UNIVERSITY OF MINNESOTA
MECHANICAL ENGINEERING

Autonomous Above Ground Gopher
Design Report

University of Minnesota – Twin Cities Autonomous Vehicle for
The 9th Annual International Ground Vehicle Competition

Submitted By:

Josh Chambers Kevin Miller Andy Pullis Paul Zachman

Faculty Advisor Contact:

Patrick Starr

1 Introduction

This is the first entry in several years by the University of Minnesota in the IGVC. Our vehicle, the "Autonomous Above Ground Gopher" (AAGG), was developed over two semesters by mechanical engineering students for their senior design project. The first semester's team designed and built the frame, drive train, and power systems for AAGG. The second semester researched, designed and developed sensor systems and control algorithms for the AAGG.

1.1 Team Organization

Over the two semesters the team broke down into seven major teams each concentrating on different aspects of the vehicle. During the first semester the team focused on the mechanics of the vehicle along with the electrical layout. This was the foundation of the breakdown that is shown below. (Ref. Table 1.1)

Table 1.1: First Semester Team Breakdown

Electronics and Controls Design	Brett Scott Jon Lenzin-Oiknin
Chassis Design and Manufacture	Jason Tiedeken Karl Biewald
Drive train Design	Michael Spargo Brian Grev
Power System Design	Tim Ourada Yoke Kong Kuan

During the second semester the vehicle control system evolved into three main subsystems: sensory, guidance, and motion control. The status of the previously built vehicle combined with the regulations of the IGVC drove the system's specifications. These specifications became the framework for conceptualizing possible design solutions. The team split into three teams as shown below in table (Ref. Table 1.2).

Table 1.2: Second Semester Team Breakdown

Sensory System	Guidance System	Motion Control System
Brian Grev	David Clausnitzer	Jon Lenzen-Oiknin
Iyabo Lawal	Todd Hanks	Andy Pullis
Shumaila Anwer	Aaron Sou	Brett Scott
Paul Zachman	Mike Spargo	
Kevin Miller	Phil Michaelson	
	Josh Chambers	

1.2 AAGG Overview

The AAGG sensory system includes a laser range finder and a stereoscopic camera. The camera captures both 2-D and 3-D images in a range of up to 25 ft, with a resolution of 6 inches at the most. While the laser rangefinder determines the distance to obstacles in a 180° arc with a range of up to 50 feet. The information received from these devices is provided to the computer, which uses it to understand the surrounding environment.

The mechanical drive system consists of a pair of geared down servo motors, independently driving the left and right wheels of the vehicle. This provides forward and backward movement at speeds up to 8 mph, and zero turn radius steering for the AAGG. The motors also have encoders, which provide position information to the computer.

2 Mechanical Systems

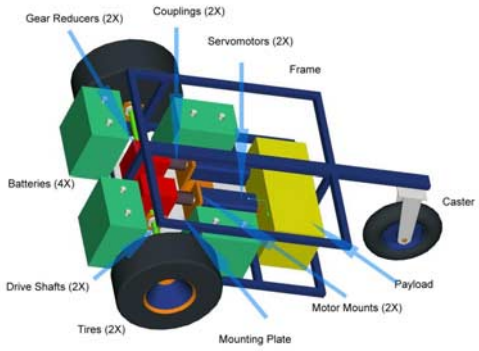

Figure 1.0: Mechanical Layout

2.1 Drive System

Our vehicle uses a differential drive system for movement. To power the differential drive we chose servomotors, because of their precise controllability and the ease with which they can be integrated into our control strategy.

The motors on our vehicle are brushless servomotors, each capable of 376 oz-in of output torque and developing 520 watts at 2000 revolutions per minute. Keep in mind these numbers are before gear down. The Kollmorgen servomotors were chosen because they met our needs and were donated to us by The Mackubin Group.

The speed reducer we chose is a single 20:1 reduction worm gear made by Boston Gear and has a transmission efficiency of 80%. The 20:1 gear down was based on the Kollmorgen motors discussed earlier. It is necessary to gear a motor so it operates at its optimal rpm during peak loading, and so the wheels have enough peak torque to overcome any barrier that may be encountered. The Boston Gear speed reducer was chosen over other options because of the output shaft diameter, company reputation, cost, and standardized mounting face size. By using a NEMA 56c mounting face we can use other motors in the future without having to custom fabricate mounting plates. Also by using a NEMA 56c speed reducer with a 1-inch diameter

female output bore we can eliminate the need for couplers and other unnecessary hardware by directly interfacing the drive axle to the speed reducer

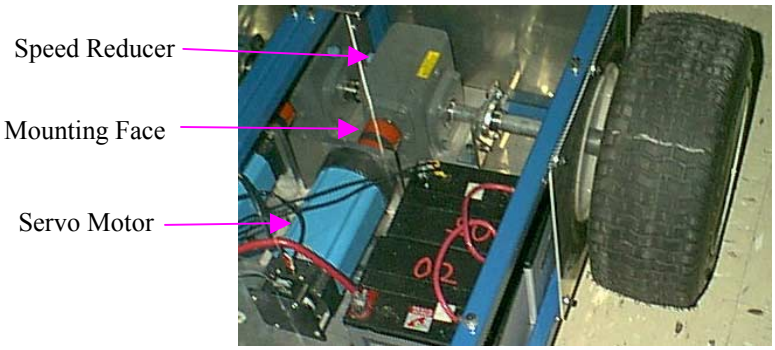

Figure 2.1: Drive Train

AAGG has a rectangular box frame, multi-level shelving to hold the electronic components and batteries, and three wheels to provide mobility. The two front wheels are propelled by computer-controlled electric motors and the third is a free rotating caster that passively trails behind.

We chose a three-wheel design, as compared to a four-wheel design, because all three wheels would be coplanar on the ground. The intent was to create a system where the two front drive wheels would always be in contact with the ground. This is especially important since the navigation system receives position feedback from the wheels for guidance purposes.

2.2 Motor Control

The motion control system of our vehicle is a based on an ISA bus motion controller card that connects directly into the motherboard of our computer. Motion controllers of this type contain onboard processors that handle the computations relating to the motors, thus leaving the main computers CPU for higher-level functions

We selected Intelligent Motion Controls POSYS704, 4-axis motion controller for our vehicle based on its availability, more than adequate capabilities, and discounted price. We selected a 4-axis controller, our current layout only requires 2, because we wanted to remain versatile and keep any future options open. The motion controller connects to the motors through a power amplifier, which moderates the power to the motors and performs the Hall effect adjustments that the motors require. We wanted our system to be able to handle voltages of 24, 36, 48, and if necessary 60 and 72 Volts. Additionally the Kollmorgen motors are brushless and need 40 amps to develop peak torque, 18 amps to develop max continuous torque. Many amplifiers met these requirements; we selected the Servo-Systems model BE40A8 in large

part because of the large discount they provided for us. Communication between the motion controller, the amplifiers and motor encoders is routed through an IMC IO789 interconnect module that makes the system more organized and easier to setup or modify.

2.3 Sensor Mounting

The camera is mounted on a telescoping pole in the center of the vehicle on the very front edge. This position will give it good visibility of the course ahead, and the telescoping system will allow for any necessary adjustment to be made quickly. The Sick Laser Range Finder is being mounted on the front of the vehicle approximately 12 inches off the ground. This position will give it an unobstructed view of the course, and allow the detection of any obstacle in front of the vehicle that is at least 12 inches tall.

3 Electrical Systems

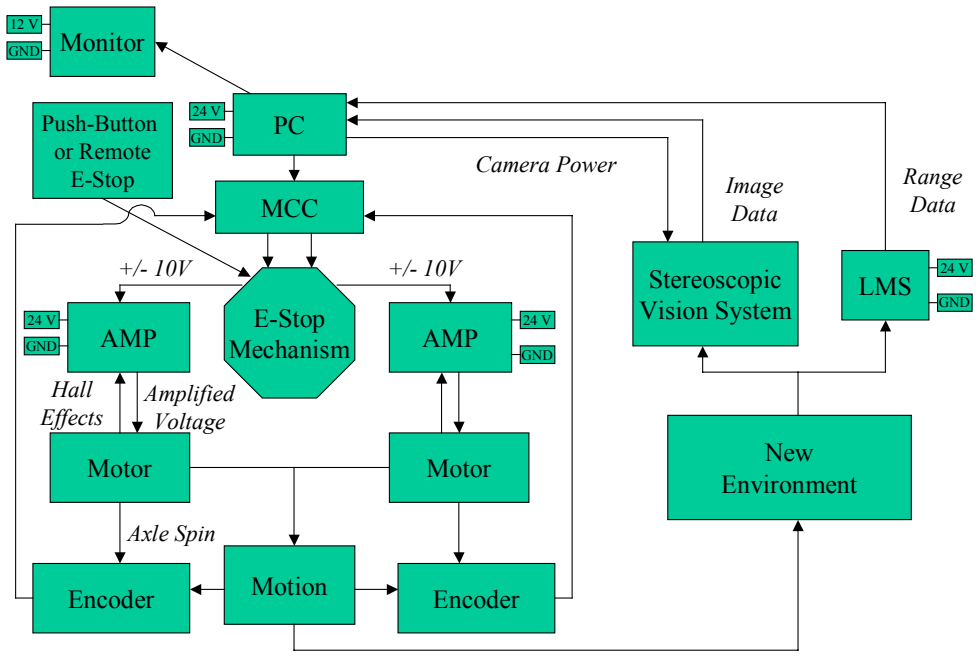

Figure 3.1: Electrical System Schematic

3.1 Power Systems

To power the electrical systems we require 24-V DC for the motors, computer and SICK Laser Range Finder, and 12-V DC for the monitor and e-stop circuitry. Because the motors and other hardware require 24-V to meet our needs, and nearly every large battery made is 12-V, we

are forced to operate with 2, 12-V batteries connected in series. At operating conditions, we expect the motors to draw approximately 35 amps of continuous current. Also we decided that we wanted the battery pack to last at least 1 hour per charge to allow adequate operational time for the competition. The solar car team at the U of MN donated a set of 10 deep cycle batteries to our project; these batteries have a life of 35 Amp-hours. We decided to dedicate a separate battery pack to the computer and laser range finder because they will require 24-V DC and around 15 Amps. The monitor and e-stop also have a 12-V battery pack; this helps avoid asymmetric discharging of the 24-V battery packs.

3.2 Computer

Vision processing, laser range detection, and motion control are computationally intensive processes. The large number of simultaneous mathematical computations requires the computer to have as much processing power as possible. A custom built computer was designed to handle the operations that would be required for running the vehicle. The computer is based on an Athlon 1.0 GHz processor, with 768 megabytes of random access memory (RAM). The processor provides the power to perform the calculations that are needed for vision processing and distance calculations, while RAM is needed for storage of the sensor information and the environment map.

Other considerations that were made in selecting components to place in the computer were compatibility with the laser range finder, vision system, and motion control card. For example, the motion control card required an industry standard architecture (ISA) slot, while the vision system was a peripheral component interface (PCI) slot. Therefore, when the computer was purchased a minimum of one ISA slot and one PCI slot were specified.

Among the components added to the computer after purchase was the Advanced Motion Control POSYS Model PC-104 Motion Control Card. This card is a PC-compatible input and output (I/O) controller, which can be used for motor control as well sending and receiving signals from external equipment.

Another component that was added to the computer was a fire wire (IEEE-1394) card. This card allows very fast data transfer rates to peripheral devices on the computer. The Mega-D Stereo Head, providing our vision sensing, was designed to work with a fire wire input, and necessitated the addition of this component to the computer.

The AC power supply on the computer was replaced with a 24-V DC power supply. Since the normal AC power supply would not work when the computer was placed on the chassis, which only has 24-V DC on board.

Also a USB compatible game pad was added, for simple manual control of the robot.

3.3 Sensors

The three sensors that the AAGG used are a stereoscopic camera, a laser range finder, and a pair of motor encoders.

Figure 3.1: Mega-D Stereo Camera Head

The stereoscopic camera is made up of two color 1.3 megapixel, progressive scan CMOS imagers mounted together in a single unit. This system has a capable resolution of 1288 H by 1032 V, along with excellent sensitivity, dynamic range, anti-blooming, and noise characteristics. The stereoscopic camera is connected to the computer through an IEEE 1394 fire wire cable, which also provides the power to the instrument. This camera system has the ability to capture both 2D and 3D images. Comparing the image of both cameras allowing the creation of the 3D images.

Figure 3.1: SICK LMS

The laser range finder is also used to detect 3D objects in the physical environment. The laser range detection uses very accurately timed laser pulses, to determine the distance to an object. In addition it is powered by a 24 volt DC supply and communicates with the computer through via a RS 232 serial port.

The servomotors that are being used are equipped with encoders that provide motor shaft displacement information. Since the vehicle wheels are connected to the shaft of the motor by a 20:1 gear reducer, we can determine the rotation and thus the speed of the wheels.

3.4 E-Stop

The vehicle has two emergency stopping devices. There is a large red button on the tail end of the vehicle and also a wireless device. The e-stop devices cut off the power from the amplifiers to the motors, thus near instantly stopping the motion of the vehicle. The wireless e-stop uses a remote entry device commonly used to unlock car doors. The range of the device was

increased to over 50 feet by greatly increasing the receiver antenna size and also increasing the power supply in the device.

4 Software

The software design was broken down into small components that could easily be worked on by sub-groups. These components were combined into a common framework that coordinated the operation of the vehicle. Some components were written entirely by the group, while other components were donated or purchased. The software team wrote the main control program. The motion control card, the laser range finder and the vision system each came with its own set of libraries that allowed access to the sensor directly through its custom software. The path

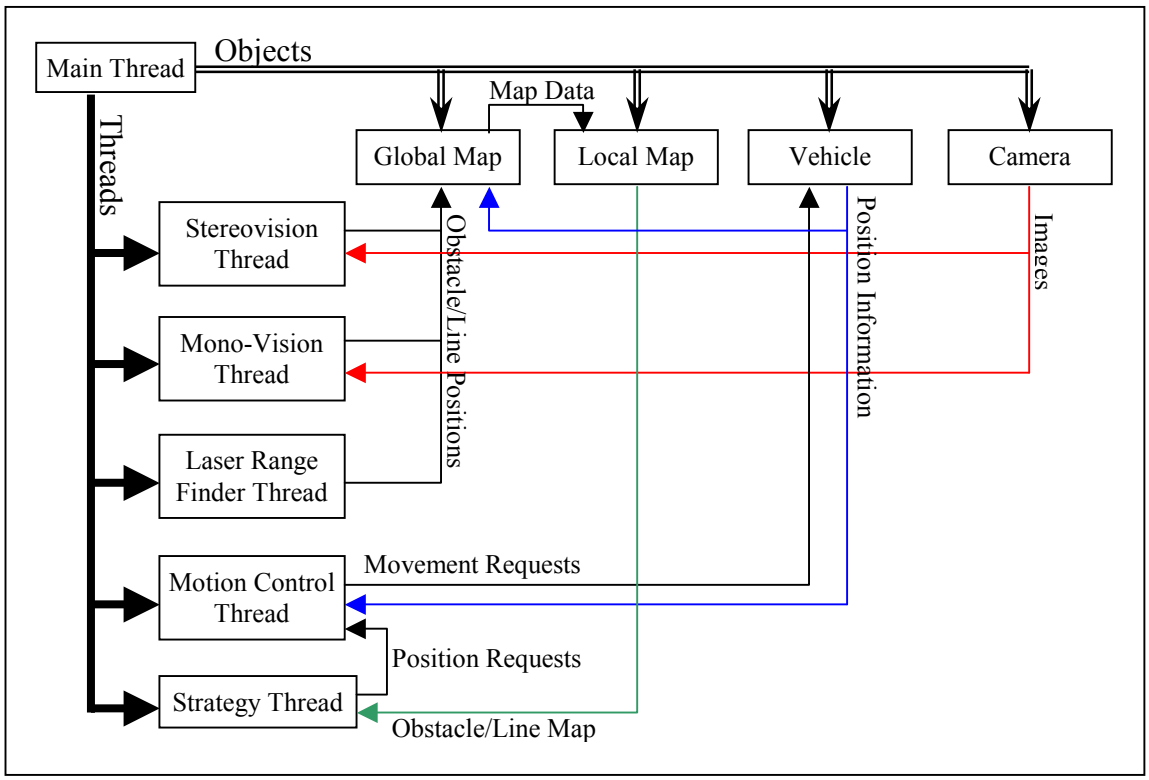

Figure 4.0: Thread and Object Communications

finding algorithms were modified versions of software freely available on the Internet. During development each software module was developed in isolation, tested to ensure that it worked correctly, then incorporated into the main program.

The framework that the vehicle's software is based on is a set of threads. The program consists of a main thread, which controls the other process threads and provides shutdown functions. The other process threads include strategy (path determination), motion control,

stereovision, mono vision, and laser range-finding threads. The main program controls all the threads, however each thread can communicate with other threads a couple of different ways. The main program creates several global variables that can be accessed by any function in the program, and therefore any thread can modify a global variable and then another thread can access the information stored in that variable. Many of these variables are actually complex objects with member functions that allow threads to change and access data without compromising data integrity. Two objects of major importance in our program are the global map, and the vehicle.

4.1 Main Thread

The main thread provides the interface between the user and the control program. It is the place that the autonomous part of the robot is initiated; it provides the emergency stop, and shutdown functions. The main thread also controls initiation of the sensor, strategy, motion control and manual control threads through standard windows menus; as well as displaying simple information about all the program threads.

The main thread is designed as a loop that runs until an error occurs within the program, or an external command is received. The main loop checks and verifies that the robot is working as intended, and if not, the vehicle will stop and shut off down.

4.2 Strategy Thread

The strategy thread contains the functions that make decisions about where to move based on information from the global map. The strategy thread holds the functions that decide where the vehicle is going to go based on the sensor information. The strategies include a general path finding algorithm which directs the vehicle in 90% of the cases, as well as a couple of other specialized functions which give direction when unusual circumstances are confronted. The strategy thread communicates with the motion control thread, sending information about the next step to the motion functions.

To move the robot from one point to another within the course, an algorithm that finds the most efficient path between two points on the map is necessary. All the different obstacles and traps that the robot might encounter must be considered while determining the path. For a large, open map there will be several, if not dozens, of possible paths that the robot could use to successfully navigate between the starting point and the ending point.

The software must quickly compute the best path, and yet still consider all the possible paths. Furthermore, the method of selecting how to rank a path must be flexible to account for different possible scenarios. Finally, the software must return results in a logical and useful format, which can be interpreted by the motion control module to move the vehicle.

Based on these criteria, the A* path finding algorithm was chosen. This algorithm, which is commonly used in video games, is based on a combination of a couple of other search algorithms – the greedy search and the uniform-cost search.

A greedy search minimizes the cost from the starting point to the goal point. Essentially, a greedy search looks at all the possible options from a particular point (on a map or grid), and takes the path that moves the user closest to the goal for that particular step. However, this process does not select the best path because a step may exist which creates a shorter total path, but for one step does not move the user closer than another option would. By not searching all possibilities this search function is said to be incomplete.

A uniform-cost search, on the other hand, is complete because it checks the length of all possible paths between the starting point and the goal. The uniform-cost search looks at the step with the lowest cost first, then looks at the other possible steps. Once the lowest cost step has been determined, that path is expanded to find the next step in the path. This process is repeated until the goal is reached, and the result is the cheapest path between the goal and the starting point. Once every branch has been recursively searched, the algorithm returns the best possible path.

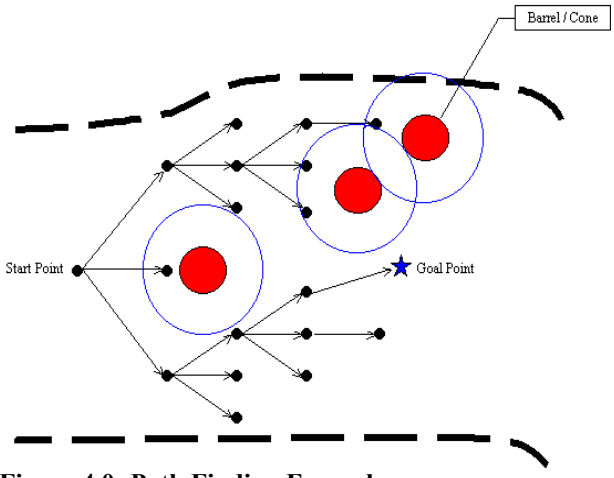

Figure 4.0: Path Finding Example

The A* algorithm takes the best of both the greedy search and the uniform-cost search by simply summing the values to determine the best path. This ensures that all paths are searched, by the uniform-cost search, and that the lowest cost path is found, by the greedy search

The weights that are used to determine the cost of each step must only follow one rule – they must always be positive. If the cost of a step were not positive, then the search algorithm would have to search every possible path, not just those with the lowest single step cost. If

negative step costs were allowed, then a single step with a large negative value could potentially be the lowest cost path.

For the A* algorithm to work it requires both a start and a goal point. The start point is easily defined as the position of the vehicle at any given time; however the goal point is more important and difficult to define. The strategy being used to determine the goal point is fairly simple. First, the forward direction of the course is guessed based on the direction of the lines near the vehicle. Second, several straight paths are projected from the vehicle in a range of angles around the forward direction, each path ends as soon as it encounters a line or the edge of the camera's vision. Third, the end of the longest path is chosen as the goal point. This goal point will be adjusted again long before it is ever reached, so this strategy should keep the vehicle on the course, and the A* algorithm will ensure that it avoid obstacles.

4.3 Motion Control Thread

The motion control thread provides the means for moving the robot while other operations are occurring. The motion control thread is the set of functions that directly drives the motors through the motion control card (MCC) installed on the computer. The MCC was provided with a package of functions that are used to output specific commands to the servo-amplifiers, which then move the motors. The motion control thread also tracks the position of the vehicle relative to the starting point. The vehicle's position is marked on the global map, and is updated via rotational displacement provided by the encoders on the motors.

The motion control system is a real-time displacement feedback system that is run by the Advanced Motion Control POSYS 704, motion control card. The motion control card provides the interface between the computer and the motors that drive the vehicle, as well as any other input that the vehicle may need, such as emergency stop signals.

The motion control thread responds to the input that it receives from the path finding thread. Once the motion control thread has the list of steps that it must take to reach the next goal point, it moves towards the goal in a systematic fashion. The motion control module gives a command to the motors to move to the next node, which is perhaps 3 inches away, and waits until it receives feedback from the motors that the step has been accomplished. The motion control module then issues the next command to move to the next node, and again waits. This process is repeated until the goal is reached or the path finding algorithm sets a new goal. If the path finding algorithm determines that a new goal is necessary, the motion control module

finishes the step on which it is currently working, and then begins the cycle over again with the new list of steps provided by the path finding module.

While this systematic method is somewhat halting in operation, it does provide more time for updating the sensors and verifying the direction and goals of the vehicle than other forms of control. At each node that the vehicle stops, the sensors can verify that the obstacles have not changed location and the path finding algorithm can make corrections to the path if necessary to avoid previously unseen obstacles. A tradeoff was made to favor consistency and accuracy rather than speed when moving on the obstacle course.

The method of point-to-point movement was easier to develop and implement than a continuous movement form of motion. Continuous motion would require that the points that the path finding algorithm created as the path to follow be transformed into vectors for the direction and speed of the vehicle. While this has previously been done, the level of sophistication is higher than the team thought it could design and implement at this time.

4.4 Monocular Vision Thread

The monocular vision system provides color analysis and object finding based on individual images. The monocular vision system thread uses a picture from one camera to analyze the course. Once a picture is captured, the thread sends it to a translation function that translates the real-life image into a 2D map of the area seen by the camera. This map is then analyzed for colors and potential objects. Lines, potholes, and other 2D objects are the items for which this sensor is searching.

Figure 4.1: Image Filtering Example

In order to get useful information from an image, it must be broken down into a very simple form that the computer can understand. To do this a series of filters and thresholds is used to get a black and white image containing only potholes and lines. Then the white features in the image are numbered and characterized, in an attempt to guess what they are. After they have been characterized the features image coordinates are transformed to ground coordinates and added to the global map.

The monocular vision thread can use a variety of filters to eliminate noise, distinguish edges, and threshold. The best combination of filter will likely depend on the actual appearance of the course. A set of filters that will likely work well is a smoothing filter, followed by a modified adaptive background threshold, followed by a dilation to improve the continuity of lines.

The white features of the filtered image are grouped by assigning the same number to all pixels adjacent to each other. Groups of pixels that are two small are considered noise and thrown away. The groups are characterized using area over perimeter squared as the criteria separating a line from a pothole.

Once feature characterization has taken place the center of lines and the edges of pothole are found and transformed. The transform assumes the ground is a horizontal plane and uses the pinhole camera estimation along with the camera parameters: tilt angle, focal length, height, and image sensor size to estimate ground coordinates of pixels. Once the centers and edges have been transformed, they are added to the global map for the strategy to look at.

4.5 Laser Range Finder Thread

The laser range finder thread controls the input and output of the laser range finder. The range finder is another device used to detect the distance to three-dimensional objects, such as construction barrels, cones, and buckets. This information, combined with the 2D maps created in both the stereovision and monocular vision systems, is used to accurately determine the distance to obstacles.

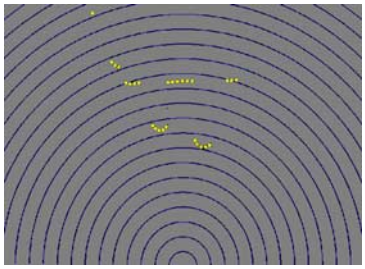

Figure 4.2: Laser Range Finding Example

The Sick Laser range measuring system is the most accurate of all our sensors. Its main function is to find obstacles with a height dimension. It does this by pulsing a laser every half-degree along its 180-degree arc of movement. After every pulse, it waits for a reflection, if a reflection is detected, the time between the pulse and the reflection is used to calculate the distance to the object. The result is a clear picture of the surrounding area given in a dot outline form. The main functions of the sick software is to initialize and import this information and then transform the data values from polar coordinates into Cartesian coordinates used the vehicle's coordinate system.

4.6 Stereo-Vision Thread

Figure 4.3: Sample Disparity Image

The stereo thread has not been fully implemented at this time. Using the Small Vision System (SVS) software provided by SRI International the pair of images from the Mega-D stereo head, are compared and a depth image is created. The depth image is very noisy, but temporal filtering eliminates much of the noise. After temporal filtering, a change in depth filter is applied. This filter eliminates small objects and depth measurement errors in large objects. The remaining depth measurements are classified as ground or obstacle by comparing there coordinates with the image transform used in the monocular vision thread. Using the depth information from the obstacles the front of each obstacle is added to the global map.

4.7 Thread User Interfaces

Another function of each thread is to control the user interfaces for each operation. During development and testing, the monitor will display what the sensors perceive, what the strategy component is deciding, and what the motion control module is doing. Each thread has a dialog box that is the users way of controlling the tread. When the dialog box is closed the thread ends. The dialog box also displays information and controls certain aspects of the thread. The modularity provided by separate dialogs improves the ease of testing each component, minimizes one module causing a different module to crash, and speeds development by breaking the program into smaller segments.

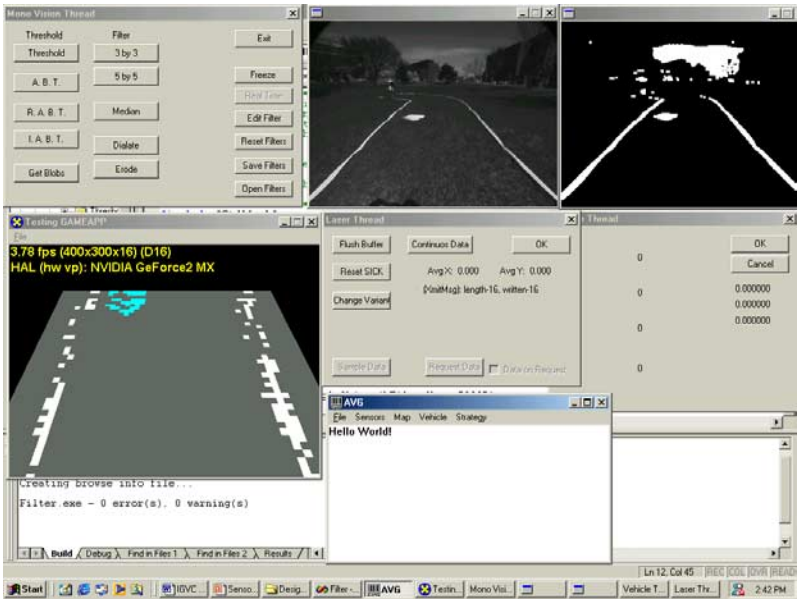

Figure 4.4: User Interface Thread Screen Shot

4.8 Data Objects

Within each thread, there are different software objects. A software object is similar to an object in real life – it has attributes and functions. Some of the objects included a global map, a path finding object, a vehicle object, a local map, and a sensor object. Each object had specific purposes for which it was designed. Coordinating how these objects interacted was a significant amount of the software development.

The global map object, for example, is a composite picture of the environment, as the robot perceives it. Every time a sensor receives new information, the global map object will take this information and add it to the map. Location, size, and types of obstacles are recorded on the map, as well as the location and orientation of the vehicle with respect to its starting point. Modules, such as the path finding algorithm, by calling the global map's member functions, could then access this information. The function will then return the information requested to the inquiring module.

The local map, which is another object, is a subset of the global map. It is limited to the area that immediately surrounds the vehicle. The local map contains most of the same information that is in the global map, just over a smaller area. The path finding algorithm uses the local map to find the best path for the next 10 to 20 feet. By using a smaller map, the computation time is significantly faster than if the larger global map were used.

5 Conclusions

This being the University of Minnesota's first attempt at an intelligent vehicle of this sort in several years, the project became quite a learning experience. Simply discovering the kinds of questions we needed answered took nearly half a semester of research. In the past few weeks various pieces of the project have begun to come together, and the results have been excellent. Finally our blood, sweet, tears and carpal tunnel are turning into a working semi-intelligent vehicle. Hopefully at this years competition we will set precedence for years to come, and the IGVC will become an annual event for the U of MN.

Appendix A: Costs

AAGG Bill of Materials				
Item	**Quantity**	**Description**	**Supplier**	**Cost**
Gear Box	2	Worm Gear	Industrial Supply	$800.00
Motor	2	Servo Motor	The Mackubin Group	Donated
Servo Amp	2	PWM Amplifier	Servo Systems	$950.00
Couplings	2	Love Joy	Industrial Supply	$44.00
Batteries	10	Lead Acid	U of M Solar Car Team	On loan
Motion Control Card	1	ISA Bus Card	Intelligent Motion Control	$700.00
Breakout Box	1	Assessory to MCC	Intelligent Motion Control	$250.00
Cable	1	Assessory to MCC	Intelligent Motion Control	$150.00
Computer	1	1 Gig Athlon	General Nanosystems	$1,215.00
Tire/Wheel	2	Tractor Wheel	Northern Tool & Equip.	$52.00
Axle Shaft	1	Threaded Axle	Northern Tool & Equip.	$41.50
Caster	1	12" pneumatic	Northern Tool & Equip.	$35.00
ATV Live Axle Hub	2	Live Axle Hub	Northern Tool & Equip.	$33.98
Live Axle Kit	2	Live Axle Bearing Kit	Northern Tool & Equip.	$40.00
33' of Framing Material	1	3/4" Mild Steel Tube	U of M tool Crib	$20.00
1/4-20 1.5" hex	13		U of M tool Crib	$1.30
1/4-20 2" hex	9		U of M tool Crib	$0.90
1/4-20 1" hex	34		U of M tool Crib	$3.40
3/8 -12 2" hex	4		U of M tool Crib	$0.40
5/16 -12 1.5" hex	16		U of M tool Crib	$1.60
1/4 -20 Locking Lug Nuts	9		Adam's Nut and Bolt	Donated
1/4-20 1" Thumb Screw	8		Adam's Nut and Bolt	Donated
AVK	43	Quick Connectors	Adam's Nut and Bolt	Donated
2' X 2' X 1/2" Aluminium	1	Used in various places	U of M tool Crib	$20.00
2' X 1/4" X 1 1/4 angle Al	1	Used for angle mount	U of M tool Crib	$0.00
1/4" x 2"x 2"	1	To mount batteries	U of M tool Crib	$0.00
35 lb of 6061 Aluminum	1	Panels		Donated
30x31x1/4" Lexan Plate	1	Top Panel		Donated
Sick LMS	1	Laser Range Finder	Sick	$5,700.00
SVS Stereo System	1	/w Mega-D Head	Videre Design	$3,000.00
Monitor	1	15" LCD Flatscreen		$500.00
Wireless Door Chime	1	For E-Stop	Radio Shack	$13.00
Remote E-Stop	1	Remotely Halts Vehicle		Donated
Manual E-Stop	1	Mounted E-Stop		Donated
Strobe Light	1	12V Halogen and cup		Donated
Power Switch	1	Amp power switch		Donated
TOTAL:				$13,572.08