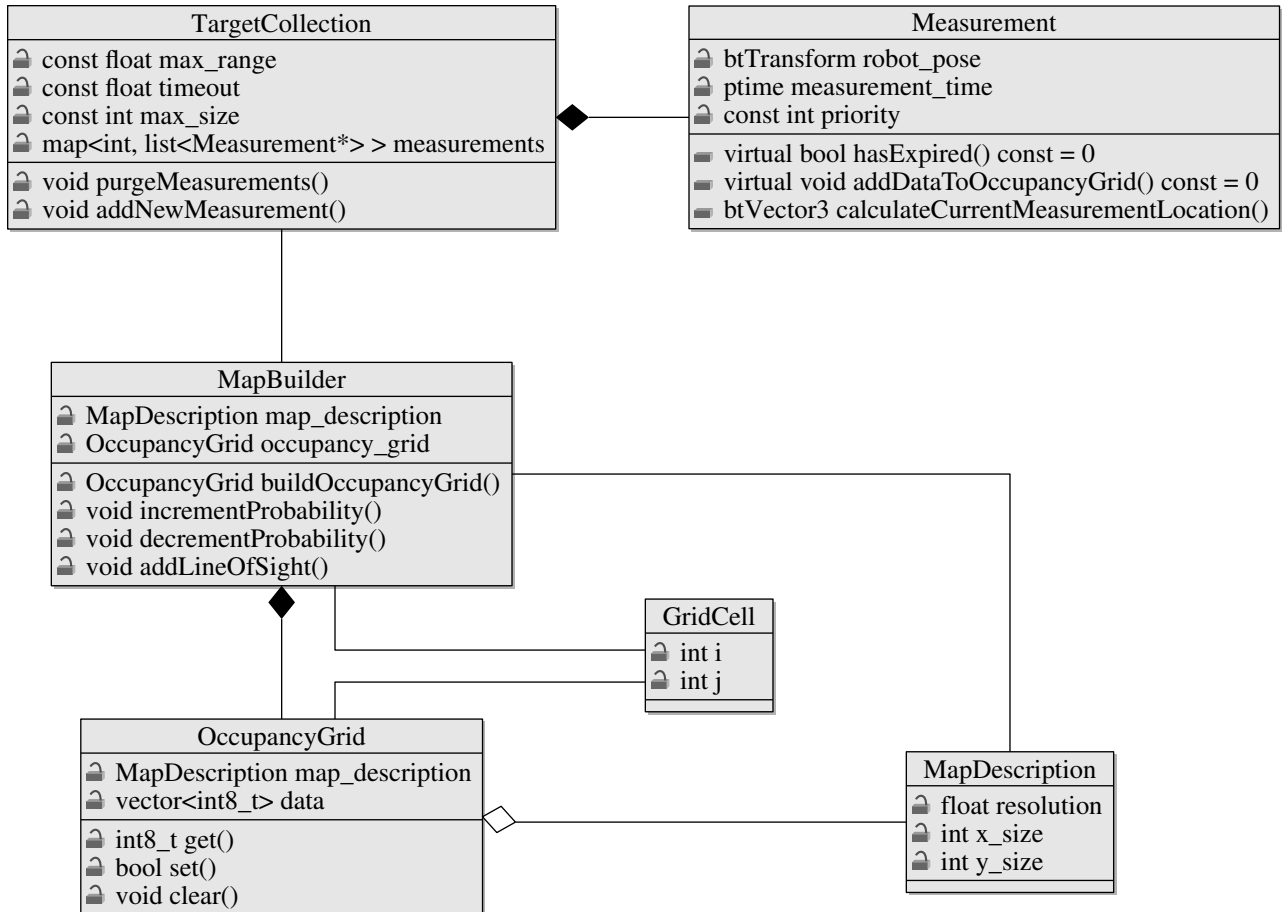


Figure 5.8: Class diagram of the core classes of the `local_map` package.

To perform this operation, a long series of transformations must be performed, as described in (Dutch, 2003). However, the transformations themselves are complicated and even minor rounding errors are likely to cause a large discrepancy in the results. For this reason, it was decided that the open source Geospatial Data Abstraction Library (Open Source Geospatial Foundation, 2011) would be used to perform the transformations.

As mentioned in (Trimble, 2004), the OmniStar HP corrections service for the AG252 publishes data using the ITRF2000 datums. GDAL does not currently support ITRF2000. However, as mentioned in (Institut Geographique National, 2007), newer realizations of the de-facto WGS84 datums used for UTM are coincident with the ITRF at the 10 cm level. This presents a negligible source of error, even more so since our local map measurements are relative measurements instead of absolute ones. In other words, as long as all the points of reference are shifted uniformly, there will be no error in the local frame.

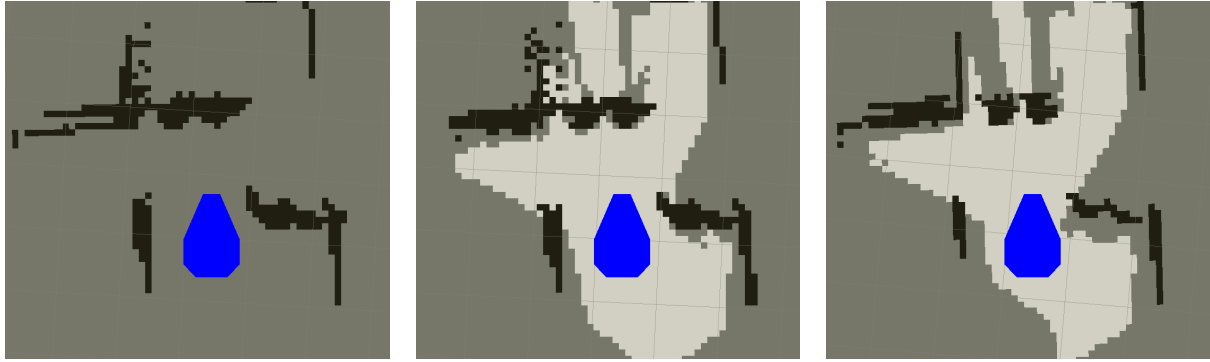
One quirk with using UTM coordinates is that since each zone uses different projections, two points very close to each other but on different sides of a grid boundary will have a "jump" between coordinate values. For this reason, it was decided that since the operational range of the robot is relatively small compared to the grid sizes, the grid zone would be determined during system initialization and the same grid zone would be used throughout.

A ROS node was implemented in Python that uses GDAL to both stream current local map coordinate data and also provide a service that does conversions from arbitrary latitude longitude pairs to local map coordinates. The GPSd library was used to handle the low-level parsing of the raw GPS data for its robust autoconfiguration abilities and ease of use (GPSd, 2011).

A final caveat concerning the current GPS is that as explained in (Trimble, 2004), the OmniStar HP corrections service converges to the advertised 10 cm level accuracy only after a period of 30 minutes. This is usually because the almanac data that contains information on satellite statuses and orbits are transmitted approximately every 13 minutes (Mehaffey, 1999). If the beginning of the signal is missed, the device must wait for the next signal and listen to it until it ends, which adds up to around 30 minutes total. The autoseed feature of the AG252 mitigates this issue partially by storing the old ephemeris and almanac data on powerdown so that at least a partial fix may be acquired immediately (or a full one if the last run of the device was very recent) and all the information does not need to be re-acquired from scratch. As a result, the GPS receiver converges faster if used recently, and thus it is a good idea to let the device converge before competition times.

5.2.3 Results

As expected, accurate odometry data is necessary to build a coherent map using the technique outlined above. In testing, it was found that poor odometry data would cause the map to smear as the same static obstacle would continuously be measured at a different location as the robot moved. To mitigate the smearing effect, the laser measurements are expired and are deleted at different times depending on their location. Laser measurements that are in front of the LIDAR sensor expire very quickly because they are likely to be observed during the next scan. However, once a laser measurement moves behind the LIDAR sensor, it takes much longer to expire. This is done because laser measurements behind the robot will likely not be observed again, but are still necessary to help the robot avoid brushing the sides of obstacles.



(a) Local map without line of sight drawn to the obstacles. (b) Local map with line of sight drawn to the obstacles. (c) Local map using a 2.5 second timeout for measurements that are in front of the LIDAR.

Figure 5.9: The progression of the techniques used for creating the local map from LIDAR data. All techniques used a maximum age of 10 seconds for the LIDAR measurements.

The local map program also performed considerably well on maps of varying resolutions. The main factor that would cause the program to slow down was the number of measurements stored in memory. It was determined that the program started to slow down significantly after about 10,000 measurements. For this reason, the number of measurements is currently limited to 10,000. If the `TargetCollection` is full, the oldest measurements are deleted to make room for new measurements.

5.2.4 Conclusion

The local map that was implemented and used for the robot performed satisfactorily during the outdoor tests. However, it still suffers from issues with smearing. When performing path planning, the robot would occasionally freeze in free, open areas before continuing on its path. This was not due to the path planner taking a long time to compute, but instead due to false positives on the map that were the result of smearing. The robot would remain stationary until the false measurements timed out, and the robot could again find a path that was drivable. In theory, the map should perform perfectly with ideal odometry data. However, since the odometry error will always have some error, this cannot be relied on. One approach to create a more accurate map would be to add filtering to the measurements.

5.3 Path Planning

Prometheus will need to perform path planning in two stages. First, before the competition, the robot needs to determine the order in which it will visit each waypoint. Next, after Prometheus has observed and created a map of its environment, it will need to make use of a control policy that provides a mapping from the robot's state observation to motions (Thrun et al., 2005). The control policy for Prometheus' itself will occur in two steps: path planning and motion planning. In path planning, robot uses the local map to plan a path to its next waypoint that avoids obstacles. In motion planning, the robot calculates the linear and angular velocities required to make the robot follow this path.

5.3.1 Background

One method of planning the best path on a grid is by using the A* search algorithm. A* is a variant of Dijkstra's algorithm, which is used to find the best path between two vertices in a graph. The A* algorithm improves the performance of Dijkstra's algorithm by using a heuristic to make a best guess at the distance from any vertex to the goal. If the heuristic always guesses a cost that is less than or equal to the actual cost, it is said to be admissible, and A* will always find a best path to the goal. When using a heuristic that is not admissible, the running time of A*, but it is no longer guaranteed to find the best path (Russell & Norvig, 2010).

One disadvantage of A* is it has a somewhat large running time. In addition, the paths it generates are often jagged and difficult to follow. Figure 5.10 shows one of the common issues of A*, which is that it typically plans paths that move along edges that are aligned in the graph. These are best paths in the graph, but when translated to the actual world, a best path would be a diagonal from the starting position to the goal.



Figure 5.10: Paths generated with A* usually do not follow a straight line from the source to the destination even when all cells are marked as unoccupied.

Another method of planning paths that avoids obstacles is by generating a set of arced paths originating from the robot's turning center. This method generally used for large, outdoor vehicles and has been effectively used by a number of competitors in the DARPA grand challenge (von Hundelshausen et al., 2008). One advantage of this technique is that the paths are extremely easy to follow. Each arced path can be summarized by two motion commands: a linear velocity and an angular velocity. Odometry information can be monitored, and a second motion command, telling the robot to stop, can be issued as soon as the robot nears the end of its path.

Once a set of arced paths is generated, each path must be evaluated for drivability. There are several methods of doing this, but in general most methods should take into account 1) the obstacles that the robot will encounter when driving on the given path, and 2) how much following the path will help the robot reach its destination.

In both path planning techniques outlined above, the robot's width will need to be accounted for. One method of accounting for the robot's width is by performing inflation, which creates a copy of the local map in which the size of each obstacle is expanded to consume additional space. The idea behind this is that the

robot is approximated as a circle about its turning center, the diameter of which is the distance from the turning center to the furthest point on the robot's base. Using this the robot will then be able to follow any path that is planned through the inflated map assuming that its base link always follows the path perfectly.

Another method of considering the robot's width is by calculating the cells that the robot will overlap when driving on a given path. This technique is difficult to implement for certain paths, but for simple arced paths, only the inner and outer radius of the path must be determined. The inside cells of the path can then be filled using a flood-fill or similar algorithm.

5.3.2 Methodology

The first algorithm implemented for path planning was tentacles because of its known effectiveness in avoiding obstacles. In addition, the algorithm creates paths that are extremely easy to generate velocity commands for. The first step for implementing tentacle following was to generate the arced paths. It was decided to represent the arc of the tentacle in an `ArcedPath` object that stores the radius, length, and a list of points located relative to the robot that make up the path. The `ArcedPath`'s are also grouped together into `TentacleSpeedSet` objects. The equations from von Hundelshausen et al. (2008) were used for determining the radius and length of each tentacle for each speed set. The parameters to the equation were modified to generate speed sets with very short lengths and a variety of turning radii. The arced paths for one of the speed sets is shown in 5.11.

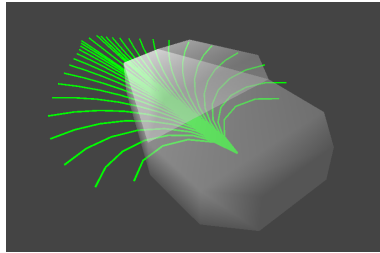


Figure 5.11: The arced paths that are part of one of Prometheus' speed sets

After generating the arced paths for a speed set, each tentacle needed to be evaluated for its drivability. Because the local map is represented as an occupancy grid, the evaluation method would need to convert tentacles to grid cells at some stage. Converting the continuous points of the tentacle to grid cells is a relatively simple problem, but accounting for the robot's width when evaluating arced paths would be more difficult. One option for doing this would be by performing inflation of the local map and then looking up the corresponding cell for each point on the arced path. However, this method would prove to be problematic for Prometheus because its turning center is approximately one meter from the further point on its base. Thus, the robot would approximate itself as two meters in diameter, and the robot would be unable to navigate between obstacles that are separated less than two meters from each other. Because of these disadvantages, each tentacle was instead inflated to account for the width of the robot. This extra padding around each tentacle is known as the classification area (von Hundelshausen et al., 2008).

To inflate each arced path, a list of points for the inner and outer radius is generated. These radii are simply the radii that are generated by the left and right edges of the robot as it follows the arced path. Because of Prometheus' semi-triangular shape, we were able to assume that the radius of the inner and outer

edges of the classification area were equal to $radius_{tentacle} \pm \frac{width_{robot}}{2}$. For added precautions, a margin was added to the inner and outer radii. This added margin made the tentacle's width more similar to the support area described in 5.11 with the exception that cells that fell outside of the classification area were not weighted less than those inside it. After the points for the inner and outer radii are generated, each of the points is converted to a grid cell, and then Bresenham's line algorithm is used to connect each of the cells to create an outline that traces the inner radius and the outer radius and connects both of their end points together. Following this, a point from the center of the arced path is selected, and a flood-fill algorithm is run to fill in each of the grid cells of the boundary. An image of the grid cells for an arced path is shown in Figure 5.12.

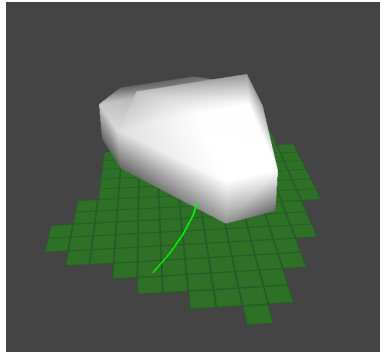


Figure 5.12: The inflated grid cells of an arced path. These cells account for the width of the robot, and can be overlaid on the local map to evaluate the drivability of a path

The actual algorithm for calculating the points on the inner and outer radii is show in Figure 5.13. One nice aspect of calculating the classification area in this manner is that the technique can be applied to other types of arced path as well. For example, if a tentacle algorithm were used where the arcs were replaced by Euler spirals, then the same algorithm should be able to also determine the classification area of the paths created by Euler spirals.

Algorithm *GenerateClassificationArea*

Input: A list, L_{path} , of vectors on the path, and the width of the robot, W

Output: A list, $L_{classification-pairs}$, of pairs of vectors that are the borders for the classification area of path

1. $L_{classification-pairs} \leftarrow \emptyset$
2. **for** v_k in L_{path}
 (* For each v_k , calculate v , the vector from the last point on the path to the v_k *)
3. **do** $v \leftarrow p_k - p_{k-1}$
4. $v_{displacement} \leftarrow v$ rotated $\frac{\pi}{2}$ radians about the z-axis
5. $v_{displacement} \leftarrow v_{displacement}$ normalized
6. $v_{displacement} \leftarrow v_{displacement} * \frac{W}{2}$
7. insert $(v_k + v_{displacement}, v_k - v_{displacement})$ into $L_{classification-pairs}$
8. **return** $L_{classification-pairs}$

Figure 5.13: Algorithm for determining the edges of the classification area of a path.

Once the grid cells have been generated, they can be overlaid on the map to evaluate each arced path. To compare different paths, a rating scheme was used where 0 meant a path was undrivable, and 1 meant a path was drivable and highly favorable. The rated score of any path was to fall inclusively between these

two numbers. For simplicity, a polarized evaluation function was created that only returned either a 0 or 1. After some experimenting, we also implemented a linear evaluation function that determined a paths scored based on the percent of cells marked as occupied on the map.

Both choosing a tentacle that does not collide with an obstacle and choose one that will eventually lead to the goal are equally important. However, mixing these two operations into one evaluation function could be dangerous, because if tuned incorrectly, a path hits obstacles to reach the goal may be rated higher than a path that takes a long route around the obstacles. For this reason, it was decided to separate the tentacle evaluation functions, which determined the scores of tentacles based on the local map, from the tentacle choosing function, which choose which of the rated arced path to drive on. As part of our experimentation, we implemented path choosing functions that simply avoided obstacles in addition to functions that took the goal into account. The most complicated of these is the A* tentacle choosing function, which ultimately is the central workings of the A*-tentacle hybrid currently used on Prometheus.

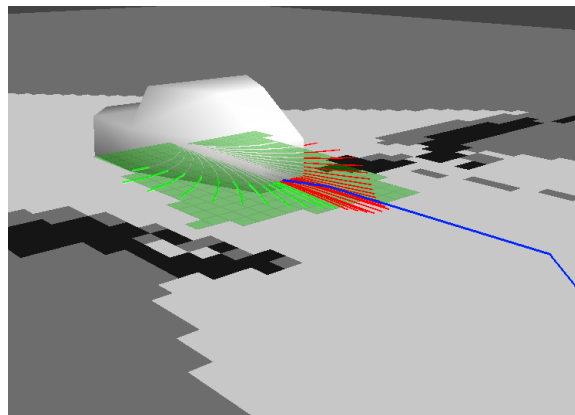


Figure 5.14: Tentacle planning and A* path planning being used together to drive to a goal. The drivable arced paths are shown in green and the undrivable ones are in red. The best path as planned by A* is drawn in blue. Prometheus uses A* to choose which drivable arced path to follow. The dark green grid cells show the cells that the robot will drive through as it follows its chosen arced path.

The A* chooser function works by first using the A* algorithm to calculate the best path from the robot to the goal. To use A* on the occupancy grid, each cell is treated as a vertex in a graph. Cells that are considered to be occupied by an obstacle are not included in this graph. Diagonal movements between cells are allowed, so each vertex has up to eight edges connecting it to its neighboring vertices. Because A* is normally able to calculate a best path through a small opening between obstacles that the robot cannot fit through, the distance value assigned to each edge is proportional to the number of nearby obstacles. Specifically, the distance value for an edge is equal to the sum of the occupancies of the cells within a specified radius. Another possible issue with A* is its slow running time. To mitigate this problem, a timeout was added to the A* calculation. If the algorithm is taking longer than a specified time, then the resolution of the grid is reduced, and A* is recalculated. This process is repeated until A* finds the best path to the goal. Once a path to the goal is found, the A* tentacle chooser function then looks at each arced path that is rated above a certain threshold. For each of these arced paths, it calculates the geometric mean of the distance of each of the points in the arced path to its closest point in the A* path. After doing this for each path, the path with the lowest geometric mean is selected to be driven on. An image of Prometheus driving with the A* tentacle chooser is seen in Figure 5.14.

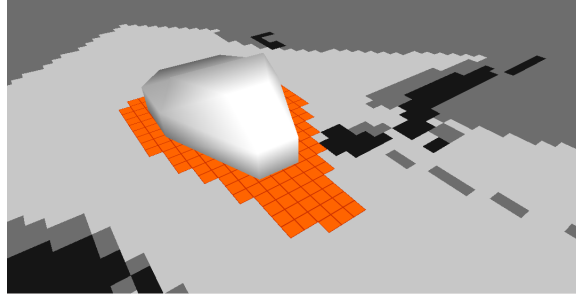


Figure 5.15: The inflated footprint being used to prevent Prometheus from continuing to follow a path that will cause it to strike an obstacle.

During testing, it was found that the Prometheus would sometimes bump into obstacles. To prevent Prometheus from following a bad path, a method for late obstacle avoidance was added to the path planning algorithms. A footprint of the robot's shape is generated using the same techniques for drawing lines and filling cells on a grid used in other algorithms. This footprint, seen in figure 5.15, can be inflated a specified amount to create a margin around the robot. Once the footprint is generated, its cells can be iterated over to search for obstacles, and the robot can stop moving and replan if it find that an obstacle has moved too close to it.

5.3.3 Results

The polarized evaluation function proved to be quite effective in simulations where only two values, occupied or unoccupied, were assigned to cells of the map. However, it performed poorly in actual tests of the robot because in these scenarios there was a gradient of values representing occupancies. In addition, unoccupied squares would sometimes be marked as unoccupied, and if a single such cell were to get in the way of the robot, it would stop moving or try to avoid it. This led to the creation of the linear evaluation function, which was more lenient about letting the robot pass through obstacles. This function was found to be quite effective both in outdoor and simulated tests of the robot.

One of the most effective tentacle chooser functions for simply avoiding obstacles was to follow the tentacle with the largest radius (that is, the one that is most similar to a straight line). This is one of the functions that performed most effectively in guiding the robot through a curved hallway. Another tentacle choosing function that performed extremely well was one that always chose the center-most tentacle from the largest grouping of drivable tentacles. Figure 5.19 shows one part of a simulated obstacle course that Prometheus navigated on using this function.

The A* tentacle chooser was found to work very well in the simulated environments. As expected, it did not perform as smoothly in outdoor environments. This is in part due to the non-ideal localization and imperfect occupancy grid on which paths are planned. In addition, because A* typically plans paths that move in straight lines along the grid (see Figure 5.10), the robot would oscillate as it followed the path to the goal. Regardless, the algorithm enabled the robot to successfully navigate around obstacles in a demonstration outdoor.

One of the more difficult aspects of the path planning configuring the speed at which the robot should drive. In the first tests, the robot was configured to always drive at the same speed, but this was found to

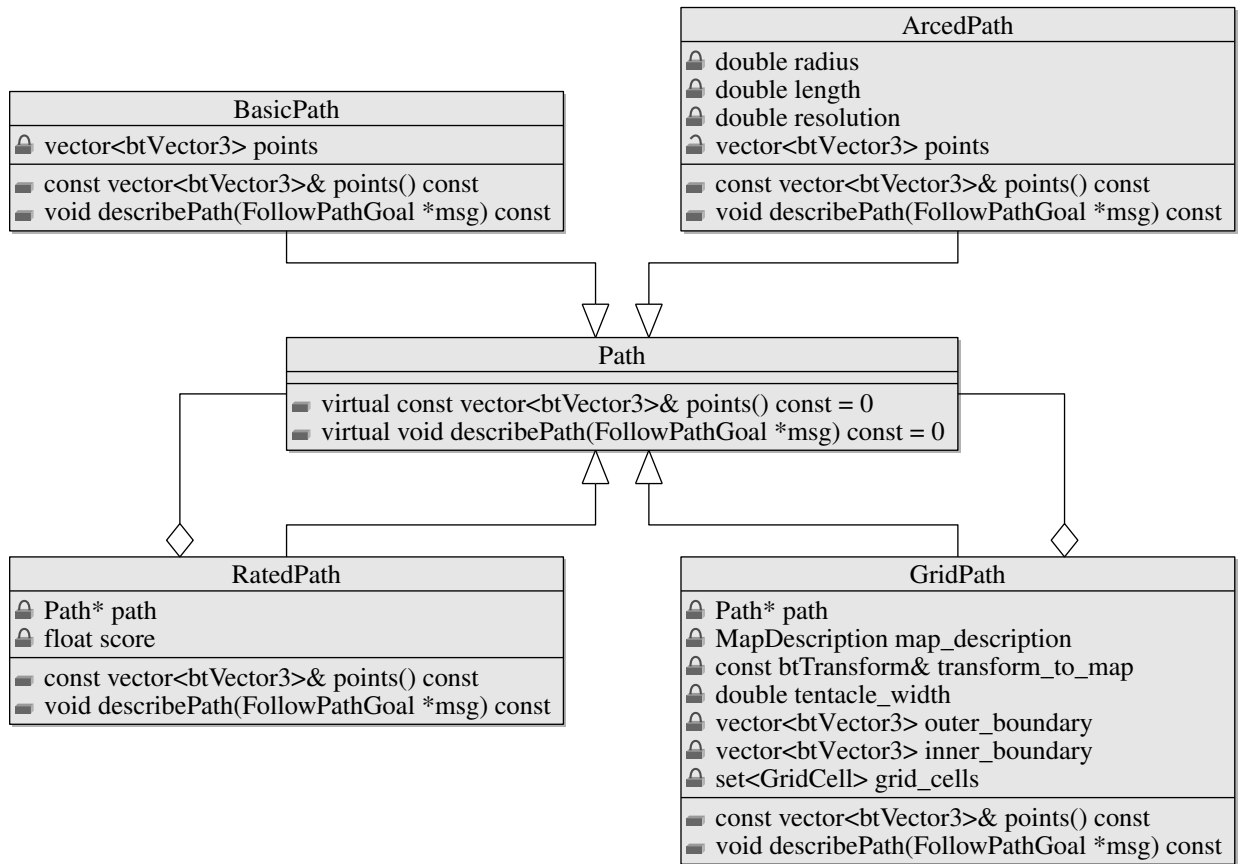


Figure 5.16: Class diagram of different `Path` objects with getters omitted for brevity. The `BasicPath` and `ArcedPath` objects are used for basic instantiations of the `Path` class. `BasicPath` and `ArcedPath` are used for the A* planner and tentacle planner respectively. The decorator pattern is used for both the `RatedPath` and `GridPath` objects. These objects append additional information to the `BasicPath` and `ArcedPath` objects

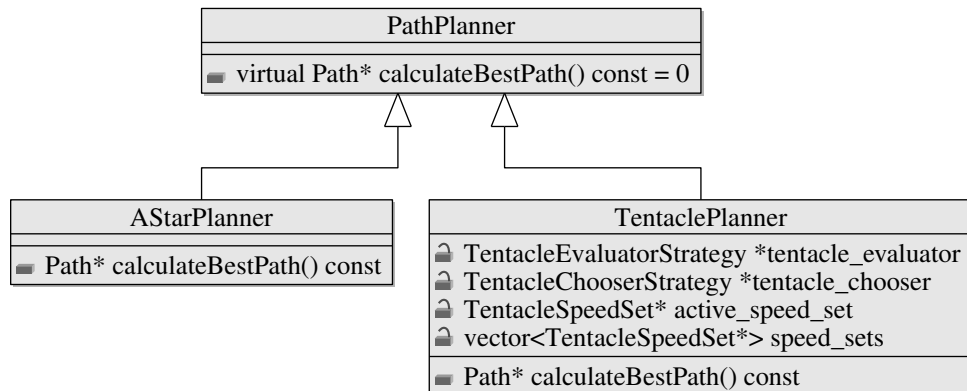


Figure 5.17: Each type of planning algorithm extends the `PathPlanner` objects. Only one type of planner can be used at a time. To combine A* with tentacles, the `TentacleChooserStrategy` object holds a reference to a `AStarPlanner` object.

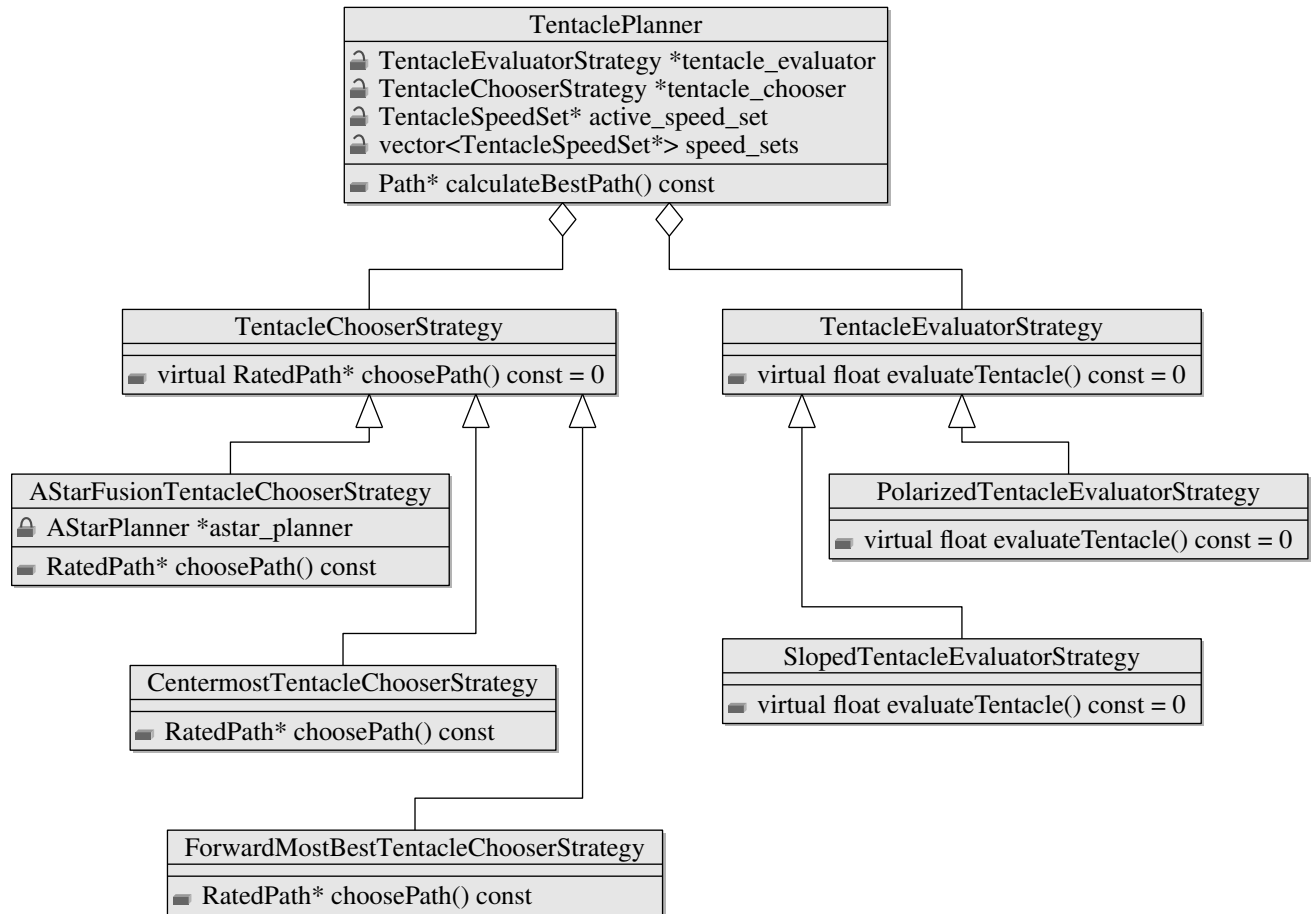


Figure 5.18: UML for some of the objects used by the `TentaclePlanner` class. Two of the objects `TentaclePlanner` is composed of make use of the strategy pattern. The reason tentacle evaluation and tentacle choosing is handled by two separate objects is that it makes it easier to implement safe choosing functions that are guaranteed to avoid obstacles.

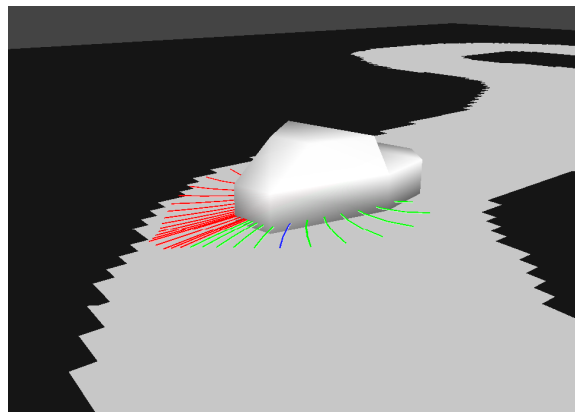


Figure 5.19: Prometheus using tentacles to traverse a curved hallway in simulation. Undrivable paths are shown in red, and drivable paths are shown in green and blue. Blue indicates the path the robot has chosen to drive on when using a tentacle choosing function that always chooses the center-most tentacle from the largest continuous grouping of drivable tentacles.

cause the robot to drive unsafely. After some discussion, the robot was configured to drive at a different speed depending on the radius of the tentacle. The robot was configured to drive slower on tentacles with smaller radii because odometry error accumulates faster when turning, which ultimately causes the map to smear more and become less accurate. In addition, the robot is less aware of obstacles in the region to which it is turning. For these reasons, the robot was configured to drive at varying speeds and was able to successfully navigate around obstacles outdoors.

5.3.4 Conclusion

The final technique that Prometheus uses for planning a path is two-fold. First, the robot generates a series of arced paths originating from its turning center. Each path is evaluated, and paths that strike obstacles are discarded. Next, a path is planned from the robot to the goal using the A* algorithm. Once A* is finished, the tentacle that is judged to be closest to A* is followed. The path planner gives the arced path to a motion planner, which sends the appropriate motor controls and stops the robot if it reaches the end of the path or veers from the path. Finally, as the robot drives along its path, an inflated footprint of the nearby cells of the robot are checked for incoming obstacles. If an obstacle enters the robot's footprint, or if the robot reaches the end of the arced path it is driving on, it recalculates the best path. This process repeats itself as quickly as possible with the only limitation being the running time of each algorithm. The reason for doing so is that the map is constantly changing, so it is important to constantly update path plans as well.

5.4 Vision

One requirement that Prometheus must fulfill is to not cross the white solid and dashed lines that delimit the part of the competition grounds that is available to the robots. Since the only easily distinguishable feature of these lines is their visual appearance, image processing must be used to detect the lines. The images are acquired through the on-board camera as described in chapter 3. Once an image is acquired from the camera, it is subjected to several steps of pre-processing to reduce noise and to make the lines in the image more prominent. Next, a line detection algorithm is run on the pre-processed image to detect the lines and their line equations will be recorded. Finally, these line equations are used to augment the local map that is used to guide the robot while avoiding obstacles.

5.4.1 Background

Pre-Processing

In an ideal world, the areas that correspond to lines in the camera images would be clearly delineated, contiguous, solid blocks of white or yellow. In the real world, the detection target, white painted lines on grass, can be challenging to detect accurately on the raw image because of the relative lack of contrast between the lines and the grass. Furthermore, the problem is exacerbated by the fact that the surface of the grass is not flat, which means that the line areas are not uniformly white and contains shadows from the grass. Also, often the paint is wearing off or is faint, which presents an additional problem because the line

areas will contain green spots. This section will term these deviations from the ideal expectation “noise”.

There are two opposing approaches to solve these noise problems. One approach is to try to detect the noisy areas from within the image and try to perform a recovery operation on them (such as trying to predict the color of the missing pixel). This approach was not selected due to its complexity. Detecting the noise is difficult since it requires a preconception of how the image should be in the first place. This is hard to form when the noise does not have a single source and does not follow a regular distribution. Furthermore, even if this operation was possible, it would be computationally intensive.

An alternative and common approach is to get rid of noise without analyzing and detecting it, at the expense of some resolution. This is usually done through operations that take into account nearby pixels and apply that information to the current pixel, thus smoothing the data. While the loss of resolution might be important for some applications, for ours an inaccuracy of up to approximately ± 15 cm is considered tolerable since this corresponds to the inaccuracies from other sensors. It is highly unlikely that the loss of resolution from the image processing will surpass that. Several different image processing operations are available that help turn the image into a form that is more usable.

Smoothing One of the most common procedures for tackling noise in image processing is smoothing. Smoothing processes subdivisions of the image and uses an averaging function to average out data in that subdivision. This way, outliers in the data set are brought closer to the prevailing mean or median. The resulting image appears blurred, and has less variance of different colored pixels within a given area. During experiments, it was found that median filtering worked best to create the desired effect of contiguous smooth areas of similar colors.

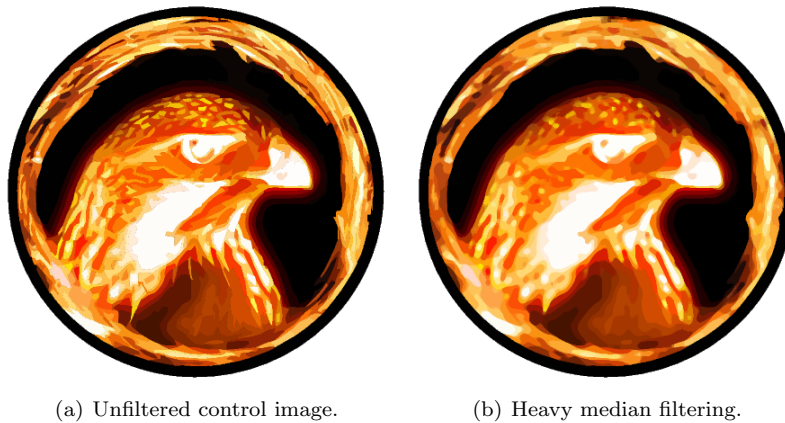


Figure 5.20: The effect of median filtering on an image. Take note of sharp edges in the control image on the left and the lack thereof on the right.

Morphological Dilation and Erosion Dilation and erosion are two morphological operations that in effect increase and decrease the sizes of contiguous foreground (bright) color regions in the image. For example, if one performs erosion on white text on a black background, the text will appear to be thinner. Dilation will have the opposite effect, thickening the lines. This operation has the use of removing unwanted small areas of foreground or background pixels, named salt and pepper noise. One problem with erosion and dilation is that while they remove salt and pepper noise, they also expand and contract the borders of the

regions in an image.

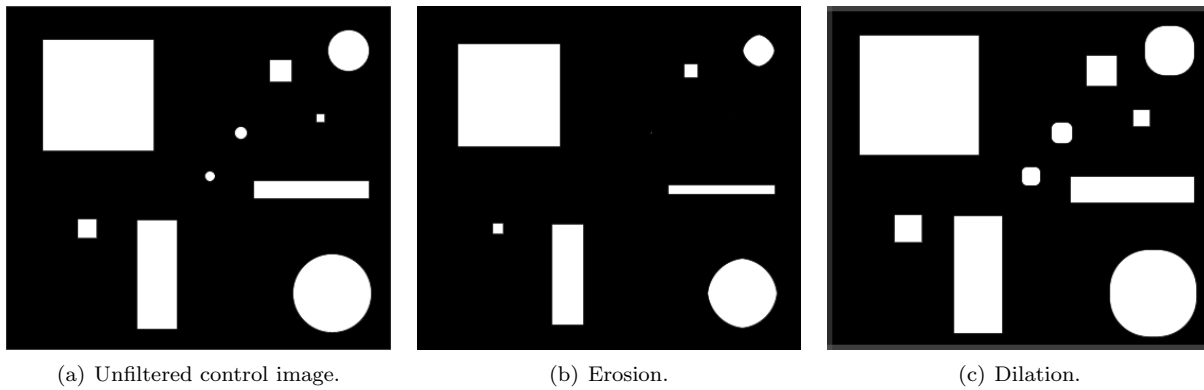


Figure 5.21: The effect of morphological erosion and dilation on an image. Notice how on the eroded image the smaller shapes are removed but the larger shapes have smaller borders. Similarly, on the dilated image, all objects are larger.

Morphological Opening and Closing When one combines dilation and erosion operations, the resulting operation is named an opening or closing depending on whether the dilation or erosion comes first. Opening and closing have the advantage of recompensating the image borders to expand or contract back to their original size, thereby providing the noise filtering functionality of dilation and erosion while still mostly preserving the image structure.

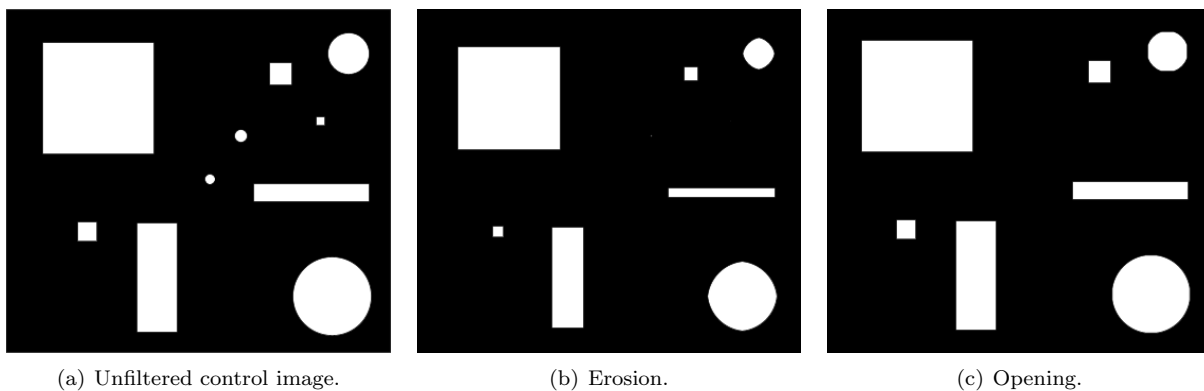


Figure 5.22: The effect of a morphological opening compared to the raw image and to a morphological erosion. While in both the erosion and opening the same objects are removed, the opening retains the original sizes of the objects while the erosion makes them smaller.

Color Space Conversion and Channel Splitting The most common color space for images is the RGB additive color model that takes Red, Green and Blue as its primary colors and defines black to be $(0, 0, 0)$ and white to be $(\text{max}, \text{max}, \text{max})$ where max is the maximum color value possible. Each channel contains all the pixels that contain that specific primary color, with each pixel containing exactly the same amount of that primary color as the original image. However, while the RGB model exposes some information on the content of each primary color in each pixel, it obscures other information. A commonly needed operation

is to extract all the pixels of a specific color. This is not equivalent to grabbing all the pixels in the green channel because a pixel in the green channel does not denote that the pixel in the original image is green, it just denotes that it has *some green in it*. There is another color space, however, that does help with such semantic distinctions. The HSV color space divides the color into three separate channels named Hue, Saturation and Value. The hue of a pixel determines its overall color in the traditional sense. The saturation of a pixel determines the color content, such that decreasing saturation brings an image closer to grayscale. The value of a pixel determines its brightness such that decreasing brightness brings an image closer to all black. For example, if one needs to extract all blue pixels regardless of the different tints and shades of blue, then one can just look at the hue channel and extract according to that.

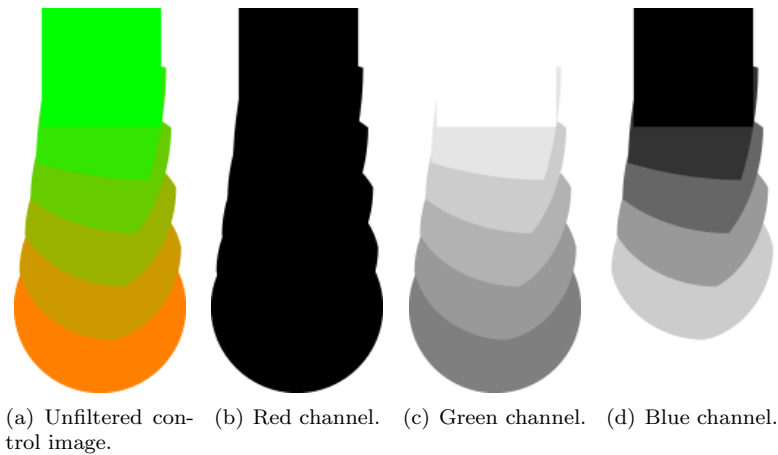


Figure 5.23: The effect of channel splitting. Adding the red, green and blue channels result in the original image.

Thresholding The thresholding operation defines a threshold of brightness or color value and then binarizes the image depending on whether each pixel has a higher or lower value. This is useful because it assists in filtering out all pixels that we know are above or below a certain brightness. For example, when the original image is channel-split to obtain the green channel, not all pixels in the channel correspond to pixels in the image that are green, because pixels of other colors might contain a green component, but may not be green overall. In such a situation, it might be desirable to filter out all the pixels that contain less green than others. A thresholding operation may be used to achieve this.

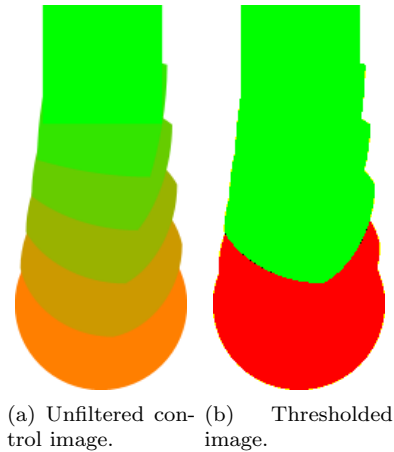


Figure 5.24: The effect of thresholding. The binarization is apparent in the right image since the only remaining 2 colors are red and green.

Canny Edge Detection Many object and shape detection algorithms operate on the edge features of objects. Edge detection has the advantage of reducing the number of pixels that most algorithms have to operate on. An edge is defined as a region in which there is a sharp change in image brightness. This change often corresponds to discontinuities in depth, surface angle and material in the real world. The output is a binary image containing the edges. In the line detection problem, the border lines of the white line to the green grass consist of “edges” in this sense. By this definition, each white line is surrounded by two edges. The Canny Edge Detector is the most prevalent edge detection algorithm, as described in (Canny, 1986).

Hough Transform

While edge detection detects edges, it will not actually do the detection of shapes or straight lines. A popular method of line detection in computer vision is a process called the Hough Transform, as described in (Hough, 1962). The basic process of the Hough Transform involves iterating through each non-empty pixel and for each pixel “voting” on what the point thinks are valid lines. Each pixel will vote for a specific number of lines going through that pixel. More lines allow for more granularity, but add to processing complexity since this process must be repeated for every pixel. Certain lines from collinear pixels will get a much higher amount of votes, which will be the “lines” detected. Optimizations to the algorithm involve speed optimizations that try to limit the amount of voting in order to reduce the overall time, and also cluster the voting process into areas to be able to detect multiple segments more accurately.

Some variants of the algorithm are as follows:

The Standard Hough Transform (SHT) The original proposition that is naive. (Hough, 1962)

Adaptive Hough Transform (AHT) After a Standard Hough Transform, an accumulator array is used to identify peaks in the parameter space (which represent lines on the image) going from coarse to fine. (Illingworth & Kittler, 2009)

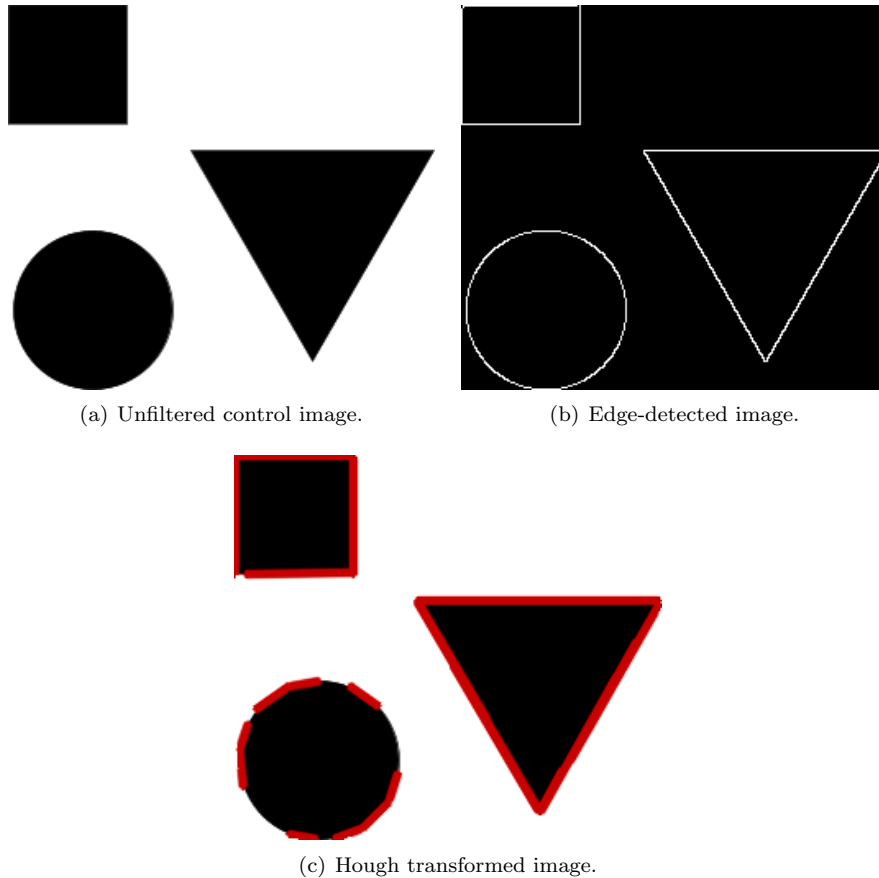


Figure 5.25: The effect of Canny edge detection and a subsequent Hough Transform.

Gradient Based Hough Transform (GHT) Uses the fact that the local gradient of the image intensity must be orthogonal to the direction of the line. This means that the angle in which the line lies can be reduced from a possible range of 180 degrees down to about 45 degrees, depending on the accuracy of the gradient calculation. This saves time by not counting redundant votes. (O’Gorman & Clowes, 2006)

Probabilistic Hough Transform (PHT) The image data is randomly sampled during the voting process. Because of the way that the random normal distribution works, this usually generates a sufficient amount of votes from the edges to detect lines, while saving time by not counting a good amount of redundant votes. (Kiryati et al., 1991)

Kernel based Hough Transform (KHT) This variant finds and groups collinear points, calculates an elliptical curve and a corresponding Gaussian kernel that encompasses them, and then uses these to limit and weight the votes. This is a much newer procedure, but is much faster than the other variants and suitable for realtime applications. Furthermore, a sample implementation is available as a guide for an implementation. (Fernandes & Oliveira, 2008)

Image Processing Libraries

OpenCV An image processing and computer vision library written by Willow Garage, the same company that wrote ROS. It takes a high-level approach to image processing, and implements many common image processing operations. Because it is tailored specifically towards computer vision, it also contains computer vision specific operations. However, it does not make use of GPU processing and performs operations sequentially. There is an effort to port several functions of OpenCV that makes use of the GPU, but it is in the experimental phase. There is also GpuCV, which is a completely different library that aims to be a drop-in replacement.

CUDA The CUDA API is the API provided by NVIDIA, the manufacturer of the NVIDIA Tesla GPU. It makes use of the fact that the GPU has many cores, but at the expense of ease of programming, since the provided operations are low-level programming operations and not high level image processing operations. CUDA is vendor-specific and closed source, and limits Prometheus to NVIDIA GPUs, and the architectural decisions that NVIDIA decide to take.

OpenCL The open alternative to CUDA that supports any GPU, OpenCL, is relatively newer. It provides a unified programming interface to a variety of graphics cards across different models and manufacturers. However, this comes at the price of having the lowest common denominator of features between all the supported cards. Finally, the OpenCL is relatively newer and is just gaining traction and community adoption.

5.4.2 Methodology

The overall high-level process of line detection is as follows: The image is first captured from the cameras. This is followed by a step where the image is perspective transformed to reflect the top-down view of the local map. Next, it is put through a pre-processing stage where noise is reduced and the irrelevant parts of the image are filtered out. After that, the lines on the filtered image are detected. Finally, the lines are placed onto the local map along with the other obstacles to be used in path planning.

Implementation Decisions

The software was initially implemented in the open source OpenCV image manipulation library to serve as a prototype for testing, because of the large number of already implemented image operators it contains and the relative ease of combining them to create more complicated operators. However, the OpenCV library performs its operations sequentially on a single thread, which does not make a completely effective use of the available hardware. There is noticeable delay between events happening in real life and the results appearing on the computer screen. While this delay is nonexistent at 15 Hz, it constitutes a visible delay at 30 Hz, which is the maximum available framerate for the cameras. Currently, the image acquisition frequency appears to be sufficient for the speed that the robot moves at, but it is worth exploring further options in case the other limitations that keep the robot from moving faster are removed.

In order to maximize the number of frames processed and minimize delay, there are two possible options.

One option is to attempt to use the drop in replacements for OpenCV listed above to get a speed gain. This seems to be the most likely since it is the least time consuming. The alternative is to make use of the raw CUDA API, which can take advantage of the low level functionality of the NVIDIA Tesla GPU and its parallel nature, but requires significant reimplementations of existing functions.

To fit with the overall architecture, the software was implemented as a standalone ROS node. The node subscribes to a topic that publishes image data and publishes line output to another topic. One major advantage of doing this is that it simplifies testing because of the ability of recording and simulating live data with the rosbag system. Another advantage is that the image capture is completely decoupled from the processing, meaning that the change or addition of different types of cameras does not constitute a significant problem.

Image Capture

The current method of image capture uses the camera1394 driver from ROS. This is a wrapper around the libdc1394 library that supports a wide variety of firewire cameras. The driver offers good integration with ROS, including configurability with ROS parameters and output to a ROS topic, as well as integration with the existing image pipeline for further features such as calibration and stereo correspondence.

Initial Pre-Processing Stage

The process of determining which pre-processing functions should be used, with what parameters, and in what order was initially determined empirically through a series of tests. The tests were conducted on pre-recorded footage of a simulated competition environment where the lines were drawn on grass with spray paint. Several recordings of the environment were made during several different days and different lighting conditions.

The first method resulting from this methodology was akin to this:

1. Smoothing
2. Dilation
3. Thresholding
4. Canny Edge Detection
5. Hough Transform

While this produced good results on some recorded data sets, it seemed to fail unexpectedly in others, overdetecting or underdetecting lines. The problem was that the thresholding parameters were very sensitive to adjustment because the intensities of the non-lines and lines would be very close to each other, especially when the paint on the lines was old and fading or there was a patch of non-grassy area with light colored soil. Thus, one value that worked in a single image would not work again in a brighter or darker one, leading to a detector that was not robust across different weather conditions.

Re-Evaluation and Current Pre-Processing Stage

To solve this, a completely new approach was taken by reasoning in the reverse direction. The colour content of a random sample of white line pixels through the camera were analyzed, along with a random sample of different sorts of non-white areas like grass and soil. Then, the differences were identified. It was noted that in the HSV (Hue, Saturation, Value) color space, the hues of the line pixels were largely distinct from most of the other kinds of pixels. This left the detector with only a few false positives, mainly in exposed soil. Through the previous analysis, it was determined that the value channel had different ranges of values for the two types of terrain, so limiting the range of value removed the false positives. For this purpose, the OpenCV `split()` and `inRange()` functions were used. Approximate limits were from 80 to 120 for the hue channel and 230 to 255 for the value channel, leaving the saturation channel intact.

Before this HSV filtering operation, an initial step of median filtering was performed with an aperture size of 5. The HSV filtering operation is followed by a single iteration of a morphological opening to make large regions more apparent while decreasing the importance of smaller regions. A 3x3 rectangular structuring element was used for the opening. Finally, another median filter with an aperture of 3 was applied.

Inverse Perspective Mapping

Since the cameras are angled down and forwards but the data we wish to obtain is from the top-down perspective, the image data must be transformed to the top-down point of view. This transformation depends on the assumption that the ground is completely planar and always at a specific angle to the robot. This is termed the “Flat Ground Assumption”. This assumption allows the placement of lines in 3D space without depth perception. While this might be unacceptable for applications where the terrain heavily varies, the competition grounds are mostly flat, and the small errors remain negligible. The IPM is implemented through the `getPerspectiveTransform()` and `warpPerspective()` functions in OpenCV. The test for correct values for IPM involves placing two parallel lines in the field of view of the camera and ensuring that in the transformed image the lines are also parallel.

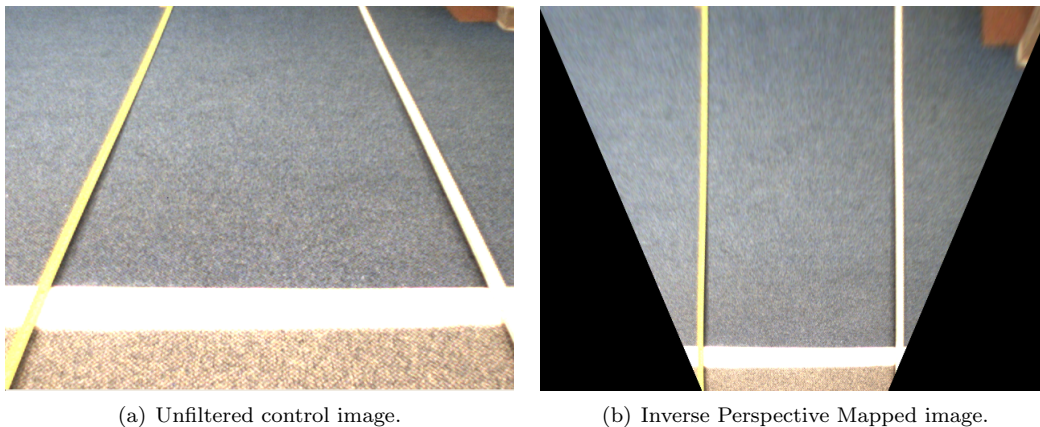


Figure 5.26: The effects of Inverse Perspective Mapping.

Line Detection

For the line detection step, the Kernel Based Hough Transform method was initially planned to be used, because of its increased performance advantages compared to the alternatives. However, this would have required an implementation from scratch based on the paper, so due to time constraints the Probabilistic Hough Transform, which was already implemented in OpenCV, was chosen instead.

The Probabilistic Hough Transform implementation in OpenCV, `HoughLinesP()`, takes a standard image matrix and outputs a series of points that denotes the beginning and end points of each line segment found. The configurable parameters include the resolution of the angle and distance of the accumulator for each voting step, the minimum length of each acceptable line and the maximum allowable gap between segments to be considered a single segment.

Local Map Placement

The resulting lines are correct in terms of relative sizes, but are usually off by a scaling factor in real-life sizes. Since the scaling factor from detected line sizes to real-world sizes does not change unless the camera placement changes, it is possible to determine them once empirically, and use those values during operation. The correct scaling factor is determined by placing a detectable square of known sizes at a known location in the field of view of the camera, and comparing the sizes and locations detected lines to the lines of the real object.

5.4.3 Results

As can be seen from Figure 5.27, we have effectively developed a line detection system that functions robustly under a wide variety of lighting conditions. While formal testing has not been conducted, during the routine testing of the robot the line detection system has functioned well in many different times of day and different amounts of cloud cover, as well as under indoor lighting. The performance limitation of the system running in a single thread on an Intel Core i7 CPU is approximately 8 fps.

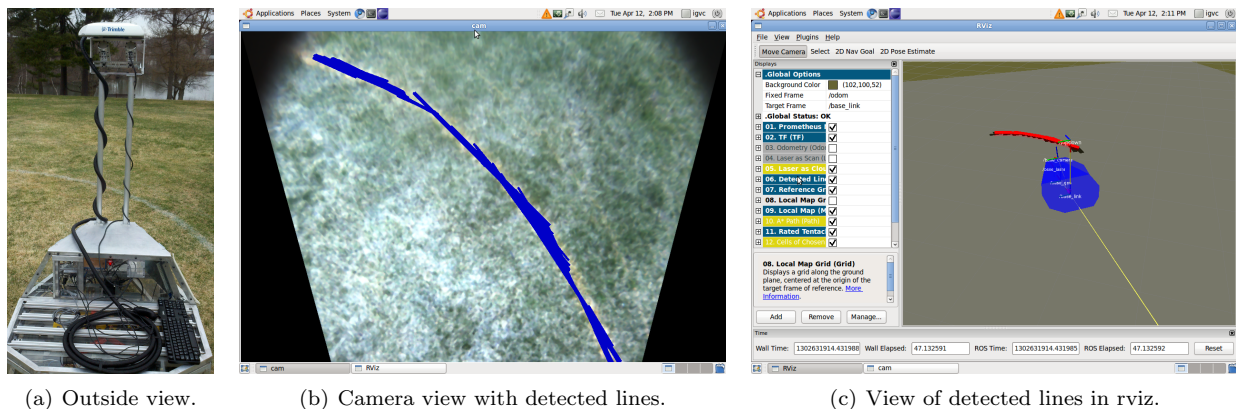


Figure 5.27: Demonstrating the three phases of detection.

Obviously, when the initial assumptions are not met, the algorithm produces sub-optimal results. Ex-

amples of these include a slight displacement in line location when dealing with a non-flat ground plane, and lack of detection when there is a lack of visibility such as in situations where even the human eye itself cannot distinguish the lines in the raw image data.

Since the cameras are being used in an attempt to calculate the location of the lines on a map, the intrinsic parameters of the lens play a role in the calculations. However, it was found that the other sources of error in odometry information, such as encoder slippage, IMU drift and GPS drift, are large enough that the implications of a non-calibrated camera are negligible. A similar source of minor error, the minimal vibration of the metal posts that hold the cameras was also considered negligible for our purposes.

5.4.4 Conclusion

While the results are satisfactory, there still are future concerns and improvements. Speed is a major area in which improvements could be implemented. As mentioned previously, using the available NVIDIA Tesla GPU to accelerate the processing is an option. Where possible, the process should be improved by using one of the GPU based drop-in replacements to OpenCV functions. This minimizes reimplementing time while maximizing performance improvements. When this is done, the frame rate can be increased to the full 30 fps available from the cameras at 1024x768. While 1024x768 at 30 fps is only available in the YUV color space, the conversion may be handled by OpenCV or through the ROS image pipeline. Finally the ability to drop frames must be implemented to ensure that if the images can't be processed fast enough, the delay doesn't accumulate over time.

On the detection side, there are many possible areas for expansion of the basic feature set. First, it is possible to come up with a better way to change the colors to be detected as lines than to vary numerical parameters by hand. One way of doing this is to utilize the touch screen to do a finger-based selection of what the user thinks is a line. This area would then be analyzed to provide a range of values that are deemed acceptable as lines, and finally would be used in the detection phase. Another way is to utilize a hill climbing search to try to dynamically maximize the detection quality. However, this method has the risk of overdetecting non line areas, and would need to be mitigated with some sort of error checker by analyzing the change in variance of the new values.

It is also possible to use the already available line information better to get more accurate results and less false positives. One idea is to use some form of statistical filtering where clusters of lines would be determined with a technique such as k-means clustering, and lines that are outliers from the closest cluster would be deemed as low probability and thus eliminated. Another idea is to join segments on the same line together to account for gaps in areas where detection failed. This would also aid in showing dashed lines as solid ones. However, since our path planning algorithms take into account the width of the robot, this does not really constitute an issue at this point. Finally, the data can be correlated temporally such that if a line appears at a location where no lines have been seen before, it can be deemed less likely. Of course, the older records of line locations would fade over time and be replaced by newer ones to accommodate the long term changing nature of the line data in situations such as turns.

Chapter 6

Conclusion

On April 8 the team gave a demonstration consisting of Prometheus upgrades and capabilities. The goal of this demonstration was to prove that Prometheus would be a qualifying robot at the 2011 IGVC. In order to do this, a course was set up in Institute Park. This course (shown in Figure 6.1) was comprised of three GPS waypoints (marked by red flags), cones for obstacles, and spray painted white lines to mark lanes.

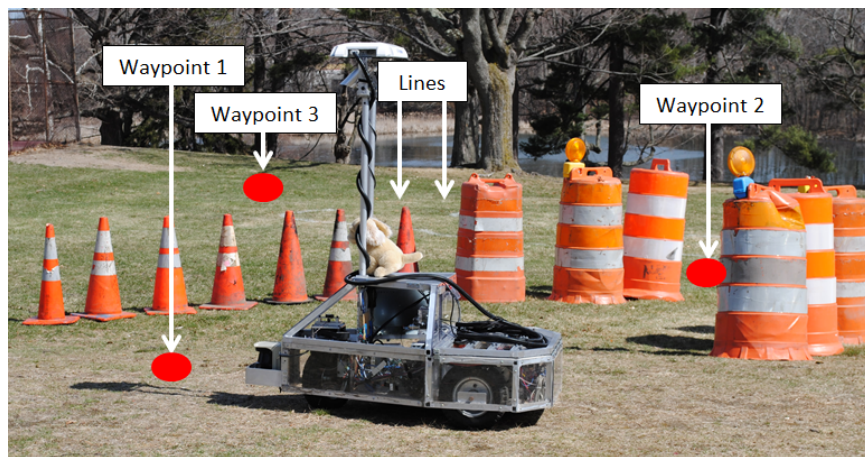


Figure 6.1: Prometheus navigating during the April 8, 2011 demonstration in Institute Park, Worcester Massachusetts

The first GPS waypoint was marked away from the robot's starting position, and at the beginning of the section of cones. The next waypoint was about ten feet after the set of cones, followed by a final waypoint at the end of a curving, spray paint-marked lane. The robot's task was to travel to each of these waypoints, while along the way, avoiding obstacles and staying within the white lines.

This task was deemed acceptable by the team because it demonstrated all three major components of qualification, as stated in the IGVC rules: "Lane Following: The vehicle must demonstrate that it can detect and follow lanes. Obstacle Avoidance: The vehicle must demonstrate that it can detect and avoid obstacles. Waypoint Navigation: Vehicle must prove it can find a path to a single 2 meter navigation waypoint" (Theisen, 2010).

After a few trial runs and alterations, the robot completed the majority of the course. It successfully

traveled to the first waypoint, avoided the cones on either side of it, got to the second waypoint, and avoided a portion of the white lines. After breaking away from the lane, the robot made its way to the third waypoint. In doing so, the robot demonstrated its ability to fully meet the qualification standards of the second and third rules mentioned previously. In terms of lane following, the robot did stay within the lanes for part of the course. However, once the robot paused to plan some of its path, its memory of the lines it saw was overwritten with new data. Since the demonstration, this memory has been increased in order to compensate for situations of the sort.

Upon completion of testing we compared our results to our project requirements (referenced in 7) and believe we have met all six requirements stated. Through our ability to perform the tasks specified in bullets two through four, we believe we have also met the competitive IGVC entry requirement of bullet one. The robot's ruggedness has been tested and established through multiple months of indoor and outdoor testing with no components breaking or sustaining damage. Finally, we feel that the manual control system developed was in fact intuitive and easy to use. Visual cues and displays significantly cut down on debugging time while more accessible ports simplified connections and reduced crowding in internal areas.

Based on the project requirements we completed and the positive results seen during testing, we are very pleased with the final outcome of this MQP. We were able to combine the various skills of six different students to produce multiple performance advancements for WPI's UGV Prometheus. These advancements will contribute to a competitive showing at the IGVC on June 4th, as well as provide a vast platform for future work and research in the area of autonomous vehicles.

Appendix A

Authorship

Mali Akmanalp wrote the Vision section of the Intelligence chapter and the GPS-related sections of the World Representation chapter.

Ryan Doherty wrote the Inertial Measurement Unit section and collaborated with Ellen to write the Global Positioning System section of the Sensors chapter.

Jeffrey Gorges wrote the introduction for Robot Structure chapter as well as sections 2.4.1 and 2.2.1 in the chapter discussing the chassis of the robot.

Peter Kalauskas wrote the Software Architecture chapter as well as the World Representation and Path Planning sections of the Intelligence chapter.

Ellen Peterson wrote the Motivation and Intelligent Ground Vehicle Competition (IGVC) sections of the Introduction chapter as well as the Sensor Fusion section of the Intelligence chapter and Conclusion chapter. Ellen also collaborated with Ryan to write the Global Positioning System section of the Sensors chapter.

Felipe Polido wrote the Paper Overview of the Introduction chapter as well sections 2.4.2 and 2.2.2 in the Robot Structure chapter discussing the usability improvements for Prometheus. He also wrote the encoder and IMU compensation portions in the Sensors chapter.

All team members contributed to writing the Starting Condition of Prometheus section of the Introduction chapter.

Appendix B

Frequency Spectrum Analysis

Two important signals experienced problems during the competition. The first was the GPS. The accuracy of the GPS was very poor during the competition such that the robot was not able to navigate to a waypoint ten meters away. This problem stemmed from the signal error being too great, significantly more than the advertised and previously tested accuracy of less than three meters. The other signal that had difficulties was the wireless emergency stop. The signal is supposed to operate at a range of 1500 feet but was only effective up to 35 feet. The minimum distance required for the e-stop at the IGVC competition is 50 feet.

One hypothesis for both of these problems was that the robot was creating so much electro-magnetic radiation that these signals were being disrupted. In order to solve such a problem, the electronics of the robot would need to be shielded from the receivers for the GPS and the wireless e-stop. It was necessary to test this hypothesis and identify different frequencies that might be creating the problem.

The radio frequencies being emitted from the robot were measured using a spectrum analyzer and a broad band rigid horn antenna directed at the body of the robot. A graph of frequencies versus amplitudes was displayed on the spectrum analyzer. The robot was then turned on to various states including off, the computer running at top speeds, the motors being run at full power, the wireless emergency stop being engaged and disengaged. All of these measurements were conducted with the cover of the robot both open and closed.

After doing these measurements with a full spectrum from 0 Hz to 3 GHz, two key areas were highlighted. First, the display was focused on the GPS frequency of 1.575 42 GHz. The spectrum analyzer image was then recorded for the different states of the robot to see if there was any effect on these particular frequencies. The same was done for the wireless e-stop frequency which was observed to be 304.9 MHz.

Figure B.1(a) shows the baseline image for the GPS frequency range without any of the robot systems turned on. The noise shown is between -65 and -67 dB. Figure B.1(b) shows the same range with the robot turned on and all motors running at top speed. There was no observable difference between the two. This means that the GPS signal is most likely not affected by the electro-magnetic radiation from the robot.

Next the spectrum analyzer was focused on the range around the wireless e-stop signal, 340.9 MHz. Figure B.2(a) shows the signal from the e-stop without any systems on. Figure B.2(b) shows the signal

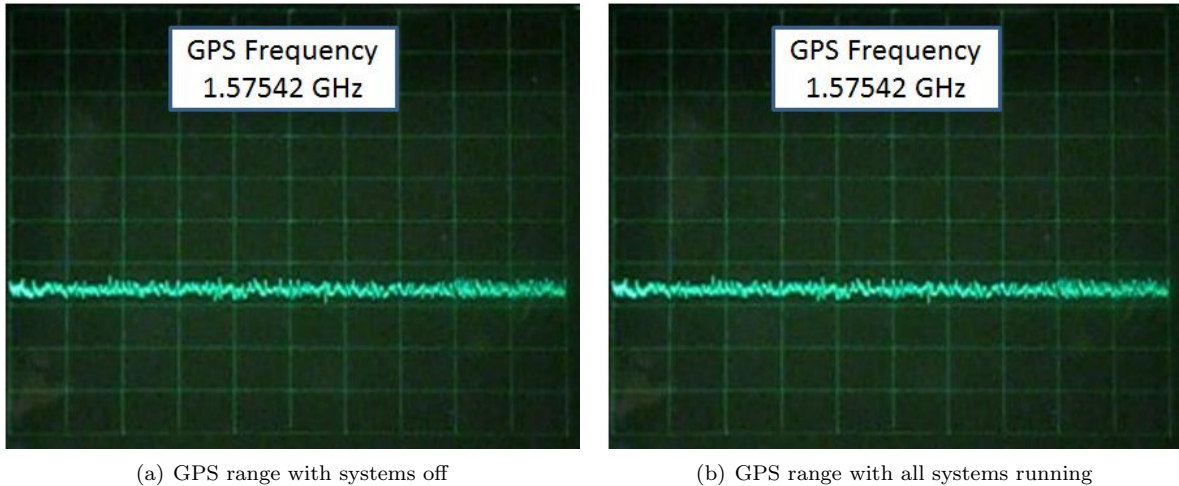


Figure B.1: Frequency spectrum analysis

from the e-stop when all the systems are running including the motors spinning at maximum velocity. A large amount of noise is generated by the robot systems in this range. The wireless e-stop signal even has a lower amplitude when the systems are running. This means that the wireless signal is being affected by the computer and motors.

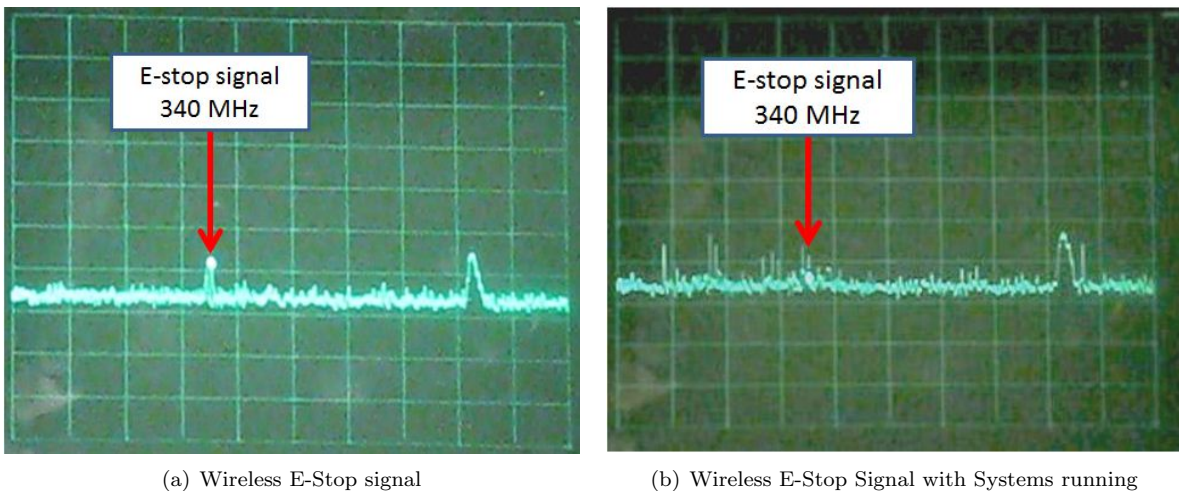


Figure B.2: Frequency spectrum analysis

The results of this experiment showed two things. The first was that there was no observable effect on the baseline noise in the range of the GPS signal with any configuration of operating parts of the robot. This meant that the poor accuracy of the GPS was most likely not due to interference from the electronics of the robot. The second conclusion was that the wireless e-stop frequency was affected by the electronics on board. Running the motors at full speed created interference at the same frequency. This helps explain the limited range of the device. Based on these findings, shielding options can be more effectively discussed. The first question is whether or not the robot needs to be shielded. Since the problem with the GPS cannot be linked to frequency interference, shielding would not in fact help the accuracy of the GPS. The wireless e-stop is

effective at the required range, and therefore shielding is not necessary for using it. If it is determined that the e-stop needs a greater range, shielding the motors would most effectively block the interfering signals based on the results of the experiment.

Appendix C

Budget

The following tables are a breakdown of the parts obtained for Prometheus 2011. There is a table for the overall budget, as well as tables of the specific budgets for both the chassis upgrades and external interface. These tables contain the quantity of each item, where it was purchased, and its cost (or value if donated by a sponsor).

Item:	Supplier:	Note:	Cost(\$):
Touchscreen 3M MicroTouch Display C1700SS (15") Serial	3M		467.86
OCZ Vertex 2 60GB SATA II MLC Internal Solid State Drive (SSD)	New Egg		129.99
External Interface (includes all components)	Plastics Unlimited	Partially paid by ECE department	79.33
Acrylic Sheet	Amazon		39.5
Futaba 6EX 2.4GHz 6-channel Remote Control Transmitter/Receiver		\$214 + shipping	214
Chassis Improvements	NI	(see attached page)	925.43
cRIO Serial Module		Replacement	579
Camera Lens			233
12V Schumacher SE-4022 Battery Charger and Tester	Amazon	x2	246
Bandago Van		Transportation	1859
		Subtotal:	4773.11
Sponsorship:			
OmniStar GPS Subscription	OmniStar		850
TCM-XB Evaluation Kit (Inertia Momentum Unit)	PNI		1,096
ThinkPad Laptop	LeNovo		1,200
		Subtotal:	3146
		Total:	7919.11

Table C.1: Overall budget.

App. Date	Part Description	Store	Cost per	Number	Total Cost
9/27/2010	Thrust Bearing	MacMaster	14.84	1	14.84
9/27/2010	Keyed Shaft	MacMaster	12.87	1	12.87
9/27/2010	Key	MacMaster	2.74	1	2.74
9/27/2010	Bearing	MacMaster	9.66	1	9.66
10/18/2010	Loctite 242	Lowes	6.87	1	6.87
10/18/2010	Loctite 271	Lowes	6.87	1	6.87
10/18/2010	Hex-wrench set	Lowes	12.48	1	12.48
10/18/2010	Screw Driver Set	Lowes	12.96	1	12.96
10/18/2010	Toolbox	Lowes	8.28	1	8.28
11/12/2010	1"x1"x72" Al. square tubing 1/8" wall	MacMaster	21.27	5	106.35
11/12/2010	24"x48"x1/4" Polycarbonate	MacMaster	57.24	2	114.48
11/12/2010	1"x1"x36" Al. square stock	MacMaster	26.98	1	26.98
11/12/2010	12"x12"x0.19" Al. sheet	MacMaster	35.19	1	35.19
11/12/2010	1"x1"x72" Al. T-slot extrusion	Faztek	33.12	2	66.24
1/11/2011	2'x2'x.25" Al. Plate	MetalsDepot	118.56	1	118.56
1/11/2011	10' Weather stripping	Lowes	7.86	4	31.44
1/11/2011	9.8 oz. Silicone Caulk	Lowes	5.97	3	17.91
1/11/2011	10-24 x 3/4" S. Steel Socket Cap Screws X25	McMaster	8.31	2	16.62
2/7/2011	2.25"x1' thickwalled aluminum tube	McMaster	27.32	1	27.32
2/7/2011	1.25"x6' thickwalled Al. tube	McMaster	43.77	1	43.77
3/15/2011	Lenses - S48WI 4.8mm F/1.8	ccddirect.com	84	2	168
3/15/2011	Filters - M37.5x0.5mm linear polarizing	edmund optics	32.5	2	65
				Total	925.43

Table C.2: Chassis budget.

Item:	Supplier	Price(\$)	Quantity:	Total(\$)
4-port USB Hub	newegg	10.39	1	10.39
Ethernet connector (coupler adapter)NewEgg	newegg	5.99	1	5.99
VOLTMETER LCD 8-50VDC 3DIGIT	DigiKey	30.14	2	60.28
SOCKET 12V AUTO W/SAFETY CAP	DigiKey	2.67	1	2.67
Status LED	ECE Shop		1	0
Computer Power Switch	ECE Shop		1	0
cRIO Reset Switch	ECE Shop		1	0
Terminal Strips	ECE Shop		1	0
Wiring	ECE Shop		1	0
ECE Shop estimated expense		25		0
Estimated Shipping		20		0
				0
		Total:		79.33

Table C.3: External interface budget.

Appendix D

Products

<i>Brand</i>	<i>Update Rate</i>	<i>Size</i>	<i>Position</i>	<i>Other</i>
eDAQ/eDAQ-lite Accessory	5 Hz DGPS (WAAS)	2.4in. diameter × .75in. height	<3 m	http://www.somat.com/products/accessories/DGPS_receiver_5_hz.html
iBlue GM2 USB DGPS Receiver	5 Hz DGPS (WAAS)	55 mm × 63 mm × 16 mm	<3 m	\$36.99, http://www.semsons.com/igmusbDGPSre6.html
Qstarz BT-Q818X Bluetooth DGPS Receiver	5 Hz hardware switch (A-DGPS)	72.2 mm × 46.5 mm × 20 mm	<3 m standalone; <2.5 m DGPS	Has a 5 Hz compatible lap timing API, \$62.99 http://www.semsons.com/qsbtblDGPSre6.html
Garmin DGPS 18x 5 Hz	5 Hz	2.4in. diameter	<3 m	\$199 https://buy.garmin.com/shop/shop.do?cID=158&pID=13195
MediaTek MT3329 GPS	10 Hz	16 mm × 16 mm × 6 mm	<3 m	\$39.95 http://store.diydrones.com/MediaTek_MT3329_DGPS_10Hz_Adapter_p/mt3329-02.htm
Sokkia Axis3 DGPS		19 cm × 12.5 cm × 5.1 cm	<1 m	http://issuu.com/palmettoequipment/docs/axis3_brochure
Trimble AG252 DGPS	1, 5, or 10 Hz		<1 m	http://www.trimble.com/agriculture/aggps-252-receiver.aspx?dtID=technical_support

Table D.1: Comparison of specifications of DGPS receivers that were considered for use with Prometheus



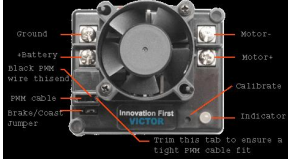

<i>Product</i>	<i>Information</i>	<i>Price</i>	<i>Picture</i>
Roboteq AX1500	<ul style="list-style-type: none"> • No closed loop (optional) • Dual motor • 30 A continuous • Serial and PWM 	\$275	
Dimension Sabertooth Dual	<ul style="list-style-type: none"> • Regenerative • Lithium cutoff mode • Simple serial and PWM • Open loop control only 	\$250	
Victor 883	<ul style="list-style-type: none"> • 30 A continuous • Only PWM • Open loop control only 	\$150	
Jaguar 24V (Current setup)	<ul style="list-style-type: none"> • 40 A continuous • Closed loop (quadrature encoder input) • Serial and PWM 	\$110	

Table D.2: Available Motor Controllers Comparison

<i>Company</i>	<i>Type</i>	<i>Specifications</i>	<i>Weight</i>	<i>Cost</i>	<i>Quantity Needed</i>	<i>Picture</i>
A123 Systems	Lithium Ion	20 A h, 3.3 V		\$1319	24	
PowerStream	Lithium Ion	55 A h, 12 V	19lbs	\$1319	2	
PingBattery	Lithium Ion	20 A h, 24 V	10.8lbs	\$344	3	
Free International Limited	Lithium Ion	100 A h, 24 V	78lbs		1	
Robot Market Place	Nickel Metal Hydride	4.5 A h, 24 V	3.12lbs	\$190	12	
Optima Batteries (Current setup)	Sealed Lead Acid	55 A h, 12 V	42.9lbs		2	

Table D.3: Lithium Ion Battery Comparison for a 24V 55Ah system

<i>Product</i>	<i>Information</i>	<i>Price</i>	<i>Picture</i>
3M Micro-Touch Display C1500SS (15") Serial	<ul style="list-style-type: none"> • Monitor only • 1024 x 768 resolution • <28 watts at normal conditions • Weight: 9.2 lbs • Communication: VGA and Serial 	\$467.86	
Kontron HMITR-104-AK-T3-24	<ul style="list-style-type: none"> • Computer solution • HMI for Transportation 10.4 inch, Touch Panel according UIC61224 / 24 VDC, EN50155 Temp Class T3 (-25C to 70C) 	\$3829	
DVM1200 All-in-one PC 12.1" touch display	<ul style="list-style-type: none"> • Computer solution • 12.1 TFT LCD touch screen • Highly durable for shock and vibration protection • Sealed, water and dust proof; IP65 rated • Fan-less, reliable operation • VESA-compliant; wall, ceiling, vehicle mountable 	\$2797	
KPanel Flange-Mounted Outdoor	<ul style="list-style-type: none"> • Computer solution • PC Housing is Rugged Aluminum • Sealed, water and dust proof; IP65 rated • Sunlight Bright 15" Industrial Flat Panel w/HI-BRITE Transflective Film and Optical Bonding • Completely Sealed Against the Elements 	\$1850	

Table D.4: Available Touchscreen Monitors Comparison

Appendix E

ION lawn mower competition

Beyond the IGVC our team will be competing on the Institute of Navigation (ION) Robotic Lawn Mower Competition. The competition involves cutting the grass on a defined area while avoiding white lines, as well as static and moving objects. Teams will be judged on area mowed in a specific time, quality of cut, and the ability to avoid obstacles.

The decision to compete derived from the fact that Prometheus currently has all the necessary capabilities to be a competitive entry, besides the lawn mower attachment itself. This opportunity allows us to demonstrate the expandability of our platform to other applications. The ION competition also gives us the ability to further extend our autonomous mapping algorithm; we will need to build on our autonomous mode to efficiently traverse the lawn, while avoiding overlap.

The lawn mower attachment will consist of an array of five (5) weed whacker blades connected directly to 12V 6000 RPM motors. This configuration was chosen because of the competition robot size limitations, motor availability, and 12V power source. The four bar linkage setup allows the cutters to always be parallel to the robot. The wheels on the cutter attachment can have their height regulated to control the height of the cut.

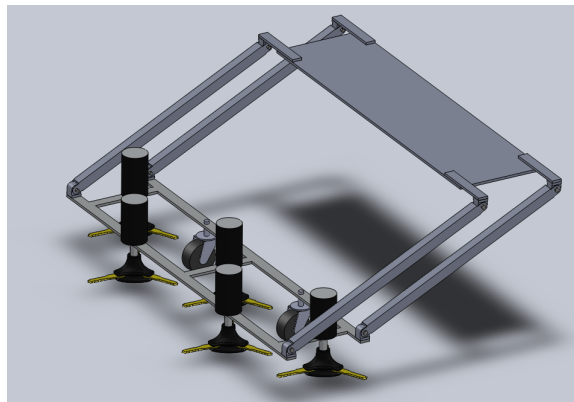


Figure E.1: Lawn mower four bar linkage

The attachment will be powered by another two (2) 12 V 55Ah batteries, this time connected in parallel to offer up to 110Ah. The electrical circuitry will go from the batteries, through resettable fuses, followed

by a relay connected to a switch and Prometheus e-stop, to the motors. This circuitry will be contained on an outdoor weatherproof box connected to the lawn mower attachment. Our intention is to have no control over the motor speeds, letting them run freely unless the switch or the emergency stop is pressed.

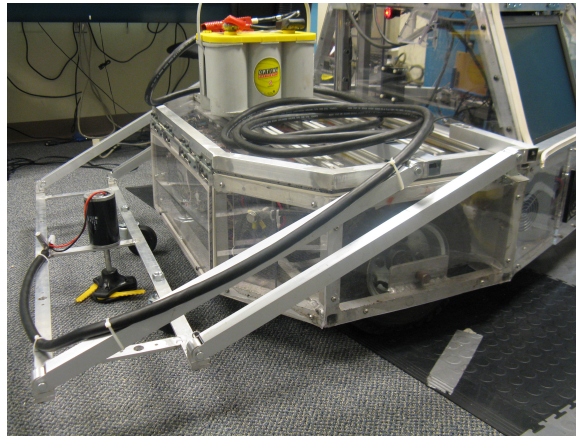


Figure E.2: Partially constructed lawnmower attachment

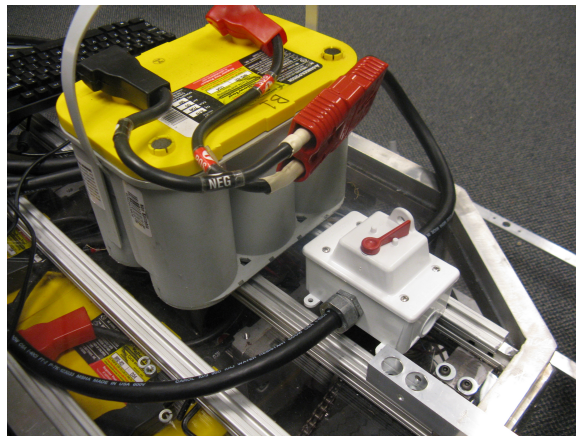


Figure E.3: Weatherproof lawn mower electrical connection box

Appendix F

Team Links

The Prometheus 2011 team has an array of information about the project available on the internet. The following links take the viewer to destinations such as the team's website, the April 8, 2011 demonstration (referenced in the Conclusion) video, promotional video, etc.

Team Website	http://www.igvc-wpi.org/
April 8, 2011 Demonstration	http://www.youtube.com/watch?v=Ij20WKGILw
Promotional Video	http://vimeo.com/22633447
Camera Mount Design	http://www.youtube.com/user/PrometheusWPI?blend=6&ob=5#p/u/0/UnVj5uPfhks
ION Qualification Video	http://www.youtube.com/watch?v=Z_bp35MyzqQ

Bibliography

- (2010). City University of New York IGVC Entry - City Alien.
Retrieved from <http://robotics.cuny.cuny.edu/blog/Research/CityALIEN>. (Accessed on 2010-12-17).
- (2011). National Instruments - Quadrature Encoder Fundamentals.
Retrieved from <http://zone.ni.com/devzone/cda/tut/p/id/4763>. (Accessed on 2011-4-26).
- Abiola, S. O., Baldassano, C. A., Franken, G. H., Harris, R. J., Hendrick, B. A., Mayer, J. R., Partridge, B. A., Starr, E. W., Tait, A. N., Yu, D. D., & Zhu, T. H. (2010). Argos: Princeton university's entry in the 2009 intelligent ground vehicle competition. vol. 7539, (p. 75390N). SPIE.
Retrieved from <http://link.aip.org/link/?PSI/7539/75390N/1>.
- Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 679–698.
Retrieved from <http://dx.doi.org/10.1109/TPAMI.1986.4767851>.
- Chen, J., Luo, C., Krishnan, M., Paulik, M., & Tang, Y. (2010). An enhanced dynamic delaunay triangulation-based path planning algorithm for autonomous mobile robot navigation. vol. 7539, (p. 75390P). SPIE.
Retrieved from <http://link.aip.org/link/?PSI/7539/75390P/1>.
- Dana, P. H. (2000). Global positioning system overview.
Retrieved from <http://www.colorado.edu/geography/gcraft/notes/gps/gps.html>. (Accessed on 2010-12-17).
- DARPA (2011). Urban challenge.
Retrieved from <http://archive.darpa.mil/grandchallenge/>. (Accessed on 2011-4-24).
- Devivo AST (2008). Jr Middleware.
Retrieved from <http://www.jrmiddleware.org/>. (Accessed on 2010-12-17).
- Dong, G., & Xie, M. (2005). Color clustering and learning for image segmentation based on neural networks. *Neural Networks, IEEE Transactions on*, 16(4), 925–936.
- Drolet, L., Michaud, F., & Cote, J. (2000). Adaptable sensor fusion using multiple kalman filters. In *Intelligent Robots and Systems, 2000. (IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 2, (pp. 1434–1439 vol.2).

- Dutch, S. (2003). Converting UTM to Latitude and Longitude (Or Vice Versa). Retrieved from <http://www.uwgb.edu/dutchs/usefuldata/utmformulas.htm>. (Accessed on 2011-4-24).
- Dyer, C. (2002). Theory of the Taylor expansion. Retrieved from http://pathfinder.scar.utoronto.ca/~dyer/csca57/book_P/node26.html. (Accessed on 2011-4-24).
- Fernandes, L., & Oliveira, M. (2008). Real-time line detection through an improved Hough transform voting scheme. *Pattern Recognition*, 41(1), 299–314.
- Frese, U. (2010). Interview: Is slam solved? *KI - Künstliche Intelligenz*, 24, 255–257. 10.1007/s13218-010-0047-x. Retrieved from <http://dx.doi.org/10.1007/s13218-010-0047-x>.
- GPSd (2011). GPSd. Retrieved from <http://gpsd.berlios.de/>. (Accessed on 2011-4-24).
- Hough, P. V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.
- IGVC (2011). The Purpose of IGVC. Retrieved from <http://www.igvc.org/objective.htm>. (Accessed on 2011-4-24).
- Illingworth, J., & Kittler, J. (2009). The adaptive Hough transform. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (pp. 690–698).
- Institut Geographique National (2007). ITRS and WGS84. Retrieved from <ftp://itrf.ensg.ign.fr/pub/itrf/WGS84.TXT>. (Accessed on 2011-4-24).
- Joint Ground Robotics Enterprise (2007). Retrieved from <http://www.jointrobotics.com/>. (Accessed on 2010-12-17).
- Kiryati, N., Eldar, Y., & Bruckstein, A. (1991). A probabilistic Hough transform. *Pattern recognition*, 24(4), 303–316.
- McKeon, R. T., Krishnan, M., & Paulik, M. (2006). Obstacle recognition using region-based color segmentation techniques for mobile robot navigation. vol. 6384, (p. 63840R). SPIE. Retrieved from <http://link.aip.org/link/?PSI/6384/63840R/1>.
- Mehaffey, J. (1999). The meaning of WARM START, COLD START, AutoLocate, and "Search the Sky". Retrieved from <http://gpsinformation.net/main/warmcold.htm>. (Accessed on 2011-4-24).
- Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the National conference on Artificial Intelligence*, (pp. 593–598). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.
- National Instruments Corporation (2010a). Ni labview mathscript rt module. Retrieved from <http://sine.ni.com/nips/cds/view/p/lang/en/nid/207267>. (Accessed on 2010-12-17).
- National Instruments Corporation (2010b). Ni labwindows/cvi. Retrieved from <http://www.ni.com/lwcvil/>. (Accessed on 2010-12-17).

- Nepal, K., Fine, A., Imam, N., Pietrocola, D., Robertson, N., & Ahlgren, D. J. (2009). Combining a modified vector field histogram algorithm and real-time image processing for unknown environment navigation. vol. 7252, (p. 72520G). SPIE.
Retrieved from <http://link.aip.org/link/?PSI/7252/72520G/1>.
- Newman, J., Zhu, H., Partridge, B. A., Szocs, L. J., Abiola, S. O., Corey, R. M., Suresh, S. A., & Yu, D. D. (2011). Phobator: Princeton university's entry in the 2010 intelligent ground vehicle competition. vol. 7878, (p. 78780Z). SPIE.
Retrieved from <http://link.aip.org/link/?PSI/7878/78780Z/1>.
- O’Gorman, F., & Clowes, M. (2006). Finding picture edges through collinearity of feature points. *Computers, IEEE Transactions on*, 100(4), 449–456.
- Open Source Geospatial Foundation (2011). Converting UTM to Latitude and Longitude (Or Vice Versa).
Retrieved from <http://www.gdal.org/>. (Accessed on 2011-4-24).
- Pike, J. (2011). Military robots / unmanned ground vehicles (ugv).
Retrieved from <http://www.globalsecurity.org/military/systems/ground/ugv.htm>. (Accessed on 2011-4-24).
- Poskanzer, J. (2003). Pgm format specification.
Retrieved from <http://netpbm.sourceforge.net/doc/pgm.html>. (Accessed on 2011-4-26).
- Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., & Ng, A. (2009). ROS: an open-source Robot Operating System. In *International Conference on Robotics and Automation*.
- Raymond, E. (2004). *The art of Unix programming*. Addison-Wesley professional computing series. Addison-Wesley.
- ROS (2010).
Retrieved from <http://www.ros.org/>. (Accessed on 2010-12-17).
- Rowe, S., & Wagner, C. R. (2008). An Introduction to the Joint Architecture for Unmanned Systems (JAUS).
Retrieved from <http://www.openskies.net/papers/07F-SIW-089IntroductiontoJAUS.pdf>. (Accessed on 2010-12-17).
- Russell, S., & Norvig, P. (2010). *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall.
- SAE International (2010). Sae standards.
Retrieved from <http://standards.sae.org/>. (Accessed on 2010-12-17).
- Siciliano, B., & Khatib, O. (2008). *Springer handbook of robotics*. Gale virtual reference library. Springer.
- Stachniss, C., Frese, U., & Grisetti, G. (2010). OpenSLAM.
Retrieved from <http://openslam.org/>. (Accessed on 2010-12-17).
- Theisen, B. (2010). The 19th annual intelligent ground vehicle competition (igvc) annual igvc rules.
Retrieved from <http://www.igvc.org/rules.pdf>. (Accessed on 2010-12-17).

- Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. Intelligent robotics and autonomous agents. MIT Press.
- Trimble (2004). AgGPS252 Receiver User Guide.
Retrieved from http://trl.trimble.com/dscgi/ds.py/Get/File-159152/AgGPS252_100A_UserGuide_55510-00-ENG.pdf. (Accessed on 2011-4-24).
- Vislab (2010). The 2010 VisLab Challenge.
Retrieved from http://viac.vislab.it/?page_id=11. (Accessed on 2010-12-17).
- von Hundelshausen, F., Himmelsbach, M., Hecker, F., Mueller, A., & Wuensche, H.-J. (2008). Driving with tentacles: Integral structures for sensing and motion. *J. Field Robot.*, 25, 640–673.
Retrieved from <http://portal.acm.org/citation.cfm?id=1405647.1405650>.
- Wang, B., Truchanovicus, V., Sacco, D. R., Roy, B. R., Panzica, A. C., Madera, R., Gamache, C. M., Fitzpatrick, R. J., & Barrett, J. M. (2010). Design and realization of an intelligent ground vehicle.
Retrieved from <http://www.wpi.edu/Pubs/E-project/Available/E-project-042910-143753/>. (Accessed on 2010-12-17).
- Welch, G., & Bishop, G. (1995). An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel Hill, NC*.
- WPI (2010). 2010-2011 undergraduate catalog.
Retrieved from <http://www.wpi.edu/academics/catalogs/ugrad2011.html>. (Accessed on 2011-4-24).



**The 19TH Annual Intelligent
Ground Vehicle Competition
(IGVC)**

**June 3RD - 6TH, 2011
Oakland University
Rochester, Michigan**

In memory of Paul Lescoe

***New Items:
Corrected Flag Part Numbers (page 8)
100ft. wireless E-stop distance.
Mapping & course memorization is allowed.***

Student teams are invited to display their vehicles at The Association for Unmanned Vehicle Systems International's Unmanned Systems North America 2011 Symposium & Exhibition Held at Washington Convention Center in Washington, District of Columbia on August 16TH – 19TH, 2011

April 21, 2011 Version

TABLE OF CONTENTS

I	COMPETITION INFORMATION
I.1	TEAM ENTRIES
I.2	VEHICLE CONFIGURATION
I.3	PAYLOADS
I.4	QUALIFICATION
I.5	INDEMNIFICATION AND INSURANCE
II	AUTONOMOUS CHALLENGE
II.1	OBJECTIVE
II.2	VEHICLE CONTROL
II.3	OBSTACLE COURSE
II.4	COMPETITION RULES & PROCEDURES
II.5	PRACTICE COURSE
II.6	TRAFFIC VIOLATION LAWS
II.7	HOW COMPETITION WILL BE JUDGED
II.8	GROUND FOR DISQUALIFICATION
III	DESIGN COMPETITION
III.1	OBJECTIVE
III.2	WRITTEN REPORT
III.3	ORAL PRESENTATION
III.4	EXAMINATION OF THE VEHICLE
III.5	FINAL SCORING
IV	NAVIGATION CHALLENGE
IV.1	OBJECTIVE
IV.2	ON-BOARD SENSORS
IV.3	GPS COURSE
IV.5	PRACTICE COURSE
IV.5	THE RUN PROCEDURE AND SCORING
IV.6	RUN TERMINATION
V	J AUS CHALLENGE
V.1	TECHNICAL OVERVIEW
V.2	COMMON OPERATING PICTURE
V.3	COMMUNICATIONS PROTOCOLS
V.4	J AUS SPECIFIC DATA
V.5	COMPETITION TASK DESCRIPTION
V.6	TRANSPORT DISCOVERY
V.7	CAPABILITIES DISCOVERY
V.8	SYSTEM MANAGEMENT
V.9	VELOCITY STATE REPORT
V.10	POSITION AND ORIENTATION REP
VI	AWARDS AND RECOGNITION
VI.1	AUTONOMOUS CHALLENGE
VI.2	DESIGN COMPETITION
VI.3	NAVIGATION CHALLENGE
VI.4	J AUS CHALLENGE
VI.5	ROOKIE OF THE YEAR AWARD
VI.6	GRAND AWARD
VI.7	PUBLICATION AND RECOGNITION

I. COMPETITION INFORMATION

I.1 TEAM ENTRIES

Teams may be comprised of undergraduate and graduate students, and must be supervised by at least one faculty advisor. Interdisciplinary (Electrical, computer, mechanical, systems engineering, etc.) teams are encouraged. Students must staff each team. Only the student component of each team will be eligible for the awards. Faculty supervisor will certify that all team members are bonafide students on the application form and will also provide contact information (telephone number and e-mail address) for him and the student team leader on the form. Business/Non-Engineering students are encouraged to join teams to promote marketing, sponsorships, and other program management functions. For a student to be eligible to compete as a team member, they are required to have attended at least one semester of school as a registered student between June 2010 and June 2011.

Team sponsors are encouraged. Sponsors' participation will be limited to hardware donation and/or funding support. Sponsors logos may be placed on the vehicle and may be displayed inside of the team maintenance area. Teams should encourage sponsor attendance at the IGVC.

Schools are encouraged to have more than one entry; but are limited to a maximum of three per school, and each vehicle must have a separate team of students and a distinct design report. Each entry must be based on a different chassis and software and must be documented by a separate application form and design report, submitted in accordance with all deadlines. All entries must have a team name and each application form must be **TYPED** and accompanied with a \$250.00 non-refundable registration fee made payable to *Oakland University*. Intention to compete must be received no later than **February 28, 2011**, by mailing your application form to:

**Gerald C. Lane
C/O Dr. Ka C. Cheok
102A SEB
SECS-ECE Dept.
Oakland University
Rochester, MI 48309-4478**

If you have any questions, please contact Bernard Theisen by telephone at (586) 282-8750, fax: (586) 282-8684 or e-mail: bernard.theisen@us.army.mil.

I.2 VEHICLE CONFIGURATION

The competition is designed for a small semi-rugged outdoor vehicle. Vehicle chassis can be fabricated from scratch or commercially bought. Entries must conform to the following specifications:

- **Design:** Must be a ground vehicle (propelled by direct mechanical contact to the ground such as wheels, tracks, pods, etc. or hovercraft).
- **Length:** Minimum length three feet, maximum length seven feet.
- **Width:** Minimum width two feet, maximum width five feet.
- **Height:** Not to exceed 6 six feet (excluding emergency stop antenna).
- **Propulsion:** Vehicle power must be generated onboard. Fuel storage or running of internal combustion engines and fuel cells are not permitted in the team maintenance area (tent/building).
- **Average Speed:** Speed will be checked at the end of a challenge run to make sure the average speed of the competing vehicle is above one (1) mph over the course completed. Vehicle slower than the minimum average speed will be disqualified for the run.

- **Minimum Speed:** There will be a stretch of about 88 ft. long at the beginning of a run where the contending vehicle must consistently travel above 1 mph. A vehicle slower than this speed is considered to “hold-up traffic” and will be disqualified.
- **Maximum Speed:** A maximum vehicle speed of ten miles per hour (10 mph) will be enforced. All vehicles must be hardware governed not to exceed this maximum speed. No changes to maximum speed control hardware are allowed after the vehicle passes Qualification.
- **Mechanical E-stop location:** The E-stop button must be a push to stop, red in color and a minimum of one inch in diameter. It must be easy to identify and activate safely, even if the vehicle is moving. It must be located in the center rear of vehicle at least two feet from ground, not to exceed four feet above ground. Vehicle E-stops must be hardware based and not controlled through software. Activating the E-Stop must bring the vehicle to a quick and complete stop.
- **Wireless E-Stop:** The wireless E-Stop must be effective for a minimum of **100 feet**. Vehicle E-stops must be hardware based and not controlled through software. Activating the E-Stop must bring the vehicle to a quick and complete stop. During the competition performance events (Autonomous Challenge and Navigation Challenge) the wireless E-stop will be held by the Judges.
- **Safety Light:** The vehicle must have an easily viewed solid indicator light which is turned on whenever the vehicle power is turned on. The light must go from solid to flashing whenever the vehicle is in autonomous mode. As soon as the vehicle comes out of autonomous mode the light must go back to solid.
- **Payload:** Each vehicle will be required to carry a 20-pound payload. The shape and size is approximately that of an 18" x 8" x 8" cinder block. Refer to section I.3 Payload.

I.3 PAYLOAD

The payload must be securely mounted on the vehicle. If the payload falls off the vehicle during a run, the run will be terminated. The payload specifications are as follows: 18 inches long, 8 inches wide, 8 inches high and a weight of 20 pounds.

I.4 QUALIFICATION

All vehicles must pass Qualification to receive standard award money in the Design Competition and compete in the performance events (Autonomous Challenge and Navigation Challenge). To complete Qualification the vehicle must pass/perform all of the following criteria.

- **Length:** The vehicle will be measured to ensure that it is over the minimum of three feet long and under the maximum of seven feet long.
- **Width:** The vehicle will be measured to ensure that it is over the minimum of two feet wide and under the maximum of five feet wide.
- **Height:** The vehicle will be measured to ensure that it does not to exceed six feet high; this excludes emergency stop antennas.
- **Mechanical E-stop:** The mechanical E-stop will be checked for location to ensure it is located on the center rear of vehicle a minimum of two feet high and a maximum of four feet high and for functionality.
- **Wireless E-Stop:** The wireless E-Stop will be checked to ensure that it is effective for a minimum of 100 feet. During the performance events the wireless E-stop will be held by the Judges.

- **Safety Light:** The safety light will be checked to ensure that when the vehicle is powered up the light is on and solid and when the vehicle is running in autonomous mode, the light goes from solid to flashing, then from flashing to solid when the vehicle comes out of autonomous mode
- **Speed:** The vehicle will have to drive over a prescribed distance where its minimum and maximum speeds will be determined. The vehicle must not drop below the minimum of one mile per hour and not exceed the maximum speed of ten miles per hour. Minimum speed of one mph will be assessed in the fully autonomous mode and verified over a 88 foot distance between the lanes and avoiding obstacles. No change to maximum speed control hardware is allowed after qualification. If the vehicle completes a performance event at a speed faster than the one it passed Qualification at, that run will not be counted.
- **Lane Following:** The vehicle must demonstrate that it can detect and follow lanes.
- **Obstacle Avoidance:** The vehicle must demonstrate that it can detect and avoid obstacles.
- **Waypoint Navigation:** Vehicle must prove it can find a path to a single two meter navigation waypoint by navigating around an obstacle.

During the Qualification the vehicle must be put in autonomous mode to verify the mechanical and wireless E-stops and to verify minimum speed, lane following and obstacle avoidance. The vehicle software can be reconfigured for waypoint navigation qualification. For the max speed run the vehicle may be in autonomous mode or joystick/remote controlled. Judges will not qualify vehicles that fail to meet these requirements. Teams may fine tune their vehicles and resubmit for Qualification. There is no penalty for not qualifying the first time. Vehicles that are judged to be unsafe will not be allowed to compete. In the event of any conflict, the judges' decision will be final.

I.5 INDEMNIFICATION AND INSURANCE

Teams will be required to submit an Application Form prior to **February 28, 2011**. The Application Form can be downloaded from www.igvc.org.

Each Team's sponsoring institution will also be required to submit a Certificate of Insurance at the time the Application Form is submitted. The certificate is to show commercial general liability coverage in an amount not less than \$1 million.

In addition, each individual participating at the competition will be required to sign a Waiver of Claims when they arrive at site and before they can participate in the IGVC events.

NOTE: The IGVC Committee and Officials will try to adhere to the above official competition details, rules and format as much as possible. However, it reserves the right to change or modify the competition where deemed necessary for preserving fairness of the competition. Modifications, if any, will be announced prior to the competition as early as possible.

II AUTONOMOUS CHALLENGE COMPETITION

All teams must pass Qualification to participate in this event.

II.1 OBJECTIVE

A fully autonomous unmanned ground robotic vehicle must negotiate around an outdoor obstacle course under a prescribed time while maintaining an average course speed of one mph, a minimum of speed of one mph over a section and a maximum speed limit of ten mph, remaining within the lane, negotiating flags and avoiding the obstacles on the course.

Judges will rank the entries that complete the course based on shortest adjusted time taken. In the event that a vehicle does not finish the course, the judges will rank the entry based on longest adjusted distance traveled. Adjusted time and distance are the net scores given by judges after taking penalties, incurred from obstacle collisions and boundary crossings, into consideration.

II.2 VEHICLE CONTROL

Vehicles must be unmanned and autonomous. They must compete based on their ability to perceive the course environment and avoid obstacles. Vehicles cannot be remotely controlled by a human operator during competition. All computational power, sensing and control equipment must be carried on board the vehicle. No base stations allowed for positioning accuracy is allowed. Teams are encouraged to map the course and use that information to improve their performance on the course.

II.3 OBSTACLE COURSE

The course will be laid out on grass over an area of approximately 330 feet long by 240 feet wide and 1,320 feet in length. This distance is identified so teams can set their maximum speed to complete the course pending no prior violations resulting in run termination. Track width will vary from ten to twenty feet wide with a turning radius not less than five feet.

The course outer boundaries will be designated by continuous or dashed white lane markers (lines) approximately three inches wide, painted on the ground. Track width will be approximately ten feet wide with a turning radius not less than five feet. Alternating side-to-side dashes will be 15-20 feet long, with 10-15 feet separation. A minimum speed will be required of one mph and will be a requirement of Qualification and verified in each run of the Autonomous Challenge. If the vehicle does not average one mph for the first 88 feet (one minute) from the starting line, the vehicle run will be ended. The vehicle will then need to average over one mph for the entire run.

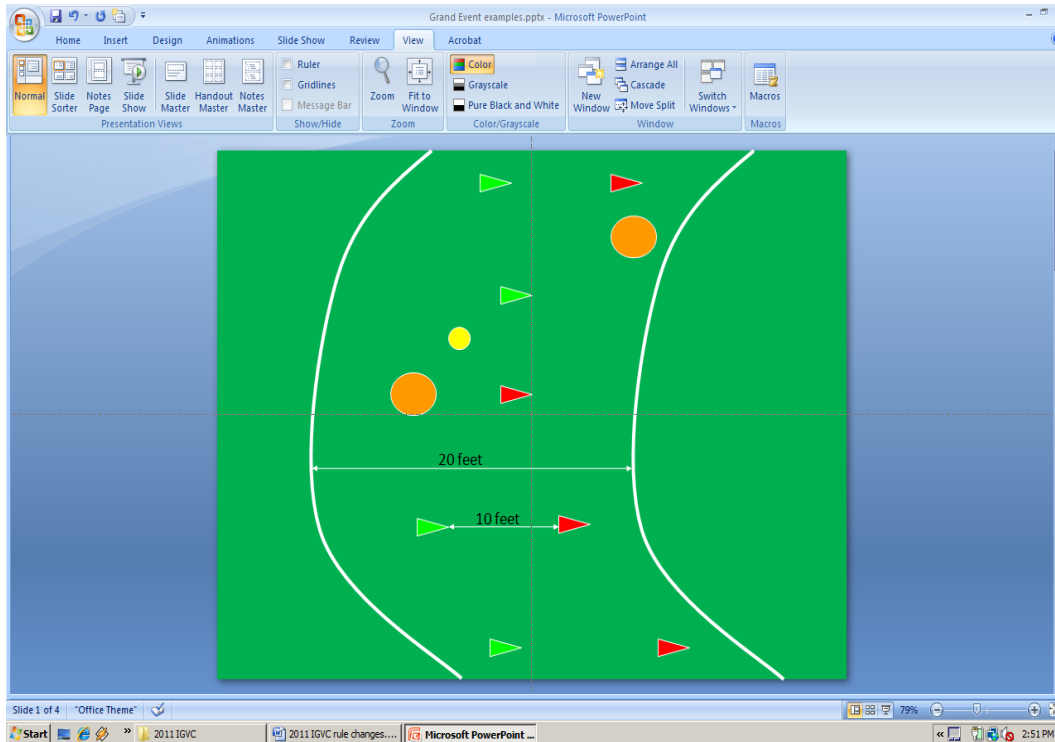
Competitors should expect natural or artificial inclines with gradients not to exceed 15% and randomly placed obstacles along the course. The course will become more difficult to navigate autonomously as vehicle progresses. Obstacles on the course will consist of various colors (white, orange, brown, green, black, etc.) of construction barrels/drums that are used on roadways and highways. Natural obstacles such as trees or shrubs and man-made obstacles such as light posts or street signs could also appear on the course. The placement of the obstacles may be randomized from left, right, and center placements prior to every run.

There will be a minimum of six feet clearance, minimum passage width, between the line and the obstacles; i.e., if the obstacle is in the middle of the course then on either side of the obstacle will be six feet of driving space. Or if the obstacle is closer to one side of the lane then the other side of the obstacle must have at least six feet of driving space for the vehicles. Also on the course there will be complex barrel arrangements with switchbacks and center islands. These will be adjusted for location between runs. Direction of the obstacle course will not change between heats.

Alternating red (right) flags (Grainger part no.3LUJ1) and green (left) flags (Grainger part no.3LUJ6) will be placed on the later part of the course. Flags will have a minimum passage width between them of six feet; i.e., if the flag is near the edge of the course then between the flag and the line will be six feet of driving space. Flags are not obstacles and vehicles can touch flags to increase speed and optimized route, vehicles are not allowed to go over flags. The objective is for the vehicle to stay to

the left of the red flags and to the right of the green flags. Flags can be staggered or the vehicle could be driving through a set of flags.

Example Autonomous Challenge Layout



Flag Configurations Map

Autonomous Challenge will contain eight Global Positioning System (GPS) waypoints, one in each corner (four total) and one at each intersection opening (four total). In the Autonomous Course figure above, the two pairs of navigation waypoints the team will use to go point to point either 2 to 3 or 4 to 1. The open space between the navigation waypoints will contain a mix of obstacles which must be avoided while staying with-in the course.

Teams will have to choose how they want to run the course; there will be two paths that are the same distance in length and difficulty. Teams can choose their first path but must alternate subsequent run paths. Examples of the paths would be start and go from point 6-2-3-8-7 or start and go from point 6-5-4-1-7. Teams do not have to cross through the actual (numbered) points, they are there to assist and use as the team seem fit.

II.4 COMPETITION RULES & PROCEDURES

- The competition will take place in the event of light rain or drizzle but not in heavy rain or lightning.
- Each qualified team will have the up to two runs (time permitting) in each of three heats.
- Judges/officials will assign a designated starting order. Teams will setup on-deck in that order. Failure to be on-deck will place you at the end of the order for the run and may forfeit you final (second) run in a heat based on heat time completion.
- No team participant is allowed on the course before the team's first run, and only one student team member is allowed on the course during a run. This shall in no case be the faculty advisor.

- At the designated on-deck time, the competing team will be asked to prepare their vehicle for an attempt. On-deck teams start in the order they arrive in the starting area unless they give way to another team.
- A Starting Official will call teams to the starting line. The Starting Official's direction is final. The Starting Officials may alter the order to enhance the competition flow of entries; e.g. slower vehicles may be grouped together to allow successive running of two vehicles on the course simultaneously.
- A team will have one minute in the starting point to prep the vehicle at the starting line and point out to the Competition Judges the buttons to start and stop the vehicle,
- The Competition Judge will start the vehicle by a one touch motion; i.e. pushing a remote control button, hitting the enter key of a keyboard, a left mouse click, lifting the e-stop up, flipping a toggle switch, etc. The Competition Judge will also carry the E-Stop.
- An attempt will be declared valid when the Competition Judge initiates the start signal at designated competing time. An attempt will continue until one of the following occurs:
 - The vehicle finishes the course.
 - The vehicle was E-Stopped by a judge's call.
 - The team E-Stops the vehicle.
 - Five minutes have passed after the vehicle run has started.
 - The vehicle has not started after one minute after moving to the start line or at the judges' discretion.
- Time for each heat will be strictly observed.
- Tactile sensors will not be allowed.
- Each vehicle will be given 5 minutes per attempt to complete the course, if the vehicle has not completed the course in the 5 minute time period, the attempt will ended by a judge's choice E-stop, with no additional penalty for that run.
- Each vehicle must navigate the course by remaining inside the course boundaries and navigating around course obstacles. For the following Traffic Violations, the appropriate ticket will be issued and deducted from the overall distance or time score. Refer to section II.5 Traffic Violation Laws.

II.5 TRAFFIC VIOLATION LAWS

	Traffic Violations	Ticket Value	E-Stop	Measurement
1	Hold-up Traffic	End of Run	Yes	>60 secs. to 88 ft
2	Leave the Course/Scene	- 10 Feet	Yes	Yes
3	Crash/Obstacle Displacement	- 10 Feet	Yes	Yes
4	Careless Driving	- 5 Feet	No	No
5	Sideswipe/Obstacle Touch	- 5 Feet	No	No
6	Student's Choice E-Stop	- 5 Feet	Yes	Yes
7	Judge's Choice E-Stop	0 Feet	Yes	Yes
8	Blocking Traffic	- 5 Feet	Yes	Yes
9	Loss of Payload	0 Feet	Yes	Yes
10	Wrong Side of Flag	-5 Feet	No	No
11	Run over Flag	-10 Feet	Yes	Yes
12	Too slow, did not average 1 mph	Disqualified	No	No

- **Hold-up traffic:** Must maintain 1 mph, there will be a speed check at 88 foot mark of the course, will result in end of run with time recorded
- **Leave the scenecourse:** All portions of the vehicle cross the boundary. The overall distance will be measured from the starting line to the furthest point where the final part of the vehicle crossed the boundary outside edge.
- **Crash:** The overall distance will be measured from the starting line to the collision point with the obstacle.
- **Careless Driving:** Crossing the boundary while at least some part of the vehicle remains in bounds.
- **Student E-Stop:** Student e-stop is used if the team feels that there may be damaged caused to their vehicle or they know that it is stuck and want to end their time.
- **Judge E-Stop:** The overall distance will be measured from the starting line to the front of the vehicle or where the final/furthest remaining part of vehicle if stopped, crossed the boundary outside edge.
- **Obstacle Displacement:** Defined as displacing permanently the obstacle from its original position. Rocking/Tilting an obstacle with no permanent displacement is not considered obstacle displacement.
- **Blocking Traffic:** Vehicles stopping on course for over one minute will be E-Stopped and measured.
- **Loss of Payload:** If the payload falls of the vehicle the run will be ended.
- **Wrong Side of Flag:** Vehicles must remain on the left side of red flags and the right side of green flags.
- **Run over Flag:** Vehicles drive over the top of a red or green flag will results in End of Run.
- **Too Slow:** If the vehicle does not maintain 1 mph minimum average speed limit throughout the course this run is disqualified.

II.6 PRACTICE COURSE

All teams that have qualified will be given six tokens. Each token represent one opportunity to use the Autonomous Challenge Practice Course. The course will be open daily for use from the time a team Qualifies till the start of the third heat of the Autonomous Challenge. The course will be run like the Autonomous Challenge with the same rules and similar obstacles. One token allows a maximum of six minutes (one minute at the start point and five minutes for the run) on the Autonomous Challenge Practice Course. In that time you must position your vehicle at the start, prep the vehicle for the judge to start, and can continue to run as long as you do not break any of the rules of the Autonomous Challenge. If so, your run and remaining time will be ended. All teams will still have unlimited access to the regular practice fields.

II.7 HOW COMPETITION WILL BE JUDGED

- A team of judges and officials will determine compliance with all rules.
- Designated competition judges will determine the official times, distances and ticket deductions of each entry. At the end of the competition, those vehicles crossing the finish line will be scored on the time taken to complete the course minus any ticket deductions. Ticket values will be assessed in seconds (one foot = one second) if the vehicle completes the course within the five minute run time.
- The team with the adjusted shortest time will be declared the winner.
- In the event that no vehicle completes the course, the score will be based on the distance traveled by the vehicle minus the ticket deductions. The team with the adjusted longest distance will be declared the winner.
- For standard award money consideration, entry must exhibit sufficient degree of autonomous mobility by passing the money barrel. The money barrel location is determined by the judges during the final/actual course layout. If a tie is declared between entries, the award money will be split between them.

II.8 GROUNDS FOR DISQUALIFICATION

- Judges will disqualify any vehicle which appears to be a safety hazard or violate the safety requirements during the competition.
- Intentional interference with another competitor's vehicle and/or data link will result in disqualification of the offending contestant's entry.
- Damaging the course or deliberate movement of the obstacles or running over the obstacles may result in disqualification.
- Actions designed to damage or destroy an opponent's vehicle are not in the spirit of the competition and will result in disqualification of the offending contestant's entry.

III. DESIGN COMPETITION

All teams must participate in the Design Competition.

III.1 OBJECTIVE

Although the ability of the vehicles to negotiate the competition courses is the ultimate measure of product quality, the officials are also interested in the design strategy and process that engineering teams follow to produce their vehicles. Design judging will be by a panel of expert judges and will be conducted separate from and without regard to vehicle performance on the test course. Judging will be based on a written report, an oral presentation and examination of the vehicle.

Design innovation is a primary objective of this competition and will be given special attention by the judges. Innovation is considered to be a technology (hardware or software) that has not ever been used by this or any other vehicle in this competition. The innovation needs to be documented, as an innovation, clearly in the written report and emphasized in the oral presentation.

III.2 WRITTEN REPORT

The report should not exceed 15 letter-sized pages, including graphic material and all appendices, but not including the title page. Reports will lose 5 points in scoring for each page over 15. Line spacing must be at least 1.5, with at least a 10 point font (12 is preferred). Each vehicle must have a distinct and complete report of its own (a report cannot cover more than one vehicle). Participants are required to submit four hard copies of the report and an electronic copy in PDF format on a CD; failure to submit either of these will result in **disqualification**. All reports, both for new vehicles and for earlier vehicles with design changes, must include a statement signed by the faculty advisor certifying that the design and engineering of the vehicle (original or changes) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course. The certification should also include a brief description of the areas in which changes have been made to a vehicle from a previous year. Everything must be mailed so as to arrive by **May 10, 2011**, addressed to:

**Bernard Theisen
21281 Curie Avenue
Warren, MI 48091-4316**

Written reports arriving after that date will lose 10 points in scoring for each business day late, electronic copies arriving after that date will lose 5 points in scoring for each business day late. Teams are encouraged to submit reports even several weeks early to avoid the last minute rush of preparing vehicles for the competition, and there will be no penalty for last minute changes in the vehicle from the design reported. The electronic copy of the report will be posted on the competition's web site in PDF format after the completion of the competition.

The paper should present the conceptual design of the vehicle and its components. Especially important to highlight are any unique innovative aspects of the design and the intelligence aspects of the vehicle. Also included must be descriptions of:

electronics	design planning process
electrical system	signal processing
actuators	plan for path following
software strategy	(both solid & dashed lines)
sensors	plan for control decisions
computers	system integration plan
mapping	high speed operations

Design of the lane following and obstacle detection/avoidance systems must be specifically described. Along with how the vehicle uses mapping techniques to perceive and navigate through its environment. Describe how the system uses GPS for waypoint navigation and localization.

Components acquired ready-made must be identified, but their internal components need not be described in detail. The steps followed during the design process should be described along with any use of Computer-Aided Design (CAD). How considerations of safety, reliability, and durability were addressed in the design process should be specifically described, as well as problems encountered in the design process and how they were overcome. The analysis leading to the predicted performance of the vehicle should be documented, specifically:

- Speed
- Ramp climbing ability
- Reaction times
- Battery life
- Distance at which obstacles are detected
- How the vehicle deals with complex obstacles including switchbacks and center islands dead ends, traps, and potholes
- Accuracy of arrival at navigation waypoints
- Comparison of these predictions with actual trial data is desirable.

Although cost itself is not a factor in judging (these are considered research vehicles), the report should include a cost estimate (not counting student labor) for the final product if it were to be duplicated. A breakdown of the cost by component is helpful.

The team organization and the names of all members of the design team, with academic department and class, should be included along with an estimate of the project's total number of person-hours expended.

Vehicles that have been entered in IGVC in earlier years and have not had significant changes in design are ineligible in either the design or performance events. Vehicles that have been changed significantly in design (hardware or software) from an earlier year are eligible, but will require a completely new design report (15 pages or less) treating both the old and new features, thus describing the complete vehicle as if it were all new.

Judges will score the written reports as follows:	Maximum Points
1. Conduct of the design process and team organization (including decision-making & software development)	50
2. Completeness of the documentation	50
3. Quality of documentation (English, grammar, and style)	50
4. Effective innovation represented in the design (as described above)	150
5. Description of mapping technique	100
6. Description of electronic design	100
7. Description of software strategy	150
8. Description of systems integration <u>Descriptions to include:</u> lane following, obstacle detection/ avoidance, and waypoint navigation (GPS or other)	150
9. Efficient use of power and materials	50
10. Attention given to safety, reliability, and durability	50
Total	900

III.3 ORAL PRESENTATION

The technical talk should relate the highlights of the written report described above and include any updates of the design since the written report. Audio or video tape presentations of the text are not allowed, but graphic aids may be presented by video, slide projection, computer projection, overhead transparencies, or easel charts. The presentation must be made by one or more student members of the team to the judges and other interested members of the audience and should last not more than 10 minutes. A penalty of 5 points will be assessed for each minute or fraction thereof over 11 minutes. After the presentation, judges only may ask questions for up to 5 minutes. The audience should be considered as a senior management group of generally knowledgeable engineers upon whom the project is dependent for funding and the team is dependent for their employment. Scoring will be as follows:

Judges will score the oral presentations as follows:	Maximum Points
1. Clear and understandable explanation of the innovations	50
2. Logical organization of the talk	25
3. Effective use of graphic aids	25
4. Articulation	20
5. Demonstrated simulation of vehicle control in performance events	10
6. Response to questions	10
7. Salesmanship	10
Total	150

Effective use of graphic aids includes not blocking the view of the screen by the presenter and simple enough graphics that are large enough to read (block diagrams rather than detailed circuit diagrams). Articulation refers to the clarity and loudness of speaking. Response to questions means short answers that address only the question. Salesmanship refers to the enthusiasm and pride exhibited (why this vehicle is the best).

Participants are responsible for providing their own visual aids and related equipment (the vehicle itself may be displayed). A computer-connected projector will be made available. Projectors may also be supplied by the participants.

During the oral presentation, the following question period and the examination of the vehicle, team members sitting the audience may participate by assisting the oral presenters, but at no time is the faculty advisor to participate in this part of the design competition.

III.4 EXAMINATION OF THE VEHICLE

The vehicle must be present and will be examined by the judges preferably immediately after the oral presentation or at another convenient time the time during the competition. Software is not included in this judging. Judging will be as follows:

Judges will score the vehicle examinations as follows:	Maximum Points
1. Packaging neatness, efficient use of space	20
2. Serviceability	20
3. Ruggedness	20
4. Safety	20
5. Degree of original content in the vehicle (as opposed to ready-made)	50
6. Style (overall appearance)	20
Total	150

III.5 FINAL SCORING

The number of points awarded by the individual judges will be averaged for each of the 23 judging areas above, and these results will be offered to each participating team for their edification. The total of the average scores over all 23 areas (max 1200) will be used to determine the ranking.

When two teams of judges are used (due to a large number of entries) each judging team will determine the top three winners in their group, and the resulting six contestants will participate in a runoff of oral presentations and vehicle examinations judged by all judges to determine an overall Design Winner. The six teams will be judged in random order.

For the Finals competition four criteria from the written report judging will be added to the normal oral presentation scoring shown above for preliminary judging. Thus, the Finals Oral presentation scoring will have maximum points as below:

Judges will score the final presentations as follows:	Maximum Points
1. Clear explanation of the innovations	50
2. Description of mapping technique	30
3. Description of Electronic Design	30
4. Description of Software Strategy	30
5. Description of System Integration	30
6. Logical organization of the talk	50
7. Effective use of graphic aids	25
8. Articulation	25
9. Demonstrated Simulation of Vehicle Control	10
10. Response to questions	10
11. Salesmanship	10
Total	300

The vehicle examination scoring will be the same as in the preliminary judging, as shown above.

IV. NAVIGATION CHALLENGE

All teams must pass Qualification to participate in this event.

IV.1 OBJECTIVE

Navigation is a practice that is thousands of years old. It is used on land by hikers and soldiers, on the sea by sailors, and in the air by pilots. Procedures have continuously improved from line-of-sight to moss on trees to dead reckoning to celestial observation to use of the GPS. The challenge in this event is for a vehicle to autonomously travel from a starting point to a number of target destinations (waypoints or landmarks) and return to home base, given only the coordinates of the targets in latitude and longitude.

IV.2 ON-BOARD SENSORS

It is expected that most contestants will use Differential GPS, but non-differential GPS is allowed as well as dead reckoning with compasses, gyros, and wheel odometers. Vision systems and/or sonar and laser rangefinders may be used for obstacle detection. There are a number of handheld GPS systems that connect to laptop computers available on the market for under \$200. Garmin, SkyMap/GPS, and Magellan are some; Earthmate even has one for Macintosh. These may not be convenient to integrate in vehicle control programs. However, differential GPS units are available from Hemisphere GPS, Trimble, Thales, Magellan, Garmin, NovAtel, and Starlink (and possibly others). Differential correction signals are available in the Southeast Michigan area from the U.S. Coast Guard. WAAS or any of the commercial suppliers of corrections are also allowed. The use of a base station to supplement the on board GPS unit is not permitted.

IV.3 GPS COURSE

The map in the figure below shows a typical course for the Navigation Challenge. This is a practice map for use by teams during development of their vehicle. Coordinates for the actual navigation course waypoints and the origin will be given to the contestants on **June 4, 2011** in degrees latitude and longitude, but no XY coordinates will be provided. There will be three 4 meter square start/ finish boxes (one for each heat) outside the main course in which teams will set up their vehicles before their run.

The competition course will be run on grass and will be approximately 50 by 65 meters (roughly 0.8 acre), and the total travel distance on the course will be on the order of 200 meters depending on the route chosen for the vehicles. The exact waypoint locations will be marked on the grass for use by the judges, but there will be no standup markers to indicate those positions. Construction barrels, barricades, fences, and certain other obstacles will be located on the course in such positions that they must be circumvented to reach the waypoints. These may be randomly moved between runs.

The course will be divided into two areas by a fence with a two meter wide opening located somewhere along it (no coordinates are provided). The opening will be randomly relocated along the fence at the start of each run. Waypoints south of the fence (the Valley) will have four meter diameter circles or squares around them (visible only to the judges) and waypoints north of the fence (the Mesa) will have two meter circles around them.

No team participant is allowed on the course before the team's first run, and only one team member is allowed on the course during a run. This shall in no case be the faculty advisor.

2
3
4
5
6
7
8

Example Navigation Map

IV.4 Practice Course

All teams will have access to the practice navigation course. The Navigation Challenge Practice Course will have similar obstacles to the ones on the Navigation Challenge. There will be a minimum of three practice waypoints on the course for teams to tune their system.

IV.5 RUN PROCEDURE AND SCORING

There will be three heats during the day with start and stop times the same as those in the Autonomous Challenge. It is intended that each team will be allowed up to two runs in each heat. There will be three starting boxes, so each team that manages three runs will start from each of the boxes during the course of three heats. In a first-come-first-served order teams should choose any free starting box that they have not been in before, until they have started in all three. After three runs they will begin the cycle of boxes again. The trial with the best performance will be used for scoring. Starting times will be first-come-first-served within each heat, except that teams up for their first trial will have priority over those wanting a second trial. It is unlikely that there will be time in the day for all registered teams to get six tries.

Vehicles will park in the starting box selected and have up to five minutes or until the course is clear for final adjustments before starting. Vehicles may seek the waypoints in any order, and the vehicle actually reaching the most waypoints (counting also the Start/Finish boxes) in the allotted five minute run time will be the winner. The vehicle must finish in the same box in which it started. If two or more vehicles reach the same number of waypoints, the vehicle doing so in the least time will be declared the leader. If two or more vehicles reach the same number of waypoints while stopped by the five minute rule, they will be declared tied and will share any awards.

If a vehicle (any part) fails to come within two meters of a target in the southern area of the course or one meter in the northern area, it will not be judged to have reached that target. In order to qualify for standard award money a vehicle must reach at least five waypoints (not counting the start/finish box).

IV.6 RUN TERMINATION

All runs will be terminated by an E-stop (by the students or the judges) signaled by a judge's whistle or bell, either:

- When the vehicle arrives back at the original start/finish point or enters any start/finish box any time after first starting.
- If any part of the vehicle leaves the perimeter of the field other than at a start/finish box.
- If the vehicle strikes any obstacle.
- If five minutes have elapsed since the start of the run (200 meters in six minutes is 1.2 miles per hour).
- In all cases the judges' call will be final.

V. JAUS Challenge

Participation in the JAUS Challenge is recommended.

V.1 TECHNICAL OVERVIEW

Each entry will interface with the Judge's COP providing information as specified below. The general approach to the JAUS interface will be to respond to a periodic status and position requests from the COP. This requires the support of the JAUS Transport Specification (AS5669A) and the JAUS Core Service Set (AS5710). The JAUS Transport Specification supports several communication protocols, the competition will use only the Ethernet based JUDP. The Core services required for the competition include the discovery, access control, and management services. The JAUS Mobility Service Set (AS6009) or JSS-Mobility defines the messaging to be used for position communications and waypoint based navigation.

V.2 COMMON OPERATING PICTURE

The COP will provide a high level view of the systems in operation that successfully implement the JAUS protocol as described above. This software is a simple validation, reporting and recording tool for the Judges to use while verifying student implementations of the JAUS standard. It provides a graphical display of the operational area in relative coordinates. Primitive graphics are loaded in the display of the COP to add perspective. Each reported status is displayed on the COP user interface and recorded for future reference. For competitions and systems reporting positional data, a 2-D map on the COP display is annotated with the updated position as well as track marks showing the previous position of the system for the current task.

V.3 COMMUNICATIONS PROTOCOLS

The teams will implement a wireless 802.11b/g or hardwired Ethernet (RJ-45) data link. The interface can be implemented at any point in the student team's system including the control station or mobility platform.

The Internet Protocol (IP) address to be used will be provided at the competition. For planning purposes, this address will be in the range of 192.168.1.100 to 192.168.1.200. The Judge's COP will have both hard-wire and 802.11b/g capabilities where the IP address of the COP will be 192.168.1.42. All teams will be provided an IP address to be used during the competition. The last octet of the IP address is significant, as it will also be used as the subsystem identifier in the team's JAUS ID. The port number for all JAUS traffic shall be 3794.

V.4 JAUS SPECIFIC DATA

The JAUS ID mentioned above is a critical piece of data used by a JAUS node to route messages to the correct process or attached device. As indicated above each team will be provided an IP address in which the last octet will be used in their respective JAUS ID. A JAUS ID consists of three elements, a Subsystem ID, a Node ID and a Component ID. The Subsystem ID uniquely identifies a major element that is an unmanned system, an unmanned system controller or some other entity on a network with unmanned systems. A Node ID is unique within a subsystem and identifies a processing element on which JAUS Components can be found. A Component ID is unique within a Node represents an end-point to and from which JAUS messages are sent and received. The last octet of the assigned IP address will be used as the team's JAUS Subsystem ID. So for the team assigned the IP address of 192.168.1.155, the completed JAUS ID of the position-reporting component might be 155-1-1 where the node and component are both assigned the IDs of 1. This is shown in the IP and JAUS ID Assignment Figure below. The Node ID and Component ID are discussed further in the JAUS Service Interface

Definition Language standard (AS5684). The COP software will be programmed with the assumption that all services required by the specific competition are implemented on a single component.

The image shows two overlapping windows. The left window is a PDF viewer displaying a document titled 'V.5 COMPETITION TASK DESCRIPTION'. It contains a diagram for 'IP and JAUS ID Assignment' showing an IP address structure (192, 168, 1, 155) and a subsystem structure (155, 1, 1). The right window is a Microsoft Word document titled 'Rules 2010.docx'. It shows a table of file metadata and a section titled 'V.1 Technical Overview'.

IP and JAUS ID Assignment

In summary, each team will be assigned an IP address by the judges. The last octet of that IP address will be the team's subsystem identifier. The COP will be a subsystem as will each team's entry in the competition. The COP will have a JAUS ID of 42:1:1 and an IP address of 192.168.1.42. The port number shall be 3794.

V.5 COMPETITION TASK DESCRIPTION

Messages passed between the COP and the team entries will include data as described in the task descriptions below. The COP will initiate all requests subsequent to the discovery process described as Task 1. A system management component is required of all teams. This interface will implement several of the messages defined by the Management Service defined in the JSS-Core. This service inherits the Access Control, Events and Transport services also defined by the JSS-Core document. The implementation of the Access Control interfaces will be necessary to meet the JAUS Challenge requirements; however no messages from the Events service will be exercised. The sequence diagram in Discovery and System Management Figure shows the required transactions for discovery including the access control setup and system control protocol. This interaction is required for every task.

The judges will evaluate each team's ability to meet the Interoperability Challenge for the tasks described below in accordance with the scoring chart.

Judges will score the task as follows:	Maximum Points
1. Transport Discovery	10
2. Capabilities Discovery	10
3. System Management	10
4. Velocity State Report	10
5. Position and Orientation Report	10
6. Waypoint Navigation	10
Total	60

V.6 TRANSPORT DISCOVERY

For any two elements in the system to communicate meaningful data there must first be a handshake to ensure both sides use the same protocols and are willing participants in the interaction. For the sake of simplicity, the team's entry shall initiate the discovery protocol with the Judge's COP, and the IP address and JAUS ID of the COP shall be fixed. The IP address and JAUS ID of the Judge's COP are defined as:

COP IP ADDRESS: 192.168.1.42:3794
 COP JAUS ID: 42-1-1 (Subsystem-Node-Component)

The discovery process, in Discovery and System Management Figure, will occur at the application layer. The student team's JAUS element will send a request for identification to the COP once every 5 seconds. The COP will respond with the appropriate informative message and request identification in return from the team's JAUS interface. After the identification report from the COP, the team entry will stop repeating the request. This transaction will serve as the basic discovery between the two elements.

The COP software will be programmed with the assumption that all services required by the specific competition are provided at the single JAUS ID. Furthermore, as per the AS5669A Specification, the team's entry shall receive JUDP traffic at the same IP address and port number that initiated the discovery protocol. Teams should note that this is different from common UDP programming approaches in which the outbound port for sent messages is not bound.

The diagram shows the following interactions:

- Transport Discovery:** Entry sends 'Query Identification' to COP. COP responds with 'Report Identification'. Note: 'The entry will request Identification every 5 seconds until it is received.'
- Capabilities Discovery:** Entry sends 'Query Services' to COP. COP responds with 'Report Services'. Note: 'The COP will request a listing of supported services at the entry's interface.'
- System Management:** Entry sends 'Query Control', 'Report Control', 'Request Control', and 'Confirm Control' to COP. COP responds with 'Query Status' and 'Report Status'. Note: 'The COP will get control of the entry and then query the Status of the entry every 5 seconds until task termination. At any time, the judge may change the state of the entry using the Resume or Standby messages.'
- System Management (Termination):** Entry sends 'User Cmd' to COP. COP responds with 'Shutdown'. Note: 'The COP will query various data in accordance with the tasking until the judge terminates the JAUS part of the mission at which point the interface is shutdown.'

Additional text in the document includes: 'System operation continues. Refer to specific sections for details of the Velocity State Report, Position & Orientation Report, and Waypoint Navigation tasks.'

The document also contains sections for 'V.1 Technical Overview' and 'V.2 Communications Protocols'.

Discovery and System Management

The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Query Identification	Report Identification	N/A

V.7 CAPABILITIES DISCOVERY

Following the completion of the Transport Discovery handshake the COP will query the entry for its capabilities. The Query Services message and Report Services message are defined in the AS5710 document and require the inheritance of the Transport service. The COP will send a Query Services message to a student team entry. Upon receipt of the message the student team entry shall respond with a properly formed Report Services message.

The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Query Identification	Report Identification	N/A

V.8 SYSTEM MANAGEMENT

The implementation of the status report is required. This interoperability task, like the discovery tasks above, is also a prerequisite for all other tasks. The task begins with the discovery handshake as described above and continues for an indeterminate period of time. The protocol is given in Discovery and System Management Figure. The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Query Control	Report Control	N/A
Request Control	Confirm Control	N/A
Query Status	Report Status	N/A
Resume	<none>	N/A
Standby	<none>	N/A
Shutdown	<none>	N/A

V.9 VELOCITY STATE REPORT

In the Velocity State Report task the COP will query the entry for its current velocity state. The COP will send a Query Velocity State message to a student team entry. Upon receipt of the message the student team entry shall respond with a properly formed Report Velocity State message.

The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Query Velocity State	Report Velocity State	Velocity X, Yaw Rate & Time Stamp [320 Decimal, 0140h]

V.10 POSITION AND ORIENTATION REPORT

For performing the task Position and Orientation Report, the discovery and status protocols described above are also required. In addition to the COP queries for status, the vehicle systems will also be required to respond correctly to local position queries. The reports will be validated for relative position and with respect to a relative time offset to ensure the time contained within each position report is valid with respect to some timer within the entry's system. In other words, the position reports must show that the travel occurred at a reasonable speed and not instantaneously. Additional variation in the position reporting using the available presence vectors is allowed. Minimally, all entries must report X, Y and Time Stamp.

The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Set Local Pose	<none>	X, Y & Yaw [67 Decimal, 0043h]
Query Local Pose	Report Local Pose	X, Y & Time Stamp [259 Decimal, 0103h]

V.11 WAYPOINT NAVIGATION

The team entry shall implement the Local Waypoint List Driver service from the JAUS Mobility Service Set (AS6009). From a starting point in the JAUS challenge test area the student entry will be commanded to traverse, in order, a series of 4 waypoints. Time will be kept and will start at the moment that the student entry exits the designated start box. Upon leaving the start box the student entry will proceed to the first waypoint in the list. Upon satisfactorily achieving each waypoint the team will be credited with 2.5 points. Time is kept for each waypoint achieved. The shortest overall time taken to achieve this task will determine the winner in the event of a tie.

The following table shows the messages sent from the COP to the team's entry, along with the expected response and minimal required fields to be set using the presence vector (PV) if applicable, required to complete this portion of the challenge:

Input Messages	Expected Response	Required Fields (PV)
Set Element	Confirm Element Request	N/A
Query Element List	Report Element List	N/A
Query Element Count	Report Element Count	N/A
Execute List	<none>	N/Speed (value of 1)
Query Active Element	Report Active Element	N/A
Query Travel	Report Travel Speed	N/A
Query Local Waypoint	Report Local Waypoint	X & Y (value of 3)

VI. AWARDS AND RECOGNITION

All schools are only eligible to win award money once per event (Autonomous Challenge, Design Competition, Navigation Challenge and JAUS Challenge); if more than one team from the same school places in the same event, only the highest placing team will be placed in a standing and receive money for that event.

VI.1 AUTONOMOUS CHALLENGE COMPETITION

Autonomous Competition Standard Awards

1 ST Place	\$25,000
2 ND Place	\$5,000
3 RD Place	\$4,000
4 TH Place	\$3,000
5 TH Place	\$2,000
6 TH Place	\$1,000

Nominal Award Money (Vehicle did not pass Money Barrel)

1 ST Place	\$3,000
2 ND Place	\$2,000
3 RD Place	\$1,000
4 TH Place	\$ 750
5 TH Place	\$ 500
6 TH Place	\$ 250

VI.2 VEHICLE DESIGN COMPETITION

Design Competition Standard Awards

1 ST Place	\$3,000
2 ND Place	\$2,000
3 RD Place	\$1,000
4 TH Place	\$ 750
5 TH Place	\$ 500
6 TH Place	\$ 250

Nominal Award Money (Vehicle did not pass Qualification)

1 ST Place	\$ 600
2 ND Place	\$ 500
3 RD Place	\$ 400
4 TH Place	\$ 300
5 TH Place	\$ 200
6 TH Place	\$ 100

VI.3 NAVIGATION CHALLENGE COMPETITION

Navigation Competition Standard Awards

1 ST Place	\$5,000
2 ND Place	\$4,000
3 RD Place	\$3,000
4 TH Place	\$2,000
5 TH Place	\$1,000
6 TH Place	\$ 500

Nominal Award Money

(Did not make 7 waypoints)

1 ST Place	\$1,000
2 ND Place	\$ 800
3 RD Place	\$ 600
4 TH Place	\$ 400
5 TH Place	\$ 300
6 TH Place	\$ 200

VI.5 JAUS CHALLENGE

JAUS Competition Standard Awards

1 ST Place	\$4,000
2 ND Place	\$3,000
3 RD Place	\$2,000
4 TH Place	\$1,000
5 TH Place	\$ 750
6 TH Place	\$ 500

Nominal Award Money

(Vehicle did not pass Qualification)

1 ST Place	\$ 600
2 ND Place	\$ 500
3 RD Place	\$ 400
4 TH Place	\$ 300
5 TH Place	\$ 200
6 TH Place	\$ 100

VI.5 ROOKIE-OF-THE-YEAR AWARD

The Rookie-of-the-Year Award will be given out to a team from a new school competing for the first time ever or a school that has not participated in the last five competitions (for this year the team would be eligible if they haven't competed since the thirteenth IGVC in 2005). To win the Rookie-of-the-Year Award the team must be the best of the eligible teams competing and perform to the minimum standards of the following events. In the Design Competition you must pass Qualification, in the Autonomous Challenge you must pass the Rookie Barrel and in the Navigation Challenge you must make three waypoints. The winner of the Rookie-of-the-Year Award will receive \$1,000 in award money; in the case the minimum requirements are not met the best of the eligible teams competing will receive \$500.

VI.6 GRAND AWARD

The Grand Award trophies will be, presented to the top three teams that perform the best overall (combined scores per below), in all three competitions. For each competition, points will be awarded to each team, below is a breakdown of the points:

Autonomous Challenge	Passed Money Barrel	Short of Money Barrel
First Place	48	24
Second Place	40	20
Third Place	32	16
Fourth Place	24	12
Fifth Place	16	8
Sixth Place	8	4

Design Competition	Vehicle Qualified	Vehicle Failed to Qualify
First Place	24	12
Second Place	20	10
Third Place	16	8
Fourth Place	12	6
Fifth Place	8	4
Sixth Place	4	2

Navigation Challenge	Completed 7 Waypoints	Short of 7 Waypoints
First Place	36	12
Second Place	30	10
Third Place	24	8
Fourth Place	18	6
Fifth Place	12	4
Sixth Place	6	2

J AUS Competition	Vehicle Qualified	Vehicle Failed to Qualify
First Place	24	12
Second Place	20	10
Third Place	16	8
Fourth Place	12	6
Fifth Place	8	4
Sixth Place	4	2

VI.7 PUBLICATION AND RECOGNITION

International recognition of all participating teams through AUVSI and SAE publications.

Student Teams are Invited to Display Their Vehicles at The Association for Unmanned Vehicle Systems International's Unmanned Systems North America 2011 Symposium & Exhibition Held at Washington Convention Center in Washington, District of Columbia on August 16TH – 19TH, 2011

All teams are invited to display the winning vehicles in the AUVSI exhibit halls.

Videos of the competition event will be distributed to sponsors, media and the public. All design reports, articles, videos and pictures will be post on the IGVC website www.igvc.org.

If you have any questions, please feel free to contact any of the following IGVC Officials:

IGVC Co-Chairs:

Bill Agnew	General Motors (retired)	agnew26@comcast.net
Ka C Cheok	Oakland University	cheok@oakland.edu
Jerry R. Lane	SAIC	gerald.r.lane@saic.com

Autonomous Challenge Lead Judges:

Jerry R. Lane	SAIC	gerald.r.lane@saic.com
Ka C Cheok	Oakland University	cheok@oakland.edu

Design Competition Lead Judge:

Bill Agnew	General Motors (retired)	agnew26@comcast.net
------------	--------------------------	--

Navigation Challenge Lead Judges:

Jeff Jaczkowski	PEO GCS RS JPO	jeffrey.jaczkowski@us.army.mil
Chris Mocnik	U.S. Army TARDEC	chris.mocnik@us.army.mil

JAUS Challenge Lead Judge:

Woody English	DeVivo AST	woodyenglish@devivoast.com
---------------	------------	--

Administrative:

Gerald C. Lane	Oakland University	geraldclane@comcast.net
----------------	--------------------	--

Director of Operations:

Bernard Theisen	U.S. Army TARDEC	bernard.theisen@us.army.mil
-----------------	------------------	--

<u>Name</u>	<u>Years as Editor</u>
Bernard Theisen	2006-2011
Greg Gill	2005-2006
Bernard Theisen	2004-2005
Dan Maslach	2003-2004
Bernard Theisen	2001-2003
Stephen W. Roberts	2000-2001
Scot Wheelock	1999-2000
Geoff Clark	1998-1999
G. Edzko Smid	1997-1998
Candy McLellan and G. Edzko Smid	1996-1997
Jerry Lane, Paul Lescoe and Ka C. Cheok	1992-1996

IGVC Rules Editors

January 24, 2011 Version

The Eight Annual Robotic Lawnmower Competition Rulebook Revision 2011.1.0

June 2-4, 2011
Dayton, Ohio



Sponsored by:



The Institute of Navigation-
Satellite Division



Air Force Research Laboratory-
Sensors Directorate

1 GENERAL OVERVIEW3

1.1 GENERAL TEAM RULES.....4

1.2 BEST CUT AWARD4

1.3 VIDEO QUALIFICATIONS5

1.4 IMPORTANT DATES5

1.5 DOCUMENT REVISION HISTORY:.....5

2 CATEGORY I: BASIC MOWING FIELD AUTONOMOUS LAWNMOWER CONTEST6

2.1 OVERVIEW6

2.2 RULES & REGULATIONS.....6

 2.2.1 Mower Design.....6

 2.2.2 Safety / Autonomous Operation Check6

 2.2.3 Mowing7

2.3 FIELD DESCRIPTION7

2.4 SCORING8

 2.4.1 Report Scoring8

 2.4.2 Technical Presentation Scoring.....9

 2.4.3 Mowing Competition Scoring9

 2.4.4 Total Scoring10

2.5 PRIZES.....11

3 CATEGORY II: ADVANCED AUTONOMOUS MOWING FIELD CONTEST.....12

3.1 OVERVIEW12

3.2 RULES & REGULATIONS.....12

 3.2.1 Mower Design.....12

 3.2.2 Safety / Autonomous Operation Check12

 3.2.3 Mowing13

3.3 FIELD DESCRIPTION13

3.4 MOVING OBSTACLE DESCRIPTION15

3.5 FLOWER BED DESCRIPTION15

3.6 SCORING15

 3.6.1 Report Scoring16

 3.6.2 Technical Presentation Scoring.....16

 3.6.3 Mowing Competition Scoring17

 3.6.4 Total Scoring18

3.7 PRIZES.....18

APPENDIX A, STATIC OBSTACLE DESCRIPTION.....19

APPENDIX B, DYNAMIC OBSTACLE DESCRIPTION20

APPENDIX C, PICKET FENCE DETAILS.....21

1 General Overview

The Institute of Navigation (ION) announces the Seventh Annual Robotic Lawnmower Competition sponsored by the ION's Satellite Division, Air Force Research Laboratory.

The purpose of this competition is to design and operate an autonomous unmanned lawnmower using the art and science of navigation to rapidly and accurately mow a field of grass.

This year the competition will consist of the following two categories: Basic Autonomous Mowing and Advanced Autonomous Mowing. The Basic Mowing Competition field will be rectangular with one static (non-moving) obstacle. The Advanced Mowing Competition field will have a non square shape and contain both static and moving obstacles as well as a border such as a fence. Teams are permitted to compete in either the basic or the advanced course- schools are permitted to have more than one team. Given favorable field conditions, a separate award will be awarded to the team which cuts the field with the "best cut." Details on the best cut award can be found in section 1.2.

Changes for 2011!

- *Video Qualifications added! Videos of an operating mower due 21 April 2011. Details can be found in section 1.3.*
- *We now have a facebook page! (<http://www.facebook.com/pages/Institute-of-Navigation-Robotic-Lawn-Mower-Competition/125482774151756>)*

The competition will take place 2-4 June, 2011 - a Thursday, Friday, and Saturday (with Sunday as a rain date), and will be held at Siebenthaler's Beaver Valley Garden Center near Dayton, OH. Presentations shall be given Thursday afternoon at **TBD** in Dayton, OH; the safety and autonomous operation inspections shall be Friday; and the mowing competition shall take place on Saturday with Sunday as a rain date.

Teams competing in the Basic and Advanced categories will be scored separately. The winning teams for both categories are based on the highest total score. Scoring details for each category in the mowing competition are provided in their respective sections. There will be cash awards presented to the top three teams in each category. In addition, special international recognition will be provided for all participating teams through ION publications and announcements at the 2011 ION Global Navigation Satellite Systems (GNSS) conference. The winning teams will also be given an invitation to display their lawnmowers during the ION GNSS, and given the opportunity to make a 15-minute presentation on their robotic lawnmower design. Finally, videotapes of the competition event will be distributed to sponsors, media, and the public.

1.1 General Team Rules

1. Teams may be comprised of undergraduate and/or graduate students, and must be supervised by at least one faculty advisor. Interdisciplinary teams are encouraged (EE, ME, CS, etc.). Only the student component of each team will be eligible for the awards. The faculty supervisor will certify that all team members are students on the application form. Business/Non-Engineering students are encouraged to join teams to promote marketing, sponsorships, and other program management functions.
2. Team sponsors are encouraged. Sponsors' participation will be limited to hardware donation and/or funding support. Sponsors' logos may be placed on the lawnmower and may be displayed inside the team's maintenance area. Teams should encourage sponsor attendance.
3. A non-secured sheltered maintenance area will be provided Thursday, Friday and Saturday nights.
4. Schools may have more than one entry. Each entry must be based on a different mower (chassis & electronics), different team, and must be documented by a separate application forms and team reports. Each form must be accompanied with a \$200.00 non-refundable registration fee made payable to: ION, Dayton Section. This fee includes up to five ION student memberships.
5. The Application Form must contain an **Indemnification Agreement** executed by an individual from the team's sponsoring institution who has authority to bind the institution for which he or she signs. Additionally, the Team's sponsoring institution is required to supply the Dayton Section of the Institute of Navigation with a Certificate of Insurance at the time the Application Form is submitted. The certificate must show commercial general liability coverage in an amount not less than \$1 million.
6. Intention to compete must be received no later than **April 21, 2011** by mailing your application form to Lisa Beaty, Institute of Navigation, 8551 Rixlew Lane, Suite 360, Manassas, VA 20109. To be entered into the competition, the application must include a link to a YouTube (or equivalent) video containing the qualification run. Details on the qualification run can be found in section 1.3.

1.2 Best Cut Award

The Best Cut Award will be presented to the team who cuts as least 75% of their field and has the most aesthetically pleasing cut. Since the definition of "aesthetically pleasing" is subjective, professionals from the lawn-care business will be on hand to do the judging. A cash award will be given to the team whose field is deemed "best cut." The value of the award is dependent on the sponsorship of the competition and favorable field conditions the day of the competition- please refer to the official Robotic Lawnmower Webpage (www.automow.com) or our facebook page (<http://www.facebook.com/pages/Institute-of-Navigation-Robotic-Lawn-Mower-Competition/125482774151756>) for the latest award amount information.

1.3 Video Qualifications

Participation in the competition now requires a video qualification. Video qualifications are due at the time of application. Upon qualification team will be invited to participate in the competition. Entry fees will be refunded for teams who are not able to pass the video qualification. Qualification videos can be uploaded to YouTube and the link to the qualification video must be included in the entry application. To qualify for the competition, the following criteria must be demonstrated in the video:

- ✓ Robotic Mower must operate under its own power (e.g. not with an extension cord or pulled by a rope) via radio control or autonomously for 3 minutes in the grass while demonstrating the ability to drive forward and turn.
- ✓ While operating via radio control or autonomously the main mower blade must be cutting the grass (which should be evident in the video). Considerations will be given for teams located in climates where grass hasn't started growing by mid-April (such as Nome, AK).

And, while no extra credit is given for creativity- it is always welcome ☺.

1.4 Important Dates

- April 21, 2011 - Application forms and **Qualification Video** due
- May 12, 2011 - Reports are due by 5:00 PM
- June 2, 2011 - Presentations in the evening
- June 3, 2011 - Robotic lawnmower safety and autonomous operation check
- June 4, 2011 - Mowing competition
- June 5, 2011 - Rain date for competition

1.5 Document Revision History:

- Revision 2011.1.0
 - Baseline Document

2 Category I: Basic Mowing Field Autonomous Lawnmower Contest

2.1 Overview

Design and operate an autonomous unmanned lawnmower using the art and science of navigation to rapidly and accurately mow a rectangle field of grass with static (non-moving) obstacles. The teams are placed based on their total scores; 80% of the total score is based on the mowing competition; 20% of the total score is based on the presentation and report.

2.2 Rules & Regulations

2.2.1 Mower Design

1. Lawnmowers shall be autonomous and unmanned and shall not be remotely controlled during the competition. Remotely controlled includes but is not limited to: commands to reset the mowers computers, commands to reinitialize the mower, commands to adjust a mowing route, etc.
2. For safety, a maximum lawnmower speed of 10 km/hr shall be enforced.
3. The lawnmower shall be equipped with both a manual and a wireless (radio frequency) remote emergency stop capability. The wireless emergency stop shall be effective for the entire field of operation plus 10 m in all directions. The manual emergency stop shall be easily accessible by a standing operator behind the lawnmower, and shall be **red** in color and have a diameter of at least **40 mm**. After the initiation of an emergency stop, the mowing function shall cease within 3 s and the lawnmower shall be stopped within a distance of 2 m. Lawnmowers that are determined to be unsafe by the judges shall not be operated in the competition.
4. Mower may not exceed 2 m in any dimension
5. Lawnmower movement shall be accomplished through direct contact with the ground. Power shall either be provided by combustible fuel, batteries, or both. Other power sources should be cleared with the judging officials prior to the competition.

2.2.2 Safety / Autonomous Operation Check

The safety check is conducted to test the functionality of the lawnmower manual and wireless emergency stop, and verify that the top speed of the lawnmower is below 10 km/h. Lawnmowers that fail to meet these requirements before the designated competition start time will be disqualified by the Safety Officials. Teams may fine tune their lawnmowers and resubmit for inspection and/or safety test. After passing the safety checks, changes to maximum speed control software and hardware are not allowed.

All entries are required demonstrate autonomous operation to compete in the mowing competition. The lawnmowers must demonstrate the ability to mow a **predetermined** path void of any obstacles. The shape and complexity of the predetermined path should be chosen by the teams and approved by the Safety Official. At a minimum the predetermined path shall include 3 turns at predetermined locations, and the path shall not exceed an area with 10 m by 10 m dimensions. In the event of any conflict, the judges' decision is final.

2.2.3 Mowing

1. The competitors will be required to start autonomous operation in the safety buffer and mow in the cutting zone. Teams may choose any location in the safety zone to start their run.
2. If any part of the lawnmower is outside the safety buffer (2 m in any direction outside the field of operation), the emergency stop shall be activated, and the run terminated.
3. The lawnmower shall start operation within 5 minutes after the assigned start time. The timer will be started from zero when the lawnmower crosses the start line. The team shall declare completion of the mowing operation.
4. Teams are permitted restarts. If a team chooses to restart a penalty will be assessed as detailed in the scoring section. If a restart is requested, teams will be given 1 hour to prepare for next attempt, and as with the initial run, teams may choose any location in the safety zone to restart their run.
5. Teams have a maximum of 20 minutes to cut the field. The 20 minutes is total cutting time, i.e. it includes the initial run and possible restart.
6. The mowers should be designed to operate in any weather condition. In the event of inclement weather the competition may be postponed. Teams may not work on their mowers while the event is postponed. The decision to postpone shall be made by the judges.

2.3 Field Description

The cutting field will be approximately 10 m by 15 m in size surrounded by a 2 m safety buffer as illustrated in Figure 2-1. The boarder of the fields will be marked with painted lines. The line separating the cutting field from the safety field will be white, and the outside of the safety buffer will either be yellow or white. The field will contain an obstacle. Diagrams of the obstacles can be found in Appendix A.

The quality of the cutting fields may vary, and each team's field will be chosen at random during the presentations. When designing their mower, teams should anticipate a rough mowing field

which may be wet. Field inclines will not exceed 1 in 10 m and the GPS mask angle shall not exceed 35 degrees in any direction. Utility poles, utility wires, and goal posts may exceed the 35 degree GPS mask angle.

Teams will have access to designated practice fields both Friday and Saturday of the competition. Teams will not be permitted to enter their assigned competition field prior to their assigned competition start time to preserve the quality of the grass.

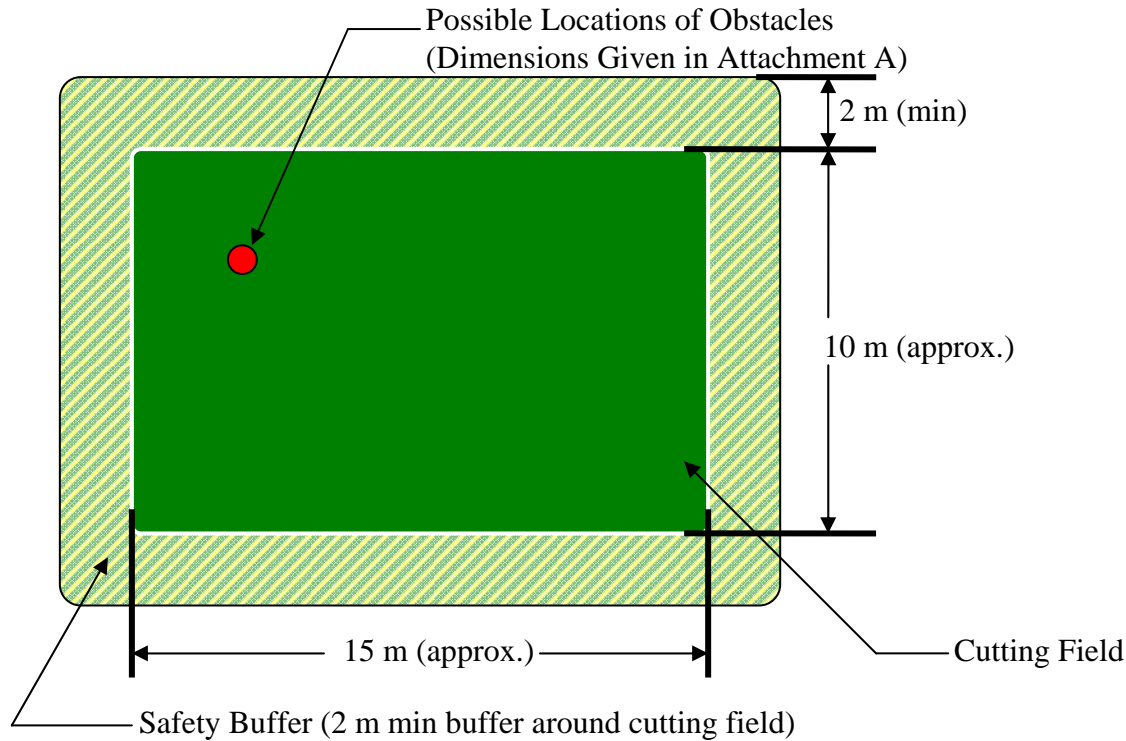


Figure 2-1, Lawn Mowing Field for the Basic Category. NOTE: Field size at the time of competition may vary.

2.4 Scoring

Competitors are placed based on their total score in the competition. The total score is a composite of the points earned on the team’s report, the team’s presentation, and the mowing competition. The individual components and the total scoring algorithms are given in this section.

2.4.1 Report Scoring

Each team shall submit a report no later than 5:00 PM on Thursday, May 12, 2011. The report shall be submitted electronically to donald.venable@wpafb.af.mil If the size of the report

exceeds size permitted by recipient's mailbox (5 MB) other arrangements should be made *before* the deadline. Late reports will be assessed a penalty of 10 points per work day from the total point score of 100.

The report shall contain a summary of the lawnmower and associated navigation design. Teams are encouraged to submit their report at the earliest opportune time. Some small changes to the final lawnmower design after the report has been submitted shall be tolerated.

The report scoring is out of 100 points and shall be as follows:

1. Overview of team organization and lawnmower design	10
2. Quality of documentation (Grammar and Style)	10
3. Effective innovation of overall lawnmower design	25
4. Completeness of overall system specifications- for example:	10
a. Cost	
b. Max speed	
c. Cutting width	
d. Dimensions	
e. Gasoline (or other energy source) usage (i.e. gallons per sqr yard cut or gallons per hr)	
f. others	
5. Description of electronic design	10
6. Description of software strategy	10
7. Description of systems integration	15
8. Attention given to safety, reliability, and durability	10
TOTAL	100

2.4.2 Technical Presentation Scoring

Each team shall give a technical presentation on Thursday, 2 June 2011. The presentation should be 15 minutes in length. A projector and computer with Microsoft PowerPoint shall be provided. Teams are not required to use the provided computer but if they do, it is recommended they email the presentation to donald.venable@wpafb.af.mil for loading, and bring a backup copy on either a CD or on a memory stick.

The technical presentation scoring is out of 100 points and shall be as follows:

1. Clarity of presentation (intro, body, summary)	40
2. Level of relevance to the technical report	40
3. Ability to capture the audience's interest	20
TOTAL	100

2.4.3 Mowing Competition Scoring

The mowing competition scoring has been greatly simplified this year. No longer will the scoring be weighted by the time required to cut or the mowing blades width. The score will be based only on the percentage of grass mowed in the cutting area with penalties assigned for grass cut in the safety buffer and for impact with obstacles. A 15% penalty will be assessed for each restart. Only two restarts are allowed. Teams have ten minutes after the end of each run to declare a restart. If a restart is not declared within the end of the teams run, the team's final score will be calculated and posted.

Percentage of the grass mowed will be determined by the following equation:

$$P_{GM} = \left(\frac{A_{CA} - A_{SB} - 0.10N_I}{A_{TA}} \right) \times 100\% - N_{RS} \times 15\% \quad (1)$$

where:

P_{GM} = Percentage of the grass mowed (score)

A_{CA} = Area of grass cut in the cutting area

A_{SB} = Area of grass cut in the safety buffer

A_{TA} = Total area of the cutting area

N_I = Number of Impacts which move an obstacle

N_{RS} = Number of Restarts

Cutting grass on the line which separates the cutting area with the safety buffer is considered cutting in the safety buffer and is a penalty. Touching a stationary obstacle is not considered a penalty if the stationary obstacle does not move. If a stationary obstacle is impacted with a force which causes the obstacle to visibly slide, a 10 percent penalty is assessed per impact as described in (1). Judges have the final decision on penalties assessed- there are no 'video replay' challenges permitted.

2.4.4 Total Scoring

All rulings and scoring by the judges shall be final.

Total score is a composite of the mowing competition (80%), technical report (10%), and technical presentation (10%).

In the event of a tie, the team which cut the grass in the least amount of time will be declared the winner.

2.5 Prizes

The prize amounts are based on the number of sponsors and values will be posted on the web page (www.automow.com). **To qualify for total dollar prize amounts the team must mow at least 50% of the cutting field during their competition run.** If less than 50% of the field is mowed the team will only receive 25% of the total prize.

3 Category II: Advanced Autonomous Mowing Field Contest

3.1 Overview

Design and operate an autonomous unmanned lawnmower using the art and science of navigation to rapidly and accurately mow an “L” shaped field of grass with moving and static obstacles. The teams are placed based on their total scores; 80% of the total score is based on the mowing competition; 20% of the total score is based on the presentation and report.

3.2 Rules & Regulations

3.2.1 Mower Design

1. Lawnmowers shall be autonomous and unmanned and shall not be remotely controlled during the competition. Remotely controlled includes but is not limited to: commands to reset the mowers computers, commands to reinitialize the mower, commands to adjust a mowing route, etc.
2. For safety, a maximum lawnmower speed of 10 km/hr shall be enforced.
3. The lawnmower shall be equipped with both a manual and a wireless (radio frequency) remote emergency stop capability. The wireless emergency stop shall be effective for the entire field of operation plus 10 m in all directions. The manual emergency stop shall be easily accessible by a standing operator behind the lawnmower, and shall be **red** in color and have a diameter of at least **40 mm**. After the initiation of an emergency stop, the mowing function shall cease within 3 s and the lawnmower shall be stopped within a distance of 2 m. Lawnmowers that are determined to be unsafe by the judges shall not be operated in the competition.
4. Mower may not exceed 2 m in any dimension
5. Lawnmower movement shall be accomplished through direct contact with the ground. Power shall either be provided by combustible fuel, batteries, or both. Other power sources should be cleared with the judging officials prior to the competition.

3.2.2 Safety / Autonomous Operation Check

The safety check is conducted to test the functionality of the lawnmower manual and wireless emergency stop, and verify that the top speed of the lawnmower is below 10 km/h. Lawnmowers that fail to meet these requirements before the designated competition start time will be disqualified by the Safety Officials. Teams may fine tune their lawnmowers and resubmit for inspection and/or safety test. After passing the safety and autonomous operation tests, changes to maximum speed control software and hardware are not allowed.

All entries are required demonstrate autonomous operation to compete in the mowing competition. The lawnmowers must demonstrate the ability to mow a **predetermined** path void of any obstacles. The shape and complexity of the predetermined path should be chosen by the teams and approved by the Safety Official. At a minimum the predetermined path shall include 3 turns at predetermined locations, and the path shall not exceed an area with 10 m by 10 m dimensions. In the event of any conflict, the judges' decision is final.

3.2.3 Mowing

1. The competitors will be required to start autonomous operation in the safety buffer and mow in the cutting zone. Teams may choose any location in the safety zone to start their run *as long as the first grass cut on the cutting field is in Zone 1*.
2. If any part of the lawnmower is outside the safety buffer (2 m in any direction outside the field of operation), the emergency stop shall be activated, and the run terminated.
3. The lawnmower shall start operation within 5 minutes after the assigned start time. The timer will be started from zero when the lawnmower crosses the start line. The team shall declare completion of the mowing operation.
4. Teams are permitted restarts. If a team chooses to restart a penalty will be assessed as detailed in the scoring section. If a restart is requested, teams will be given 1 hour to prepare for next attempt, and as with the initial run, teams may choose any location in the safety zone to restart their run *as long as the mower enters the field in Zone 1*.
5. Teams have a maximum of 20 minutes to cut the field. The 20 minutes is total cutting time, i.e. it includes the initial run and possible restarts.
6. The mowers should be designed to operate in any weather condition. In the event of inclement weather the competition may be postponed. Teams may not work on their mowers while the event is postponed. The decision to postpone shall be made by the judges.

3.3 Field Description

The cutting field will be a three zoned irregular shape surrounded by a 2 m safety buffer as illustrated in Figure 3-1. The boarder of the fields will be marked with painted lines. The line separating the cutting field from the safety field will be white, and the outside of the safety buffer will either be yellow or white. The field will contain one moving obstacle; details on the moving obstacle can be found in the next section. Diagrams of the moving obstacles can be found in Appendix B.

The quality of the cutting fields may vary, and each team's field will be chosen at random during the team presentations. When designing their mower, teams should anticipate a rough mowing field which may be wet. Field inclines will not exceed 1 in 10 m and the GPS mask angle shall not exceed 45 degrees in any direction. Utility poles, utility wires, and goal posts may exceed the 45 degree GPS mask angle.

Teams will have access to designated practice fields both Friday and Saturday of the competition. Teams will not be permitted to enter their assigned competition field prior to their assigned competition start time to preserve the quality of the grass.

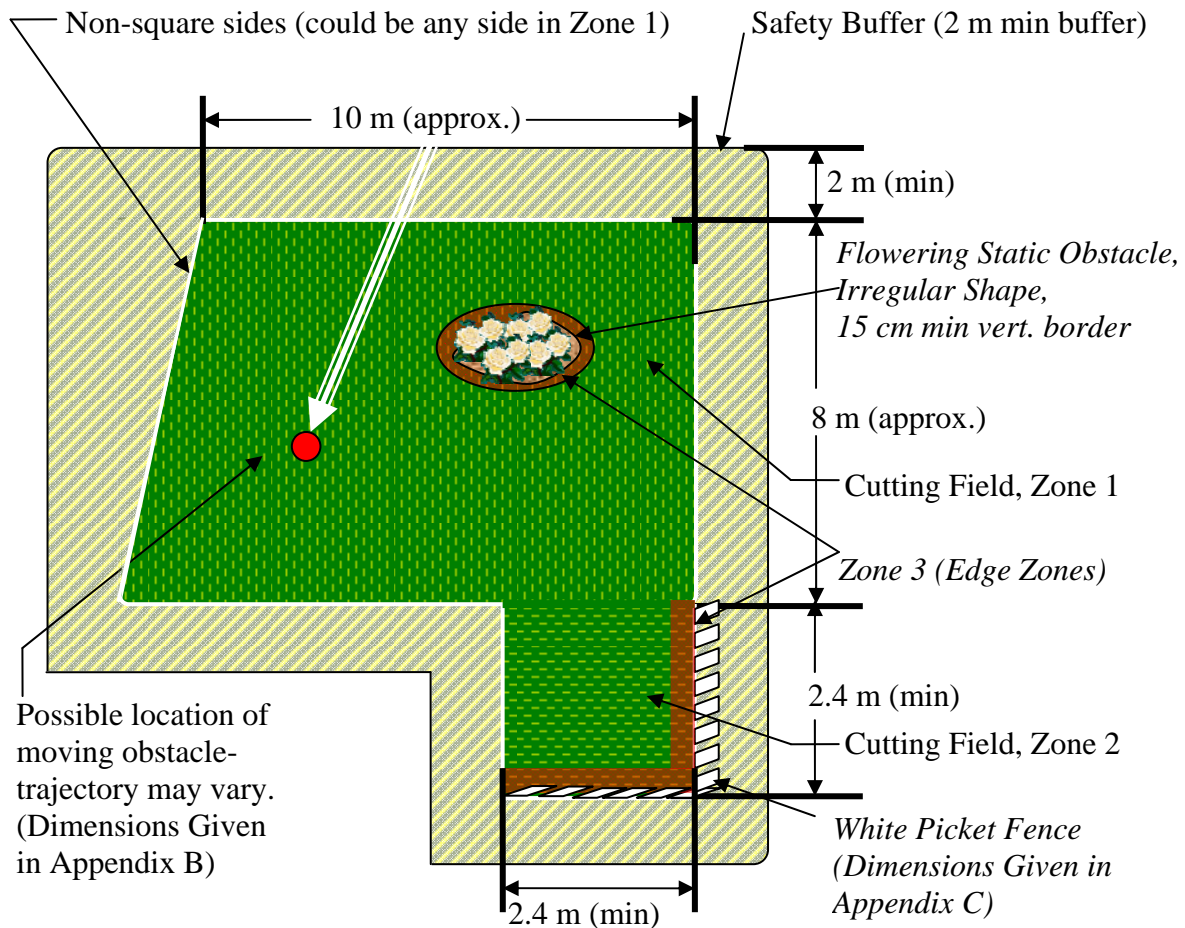


Figure 3-1, Autonomous Lawn Mowing Field for the Advanced Category. NOTE: field size, location and movement of obstacles may vary from picture.

3.4 Moving Obstacle Description

The moving obstacle will travel in a straight line towards the front (± 90 deg from mower's velocity vector) of a mower and will engage the mower once at a time chosen by the moving obstacle judge. The obstacle will move at a maximum speed of 3 m/s and its shape is described in Appendix B. The goal is to design a system which will prevent a team's mower from impacting the moving obstacle- thus, it can be assumed that the moving obstacle will not run into a team's mower (but a team's mower could run into the moving obstacle!)

3.5 Flower Bed Description

The flowerbed will be bordered with 15 cm high black plastic edging. The maximum dimension on the flower bed will not exceed 5 m, and the flowerbed will have a concave portion which has a minimum radius of 2 m. Non-concave portions of the flowerbed may contain corners. Figure 3-2 provides a notional flowerbed shape. Figure 3-3 provides a picture of the flowerbed from 2008; this year's flowerbed will be similar. There will be flowers in the flowerbed which may be taller than the plastic edging. The flowerbed will be positioned in Zone 1 of the cutting field. The position and orientation of the flower bed will not be set until 5 minutes before the start of run, and the teams may not input information to their robot which provides information on the position and orientation of the flowerbed. There will be a minimum of 2 m between all edges of the flowerbed and all edges of Zone 1 of the cutting field.

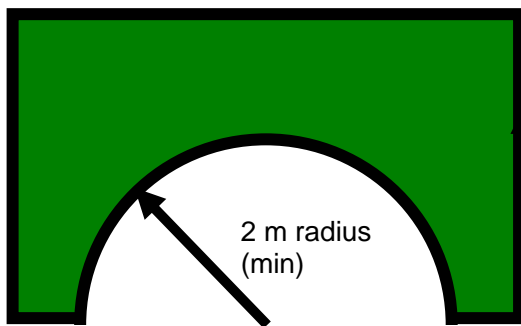


Figure 3-2, Notional flowerbed shape



Figure 3-3, 2008 Competition Flowerbed

3.6 Scoring

Competitors are placed based on their total score in the competition. The total score is a composite of the points earned on the team's report, the team's presentation, and the mowing competition. The individual components, total, and an example scoring are given in this section.

3.6.1 Report Scoring

Each team shall submit a report no later than 5:00 PM on Thursday, May 12, 2011. The report shall be submitted electronically to donald.venable@wpafb.af.mil. If the size of the report exceeds size permitted by recipient's mailbox (5 MB) other arrangements should be made *before* the deadline. Late reports will be assessed a penalty of 10 points per work day from the total point score of 100.

The report shall contain a summary of the lawnmower and associated navigation design. Teams are encouraged to submit their report at the earliest opportune time. Some small changes to the final lawnmower design after the report has been submitted shall be tolerated.

The report scoring is out of 100 points and shall be as follows:

9. Overview of team organization and lawnmower design	10
10. Quality of documentation (Grammar and Style)	10
11. Effective innovation of overall lawnmower design	25
12. Completeness of overall system specifications- for example:	10
a. Cost	
b. Max speed	
c. Cutting width	
d. Dimensions	
e. Gasoline (or other energy source) usage (i.e. gallons per sqr yard cut or gallons per hr)	
f. others	
13. Description of electronic design	10
14. Description of software strategy	10
15. Description of systems integration	15
16. Attention given to safety, reliability, and durability	10
TOTAL	100

3.6.2 Technical Presentation Scoring

Each team shall give a technical presentation on Thursday, June 2, 2011. The presentation will be 15 minutes in length.

A projector and computer with Microsoft PowerPoint shall be provided. Teams are not required to use the provided computer but if they do, it is recommended they email the presentation to donald.venable@wpafb.af.mil for loading, and bring a backup copy on either a CD or on a memory stick.

The technical presentation scoring is out of 100 points and shall be as follows:

4. Clarity of presentation (intro, body, summary)	40
5. Level of relevance to the technical report	40
6. Ability to capture the audience's interest	20

TOTAL

100

3.6.3 Mowing Competition Scoring

The mowing competition scoring has been greatly simplified this year. No longer will the scoring be weighted by the time required to cut or the mowing blades width. The score will be based only on the weighted percentage of grass mowed in the three specified cutting areas (Zone 1, Zone 2 and Zone 3 in Figure 3-1) with penalties assigned for grass cut in the safety buffer and for collision with the obstacles. A 15% penalty will be assessed for each restart. Only two restarts are allowed. Teams have ten minutes after the end of each run to declare a restart. If a restart is not declared within the end of the teams run, the team's final score will be calculated and posted.

Percentage of the grass mowed will be determined by the following equation:

$$P_{GM} = \left(0.3 \left(\frac{A_{CA_Z1}}{A_{TA_Z1}} \right) + 0.4 \left(\frac{A_{CA_Z2}}{A_{TA_Z2}} \right) + 0.3 \left(\frac{A_{CA_Z3}}{A_{TA_Z3}} \right) - \frac{A_{SB}}{A_{TA}} - 0.10N_I \right) \times 100\% - N_{RS} \times 15\% \quad (2)$$

where:

P_{GM} = Percentage of the grass mowed (score)

A_{CA_Z1} = Area of grass cut in the cutting area in Zone 1

A_{CA_Z2} = Area of grass cut in the cutting area in Zone 2

A_{CA_Z3} = Area of grass cut in the cutting area in Zone 3 (designated red edge zones)

A_{SB} = Area of grass cut in the safety buffer

A_{TA_Z1} = Total area of the cutting area in Zone 1

A_{TA_Z2} = Total area of the cutting area in Zone 2

A_{TA_Z3} = Total area of the cutting area in Zone 3 (designated red edge zones)

A_{TA} = Total area of the cutting area in Zone 1, 2 & 3 $\therefore A_{TA} = A_{TA_Z1} + A_{TA_Z2} + A_{TA_Z3}$

N_I = Number of time the team's lawnmower collides with moving or static obstacle

N_{RS} = Number of Restarts

As defined in (2), grass cut in Zone 1 (the largest zone) accounts for 30% of the mowing score, the grass cut in the smaller Zone 2 accounts for 40% of the mowing score, and the grass cut in Zone 3 accounts for 30% of the mowing score. Grass cut in the safety buffer penalizes the total score by percentage of area cut in the safety area scaled by the total cutting area. In addition, collisions with the moving obstacle will reduce the score by 10% for each collision.

Cutting grass on the line which separates the cutting area with the safety buffer is considered cutting in the safety buffer and is a penalty. The edge zones (Zone 3) will be 30 cm in width. Touching a stationary obstacle is not considered a penalty if the stationary obstacle does not

move and is not damaged. If a stationary obstacle is impacted with a force which causes the obstacle to visibly slide, a 10 percent penalty is assessed per impact as described in (2). Judges have the final decision on penalties assessed- there are no 'video replay' challenges permitted.

3.6.4 Total Scoring

All rulings and scoring by the judges shall be final.

Total score is a composite of the mowing competition (80%), technical report (10%), and technical presentation (10%).

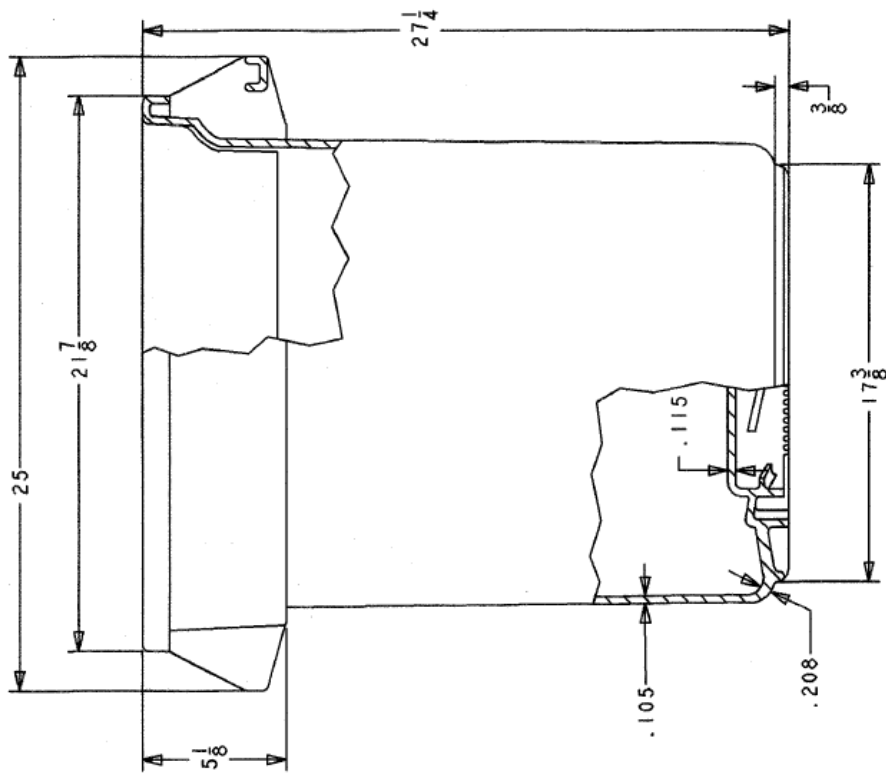
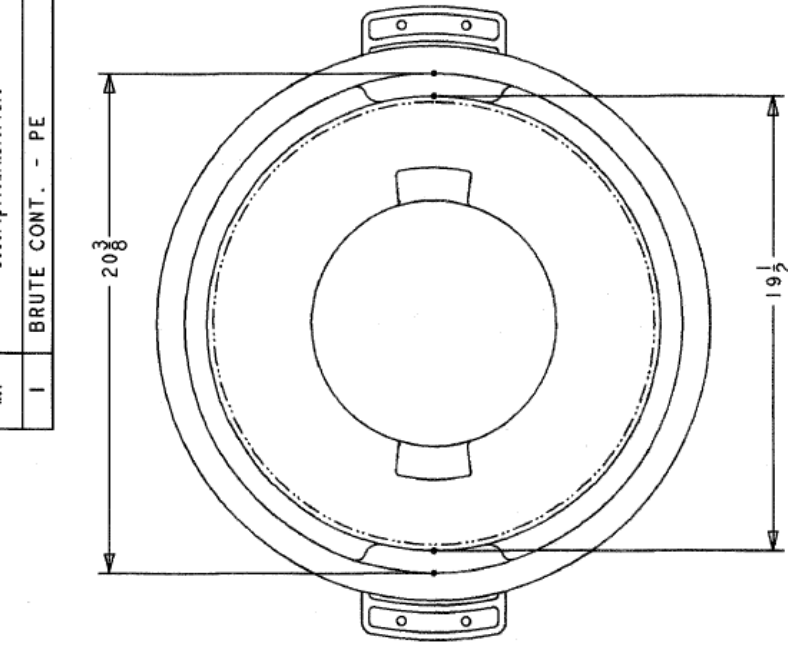
In the event of a tie, the team which cut the grass in the least amount of time will be declared the winner.


3.7 Prizes

The prize amounts are based on the number of sponsors and values will be posted on the web page (www.automow.com). **To qualify for total dollar prize amounts the team must mow at least 50% of the cutting field during their competition run.** If less than 50% of the field is mowed the team will only receive 25% of the total prize.

Appendix A, Static Obstacle Description

No.	Description/Materials
I	BRUTE CONT. - PE



product no. 2632	
Product Specifications 	
total part wt. 8.0 LBS.	NSF STD. #2 & #21 USDA MEAT & POULTRY, CSFM WHEN USED WITH 2637
capacity 32 GALS.	agency approvals _____ miscellaneous _____
	product name BRUTE CONTAINER
	product no. 2632 revision 6

Appendix B, Dynamic Obstacle Description

The moving obstacle will be a radio controlled truck with stuffed dog attached on top. The feet of the stuffed dog will be no more than 4 inches (~ 10 cm) above ground level.

The radio controlled dog will be driven in front (+/- 90 deg of the mowers' velocity vector) of the teams' mowers once during their run at a time determined by the "driver" of the dog.

The dog will stop in front of the mower for 30 seconds.

Moving Obstacle Specifications:

Melissa and Doug Lifelike and Lovable Plush Standard Poodle

Item #: 924013

SKU: C7E27438

UPC/EAN/ISBN: 000772048613

Manufacturer #: 4861

Dimensions: 27 x 21 x 6

<http://www.toysrus.com/product/index.jsp?productId=2668151>

Note : Amelia will not be riding the obstacle during the competition.



Appendix C, Picket Fence Details

Fence: 8' by 8' U shaped section off the main field

The fence will be a white plastic picket fence. Below is a link to the fence at Lowes.

Fence Specifications:

Freedom

36" x 8' White Classic Gothic Vinyl Picket Fence Panel

Item #: 56507 Model: 56507

<http://www.lowes.com/lowes/lkn?action=productDetail&productId=56507-73428-56507&lpage=none>

