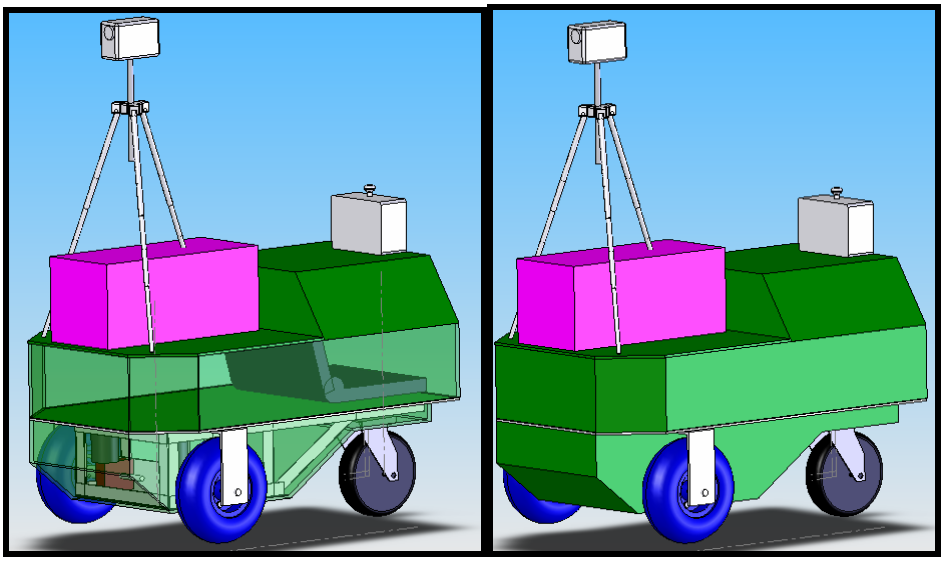
# Lawrence Technological University

# Think-Tank

## IGVC 2005

## Autonomous Vehicle




**Team Members**

David Bruder, Eden Cheng, Johnny Liu
Mark Salamango, and Maurice Tedder


**Faculty Advisor Statement**

I, Dr. CJ Chung of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on Think-Tank has been significant and each team member has earned credit hours for their work.

Signed,

_____ _____

Dr. CJ Chung (chung@ltu.edu) Date

# 1. Introduction

Intelligent Systems have become increasingly prominent in our daily lives. Innovations in autonomous systems such as robotic lawnmowers, vacuums, space exploratory vehicles, and military vehicles have an important role in advancing technology.

The IGVC competition fosters original ideas in intelligent systems. This year LTU presents Think-Tank, a unique entry in the 2005 IGVC competition. The robot is built with solid engineering principles and maximizes portability and scalability through hardware and software design. The robot uses the Java programming language as its core intelligence building block, and was built from scratch to fulfill the requirements of the competition. This paper describes the Think-Tank design, considerations, and improvements over previous submissions, and core functionality.

# 2. Design Process

## *2.1 Project Planning Process*

The Think-Tank team examined the requirements of the customer; in this case, the judges of the competition. Official IGVC competition rules provided the baseline requirements document necessary to began the design planning process. A design process was selected after careful review of the requirements document and team capabilities.

The selection of a design process for this project was guided by the traditional three constraints of time, budget, and quality. Based on these constraints and the desire to deliver a functional system as soon as possible, an agile design philosophy was selected.

**Figure 1** illustrates our agile cycle of 1. speculation, 2. collaboration, 3. learning, and 4. eventual release of a system increment.

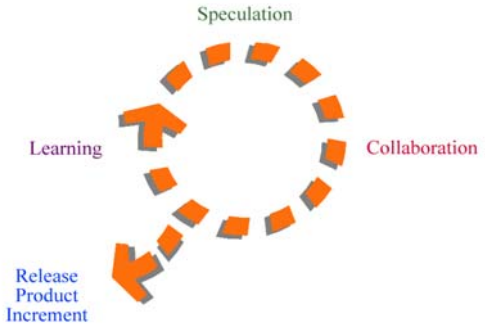



**Figure 1 Agile Design Cycle**

The agile philosophy is characterized by small self-organizing, self-managing teams, rapid delivery of incremental work products, an iterative

development cycle, and working products as the primary measure of success. Agile methods stress product delivery over analysis and design, with design and construction occurring concurrently.

### 2.2 *Project Development Process*

Development of the Think-Tank autonomous vehicle consisted of iterative development of software and hardware prototype increments. We built a fully functional vehicle prototype to use for software development and vehicle design evaluation. From this prototype we discovered drive train motor mounting and alignment problems. These issues were reviewed and corrective action was taken by mounting the motors vertically and moving them back four inches to improve handling. These changes were implemented on the prototype vehicle and incorporated into the final competition vehicle.

### 2.3 *Project tracking and Quality Metrics*

Both hardware and software metrics were tracked to improve quality. Software quality metrics were further broken down into product, process, and project metrics. Process and product metrics focused on improving the defect removal rate and the response to fixing software defects. Project metrics (number of developers, skill levels, schedule, etc.) were more difficult to track.

Hardware was also measured by conformance to requirements and performance parameters.

### 2.4 *Collaborative Team Organization*

The 2005 Think-Tank team organization can be found below in **Figure 2**. Each functional block represents a self-organizing and managing unit responsible for the three competition events. Thus far, approximately 1,595 total work hours were contributed to the project by all members of the team.

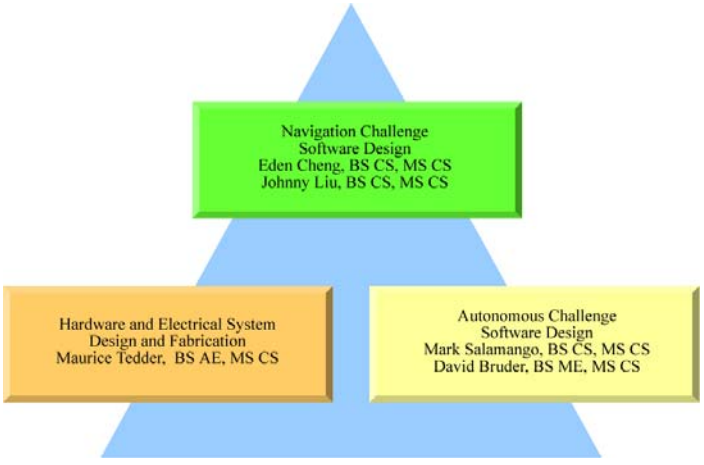



**Figure 2 Team Organization**

## 2.5 *Vehicle Conceptual Design*

Think-Tank is designed to fulfill the dimension, safety, and vehicle performance requirements as specified in the 2005 IGVC rules. Additional input came from reviews of previous LTU IGVC vehicle hardware designs from the 2003 and 2004 competitions. The review of previous vehicles revealed the following problems in our previous IGVC designs: poor traction, inaccurate control, lack of all-weather performance, and excessive workload on the camera sensor. The review also revealed successful features that the new vehicle should inherit from previous designs such as the use of lightweight, low cost materials and a simple mechanical design that is easily constructed using basic fabrication processes such as drilling, cutting, and nut and bolt fasteners.

Think-Tank's conceptual design is based on extensive use of CAD models using SolidWorks CAD software. Two essential design decisions were made before modeling the vehicle in CAD:

- A semi-monocoque vehicle frame utilizing only flat panels to form the vehicle shell
- A two piece vehicle shell with a lower drive train chassis platform and a removable command and control upper platform to house all of the control equipment and components

The CAD model allowed us to analyze the dimensions and placement of motors, wheels, laptop computer, and camera sensor and to check for interference between components while maintaining the dimension constraints imposed by the IGVC rules. After several iterations of the CAD model, a satisfactory design was obtained and is shown in **Figure 3.** Blueprints were created from the solid model to begin construction of the prototype vehicle.

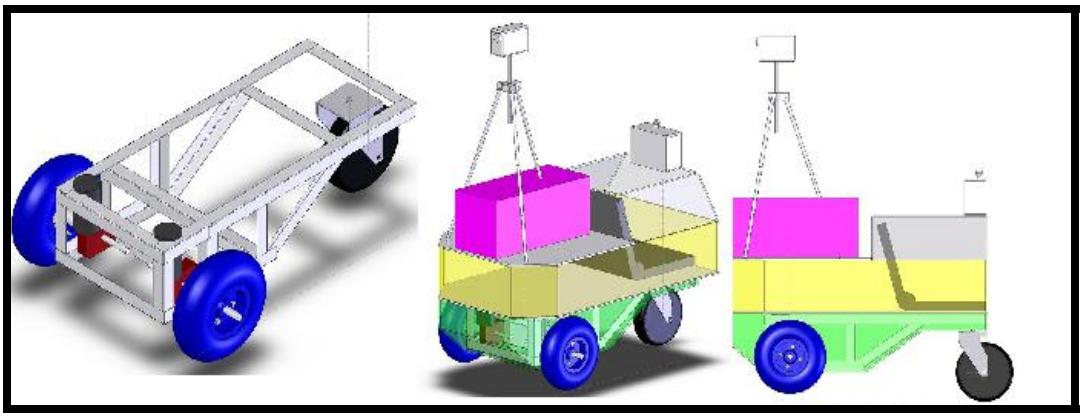


**Figure 3. CAD Models of Vehicle Structure**

# 3. Hardware Design

## *3.1 Robot Structure*

The three wheel base configuration with tank-style differential steering and rear caster wheel was inherited from previous 2003 and 2004 designs. It has proven to be an effective configuration that provides zero-turn radius maneuverability and simple vehicle control dynamics.

## *3.2 Drive Train*

Design reviews of the 2003 and 2004 LTU IGVC vehicles revealed that our ability to win the autonomous challenge and navigation challenges were severely limited by a lack of traction and precise motion control. Both of these problems were solved by using two high performance 12 volt DC 40 amp electric motors with 21:1 worm gear ratio and .25 horsepower at 150 RPM.

The addition of 400 CPR incremental optical encoders to the output shaft of each motor of the 2005 LTU vehicle allows precise feedback speed and position control not available in previous vehicles. The encoders solve the traction problem encountered in previous vehicles by allowing our vehicle to maintain the same vehicle speed regardless of the terrain or grade of the incline. Think-Tank does not have the loss of control traveling down inclines that plagued the 2004 vehicles. The poise of the robot can be precisely controlled to navigate through and around obstacles.

## *3.3 Actuators*

Think-Tank has vastly improved motion control capabilities compared to vehicles from previous years. Its motors are driven by the AX3500 two channel motor controller by Roboteq.

The new motor controller for the 2005 vehicle adds the capability to precisely control speed or position using feedback motion control with optical encoders. It also has key safety features such as independent power source inputs for the controller electronics and motor outputs, which allow us to turn off power to the motors for an emergency stop.

# 4. Electrical System

## *4.1 Power System*

There are currently 3 independent power sources available in Think-Tank's power system. An independent 12-volt DC battery power source is dedicated to powering the motors to eliminate voltage and current fluctuations during motor operation in other circuits. Each electric motor circuit is protected by a 30-amp circuit breaker that resets itself after current levels return to normal conditions. A second 12-volt battery provides electrical power for the Ax3500 motor controller board, Novatel ProPack-LB DGPS unit, camera, camera image converter, and a laptop computer. Two 12-volt DC batteries are connected in series to provide a 24-volt power bus for the laser scanner. Batteries are changed when a low battery voltage warning is displayed in our main control software based on voltage information monitored by the Ax3500 motor controller. The power and communications control system schematic for Think-Tank vehicle is shown in **Figure 4**.

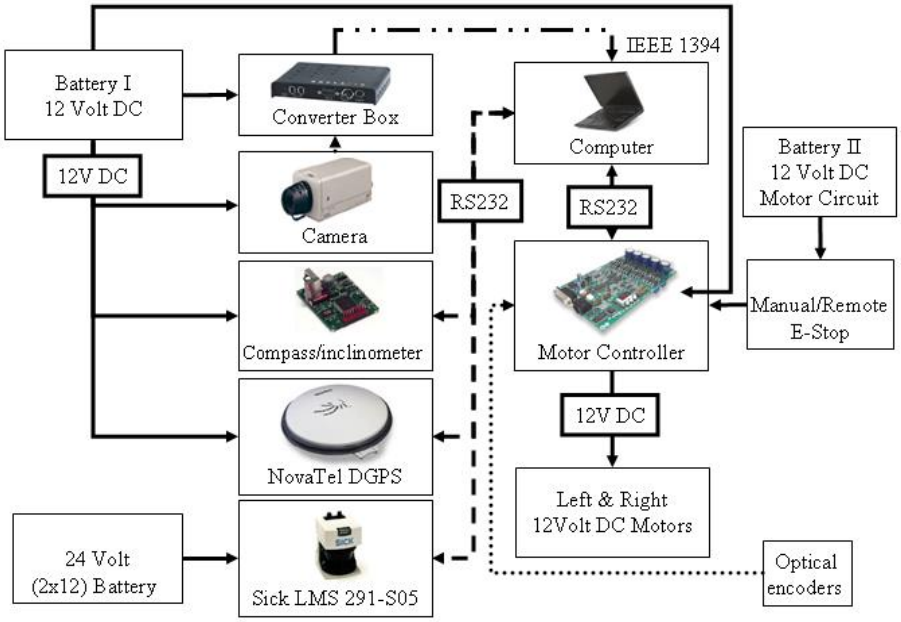



**Figure 4 Power and Communications Control System Schematic**

## *4.2 Computer Hardware*

Think-Tank's computer hardware consists of a MPC TransPort T3000 laptop. The MPC T3000 is a 1.6 GHz, x86 Pentium processor, with 512MB memory, and Windows XP operating system. Additionally, it has all the required data ports necessary to interface to Think-Tanks other components.

### *4.3 Sensors*

Think-Tank has increased sensor capability and updated the software architecture to promote a "plug-and-play" paradigm. Additional sensors include incremental optical wheel encoders, a sub-meter accuracy differential NovaTel Propack LB DGPS unit, a laser range finder (Ladar), and a digital compass/inclinometer. **Table 1** summarizes the variety of sensors used in the Think-Tank vehicle.

| **Sensor Component** | **Function** |
|---|---|
| Incremental optical wheel encoders | Controls motor speed within .1 mph and 2" distance |
| NovaTel ProPack-LB DGPS unit | Capable of .8 sub-meter accuracy with OmniSTAR |
| Digital Compass/Inclinometer | Provides heading, roll, and pitch information |
| High Dynamic Range DV Camera | Detects course lines and potholes in a 92 degree field of view |
| Pixelink PL-A544 Video Converter | Compresses a raw 640x480 video image stream to 160x120 |
| Sick Laser Scanner | Creates a 180-degree 2D map of obstacles and safe corridors |

**Table 1 Vehicle Sensor Summary**

## 5. Software Design

### *5.1 Software Strategy*

The Think-Tank software team applied a layered design process which focused on code reuse, scalability, and portability. For these reasons a true object-oriented language, Java, was selected to receive input, manipulate data, and control the robot.

All robot code has been written in Java for maximum portability. As is the case in the "real world," many times hardware becomes obsolete and developers are left with laborious, expensive integration efforts to port code from one hardware platform to another. By using Java, Think-Tank is more suited to adapting as hardware and customer requirements change.

To satisfy our goal of highly reusable code, our architecture was structured so modules such as Median Filter, Hough Transform, Black and White filter, Sobel transform, etc. could be plugged in or pulled out very quickly and easily. The notion of Java interfaces was exploited to make cose reusability and systems integration easy to implement.

Multi-threaded code can often lead to improved performance and scalability. This fact persuaded the software team to make each input to the system run in its own thread. The threads are joined later and can

then be used for decision making. The thread model allows the robot to scale up to accept and process input from many more sources simultaneously.

## *5.2 Autonomous Challenge*

The autonomous challenge requires extensive use of both the video processing and Ladar subsystems of the robot. They deliver data to the system for processing, aggregation, and then decision making.

### 5.2.1 Controller Subsystem

It is very important that data coming in from multiple sources is synchronized so that the robot is making decisions based on data that represents a fixed moment in time. For that reason, Think-Tank creates, and tracks individual threads that are spawned at the Capture Layer seen at the left side of **Figure 5** below. Those threads can continue to do their jobs in both background and foreground modes or Graphical User Interface (GUI) mode. When they are not updating to the screen, they are collecting and processing the incoming data. The background/foreground modes offer an important performance optimization.

The software model used to get filtered data into the data aggregator and then to the "Brain" of the system is also depicted in **Figure 5** below.

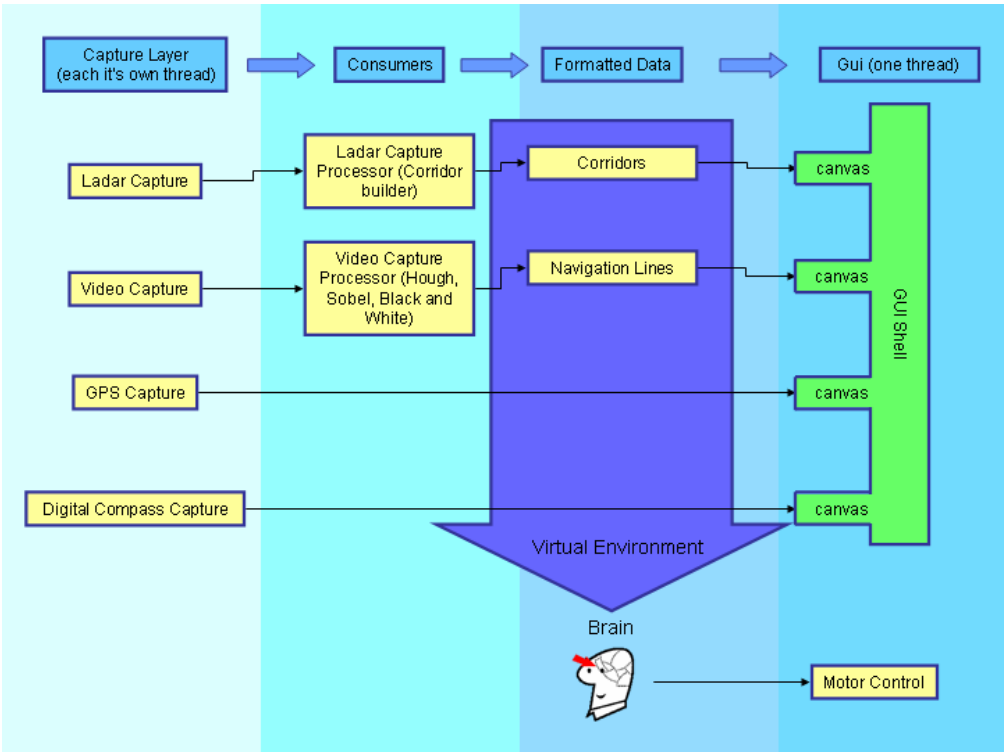



**Figure 5 Think-Tank Software Model**

Data from the Ladar, Video, GPS, and the Digital Compass are all taken and presented to data consumers which then process the data. For example, the Ladar Capture Processor looks for corridors and obstacles as it receives data from the Ladar, and the Video Capture Processor takes camera frames and manipulates them through different filtering modules.

Once the data is processed, it is passed to the Virtual Environment which registers interest in all the data inputs. Once there is new data from each of the inputs of interest, the Virtual Environment gets notified and aggregates each piece of information.

Finally, when all information is entered into the Virtual Environment for a given point in time, the "Brain" module can then make a decision on the best path for the robot to follow.

### 5.2.2 Video Subsystem

Video is used to detect the white lines. To accomplish this, the original video image goes through the following steps:

#### *5.2.2.1 Filtering*

Filtering is accomplished using a scalable, modified, color median filter. The median filter was chosen for its ability to preserve edges in the image while reducing salt and pepper video noise. The filter is set up with a 3X3 convolution window, but can be set to an arbitrary size

#### *5.2.2.2 Thresholding*

Thresholding is used to prepare a binary image for the Hough Line Transform. The image must be reduced to just 1 or 2% of its original pixels since the Hough transform does many calculations on each pixel.

#### *5.2.2.3 Hough Line Transform*

The Hough line transform [Hough 1959] is a method for finding dominant lines in noisy images. Every point in the binary image is transformed to Hough-space which represents the polar coordinates of potential lines through each point in the image. All of the potential lines through a given point in the image show up as sinusoidal curves in the Hough-space. The intersections of the curves represent collinear points in the image. By locating the peak intersection in the Hough-space, the coordinates of the dominant line is extracted.

#### *5.2.2.4 Hough Circle Transform*

The Hough Circle Transform uses similar techniques as described above to find circles of a specified radius in a noisy image. This is helpful in determining the location of potholes on the course.

#### *5.2.2.5 Perspective Correction*

In order to make the line information extracted from the image more useful for navigation, the image is transformed to look like a projection onto level ground. This is accomplished by entering five physical dimensions from the camera position on the robot and performing the trigonometric manipulation based on these. The parameters and correction diagram can be found in **Figure 6**.

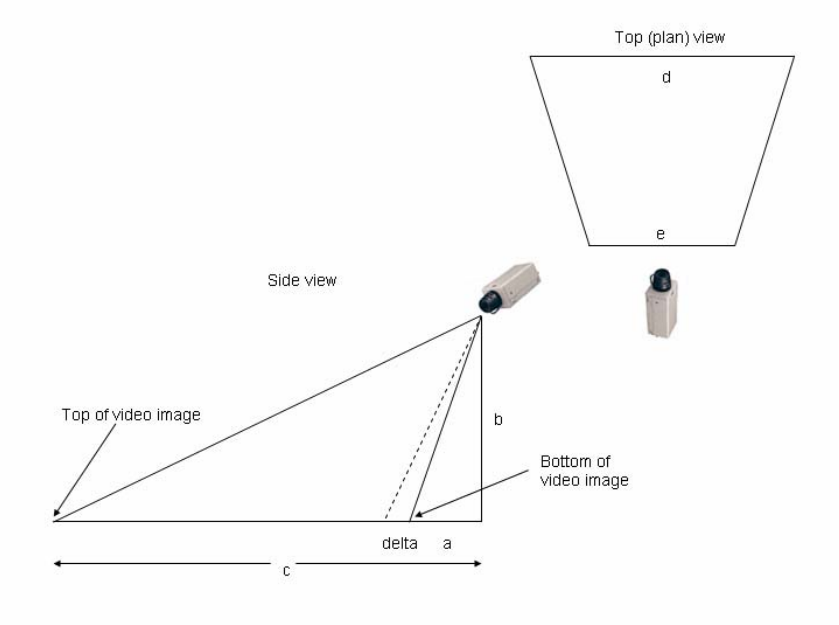



**Figure 6 Perspective Correction**

#### *5.2.2.6 Map to Virtual Environment*

The last step in the process is mapping to a specific field size in front of the robot which represents a common virtual environment.

### 5.2.3 Line Following Controller

The line following controller is a single Java class which combines several other classes to accomplish the following tasks:

1. Get heading error signal
2. Get centering error signal
3. PID controller
4. Motor control
5. AX3500 motor driver

All of the tunable parameters are in one block in the controller class and there is no cross coupling or dependency between the classes in keeping the object oriented design approach.

### 5.2.4 PID Controller

Once the heading and centering errors are determined, they are combined into a single error signal for the PID controller. Filtered video data provides input to the PID closed loop controller, which controls the vehicle's yaw-rate.

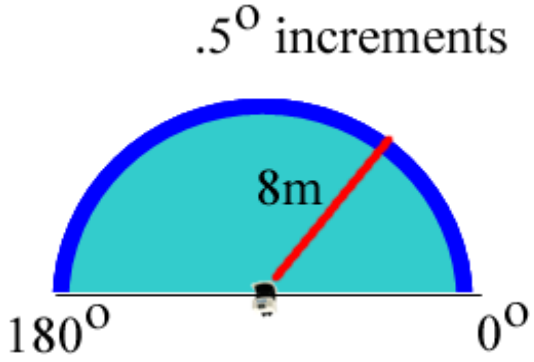
### 5.2.5 Motor Control

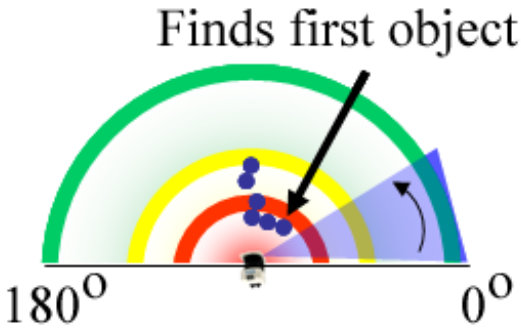
The motor control module is a smart actuator that receives speed and yaw-rate commands in engineering units; and provides retrieval functions for left and right motor command values. These can be directly transmitted to the motor driver software.

### 5.2.6 Ladar Subsystem

The Ladar (Sick LMS 291) is the predominant robot subsystem that feeds the controller with information about where the robot can or cannot go due to obstacles. Java Ladar processing code classifies objects as obstacles or corridors and feeds that information to the decision making module, or "Brain." The Obstacle and Corridor Detection algorithm is outlined in the steps below.

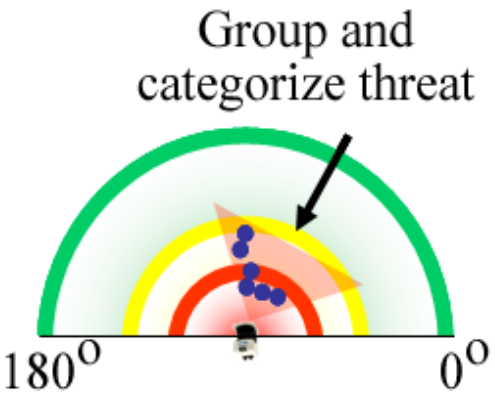
1. Start iterating through all the points of the Ladar sweep.

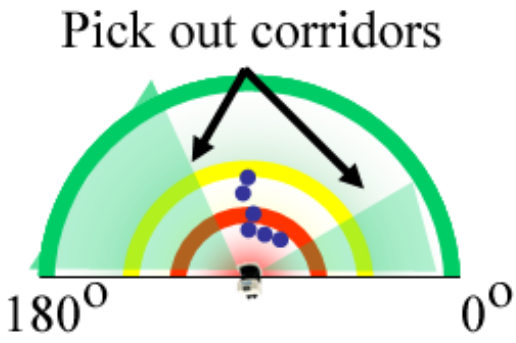



2. Search: If you haven't found an obstacle yet, keep searching. When you find one, check to see if the current point you're looking at is within the "Yellow Zone." If it is, go to the scanning obstacle mode. Otherwise, let's look to see if there is a possible corridor.

3. If isObstacle: Keep track of the maximum and minimum radii as well as the angle. Search to the left of the obstacle to see if there is an obstacle that is continuous to the last point. If there are, "expand" the angle and continue tracking the maximum and minimum angles. Once you have a gap between two points, go back to 2. (Search).




4. If Scanning a Corridor: Check all the points collected to see if they are greater than or equal to the tunable parameter CORRIDOR_THRESHOLD which sets the optimal value for a "Safe" corridor. If you find an object, switch back to isObstacle mode in Step 3.

Because the system picks out corridors up to a range of 8 meters, the robot can avoid cul-de-sacs that are less than 8 meters deep.

**Figure 7** below depicts the categorization of corridors and obstacles. The key at the bottom describes the threat levels of each zone the Ladar sees.

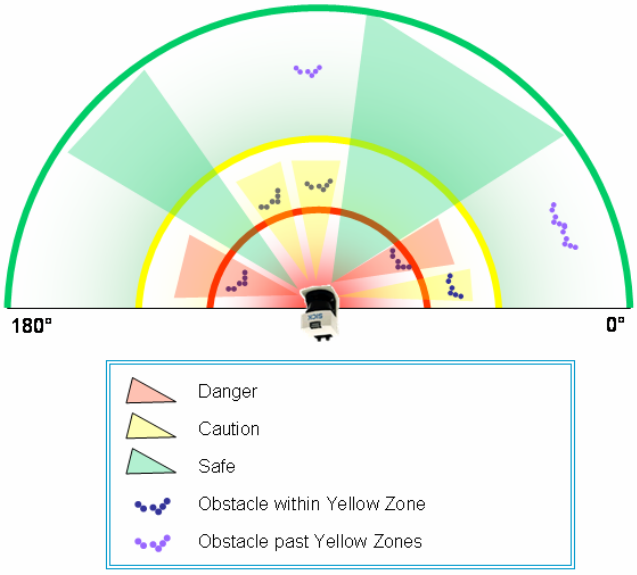



**Figure 7 Categorization of Corridors and Obstacles**

### *5.3 Navigation Challenge*

Previous LTU vehicles did not have a compass or accurate GPS sensors to determine the orientation of the vehicle. The compass and sub-meter accurate GPS on this vehicle makes the robot position information more accurate and faster because it can determine the direction to the goal and precisely move there. The robot uses the controller and Ladar subsystem code found in sections 5.2.1 and 5.2.6 to accomplish the tasks of the Navigation Challenge.

#### 5.3.1 Navigation Challenge Algorithm

Our method of reaching the GPS waypoint goal is to first start by heading in the direction of the goal. This is determined using GPS and compass data. When an obstacle is encountered, the robot will use Ladar data to avoid it and recalculate the path to the waypoint. We can calculate the angle, θ, between our robot direction and the goal direction, which is shown in **Figure 8**. Our navigation algorithm reduces the angle error between the robot and the goal to an angle θ close to zero, which means the robot is headed straight toward the goal.

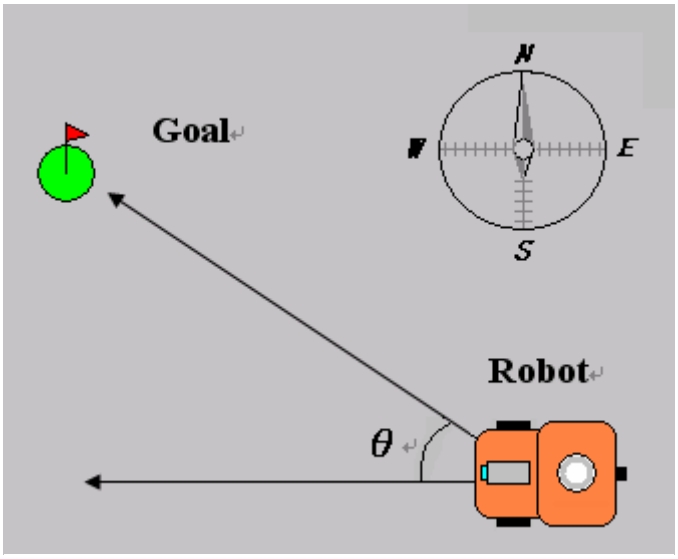



**Figure 8 The angle θ between our robot and the goal**

#### 5.3.2 Radar Graphical User Interface

A graphical interface in the form of a radar display, shown in **Figure 9**, presents compass and GPS data in a graphical tool used for testing the robot. The N represents north, the blue point is the robot, and the red line depicts the direction to the goal.

The radar graphical interface presents easy to understand graphics of the navigation. A red line on the radar display links the robot position to the waypoint indicating the path the robot should follow to reach the waypoint. When the robot reaches the goal, the red line disappears from our radar and a new link to the next waypoint is displayed.

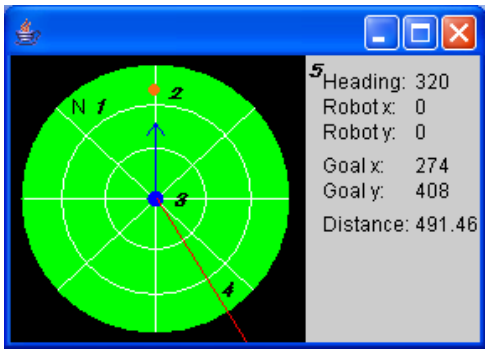



**Figure 9 Radar GUI**

# 6. Performance Analysis

A summary of vehicle predicted and trial data is listed in **Table 2**. The discrepancies between predicted and actual result are mainly due to the "real world" effects of varying terrain and sensor noise not included in prediction calculations.

| Performance Measure | Performance Prediction | Performance Results |
|---|---|---|
| Maximum Speed | 8 Mph | 7 Mph |
| Ramp Climbing Ability (mu = .3516) | 60% | 20% |
| Reaction Time | 1 – 2 Hz | 1.5 Hz |
| Battery Life (104W required) | 1 Hour | 45 Minutes |
| Object Detection | 8 meters | 8 meters |
| Dead-end and Traps | 8 meters deep | 8 meters deep |
| Waypoint accuracy (with Omnistar) | .1 to .8 meters | .6 to .8 meters |

**Table 2 Performance Analysis**

## *6.1 Safety*

The issue of safety was considered early in the initial speculation phase of our agile design process in which requirements for the vehicle were gathered. Electrical safety was addressed by installing circuit breakers inline with the motor power circuit and clearly labeling electrical wiring. The E-stops can be engaged in emergency situations when the robot has to be immediately disabled. After an E-Stop, the vehicle can only be restarted by resetting it manually.

## *6.2 Reliability*

The Mean Time Between Failure (MTBF) represents the average time between failures of a system. The Think-Tank robot hasn't had any hardware failures to date, so the MTBF in this case represents the total time the robot has been assembled. Battery redundancy increases reliability in the event one fails. Also, multiple sources of obstacle avoidance inputs from the Ladar and camera offers increased reliability if one sensor should fail.

## *6.3 Durability*

Durability of the Think-Tank vehicle was measured as the number of times any component on the vehicle required repair or maintenance during testing. None of the components required any repair or significant maintenance during testing of the vehicle.

## *6.4 Testing*

Hardware and software tests were conducted under a three phase test plan which included: unit testing of each component by the developer as the components were created, integration testing as each component

was integrated into the entire system to check that the components functioned properly as a system, and regression tests to verify that previously tested components didn't introduce bugs after integration with other newer components. Practice courses were set up to test the functionality of the robot for both the Autonomous and Navigation Challenges.

### *6.5 Systems Integration*

The systems integration model was a direct benefit from the object-oriented programming model and was aided by the use of Java interfaces and a custom written threading model. Hardware integration was facilitated by the modular design of the chassis and the use of electrical buses to deliver power to each subsystem. Hardware and software systems integration was performed during each increment of the development.

### *6.6 Vehicle Cost Summary*

**Table 3** summarizes the total material cost for the Think-Tank vehicle. Items with a team cost of $0 are components that were loaned to us for the competition.

| Component | Total Cost | Team Cost |
|---|---|---|
| **Chassis Materials** | $82 | $82 |
| (2) 12V DC motors, 285 rpm | $310 | $310 |
| (2) 4" Tire & Wheel Assembly - 10.5" Knobby Tire | $48 | $48 |
| (1) NovaTel ProPack-LB DGPS & GPS-600-LB Antenna | $2,700 | $2,700 |
| (2) Incremental Rotary Optical Encoder Kit | $122 | $122 |
| (1) Digital Compass/inclinometer | $400 | $400 |
| (1) JVC TK-C1480U Color Video Camera | $1000 | $0 |
| (1) PixelLink PL-A544 Video Converter box | $500 | $0 |
| (1) Sick LMS 291-S05 Ladar | $7,000 | $0 |
| (1) Roboteq AX3500 Speed controller /w optical encoder inputs | $395 | $395 |
| (4) Main Battery 12 Volt 7 Ah Power Sonic Sealed Lead Acid | $48 | $48 |
| **Electrical Hardware** | $136 | $136 |
| **Hardware (nuts, bolts, etc…)** | $75 | $75 |
| **Miscellaneous (paint, Velcro, etc…)** | $14 | $14 |
| (1) MPC Laptop | $2,215 | $0 |
| **Total** | **$15,045** | **$4,330** |

**Table 3 Vehicle Cost Summary**

## 7. Conclusion

Think-Tank is an autonomous vehicle designed and constructed by LTU students that incorporates modular extensible software architecture with a robust design capable of challenging the top ranking schools at the 2005 IGVC. New to this years vehicle is a host of sensor capabilities integrated using a virtual environment map. Various design improvements based on previous LTU designs have increased the control, navigation, and vision system accuracy to levels comparable to past event winners.