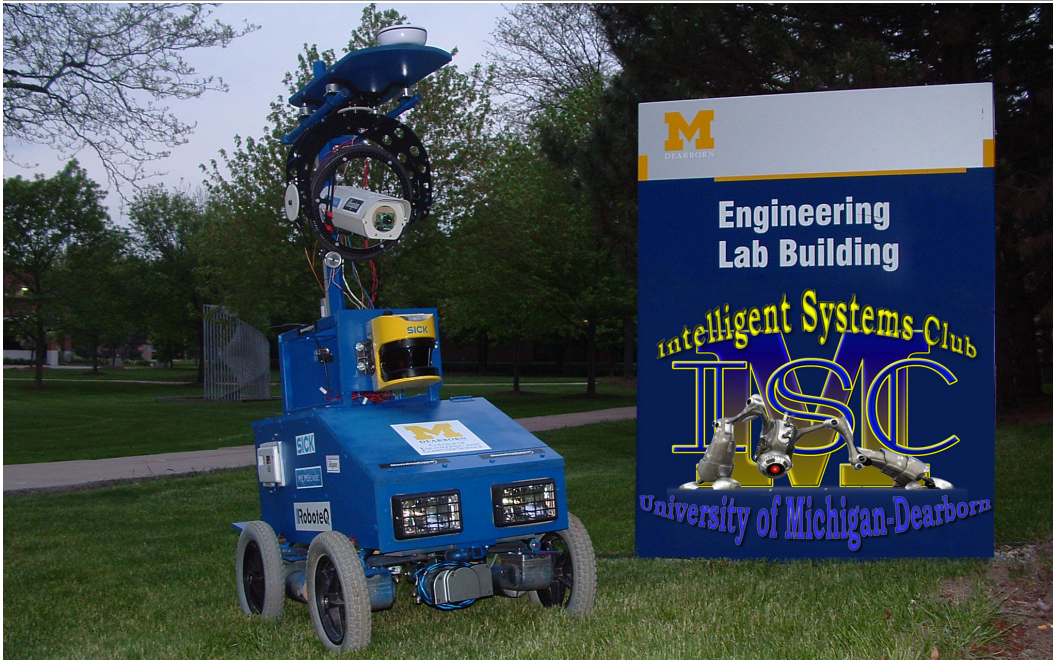


THE UNIVERSITY OF MICHIGAN - DEARBORN

# R A P T O R



## Intelligent Ground Vehicle Competition 2009

### TEAM MEMBERS

Allen Akroush, Gregory P. Czerniak, Nick Evans, Gierad Laput, Lisa Linna,  
Anthony Lucente, Ross Marten, Jason Smith



# Chapter 1

## Design Report

### 1.1 Introduction

The University of Michigan - Dearborn presents *Raptor*, a contender for the 17th Annual Intelligent Ground Vehicle Competition.

A "varsity" group of students from the University of Michigan - Dearborn having at least one year of IGVC experience were assigned to develop a fully autonomous ground vehicle to compete in 2009. The team sought to reduce systems integration overhead by improving the already-proven Raptor robot platform. Due to the Next-Generation Robot Platform (NGRP), the modular software architecture developed in 2008, it was possible to continuously improve and integrate new subsystems to Raptor's software with an iterative design process.

### 1.2 Design Process

#### 1.2.1 Competition Analysis

After Raptor competed in the 2008 IGVC, the team's first action this year was to evaluate what design improvements were necessary for better competition performance. The team's main findings were as follows:

- Raptor's front casters were inadequate for overcoming rough terrain, as well as gaps such as curbs or the beginnings of ramps.
- The GPS unit was mounted in front of a large metal pole, which restricted line-of-sight with satellites.
- Systems integration was an afterthought rather than a major design criteria.
- Due to time constraints and an attempt to write a new code base, a lot of time was spent on the new framework, and not enough was spent on the core path planning code.

- Time constraints forced the software developers to write the majority of the control into one disorganized module instead of a logical data flow.

After some organizational and design introspection, the team decided that despite its flaws, Raptor in 2008 was a suitably-stable unmanned vehicle. Therefore, instead of scrapping Raptor and starting over, the team would continuously improve Raptor's systems with innovations in 2008-2009. To clearly indicate the team's dedication to continuous improvement, the small light bulb icon at the margin indicates that the paragraph contains an improvement to Raptor.



### 1.2.2 Team Formation

The team operated in a very flat, democratic structure. Meetings were held every week where team members would present their newest work or findings. All members reached a consensus on major design decisions before action took place, mirroring Japanese management techniques. That extra collaboration and synergy resulted in better ideas, as well as preventing mistakes early in the design process.

- Lead Autonomous and Navigation Challenge Developer: Gregory P. Czerniak, ECE Graduate
- Mixed Teleoperation/Autonomy Coordinator: Nick Evans, ECE Graduate
- Testing and Quality Control: Gierad Laput, ECE Graduate
- Vision System Developer: Jason Smith, ECE Master's Student
- Camera System Developer: Lisa Linna, ECE Graduate
- Embedded Systems Developer: Anthony Lucente, ECE Graduate
- Contributing Developer: Allen Akroush, ECE Graduate
- Contributing Developer: Ross Marten, ECE Senior

### 1.2.3 Development Process and Systems Integration Plan

To highlight the continuous improvement philosophy of the Intelligent Systems Club, Raptor used an iterative development process, and the team simplified systems integration by enforcing a continuous integration policy. In other words, features would be added to Raptor in iterations, and the features would be integrated and tested into the larger system as soon as they were functional. This afforded the team many benefits:

- Raptor was in a functional, ready-to-compete state at all times.
- Features were integrated one-at-a-time in an iterative development process. Testing was easier since only one unproven component was in the system at any one time.

- By integrating systems throughout the design process, any design flaws in modules that would complicate systems integration appeared gradually throughout the process instead of all at the end.

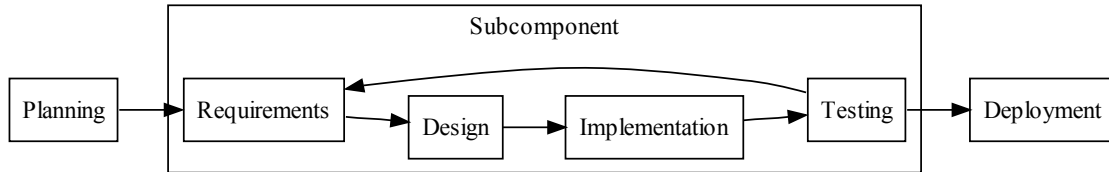


Figure 1.1: A block diagram of the iterative design process.

### 1.3 Hardware

Raptor’s hardware was mostly purchased and integrated during last year’s competition. Since the hardware performed extremely well last year, the team decided to make only minor modifications to the hardware rather than an overhaul. By maintaining a stable hardware base, it was possible to perform continuous software integration without questioning the reliability of the hardware.

Raptor was designed to be a 4-wheel motor vehicle equipped with differential steering and rear-wheel independent suspension. It is constructed in two halves: a chassis welded with square steel struts which housed all internal components, and an adjustable frame structure bolted above the chassis to hold the 3-axis gimbal and the GPS.

As mentioned in the Competition Analysis section, the enclosure that encapsulated the GPS base station caused the GPS system to malfunction. By removing the enclosure, the GPS system was able to function correctly. This critical change was implemented into this year’s entry.

Raptor’s chassis was designed to provide easy-access to its internal controls such as the motor controllers, batteries, emergency stop and power switches. The goal was to have a compact but functional structure wherein the components can flow in a manageable arrangement. Wooden door panels were constructed to provide easy access to the electronic components and the laptop. Moreover, a steel casing equipped with a front access door was mounted on top of the chassis to accommodate the payload. As an extra feature, two headlamps were mounted in front of the chassis to aid the vision system by eliminating shadows and allowing night time control.

### **1.3.1 Electronics**

Similar to the mechanical aspect of Raptor, no changes were made to the electrical and electronic components from the previous year. Since the focus of 2009 was improving the software and intelligent algorithms, electronic components such as the motor, motor controllers, and sensors remained unchanged.

#### **Power and Electrical System**

Raptor was powered by two 12-volt 85 amp-hour marine deep cycle batteries connected in series. This provided 24 volt DC power accessible to all robot subsystems. For all other subsystems needing a different voltage supply other than 24 volts, DC/DC converters rated at 24V-to-12V and 24V-to-5V were used to supply power to subsystems requiring 12V and 5V respectively. This configuration allowed flexible integration between modules with varying power requirements.

#### **Motor**

Raptor used widely-available wheelchair motors. The motors ran at approximately 140RPM with a 12.5-inch wheel, allowing Raptor to meet the IGVC speed requirements of 5mph. The two motors were mounted on the left and right side of the robot and positioned directly opposite each other. With a load of 60 in-lb, the motor ran at about 130RPM at 7.9A. It had a max rated torque of 120 in-lb with an output of 94RPM at 13.2A.

#### **Motor Controller**

The Roboteq AX2850 was the motor controller of choice because it is extremely easy to use and it comes packaged with the necessary features that allowed seamless integration with other subsystems. The controller was equipped with safety features such as the watchdog timer, and several protection features to handle variations between voltage, current, and temperature. Its two channels operate independently which allowed precise control of the direction and speed of the robot. Two 120A circuit breakers were placed on each channel to provide over-current protection.

#### **Sensors**

The robot acquired information from the outside environment through input sensors. These sensors quantized critical information such as lanes, or distances of objects at a certain field of view. The robot collected, combined, and calculated sensor information and decided the appropriate action at any given moment. These sensors provided the robot with the necessary "abilities" it needed for different aspects of the competition. The list below summarizes the different sensors used in Raptor and their corresponding functions:

- Odometry
  - Quad Optical Encoders

- Vision
  - Unibrain Fire-i Digital Camera
  - Used to detect lanes and other obstacles
- Positioning
  - Hemisphere VS-100 Deferential GPS
  - Provided geospatial information used in the Navigation Challenge
- Orientation and Balance
  - Memsense TriRate 3-axis Gyroscope
  - Provided relative angular displacement information
  - Allowed the robot to "grasp" a sense of orientation
  - Mainly used to stabilize the camera
- Accelerometer
  - DE-ACCM3D 3-axis Buffered 3D Accelerometer
  - Provided relative acceleration information
  - Used in conjunction with the gryroscope to stabilize the camera
- Obstacle Avoidance
  - SICK Laser Range Finder
  - Provided 190 data points at a 190-degree field of view (1 degree resolution)
  - At each specific angle, the SICK returned the distance of the closest object
  - Used in autonomous mode to detect and avoid obstacles

### 1.3.2 Computers

For the choice of laptop computer, the following specifications were of main concern: fast processor, large memory, high-performance graphics card, availability of firewire port, and long battery life. Although any computer running Windows OS would work, it was essential that it met the aforementioned specifications to ensure fast processing and low-latency operation. A HP Pavillion DV6700T was chosen since at the time of purchase in 2008, it was the best computer that met all the requirements at the most ideal price. The specifications for the HP Pavilion DV6700T were as follows:

- Processor: Intel Pentium Dual Core T2330 / 1.6 GHz
- Memory: 3 Gigabytes DDR II SDRAM


- Graphics Card: Geforce 8400M GS Graphics Card
- Battery: Lithium Ion Battery with 4 hours of Battery Life
- I/O Ports:
  - 1x FireWire Port (IEEE 8394)
  - 3x USB 2.0 Port
  - 1x Ethernet Port
- Other Specs:
  - Dimensions: 10 x 14 x 1.7 in
  - Weight: 6.1 lbs

## 1.4 Software

### 1.4.1 Strategy


In 2008, the Next Generation Robot Platform (NGRP) framework was written as a software base for Raptor. For this year's competition, the team decided to improve NGRP rather than write a new code base from scratch.

NGRP was originally written to run as a set of modules, each one running as a separate thread of execution on the computer system. Unfortunately, as mentioned in the Competition Analysis section, time constraints in 2008 forced the developers to create one monolithic module in order to have a functional robot in time for competition. This created high coupling and low cohesion in the software, which reduced readability and orthogonality.

The first task for the 2009 version of NGRP was to split the monolithic module into a series of small, single-purpose modules connected in a pipeline. Figure 1.2 shows a data flow diagram of this new pipelined AI approach. 

### 1.4.2 Signal Processing

#### Vision

Raptor uses the Unibrain Fire-I web camera to capture images. This camera was selected for its ease-of-use, reliable 30 frames/sec frame rate, image quality, and accessibility to hardware settings such as the shutter speed. A polarizing filter was mounted in front of the lens to reduce glare. 

A large portion of image processing is offloaded to the GPU using the NVidia Cg language. The parallelized architecture of the GPU is particularly well-suited to pixel-neighborhood operations such as noise removal, morphology, and edge detection. Raptor's vision code also used the GPU for local thresholding and moment of inertia operations, which are not typically possible in real time using only the CPU.

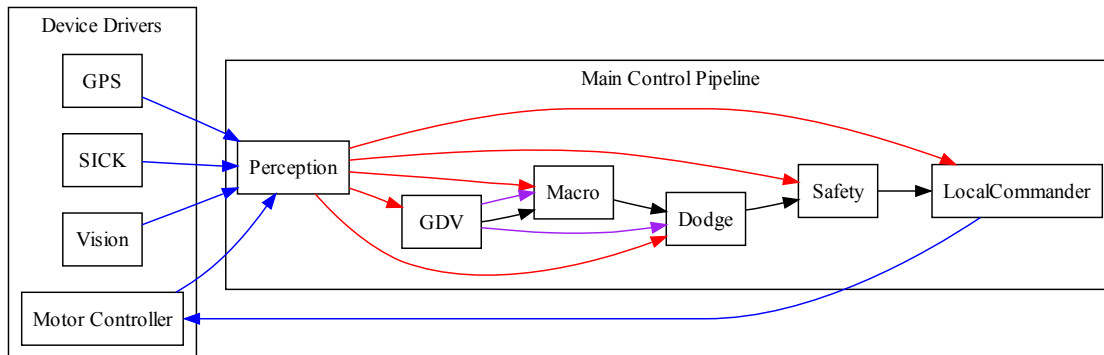




Figure 1.2: A data flow graph of NGRP. Blue arrows are UDP, red arrows are map data, black arrows are differential drive commands, and purple arrows are artificial obstacles.

The Intel OpenCV library was used for CPU-based image processing.

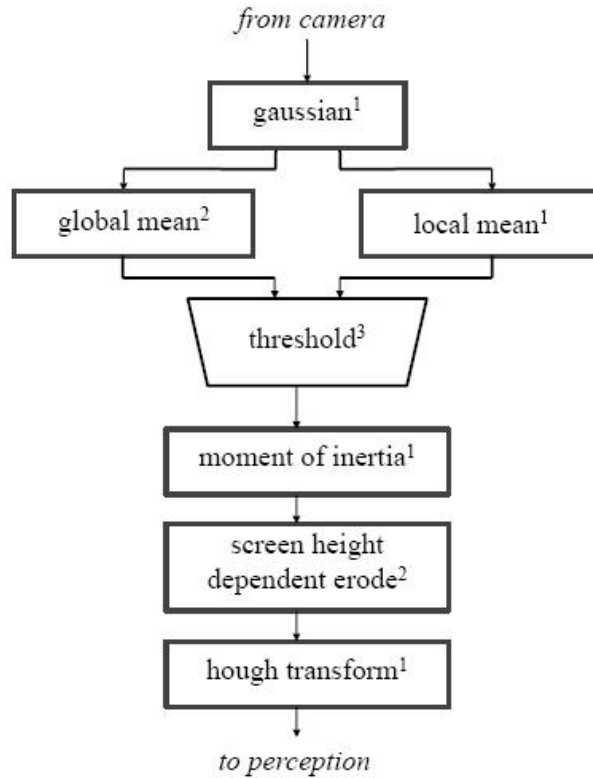
Figure 1.3 shows the sequence of operations used to extract lane markings from the image.

The image is first convolved with a Gaussian kernel to remove noise. The lane markings are white and are typically brighter than the grass, so the next step was to perform a binary threshold that only allows the highest intensity pixels to pass through. A global threshold is determined by computing a histogram for the image and finding the 95th percentile. Since shadows prevent some regions of the image where there is a white line to meet the global threshold, Raptor also uses local thresholding. The parallelized architecture of the GPU makes it possible to use a different threshold for each pixel in the image without a significant loss in performance. The final threshold value for each pixel is between the global and local thresholds. The overall strategy is to use the proven reliability of global thresholding but to make local adjustments to this threshold in case there are shadows. 

Sometimes large groupings of pixels make it through the threshold stage when glare reflects from a sand-trap or ramp. Raptor uses a moment of inertia operator to eliminate these large blobs. The moment of inertia algorithm goes through every pixel and looks at its neighbors along the major and minor axes. A value is calculated for each axis, similar to an inertial moment. This is done repeatedly as the major and minor axes are rotated. If the moment of one axis is significantly greater than the other, the pixel of interest is probably part of a line, so the algorithm allows it to pass through. Otherwise, the pixel is rejected, since a "blob"-like region typically has more symmetric inertial moments along each axis. This is a computationally-intense process and requires a GPU or other similar dedicated hardware for real-time applications. 

Due to perspective, white lines often appear thicker at the bottom of the frame compared to the top. The vision system divides the frame into 3 regions and performs increasing numbers of





- <sup>1</sup> GPU (NVidia Cg)  
<sup>2</sup> CPU (Intel OpenCV)  
<sup>3</sup> decision function performed on GPU that arbitrates between global percentile thresholding and local thresholding based on predetermined weights

Figure 1.3: Raptor's computer vision pipeline.

morphological erosions toward the bottom of the frame. This also has the benefit of eliminating small patches of dead grass that are often visible directly in front of the robot.

Finally, the code uses the OpenCV implementation of the Hough transform to extract lines from the remaining image. The endpoints of the Hough lines are placed in UDP packets and read by the Perception module.

## Mapping

After signal processing, each component supplies a piece of the overall situation. However, the fusion of this information into a map is greater than the sum of its parts. To perform mapping duties, NGRP contains a module called Perception, which receives all incoming sensor information and performs mathematical transforms to bring it all to a single point of reference (the map).

This may sound simple, but due to the fact that all sensors have error, it is actually a very hard problem. The small errors in all sensor data compound on itself, and over time the map becomes absolutely incomprehensible. To mitigate the problem of compounding errors, Perception

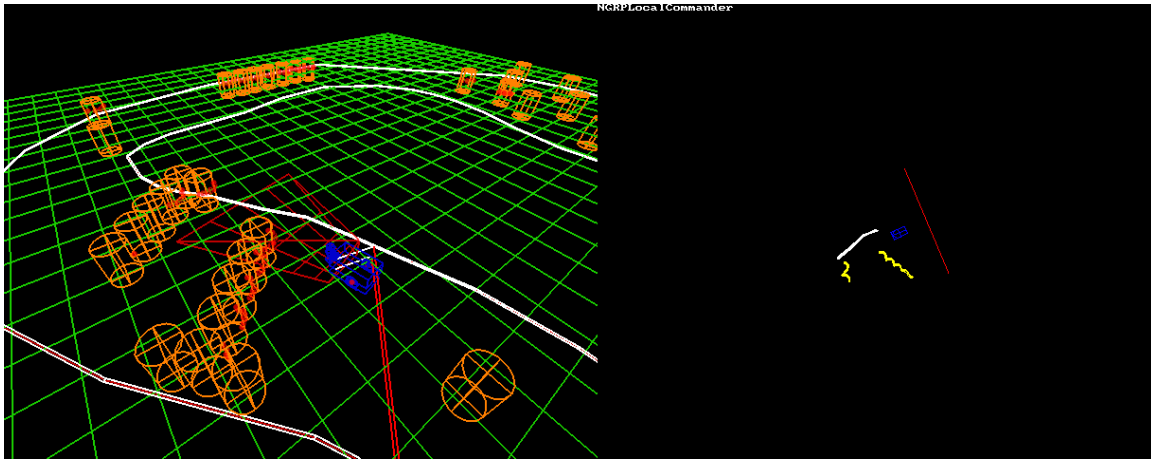




Figure 1.4: A simulated IGVC course and the real-time map of it produced by NGRP.

deliberately forgets map information after a period of time specified in the robot's configuration file. Since Autonomous Challenge courses are typically linear one-way paths, this forgetfulness is an acceptable design tradeoff, as proven by extensive simulated and real-world testing.

### 1.4.3 Autonomous Path Planning


After NGRP generates a local map of the environment, the next task is to determine the robot's trajectory. To do this, the 2009 version of NGRP uses a pipelined approach, starting with a very generalized direction vector and feeding it through a series of behavioral filters that make the trajectory more and more specific to the immediate situation. 

#### Long-term (GDV)

In the Autonomous Challenge, it is very common for robots to turn the wrong way on the course, and Raptor was no exception in 2008. This is caused by control software that lacks knowledge of the general direction that the robot should be headed. The Intelligent Systems Club terminology for this direction is the General Direction Vector (GDV). The first step of the NGRP 2009 path planning pipeline is a module that records and maintains the most current GDV. 

To calculate the GDV, NGRP assumes that the robot will initially be pointed in an initially-correct GDV at the start of an autonomous run. Since IGVC autonomous courses are not straight, the GDV must change to account for turns. This is done by assuming that if the robot is going straight on its own volition, the GDV should be changed to the robot's current direction. This is based on an assumption that if the robot is not turning, it is most likely on the right track. Both simulator and real-world testing has repeatedly shown that this assumption is sufficient for the majority of cases.

Although the GDV would theoretically always keep the robot in the correct direction, testing has shown that sometimes it is not enough. To mitigate this, a line segment behind the robot

perpendicular to the GDV known as the Virtual Wall functions as a "virtual obstacle" to prevent the robot from turning around or moving to the wrong direction. When the GDV changes, the Virtual Wall updates accordingly. This provides second layer of turnaround prevention on top of the GDV. 

### Mid-term Planning (Macro)

After a General Direction Vector had been determined, the result is sent to a mid-range planner known as Macro. The Macro algorithm is a sweep algorithm, which casts 180 rays from right to left in front of the robot in a behavior similar to the SICK LIDAR, but using the full map information. These lines are returned in the form of distances to the nearest obstacle. The 2008 version of Raptor assumed that the direction of the line with the farthest distance was the direction the robot should go. The problem with that approach was that it did not account for the general direction the robot should be going.

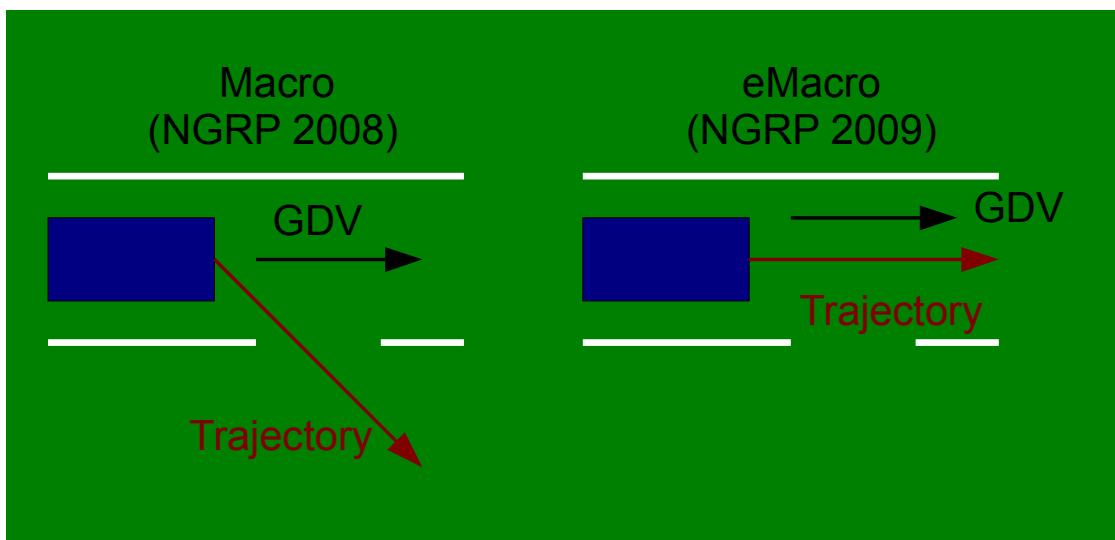




Figure 1.5: A comparison of Macro and eMacro.

Simply following the direction that leads to the farthest distance without taking the GDV into account can result in the robot making fatal decisions such as going off course on dashed lines, as Figure 1.5 shows. As a result, NGRP 2009 has a new version of Macro known as Enhanced Macro (eMacro). Rather than simply using distance as the determining factor for direction, eMacro takes  the dot product of the GDV and each sweep line. This modification causes eMacro to go in the direction that leads to the greatest forward distance toward the general direction vector.

## Obstacle Avoidance (Dodge)

After Macro completes execution, the trajectory is fed into a module named Dodge. Dodge's logic is extremely simple: it simply turns away from the direction of the closest obstacle.

In the 2008 NGRP, Dodge's effective range was unrestricted, which led to Raptor overreacting to far-away obstacles. In the 2009 NGRP, Dodge's influence is restricted to only obstacles one meter away or less. This greatly improves the overall AI pipeline by only causing reactive behavior when it is absolutely needed. 

### 1.4.4 Navigation Challenge

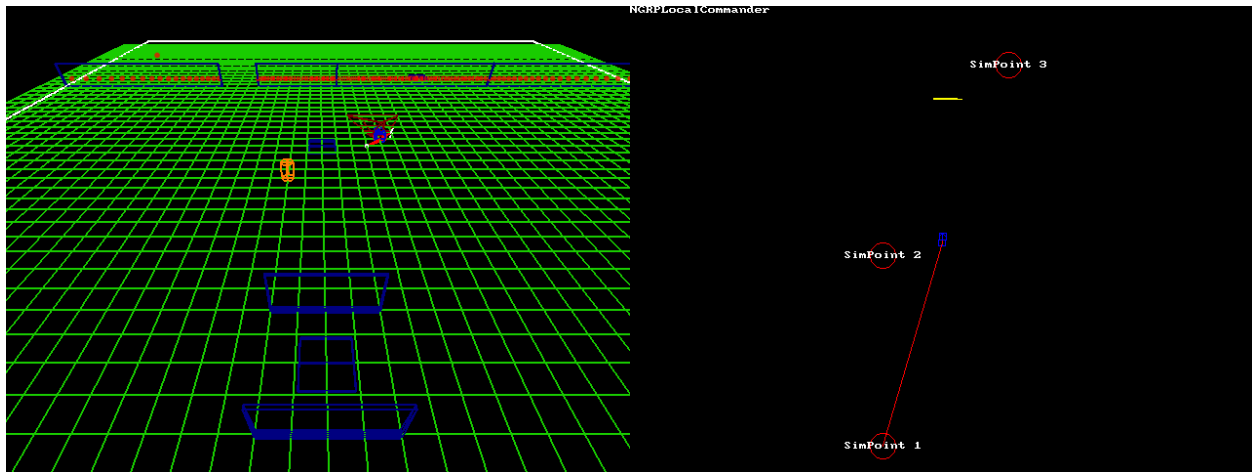


Figure 1.6: A simulated instance of the Navigation Challenge on the left, and NGRP on the right.

The pipelined approach to writing NGRP made the design for the Navigation Challenge easier, since the majority of the control pipeline could be reused. In order to implement the functionality of the Navigation Challenge, a module was created that would read a list of GPS coordinates from a file. The robot would then proceed to each waypoint in the same order as the list in the file.

After the initial loading procedure, the module would constantly do three things in an infinite loop:

1. Check to see if the robot has reached the current waypoint.
2. If the robot has reached the waypoint, set the current waypoint to the next waypoint on the list. If there are no remaining waypoints, stop.
3. If the robot has not reached the waypoint, set the General Direction Vector to the as-the-crow-flies trajectory between the robot's location and the waypoint.

### 1.4.5 Simulation

A major issue with testing pertains to the fact that multiple developers cannot simultaneously access the robot. Moreover, evaluating the robot's performance in an obstacle course requires access to a large area. Setting up a large course (similar to the ones used in competition) is not feasible.

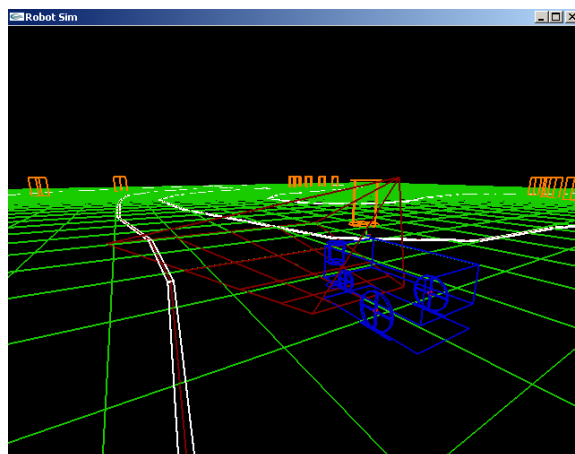



Figure 1.7: A screenshot of the simulator.

Software simulation resolves these critical issues by providing a virtual testing environment which can run on a computer equipped with Windows and a 3D-accelerated graphics card. Through the use of software simulation, software changes are readily implemented and evaluated without requiring any physical access to the robot. Furthermore, the simulator is capable of sending data in exactly the same manner as the real sensor data.

The software simulator renders a virtual environment complete with obstacles that are commonly found in the competition such as lanes, barrels, and ramps. Furthermore, a program written in conjunction with the software simulator provides the ability to easily design virtual courses. With this feature, a course can be designed to evaluate the performance of a specific algorithm. This was an improvement from the previous year where courses were designed by manually hardcoding data points to the simulator. 

## 1.5 Safety

The University of Michigan - Dearborn Intelligent Systems Club placed safety as its absolute highest priority. The safety ramifications of major design decisions were discussed throughout the design process.

The Intelligent Systems Club considered the safety requirements enumerated in the IGVC rules to be a minimum baseline for physical safety, and Raptor met these standards. There was a mechanical emergency stop on the rear of Raptor that functioned as an electrical switch to the

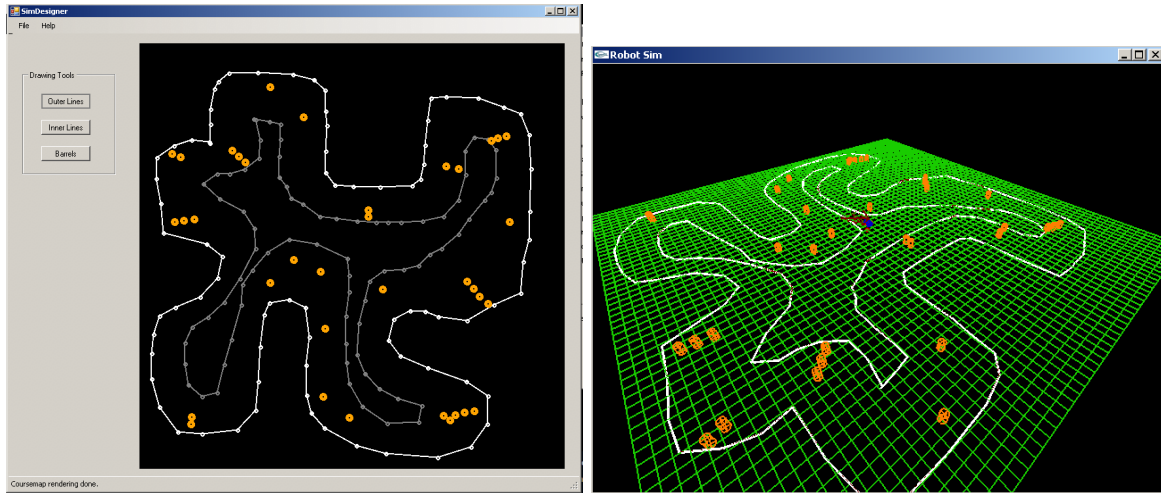




Figure 1.8: Screenshot of a program for designing simulation courses, and a screenshot of the course in the simulator.

main power circuit, and would cut power to everything if the button was pressed. There was also a wireless emergency stop effective to 50 feet that would cut electrical power to the vehicle when engaged. In addition to the IGVC requirements, in order to use a manual controller to drive the robot outside the autonomous modes, one had to hold down a dead-man's button to prevent accidental movement commands from causing crashes.

To supplement the standard IGVC safety requirements, the software also contained many safety features. The final module in any autonomous control pipeline on NGRP was called Safety, and it contained several final sanity checks made to trajectories before they were sent to the motor controller.

The first sanity check was a forward crash detector, which stopped Raptor if the robot was being instructed to drive directly into an obstacle. In 2009, a sanity check was added for keeping the robot from making turns into obstacles. Also, to prevent the robot from turning so fast that it may not have been able to issue a corrective command in time to prevent a crash, there was a check that placed upper bounds on the robot's turning rate. These safety checks significantly reduced the number of accidents that occurred during testing runs.  

## 1.6 Performance and Cost Estimate

Tables 1.1 and 1.2 at the end of the paper contain the information for cost and performance characteristics, respectively.

## 1.7 Conclusion

The University of Michigan - Dearborn Intelligent Systems Club believes that its approach to the design of Raptor in 2009 will result in significantly higher performance in the 17th Annual International Ground Vehicle Competition. By focusing on improving the existing stable Raptor platform with minor modifications to the hardware, more time could be spent developing more intelligent algorithms for path planning and navigation. Embracing an iterative design process coupled with continuous integration resulted in a more stable platform throughout the product lifecycle. The team spent approximately 2000 man-hours total on this project incorporating these changes, and the team believes that Raptor will perform well in 2009.

## 1.8 Acknowledgements

The team would like to thank Larry Sieh for his help and support, as well as our many sponsors for their invaluable financial and equipment donations.

## 1.9 Signature

I certify that the design and creation of Raptor has been significant and is equivalent to what might be awarded credit in a senior design course.

---

Dr. Narasimhurthi Natarajan

Professor of the Department of Electrical and Computer Engineering, University of Michigan - Dearborn

---

Date

#	Description	Actual Cost	Cost To Team
<b>Chassis Construction / Mechanical Design</b>			
1	Square Metal Tubing and Wood Panels	288.50	288.50
4	Electric Wheelchair Wheels 12.5 Wheel	743.80	60.00
4	Wheelchair Electric Motors	1,864.00	240.00
2	Two Ton Jacks	120.00	120.00
4	Shock absorbers (off-Go Kart, ATV, Mini Bike, Etc)	80.00	80.00
<b>Motor Control Circuitry</b>			
1	Roboteq Motor Controller Ax2850	620.00	575.00
<b>Robot Computer</b>			
1	Dell Laptop Computer	999.00	999.00
<b>Microcontrollers</b>			
1	Camera Gimbal, Embedded Arm Board	332.00	332.00
<b>Sensors</b>			
2	Wheel Encoders: Quad Optical Encoders, same ones as on WOLF	338.00	316.00
1	Sick Laser Sensor	5273.00	389.00
<b>GPS</b>			
1	Hemisphere VS100 Series DGPS	4,195.00	0.00
<b>Camera</b>			
1	Fire-i Digital Board Camera-Camera with Wide-Angle Lens	130.00	130.00
1	CopterWorks 3-Axis Gimbal - Pan, Tilt, Yaw	2,520.00	650.00
1	MEMSense 3-Axis Rate Gyroscope	810.00	0.00
1	3-Axis DE-ACCM3D Accelerometer	35.00	35.00
1	Servo Controller	39.95	39.95
1	Linear Actuators	109.00	109.00
1	Camera Enclosure	54.95	54.95
<b>Controller Transmitter and Receiver</b>			
1	Wireless Emergency Stop Receiver	60.00	60.00
1	Wireless Emergency Stop Transmitter (1500 feet)	37.00	37.00
<b>Accessories</b>			
2	Transcontinental Deep Cycle Batteries 12V 85AH	130.00	130.00
1	Multi-Bank Battery Charger: 5/5amp 2 Bank Charger	189.99	189.99
1	Mean Well USA, 24v to 12v DC/DC Converter	185.00	0.00
1	Mean Well USA, 24v to 5v DC/DC Converter	185.00	0.00
1	Exterior lights for eye catching appeal	100.00	100.00
1	Random Switches, Wires, Connectors, Electronic Parts, Enclosures, Etc	500.00	500.00
<b>6% Sales Tax</b>		\$1,196.34	\$326.12
<b>Total</b>		\$21,135.34	\$5,761.52

Table 1.1: The table of expenses for Raptor.

Attribute	Design Prediction
Maximum Speed	5.0mph
Climbing Ability	30 degree ramp
Nominal Power Consumption (Watts = Amps x Volts)	500 watts
Battery Operating Time (24v 85AH Battery System)	4 hours
Distances at which objects can be detected	5.5 meters
Waypoint accuracy (DGPS)	<60cm 95% of the time
Reaction Times	60-120 ms depending on configuration
How vehicle deals with complex obstacles	Reactive Fuzzy Logic approach

Table 1.2: Performance attributes of the Raptor.