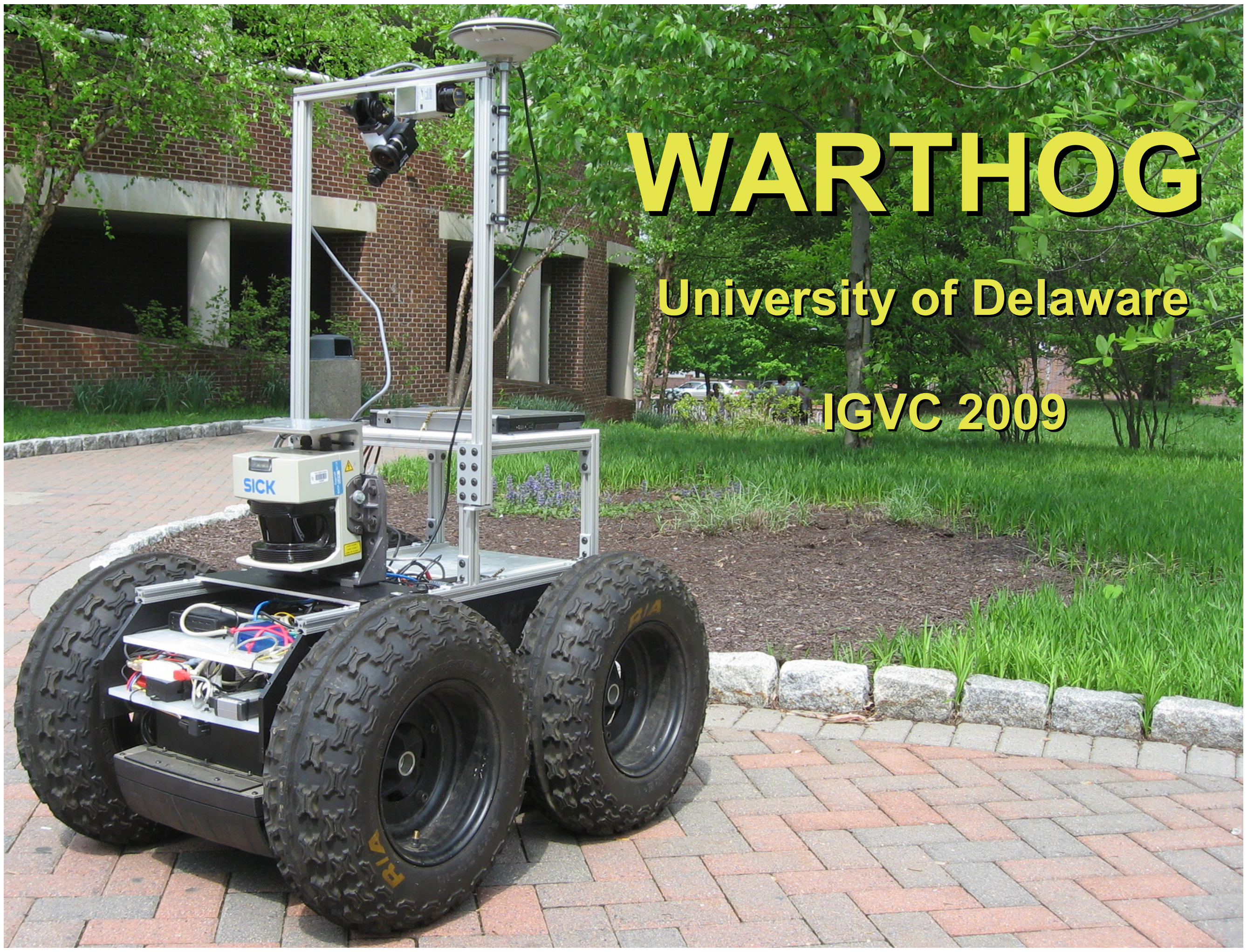


Team members

Conor Gilsenan (Junior, CIS)
Kevin Graney (Sophomore, CIS)
Kevin Schultz (Junior, ME)

Yan Lu (Grad student, CIS)
Mehmet Kocamaz (Grad student, CIS)

Faculty adviser statement

I certify that the engineering design in the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Prof. Christopher Rasmussen
Dept. Computer & Information Sciences
University of Delaware

1 Introduction

Warthog is entering the Intelligent Ground Vehicle Competition for the third time, its rookie entry having been in 2007. In the first year much time was spent on hardware development and headaches (like two major motor failures in the month before the competition), while last year we had more time for software development and testing. Our performance in the navigation challenge was good, modulo some issues discussed below, but despite considerable efforts we failed to improve on a poor showing in the autonomous challenge. This year we have devoted considerably more focus to the autonomous challenge issues. Given the already-done work from previous years, the team this year could be smaller and with a flat organization. Many decisions were made by the entire team in weekly meetings, but there was plenty of time for students to work on miscellaneous hardware-oriented projects related to new sensors and for a focused rethink of our computer vision and motion planning algorithms. We estimate that an aggregate of 300 person-hours have been put into the project this year.

Hardware and software issues, milestones, goals, and resources were tracked through a password-protected wiki page. In addition, several key project management tools were used. All code was kept in an SVN repository (http://subversion.tigris.org), Trac was used for bug tracking and as an interface for browsing the code (http://trac.edgewall.org).

Innovations

The major hardware innovation this year is the replacement of the primary camera with an omnidirectional camera mounted considerably higher. The omnidirectional capability comes not from a catadioptric set-up, but rather a fisheye lens on a high-resolution camera. The wide field of view and improved color and exposure performance are expected to yield a significant improvement in our autonomous challenge capabilities. As can be seen in the image on the cover page, an aluminum superstructure has been added to the top of the robot to facilitate sensor mounting, additional computers, and to better hold the payload. The sensor “mast” was designed to be easily raised and lowered by several feet in order to stow the robot for travel. Internally, several sliding rods have been added to allow two difficult-to-access power buttons to be pressed more easily. Several fans with LED lights have also been added to keep electronics devices cool and help illuminate dark recesses. For safety, we have also added front and rear bumpers to the e-stop system to prevent crashes where the motors keep driving, and all-weather sonars as low-voltage proximity sensors for guarded motion even under manual control (these were added after the picture was taken).

Many changes have been made to the computing subsystem. An analysis of last year's performance indicated that our single laptop was overloaded with tasks, so we have gone to a distributed system of 3 laptops: 1 low-power netbook to handle motor control and low-level safety monitoring, and 2 workstation-class laptops to divvy up image processing and ladar processing/motion planning tasks, respectively. This has been fairly straightforward given last year's move to the IPC message-passing library (http://www.cs.cmu.edu/~ipc). The netbook is always on when the robot is in operation (even under manual control) but the workstation laptops are only needed for autonomous activities. Only the ladar/motion laptop is used for the navigation challenge, while both are needed for the autonomous challenge.

In terms of Warthog's software, there have been three major sets of changes in the areas of general safety, the navigation challenge, and the autonomous challenge. First, a software module on the netbook now monitors the ladars and sonars for danger-close obstacles and has the power to slow or stop the robot, independent of the high-level motion planner that handles steering. This should prevent accidents like last year where an overloaded module leads to motor-control lags that may cause a crash. The navigation challenge module was working fairly well last year, but needed two key changes. First, the function which evaluates whether a hypothetical motion trajectory will hit an obstacle was rewritten and now runs more than an order of magnitude faster. Moreover, the function that evaluates alternative trajectories to pick the best one is pre-emptible—it is aware of how much time it is taking and will return a “best so far” trajectory when the deadline is reached. (Last year it was this function running past its deadline in certain situations that caused us to crash into the long fence). Second, the planner which generates forward trajectories obeying differential constraints (including a minimum turning radius) between start and end configurations in the absence of obstacles has been improved—it returns a plan in every situation now, whereas last year there were a few configurations that were wrongly flagged as infeasible.

The area with the most software changes is our autonomous challenge code. Our new omnidirectional camera, mounted in a new location, has required considerable efforts to calibrate both internally and externally. We have been experimenting with algorithms for manual control of the camera's exposure settings, but empirically the auto-exposure function seems to work much better than our previous camera. We have a variety of new front ends for segmenting the painted white lines robustly including bilateral filtering and and k-means clustering in color space—we are still evaluating which gives the best performance. Our line tracker has been significantly modified from last year as well. Rather than a bottom-up approach using the Hough transform and a variety of states to handle different robot orientations with respect to the lines, we have a unified top-down particle filter that hypothesizes pairs of lines (or curves, in a higher-order version) and finds the maximum likelihood pair. With our omnidirectional images, this approach allows a cleaner formulation since no line is ever out of view regardless of how the robot is oriented.

2 Hardware

2.1 Chassis and drive system

Warthog is based on a Segway RMP 400. The RMP 400 is a 4-wheel, differential drive or "skid steer" vehicle with 21" ATV tires. Each pair of wheels (front and rear) constitutes a *powerbase*. In contrast to other Segway products which have only one powerbase, the RMP 400 is *statically* stable rather than dynamically---it does not require motion to achieve balance. An independent electric motor supplying 0.071 Nm/amp of torque at the motor shaft drives each wheel through 24:1 gearing. The motors are capable of 70 amps peak current per wheel and 24 amps continuous current. The top speed of the RMP 400 is 18 mph; this is limited in hardware for the competition to 5 mph. The robot can climb 45 degree slopes and make zero-radius turns (aka "spin in place"). In 2007 our motor failures seemed to be associated with making such turns, which put maximum stress on the motors because of friction as the wheels skid; since then we have implemented a minimum radius for turning, which makes path planning somewhat trickier.

Two 72V lithium ion batteries run the motors in each powerbase. Across both powerbases, these have a total capacity of 1600 watt-hours and provide an average run time of 12 hours under good terrain and temperature conditions. A charger integrated into each powerbase recharges the batteries from empty in an average of 8 hours. Two buttons control operation of each powerbase. One supplies power to the User Interface (UI) electronics, and the other activates the motors. Both must be pressed before a powerbase can move, so four buttons must be pushed before the entire robot is ready to receive and act upon motor commands. As alluded to above, the buttons for the front base are in a difficult to reach spot inside the chassis. This year we installed two sliding rods with rubber ends to create “virtual” buttons on the outside of the chassis to make starts and restarts much easier. We devoted considerable effort to researching and spec'ing out a solution involving solenoids, but ultimately decided against it due to the complexity of the power-conditioning requirements.

As delivered, the RMP 400 had a length of 44.5", width of 30.5", and height of 21". Ground clearance is about 3.5". Our physical modifications consisted of installing an internal polypropylene and aluminum shelving system to secure electronic devices, batteries, and the control computer, as well as external mounting of the sensors and some switches and buttons on the top and rear plates, respectively. This year we have an 8020 aluminum superstructure which raises the primary camera by about 18" for a better angle for line detection, as well as adding a shelves for the payload and two laptops. The sensor mast is adjustable to a maximum height of about 55" (the top of the GPS antenna) while leaving the length and width unchanged.

2.2 Safety

A large red e-stop button is mounted on the rear of the vehicle and is attached directly to the Segway-provided e-stop circuits of both powerbases . This button physically latches--it must be pulled out to disengage it. However, the Segway e-stop circuit actually shuts the motors off rather than simply pausing them. Thus, in addition to unlatching the e-stop button, the motor activation buttons must be pressed to bring the robot out of e-stop. Wireless e-stop functionality is provided by an aftermarket automotive keyless entry system which interfaces directly with the Segway-provided e-stop circuits. The nominal range of this device is 100 feet in open air. The remote e-stop is connected in series with the manual e-stop such that triggering either one causes an e-stop.

This year we have also added several all-weather sonars from MaxBotix to use as low-voltage proximity sensors, even when the robot is being manually joysticked. We also have mounted 24"-wide front and rear bumpers from Tapeswitch wired in series with the other e-stop devices so that any crash stops the motors automatically. These bumpers require 10 lbs. of force to activate.

2.3 Computing

Up to three computers control Warthog; all run Ubuntu 8.10 for ease of unified library updates, etc. All of the laptops are networked together via Ethernet. The "base" computer is a Dell Mini-9 netbook with an Intel Atom N270 CPU, 512 Gb RAM, and a 32 Gb solid state drive. The internal battery provides about 3 hours of runtime, but we have a DC-DC converter attached to our auxiliary battery that allows all-day operation. Each of the workstation laptops is a Dell M2400 Precision with an Intel Core Duo T9600 at 2.8 GHz, 4 Gb RAM, and a 7200 rpm hard drive. The internal lithium-ion batteries in these provide an average run time of about 2.5 hours and must be swapped for continued operation. The wattage requirements of the workstation laptops are too high for our DC-DC converter to tap into the auxiliary battery. The Mini-9 computer is connected to the RMP 400 via USB (technically, two cables through a hub--one per powerbase, since duplicate commands are sent to the front and rear powerbases). Low-level motion commands are sent in the form [*desired forward speed, desired turn rate*]. The UI electronics take care of PID control to achieve and maintain the commanded values.

A wireless Logitech Rumblepad joystick is used to control the robot remotely when necessary. In open air, it offers an effective range of 50-100 feet. For higher-level remote commands and telemetry, we use a Fujitsu P1610 tablet PC running Ubuntu communicating with a wireless router onboard the robot. This computer is extremely portable, weighing about 1.0 kg, and has an 8.9" touch-sensitive screen. It makes it easy to walk along with Warthog over all kinds of terrain while monitoring the state of its software as it processes sensor readings and plans motions.

2.4 Sensors

Warthog's primary sensors are a SICK LMS-291 laser range-finder, a Unibrain Fire-i400 Firewire color camera, a Novatel Propak-V3-HP GPS, and a PNI TCM2-50 digital compass. New this year is a Pt. Grey Flea2 camera with a fisheye lens for omnidirectional imagery, as well as two sonar devices for proximity warnings. The Segway RMP 400 base also provides extensive proprioceptive sensing regarding odometry, wheel torques, pitch and roll angle and rotational velocities, remaining battery life, and so on. A diagram of all IGVC-relevant external devices and how they are connected to the control computer is given below.

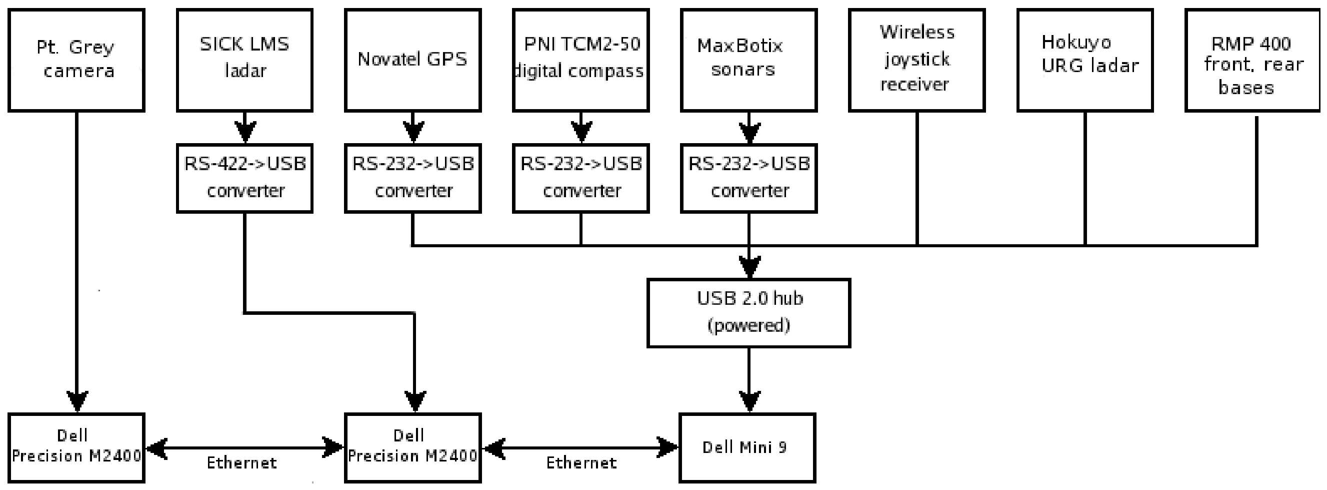


Figure 1: List of major sensors and how they connect to the control computers

2.5 Auxiliary electrical system

While the RMP 400's motors are powered by Segway-provided batteries integral to each base, all other electrical devices except the workstation laptops require a separate power system. Two 12V, 32 Ah AGM deep-cycle lead-acid batteries are connected in series to create a 24V “auxiliary” battery. These weigh 50 lbs. total. AGM batteries are excellent for the rugged conditions created by autonomous vehicles; they cannot spill even if broken and can be mounted in any orientation. An externally-mounted Xantrex battery monitor with an LCD display shows the state of the battery. It has a serial connection to the control computer to report charge remaining for graceful shutdowns. A 24V, 8 A smart charger also rides onboard Warthog, permanently attached to the battery. Plugging in its cable to an AC outlet commences charging that does not need to be monitored.

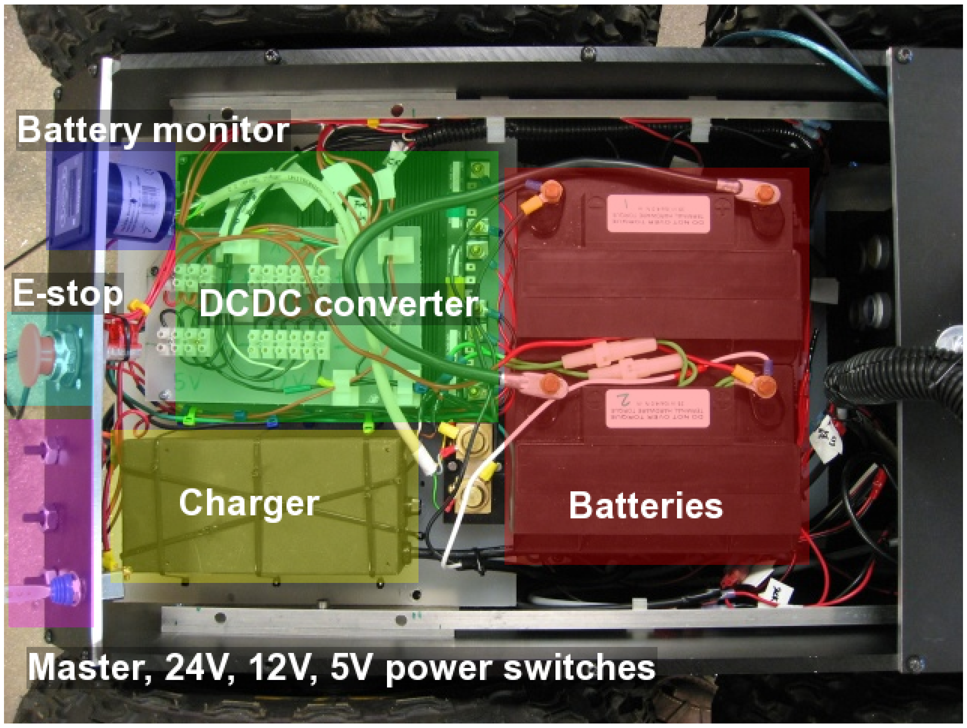


Figure 2: Auxiliary electrical system components

The batteries are connected via a fuse to a Vicor ComPAC DC-DC converter which outputs 24V, 12V, and 5V power at up to 100 watts per output. The Vicor offers EMC filtering, transient protection, and reverse polarity protection at about 85% efficiency. The arrangement of the various devices in the auxiliary electrical system is shown in the figure above.

One new component in the system this year is a 12V to 19V DC-DC converter. It allows the Mini 9 laptop to be run from these batteries, extending runtime virtually indefinitely.

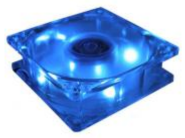
Below is a table of the electronic devices which receive their power from the DC-DC converter (the battery monitor draws power directly from the battery). As can be seen, the watts drawn from the DC-DC converter are nowhere near its limits. The total average current draw, with a 10% safety factor added, is about 2.4 A. Given the capacity of the batteries and the efficiency of the converter, this translates into an expected run time of almost 6 hours to go from a full charge to a 50% charge.

24V devices	**Average power**	**Average amps**
SICK ladar	20 W	0.84 A
Mini 9 laptop (19V)	30 W	1.6 A
12V devices		
Novatel GPS	2.5 W	0.21 A
Firewire hub (old camera)	1 W	0.084 A
Wireless e-stop receiver	1.2 W	0.1 A
LED fan	2.4 W	0.2 A
5V devices		
USB hub (wireless joystick)	1 W	0.20 A
TCM2-50 digital compass	0.1 W	0.02 A
Hokuyo ladar	2.5 W	0.5 A
Wireless router	1 W	0.2 A
Sonar x 2	0.5 W	0.1 A
Firewire card (new camera)	1 W	0.2 A

Table 1: Power consumption of electronic devices

2.6 Major component list

The table on the next page lists the major components detailed in the previous subsections that went into the construction of Warthog, their retail cost, and the cost to the team this year (items carried over from previous years are not counted in the latter category).

Components	Item	Retail cost	Cost this year
	Dell Mini 9	$400	$400
	Dell Precision M2400 x 2	$1,700	$1,700
	Tapeswitch x 2	$350	$350
	MaxBotix sonar	$200	$200
	Pt. Grey Flea2	$800	$800
	MassCool 80 mm LED fan	$4.50	$4.50
Existing components			
	Segway RMP400	$28,500	-
	SICK LMS-291	$5,000	-
	Novatel Propak-V3-HP	$5,500	-

Components	Item	Retail cost	Cost this year
	PNI TCM2-50	$800	-
	D-Link USB hub	$27	-
	2 x Concorde SunXtender PVX-340T battery	$188	-
	Vicor ComPAC DC-DC converter	$436	-
	Soneil 2416SRF charger	$160	-
	Xantrex battery monitor	$225	-
	Fujitsu tablet PC	$1,629	-
	Logitech Rumblepad	$20	-
	D-Link DGL-4300 wireless router	$120	-
	Hokuyo URG-04LX	$2,500	-

Table 2: Major hardware components and their costs

3 Software

Warthog's software architecture consists of a set of *modules*, which this year are separate processes. Many of the modules are associated with device drivers—they talk directly to sensors and stream raw data, as well as providing log writing and reading functionality. A set of middle level modules filter the output of the sensor modules, but make no actual decisions about what to do. Finally, at the top level are the *pilot* modules, which are the only ones allowed to send motor commands. Only one of these is running at any given time. All modules used are listed below in alphabetical order. Unless otherwise noted, each module is used for both challenges ("AC" = autonomous challenge, "NC" = navigation challenge).

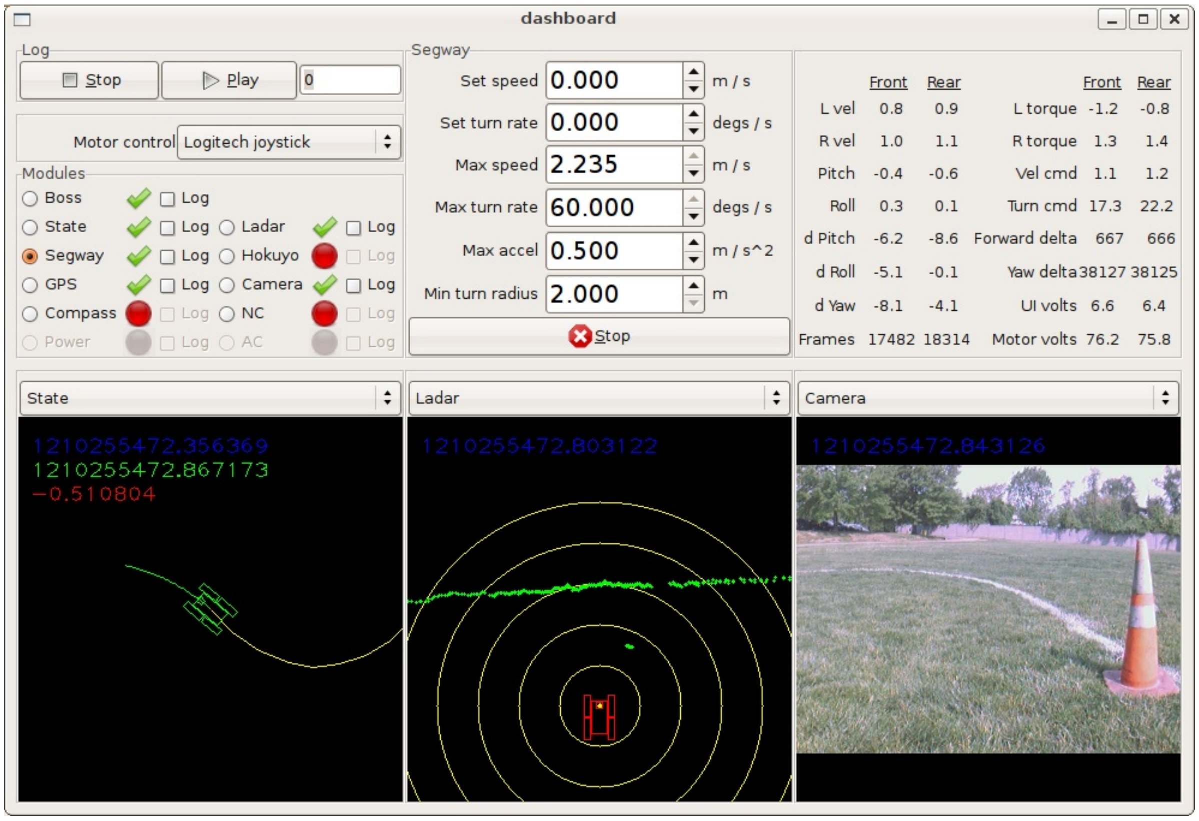
Module	Purpose
boss	Process health monitor. Starts and stops modules, kills and restarts hung modules. We are using the Upstart task/service daemon (http://upstart.ubuntu.com) to automatically restart tasks that die.
compass	Digital compass data capture, including heading, pitch/roll angle, and temperature
dashboard	GUI interface for module management, real-time telemetry, and visualization of sensor data. Typically runs on wirelessly-connected laptop or tablet PC
gps	Novatel GPS data capture, including UTM coordinates, heading, and velocity
hokuyo_ladar	Hokuyo ladar range data capture and transformation to vehicle coordinate frame. Useful for slope estimation, negative obstacle detection.
ladar	SICK ladar range data capture and transformation to vehicle coordinate frame
pilot_auto (AC only)	Executive module for Autonomous Challenge. Performs image capture, internal/external camera calibration, low-level image processing, line detection and tracking. Integrates input from ladar, and state data to decide which direction to drive.
pilot_nav (NC only)	Executive module for Navigation Challenge; reads list of waypoints and integrates waypoint homing with obstacle avoidance.
segway	Segway RMP 400 data capture and motor control. Embedded in this module is joystick driver for manual piloting
state	Measure and filter robot position, heading, velocity, and other positional variables

Table 3: Software modules used for both challenges

External libraries used include OpenCV for computer vision (http://opencvlibrary.sourceforge.net) and the Bayes++ Bayesian Filtering library for state filtering (http://bayesclasses.sourceforge.net). A few selected modules are explained in more detail in the following pages.

3.1 dashboard

Expanded and improved this year is a GTK+ GUI program called "dashboard" for monitoring low-level robot health as well as setting high-level goals. A screenshot of the program is shown below. Running modules are indicated in the table at upper-left, while the other top-level panels show numerical and textual data about the selected module. Messages and instructions may also be sent to the modules through these panels, as for example with the "speed" and "turn rate" fields below. The bottom panels display timestamped, graphical information in a user-choosable fashion. Here the bottom-left panel is showing the state of the robot and its path history, the middle panel is showing raw ladar returns (in this configuration the ladar was pitched down, so the line behind the cone is groundstrikes), and the bottom-right panel shows what the camera sees (the camera image is from last year's narrow-angle camera).



Motor control: Logitech joystick

Modules

Boss	✓	Log			
State	✓	Log	Ladar	✓	Log
Segway	✓	Log	Hokuyo	●	Log
GPS	✓	Log	Camera	✓	Log
Compass	●	Log	NC	●	Log
Power	●	Log	AC	●	Log

Set speed	0.000	m / s
Set turn rate	0.000	degs / s
Max speed	2.235	m / s
Max turn rate	60.000	degs / s
Max accel	0.500	m / s^2
Min turn radius	2.000	m

	Front	Rear		Front	Rear
L vel	0.8	0.9	L torque	-1.2	-0.8
R vel	1.0	1.1	R torque	1.3	1.4
Pitch	-0.4	-0.6	Vel cmd	1.1	1.2
Roll	0.3	0.1	Turn cmd	17.3	22.2
d Pitch	-6.2	-8.6	Forward delta	667	666
d Roll	-5.1	-0.1	Yaw delta	38127	38125
d Yaw	-8.1	-4.1	UI volts	6.6	6.4
Frames	17482	18314	Motor volts	76.2	75.8

```
1210255472.356369
1210255472.867173
-0.510804
```

```
1210255472.803122
```

```
1210255472.843126
```

Figure 3: dashboard module screenshot

3.2 pilot_nav (Navigation Challenge)

Our SICK laser range-finder is set to a maximum range of about 8 m for maximum range resolution, a 180 degree field of view, and a 30+ fps refresh rate. It is mounted about 0.5 m (20") above the ground with a level scan plane. This allows easy obstacle detection (vs. a downward-pitched scan plane) and longer-range motion planning, but makes shorter obstacles invisible. In general, we would want additional ladars or camera image-based obstacle detection algorithms to fill in blind spots, but we have not found these to be necessary at IGVC.

Our base motion planner is derived from a Dubins car model, which accounts for differential constraints on Warthog's motion in the form of a minimum turning radius and rules out reverse motion (as we do not have adequate rear-facing sensors). Under this model, the only maneuvers permitted are straight-line and circular arc (left or right) segments. New this year is a complete Dubins planner which works for all start and end configurations in the absence of obstacles. This plan is always the first one we consider during the NC. However, we must also incorporate obstacle avoidance, and our primary method here (used in both the NC and AC) is a motion planner based on a modified rapidly-exploring random tree (RRT). Our RRT-inspired approach generates a set of random short-term plans from the current location consisting of Dubins maneuvers. These maneuver sequences are converted to trajectories and checked against the local obstacle map (ladar-only in NC). Trajectories which collide with obstacles (i.e., come within a safety margin based on Warthog's size) are removed from consideration. Among those remaining, a “best” plan is selected and executed which gets the robot closest to a nearby goal point. In the case of the NC, this is just the next waypoint. The critical function which caused us to crash last year was the trajectory collision checker—it was too slow when there were many obstacles (like a all of the ladar hits along a fence). This function has been rewritten and is an order of magnitude faster this year.

3.3 pilot_auto (Autonomous Challenge)

Whereas our NC module from last year has merely been upgraded and streamlined, our AC approach this year is completely different both from a hardware and software point of view. As previously mentioned, we are now using a different camera with better color, exposure performance, and resolution, and we have fitted it to an omnidirectional (aka “fisheye”) lens. Our lens and camera orientation yield a field of view of about 180 degrees in front of and behind Warthog, and 145 degrees to the left and right. Our camera height has also been raised, yielding a better angle for line paint detection and ensuring that both the left and right lines are visible at all times (modulo occlusion by barrels). A sample raw image from the camera is shown below at right. On the left in purple are SICK ladar obstacles in vehicle coordinates; concentric circle radii differ by 1 m. Overlaid red points are obstacles detected by the SICK ladar and transformed to image coordinates.

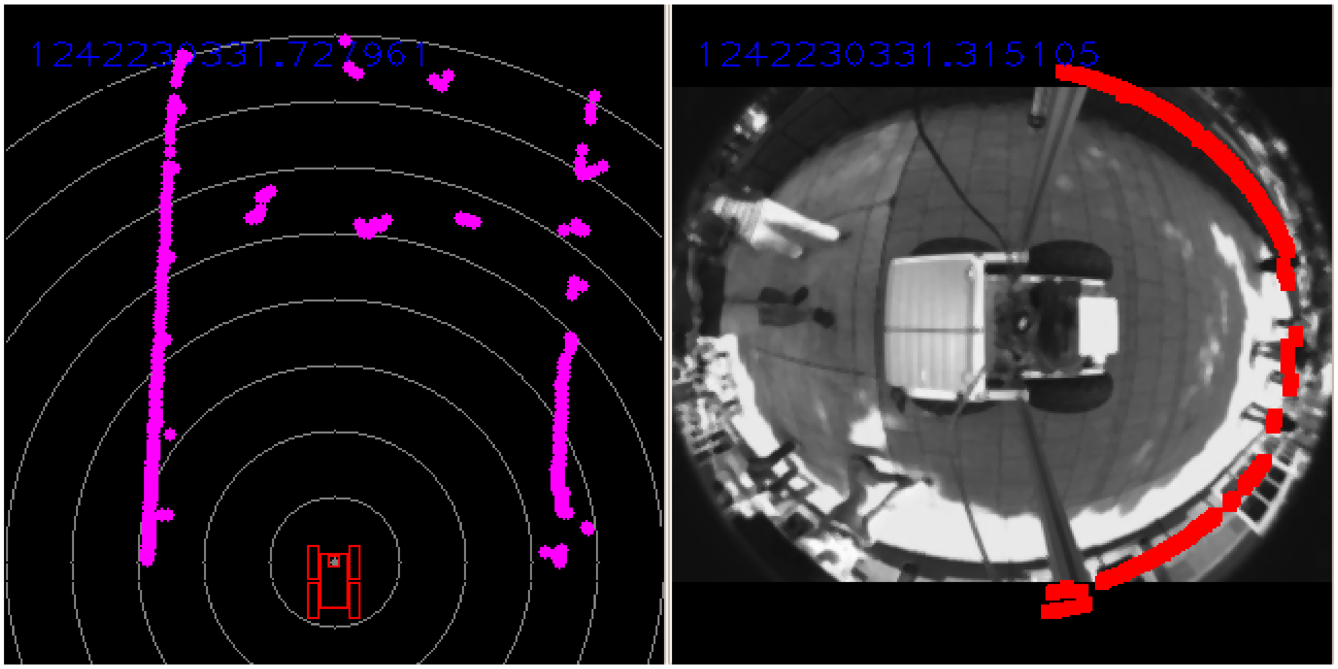


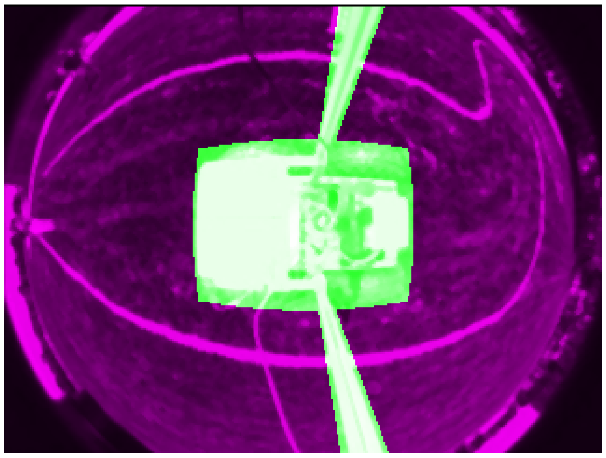
Figure 4: Ladar obstacles and omnidirectional camera view with ladar overlaid

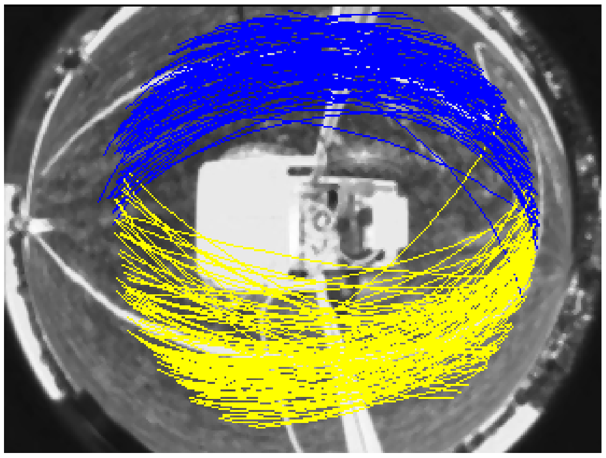
There are two major tasks for the AC. First is the detection of obstacles—where *not* to go. Second is the setting of a goal location, which is far from straightforward with no GPS waypoints as there are in the NC. For the first problem, the SICK ladar functions identically here as in the NC, creating a map of obstacles for the RRT planner to avoid. The critical challenge in the AC, of course, is to also add the painted lines to the obstacle map. This year we have opted for more of a top-down approach. Following work on the similar problem of lane-tracking for autonomous driving, we model the approaching section of path as a curved clothoid “ribbon” and track it using a *particle filter* with some custom postprocessing. The state itself has four variables, all in vehicle coordinates: (1) angle between the path curve tangent and the vehicle heading, (2) lateral offset between the path center and the vehicle center, (3) width of the path, and (4) path curvature (inverse radius of best-fitting circle arc segment).

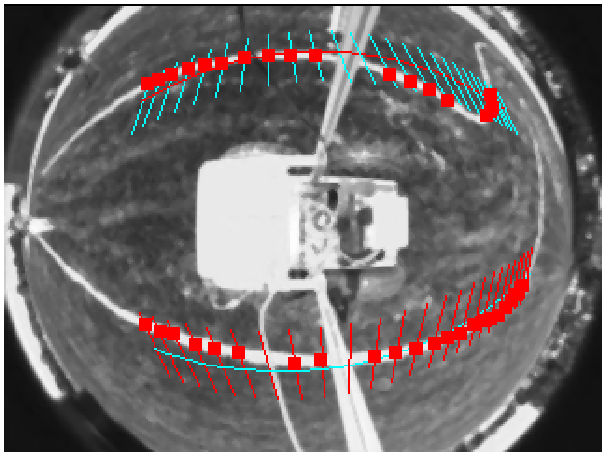
Particles consist of state hypotheses which are scored using a likelihood function which is where the image processing occurs. In this function a given hypothesis consists of two curve segments mostly in front of the robot derived from the state, transformed to image coordinates and sampled at discrete intervals. At each interval along a given curve (typically there are 15-20 each for the left and right lines), pixels in a search window (about 1 m wide) orthogonal to the curve are examined. Each search window is assigned a score based on the ratio of the maximum to minimum intensity found, with the intuition being that windows containing a painted line will have high ratios and windows with just grass will have low ones. The average score over all search windows across both the left and right curves is the likelihood for the whole hypothesis.

Intensity is measured via the blue channel of the color camera image after it has been median filtered to reduce noise. In order to not be distracted by contrasting colors on barrels or other obstacles, or the robot itself (since it appears in the omnidirectional image), a *mask* is created for each new frame based on the SICK ladar and knowledge of the camera position and robot dimensions. Any non-zero pixels in the mask are assumed to not belong to the ground, and thus are skipped in calculation of the likelihood function. Any search windows that have too many masked-out pixels are omitted from the mean score for the hypothesis because of insufficient data to be reliable.

Since circular arcs do not perfectly represent the freehand lines painted for the AC, a final refinement step is carried out to locally warp the clothoid path estimate from the particle filter. In this step, search windows along the state estimate curves are analyzed. It is assumed that most but not all of these search windows do contain a painted line; here we wish to make this classification and fix the exact location of the line. Again taking into account the obstacle/robot mask described above, the ratio of the maximum to minimum intensity in each search window is thresholded (at about 2.0). If below threshold, that window is considered to not have a line in it; if above, the location of the maximum is taken to be the location of the line. Each of these filtered line detections is then placed into the vehicle coordinate obstacle map as a virtual obstacle (along with the SICK obstacles). Some sample images illustrating steps in the process are shown below (see caption on next page).


(a)


(b)



(c)

Figure 5: Steps in AC line finding. (a) Obstacle/robot mask (in green) used to exclude pixels from consideration; (b) 100 zero-curvature path hypotheses from particle filter; (c) State estimate from particle filter with search windows shown + refined line locations (red dots)

This combined obstacle map is the one used for RRT motion planning as described above in pilot_nav section. The second major task above, setting a goal location for motion planning to terminate at, is taken directly from the state and obstacle map as follows. The nominal goal location is a point within the path some constant distance ahead (roughly 5 m) and on the centerline. If this point is too close to an obstacle, it is moved to the nearest point in free space. Since the goal is defined relative to Warthog's current location, it constant "recedes" as the robot advances.

At a higher level there is a potential problem at hairpin turns and complex barrel switchbacks with the robot making a U-turn and heading back toward the start. This is prevented by using GPS information on where the robot has already been to "push" it toward novel regions.