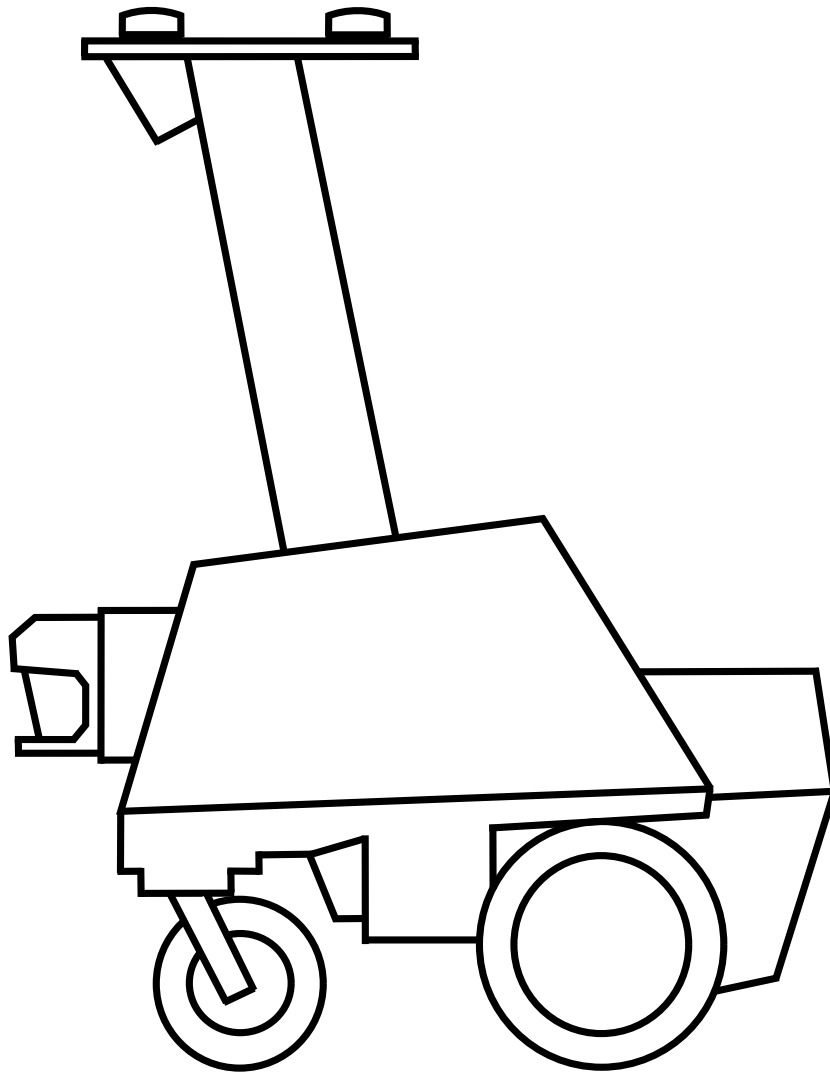


University of Michigan-Dearborn

WOLF 2010



18th Annual Intelligent Ground Vehicle Competition

Allen Akroush, Gregory P. Czerniak, Ross Marten, Jon Smereka, Jason Smith, Jakob Yonan

1 Introduction

The University of Michigan - Dearborn Intelligent Systems Club presents *Wolf*, a contender for the 18th 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 2010. The team sought to reduce systems integration overhead by further developing the already-proven “Wolf” robot platform.


2 Design Process

2.1 Competition Analysis

After *Wolf* competed in the 2009 IGVC, the team’s first action was to evaluate what design improvements were necessary for better performance. The team’s main findings were as follows:

- Although performance was not hindered by the rainy weather experienced at the competition, the chassis should be more water resistant.
- The camera protruded too far in front of the vehicle, increasing the risk of collision.
- Odometry data was not adequately filtered, resulting in erroneous local mapping.
- Obstacle avoidance and path planning routines worked well in the simulator but were not adequately tested in the field.
- Last minute changes to the software introduced unpredictable behavior.

2.2 Innovations

Despite its flaws, *Wolf* has been an extremely dependable robotics platform. Rather than start over with a new chassis, the team decided to improve *Wolf* with a combination of hardware and software revisions. A small light bulb icon has been placed throughout this document to highlight notable innovations over previous designs. Those innovations are summarized below. 

- A Simultaneous Localization and Mapping (SLAM) algorithm has been implemented (Section 4.3.3)
- Newly developed data logging software (Section 4.2)
- Improved vision algorithm (Section 4.3.1)
- Improved autonomous path planning (Section 4.4)
- Previous GPS unit replaced with a DGPS unit (Section 3.1)
- Reduced CPU usage in the constantly-running data visualization software (Section 4.6)

2.3 Team Formation

The Intelligent Systems Club is a volunteer-based student-run organization. The club provides an atmosphere of mentorship, where faculty advisors instruct students on topics not necessarily covered in standard engineering courses. Some students receive academic credit for their contributions; many participate simply for the opportunity to solve interesting problems while furthering their engineering and research skills.

Since most students do not live near campus, the team has come to rely heavily on online collaborative tools including Google Wave and Subversion. Weekly meetings are also held, where members discuss ideas for new designs or assist each other in troubleshooting a problem. Since the team is largely volunteer-based, authority is delegated equally to all members. Major design decisions are reached collectively when all team members feel comfortable with a particular course of action.

- Allen Ackroush: Contributing Developer, Testing, Quality Control
- Gregory P. Czerniak: Autonomous and Navigation Challenge Developer
- Ross Marten: Contributing Developer
- Jon Smereka: SLAM Developer
- Jason Smith: Vision System Developer
- Jakob Yonan: Contributing Developer

2.4 Development Process and Systems Integration Plan

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

- Wolf 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.

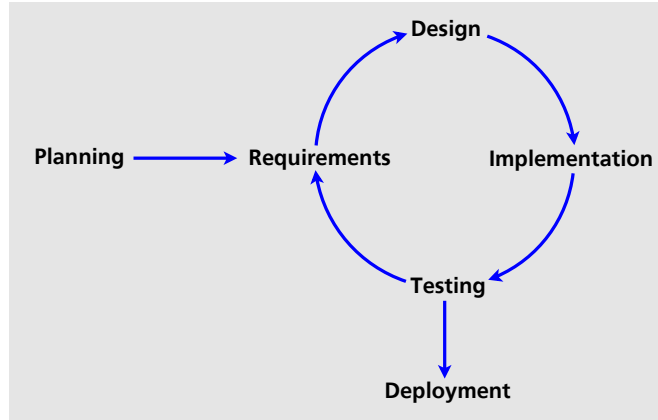


Figure 1: The iterative design process.

3 Hardware

Wolf's chassis was designed to provide easy-access to its internal components 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. The overall design uses front mounted casters and two independent drive wheels for differential steering.

3.1 Electronics

3.1.1 Power and Electrical System

Wolf is powered by two 12-volt 85 amp-hour marine deep cycle batteries connected in series. This provides 24 volt DC power accessible to all robot subsystems. For all other subsystems needing a 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.

3.1.2 Motors

Wolf used widely-available wheelchair motors. The motors can run at approximately 140RPM with a 12.5-inch wheel mounted directly on the shaft, allowing Wolf to travel up to 5mph. The two motors are mounted on the left and right side of the robot and are positioned directly opposite each other. With a load of 60 in-lb, the motors ran at about 130RPM at 7.9A each. Each motor has a max rated torque of 120 in-lb with an output of 94RPM at 13.2A.

3.1.3 Motor Controller

The Roboteq AX2850 was chosen because it is reliable and integrates easily with other subsystems. It is equipped with a watchdog timer and several other protective measures to handle variations

between voltage, current, and temperature. Its two channels operate independently, allowing precise control of the direction and speed of the robot.

3.1.4 Sensors

Wolf acquires information from the outside environment through a variety of sensors. These sensors provide critical information such as lane detection or proximity to obstacles. The list below summarizes different sensors used in Wolf 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
- Obstacle Avoidance
 - SICK Laser Range Finder
 - Provides 180 data points at a 180-degree field of view
 - At each specific angle, the SICK returned the distance of the closest object
 - Used to avoid collisions

3.2 Computer

The specifications for the HP Pavilion DV6700T used to run both NGRP and the vision software are as follows:

Processor	Intel Pentium Dual Core T2330 / 1.6 GHz
Memory	3 Gigabytes DDR II SDRAM
Graphics Card	Geforce 8400M GS
Battery	Lithium Ion / 4 hours of life
1x	FireWire Port (IEEE 8394)
3x	USB 2.0 Port
1x	Ethernet Port
Dimensions	10 x 14 x 1.7 in
Weight	6.1 lbs

Table 1: Laptop specifications.

4 Software

4.1 Strategy

In 2008, the Next Generation Robot Platform (NGRP) framework was written as a software base for Wolf. 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. This was originally architected as one monolithic module. This created high coupling and low cohesion in the software, which reduced readability and orthogonality. The software was later revised in 2009 so that the single monolithic module could be split into a series of smaller modules connected in a pipeline. Figure 2 shows a data flow diagram of this improved pipelined AI approach.

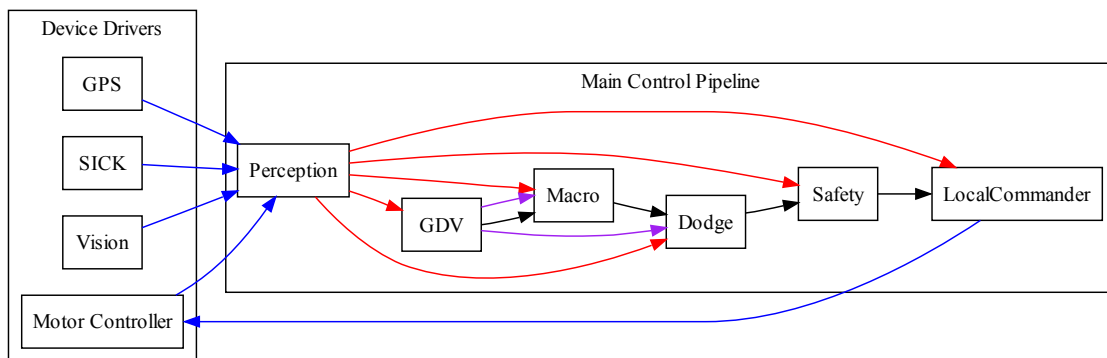



Figure 2: A data flow graph of NGRP. Blue arrows are UDP, red arrows are map data, black arrows are differential drive commands, and purple arrows are artificial obstacles.

4.2 Data Logging

Since its inception, NGRP lacked recording functionality. Without this feature, testing was ad-hoc since the team could not isolate bug-causing conditions. This year, the team added this feature to NGRP. 

Several design considerations went into the logging subsystem:

- **Extensibility:** Robot technology constantly changes. The logger should be able to support new data streams.
- **Future-proofing:** As the logging format is extended, older logs should not become unreadable.

- Consistency: Recordings should play back in the same order and at the same speed as the original event.
- Performance: Log recording must not exceed the throughput limits of a hard drive.
- Long-term storage: Logs must not take up excessive disk space.

In the end, the team chose to design an ASCII text-based format based on the recommendations of Chapter 5 of *The Art of Unix Programming* by Eric S. Raymond. Text formatting future-proofs the format compared to a binary format by not defining fixed sizes for data. The format is extensible by having each packet of data occupy one line of text with a beginning keyword as the first entry of the line. To add new data types, the programmer simply adds a new keyword to the format. All lines are timestamped with millisecond-level precision for consistency.

Several team members questioned the performance ramifications of a text-based format. After analysis, the conclusion was that a text stream containing all of Wolf’s raw sensor data would require approximately 900 kilobytes per minute of data transfer. This is well within the performance characteristics of even the slowest hard drives on the market.

Analysis also indicated that the text-based recordings had a 1:5 compression ratio on average. An hour of log data occupies 52.7 megabytes uncompressed but only 10.5 megabytes compressed. In the text-based format, a 650 MB compact disc can store nearly 62 hours of raw sensor data. The team decided that this was acceptable.

Figure 3 shows a sample text file generated by the data logger.

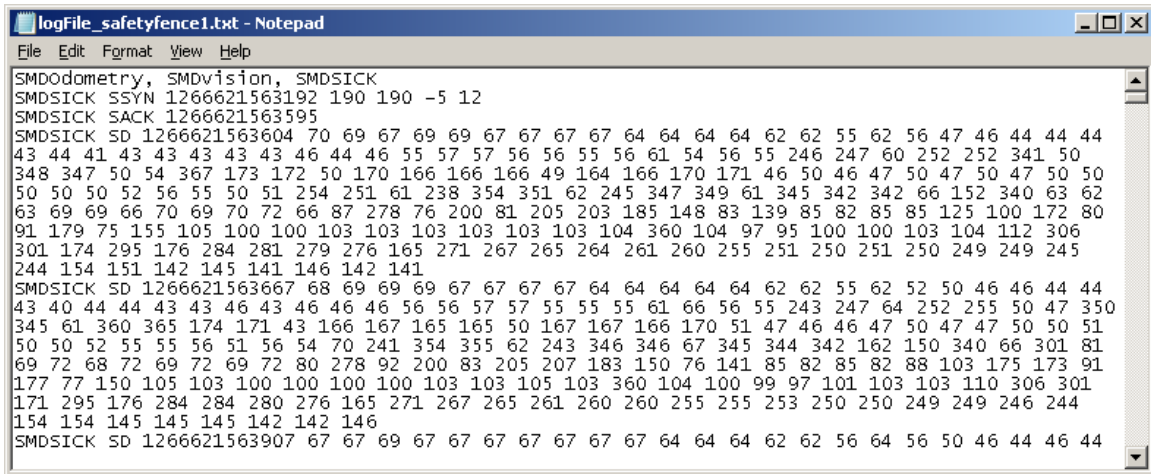


Figure 3: Sample log file.

4.3 Signal Processing

4.3.1 Vision


After reviewing the performance of the image processing system used in 2009, the team focused on improvements in noise removal and glare reduction. The current version of the software now includes a bilateral filter, which removes noise while preserving edges, and a newly modified moment of inertia operator. 

Image capture was performed with a Unibrain Fire-I web camera. 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. The vision system also uses the GPU for bilateral filtering and moment of inertia operations, which are not typically possible in real time using only the CPU.

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

Figure 4 shows the sequence of operations used to extract lane markings from the image.

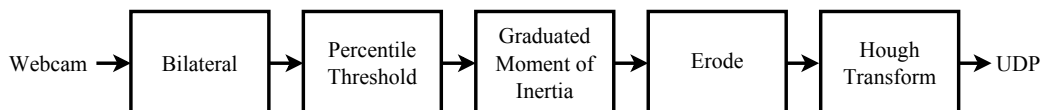



Figure 4: Wolf’s vision pipeline.

The image is first processed with a bilateral filter to remove noise while preserving edges. The lane markings are white and are typically brighter than the grass, so most of the grass can be eliminated by thresholding. A global threshold is determined by computing a histogram for the image and finding the 95th percentile.

Sometimes large groups of unwanted pixels make it through the threshold stage when sunlight reflects off of the ramp. The vision system 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. For 2010, the moment of inertia operator has been modified to include a graduated threshold. Since objects at the top of the frame are more distant and appear smaller, the requirements for determining an object’s likeness to a white line vary as the algorithm descends vertically through the image. 

Finally, the vision system performs a morphological erosion to get rid of any stray pixels and then uses the OpenCV implementation of the Hough transform to extract lines from the remaining

image. The endpoints of the Hough lines are stored in UDP packets that are read by the perception module.

4.3.2 Mapping

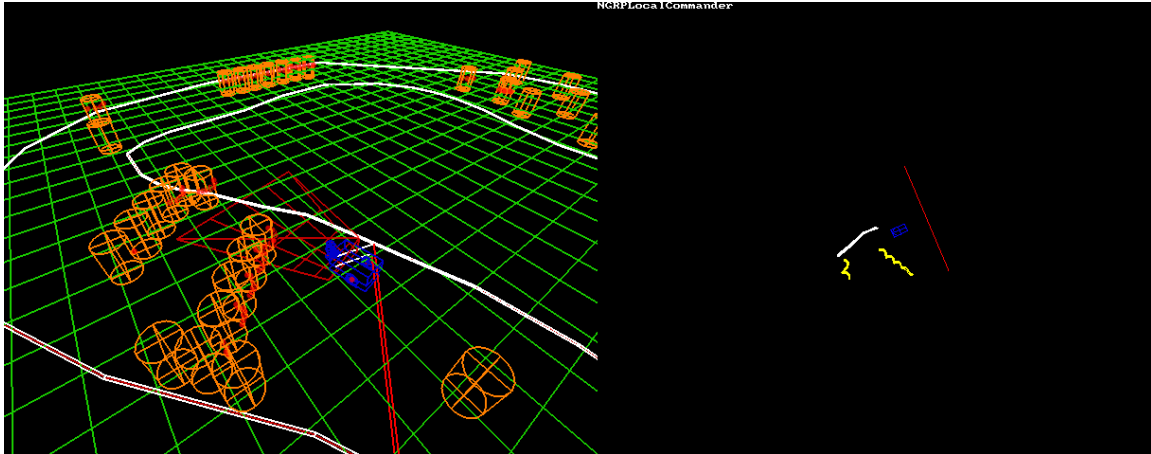



Figure 5: A simulated IGVC course and the real-time map of it produced by NGRP.

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).

4.3.3 SLAM

Simultaneous localization and mapping (SLAM) is an approach used by robots to “discover the world”. Sensing data is used to gather landmarks and continuously adjust the map to the movements of the robot. Of the several benefits for this, the most prominent is the increased ability to apply path finding algorithms to autonomous vehicles. 

There are several approaches, however the most popular is to use an Extended Kalman Filter. The Kalman Filter is a set of mathematical equations that provide a control system-like approach to predicting movements and averaging sensor data. The Extended Kalman Filter does this by taking a 'snap-shot' of a non-linear system, such as an autonomous vehicle, per time step, by taking the derivative of the equations (finding the Jacobian matrices within the sets of equations).

As the system is still experimental and still evolving, the product that is presented at the competition may have a different approach or even not have the SLAM included due to performance constraints. However, the steps to implementing the SLAM are as follows:

- Get sensor data - in this case, we used a SICK at the front of the robot.

- Compare with current data - expanded lines into rectangular planes of varying size, use the separating axis theorem for planar intersection, and if an original line, add to the database of landmarks.
- Update Kalman Filter - compute Kalman Gain, update the measurement matrix for the current position, and update the error covariance matrix.
- Predict ahead using Kalman Filter - update state and projected error covariance.

Each item has a vast amount of work that can go into perfecting the overall system. For example, by using several different data collecting sources, we can narrow down false positives in sensor data as well as remove an unknown sensor noise factor. When comparing data obtained by the sensors, spike landmarks can be singled out and clusters of such can be defined using the k-mean algorithm, to be treated as one individual landmark.

Finally, the entire system can be compartmentalized such that there is one overarching filter controlling several small filters in a tree-like fashion. Then, rather than updating one large matrix, several small matrices are updated by relation to the overarching filter's updates. This approach was used on the winning vehicle for the DARPA Grand Challenge, and is planned for development in the future.

4.4 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, 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.

4.4.1 Long-term (GDV)

In the Autonomous Challenge, it is very common for robots to turn the wrong way on the course. 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 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.

4.4.2 Mid-term Planning (Macro)

The General Direction Vector (GDV) is then sent to a mid-range planner called Macro. Macro is a sweep algorithm that casts 180 rays from right to left in front of the robot and chooses the ray with the longest distance as the steering direction. In 2009 we improved Macro by comparing the dot product of each ray against the GDV to make it favor directions closer to the GDV. This kept the robot inside the track much more often.

Despite this improvement, Macro still had a major problem: Wolf would consistently hug one side of the track and sideswipe obstacles as the robot turned. Macro did this because it was not designed around the predominant shape of the Autonomous Challenge track: corridors.

As you can see in the red ray graphs in Figure 6, openings in the track appear as columns of rays with the same distance. What happens if a column of rays are the longest? Macro 2009 would steer to the side of the column, since it would detect the first "longest ray"s as the absolute longest and ignore all the others. In contrast, Macro 2010 sees the sweep as a series of columns and aims the robot toward the center of the longest column. The result can be seen in Figure 6: the robot no longer hugs walls and obstacles. 

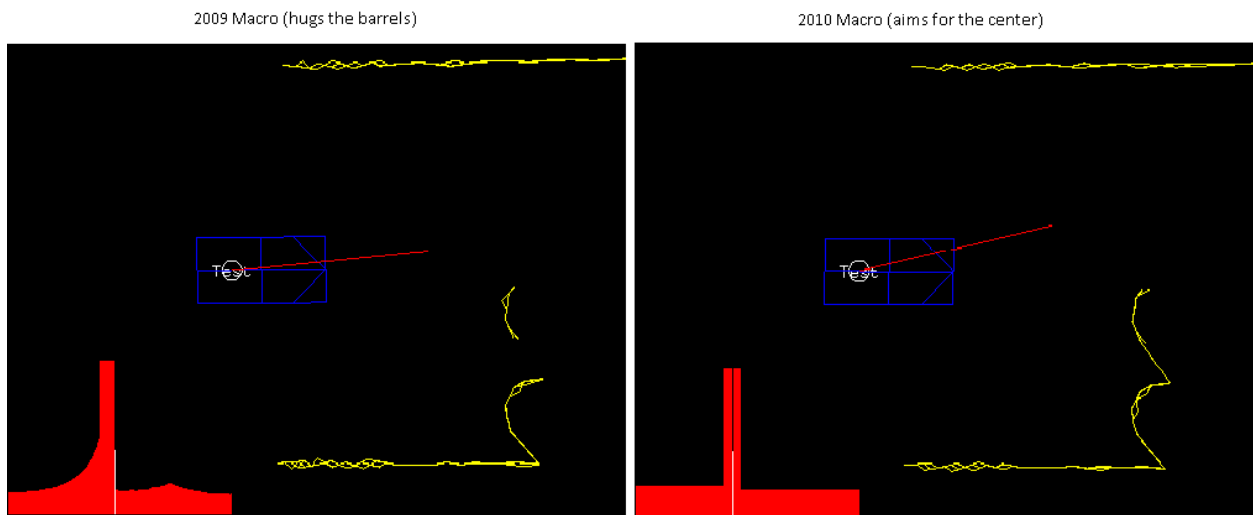


Figure 6: A comparison of 2009 Macro and 2010 Macro.

4.4.3 Obstacle Avoidance (Dodge)

After Macro completes execution, the trajectory is fed into a module named Dodge. Dodge’s logic is minimal: it simply turns away from the direction of the closest obstacle. Currently Dodge’s influence is restricted only to obstacles one meter away or less. This way, simple reactive behavior only occurs when it is absolutely needed.

4.5 Navigation Challenge

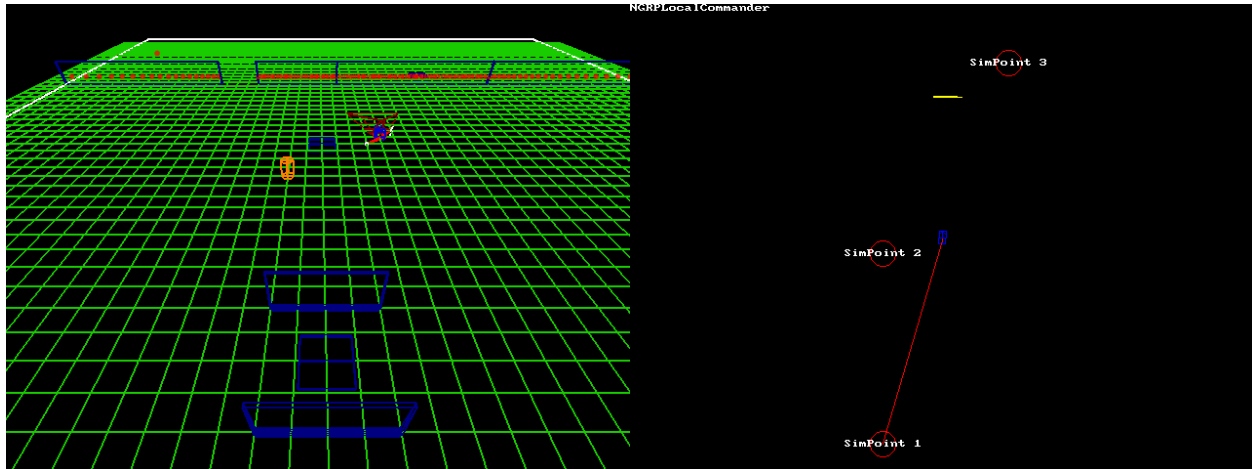


Figure 7: A simulated instance of the Navigation Challenge on the left, and NGRP on the right.

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.

4.6 Plotting

After profiling NGRP’s code for performance, the team discovered that NGRP was using 90% of available CPU power on the onboard computer, and the vast majority of that 90% was used to

draw the map to the screen with a software renderer. At the same time, testers and students who received demonstrations of the robot would consistently complain that NGRP's two-dimensional map was hard to understand. This led to two problems at the same time: the laptop would lose its charge too quickly, and the team's turnover rate was too high since many students would give up trying to understand what was going on.

To fix this, the team wrote a new plotter module that used OpenGL instead of software rendering. This diverted the task of drawing the map to the hardware-accelerated graphics card, which reduced Wolf's CPU usage to less than 10%. At the same time, the team added a new 3D plotting view. The 3D plot is also considerably more simple to understand, leading to more students and testers interested in the project.



Figure 8: New OpenGL-based plotter.

4.7 Simulation

A simulator has been developed so that high level path planning algorithms can be tested when it is either impractical or impossible to use the robot outdoors. The simulator also provides a type of immediacy not available with physical testing: innumerable course configurations can be tested one after the other with no set-up time.

The simulator sends data in exactly the same manner as the real sensor data so that the actual software that runs on the robot can be tested with no special modifications. Having a robust simulator coupled with newly developed data logging software has made it possible to analyze high level behaviors and isolate bugs more closely than ever before.

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 Wolf met these standards. There was a mechanical emergency stop on the rear of Wolf that functioned as an electrical switch to the 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 Wolf 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.

6 Performance and Cost Estimate

Tables 2 and 3 at the end of the paper contain the information for cost and performance characteristics, respectively.

7 Conclusion

The University of Michigan - Dearborn Intelligent Systems Club believes that its approach to the design of Wolf in 2010 will result in significantly higher performance in the 18th Annual International Ground Vehicle Competition. By focusing on improving the existing stable Wolf 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 Wolf will perform well in 2010.

8 Acknowledgements

The team would like to thank Larry Sieh, Professor Nattu, Anthony Lucente, and Jonathan Hyland for their help and support, as well as our many sponsors for their invaluable financial and equipment donations.

#	Description	Actual Cost	Cost To Team
Chassis Construction / Mechanical Design			
1	Chassis	\$250.00	\$250.00
2	Wheels	\$80.00	\$80.00
2	Casters	\$40.00	\$40.00
1	Painting Supplies	\$28.90	\$28.90
1	Mounting Supplies	\$49.47	\$49.47
Motor Control Circuitry			
1	Roboteq Motor Controller	700.00	700.00
Computer			
1	HP Laptop Computer	999.00	999.00
Sensors			
2	Wheel Encoders: Quad Optical Encoders	338.00	316.00
1	Sick Laser Sensor	5,273.00	389.00
GPS			
1	Hemisphere VS100 Series DGPS	4,195.00	0.00
Camera			
1	Fire-i Firewire Webcam with Wide-Angle Lens	130.00	130.00
Controller Transmitter and Receiver			
1	Wireless Emergency Stop Receiver	60.00	60.00
1	Wireless Emergency Stop Transmitter (1500 feet)	37.00	37.00
Accessories			
2	Batteries	320.00	320.00
	6% Sales Tax	\$750.02	\$203.96
	Total	\$13,250.39	\$3,399.37

Table 2: Total expenses.

Attribute	Design Prediction
Maximum Speed	5.0mph
Climbing Ability	30 degree ramp
Nominal Power Consumption (Watts = Amps x Volts)	240 watts
Battery Operating Time (24v 55AH Battery System)	6 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 3: Performance attributes.

I certify that the design and creation of Wolf 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