Team members

Vishal Anand (Grad student, CIS)
Brad Fletcher (Freshman, CIS)
Conor Gilsenan (Sophomore, CIS)
David Lopez (Grad student, CIS)
Kevin Schultz (Junior, ME)
Donald Scott (Senior, CIS)
Kanik Sem (Grad student, CIS)
Miao Tang (Grad student, CIS)
Yuqi Wang (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 second time, its inaugural entry having been in 2007. Last year much time was spent on hardware issues (we had two motor failures in the last month before the competition), but we managed a respectable showing. This year we have had more time for software development and testing and hopes are high for a better showing. The team organization this year has been more informal, with students working in pairs and groups of three on specific issues like ladar-image fusion for obstacle detection, improving long-range motion planning through SLAM, and really focusing on robust and efficient line detection and tracking. We estimate that an aggregate of 500 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

Much of last year's vehicle is physically unchanged. The remote e-stop has been replaced after failing, a 24 V inverter has been added to allow the laptop to run off of the auxiliary battery (see section 2.5 below), and the electrical distribution board has improved connectors and diodes to prevent back-feeding from the USB and Firewire hubs. A low-power Hokuyo URG-04LX ladar has been added with a sagittal scanning plane to serve as a virtual bumper to prevent collisions even under manual control. However, it is not being used for the IGVC competition. Also, a wireless router has been added to allow easy control and telemetry via a tablet PC.

Most of the changes this year are to Warthog's software. A low-level yet fundamental change has been a move away from the Player/Stage robot library (http://playerstage.sourceforge.net) used last year to a more decentralized system which uses the CMU IPC message-passing library (http://www.cs.cmu.edu/~ipc). The Player/Stage library uses a client-server model which necessitated a single (albeit multithreaded) perception/planning/control program. This worked fine for our Navigation Challenge algorithm, but the computer vision processing necessary for the Autonomous

Challenge was CPU-intensive and caused serious errors due to “buffer overruns” in the library. The algorithm described in our technical paper last year, which worked well on offline data, turned out to be unusable for controlling the robot due to this issue and we had to revert to a more simple and ultimately ineffective alternative. The IPC approach has led to new drivers (especially for the SICK ladar—we are using a driver from the UPenn DARPA Urban Challenge team that is much more stable than the Player/Stage version) and a looser collection of modules as separate processes which has sped development.

The algorithmic components we are using have changed significantly this year as well. After analyzing our performance last year, we have made several key improvements. First, we have a new state estimator which fuses GPS and odometry. Second, we have replaced our reactive motion planner, which failed in several tight obstacle situations, with an efficient deliberative planner adapted from Rapidly-Exploring Random Trees (RRTs). This planner is global (also obviating the TangentBug method used last year to avoid getting stuck in local minima), can handle intricate multi-stage maneuvers through obstacle fields (including reversing when necessary due to a minimum turning radius constraint), and is also applicable to the Autonomous Challenge. Finally, we are using a much more sophisticated method in the Autonomous Challenge than we were able to field on the day of competition last year.

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"). Last year 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, so this year 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 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. These increased the height to 42" (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.

2.3 Computing

The sole computer controlling Warthog is a Dell XPS M1710 laptop with an Intel Core Duo T2600 2.16 GHz CPU and 2 Gb RAM, running the Debian Linux operating system. The internal lithium-ion battery provides an average run time of about 2.5 hours. The computer is connected to the RMP 400 via two

USB cables, one per powerbase (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. 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.

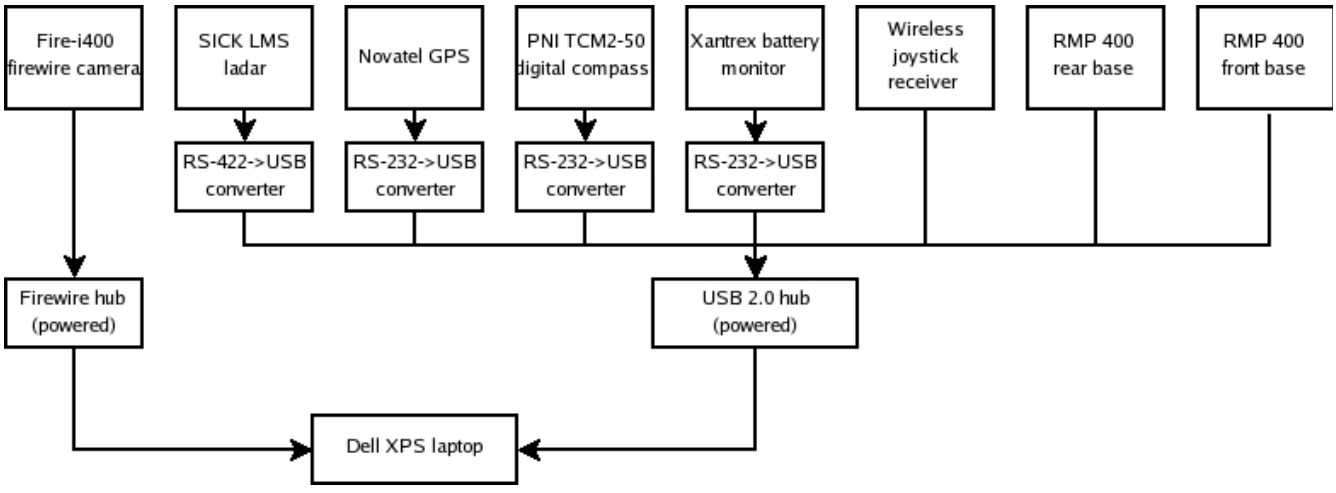

Figure 1: List of sensors and how they connect to the control computer

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 computer 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.

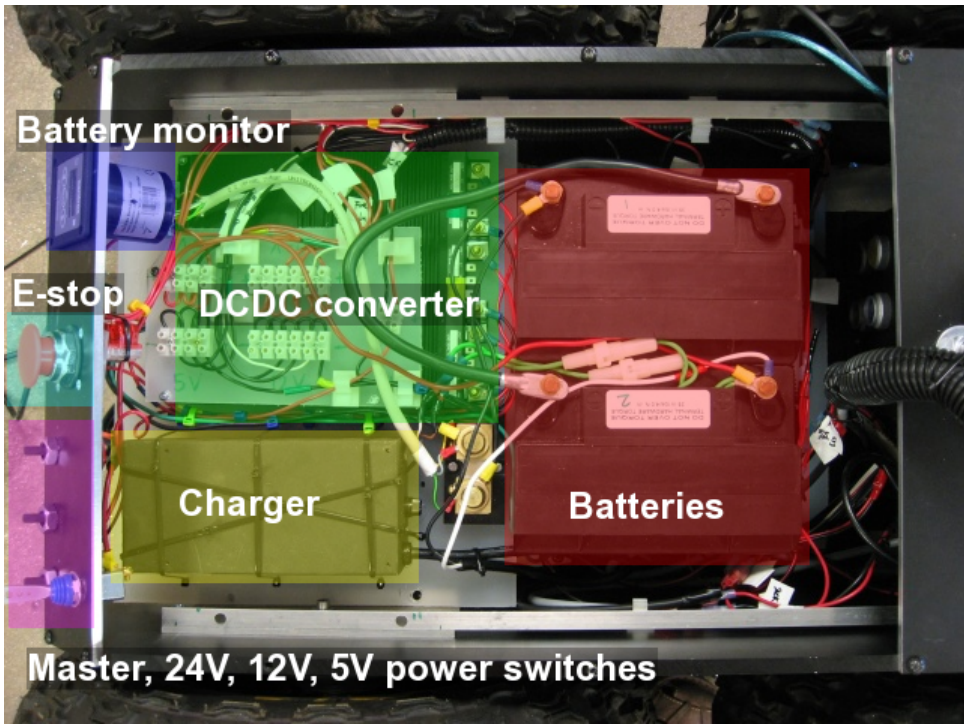

Figure 2: Auxiliary electrical system components

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 24V, 400-watt inverter which is attached directly to the auxiliary batteries, bypassing the DC-DC converter. It allows the laptop to be run from these batteries, extending runtime for several hours.

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
12V devices		
Novatel GPS	2.5 W	0.21 A
Firewire hub (camera)	1 W	0.084 A
Wireless e-stop receiver	1.2 W	0.1 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

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).

	Item	Retail cost	Cost this year
	Segway RMP400	$28,500	$0
	SICK LMS-291	$5,000	$0
	Unibrain Fire-i400	$425	$0
	Novatel Propak-V3-HP	$5,500	$0
	PNI TCM2-50	$800	$0

	Manual e-stop button	$80	$0
	D-Link USB hub	$27	$0
	2 x Concorde SunXtender PVX-340T battery	$188	$0
	Vicor ComPAC DC-DC converter	$436	$0
	Soneil 2416SRF charger	$160	$0

	Manual e-stop button	$80	$0
	Xantrex battery monitor	$225	$0
	Dell XPS M1710	$4,155	$0
	Fujitsu tablet PC	$1,629	$0
	Logitech Rumblepad	$20	$0
	D-Link DGL-4300 wireless router	$120	$120
	Hokuyo URG-04LX	$2,500	$0
	PowerBright 400 watt 24V inverter	$100	$100
	Total	$49,905	$260

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.

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.
camera (AC only)	Image capture, internal/external camera calibration, low-level image processing. Line detection and tracking for the AC is performed inside this module because full-resolution images are too large to pass as messages.
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
ladar	SICK ladar range data capture and transformation to vehicle coordinate frame
pilot_auto (AC only)	Executive module for Autonomous Challenge. Integrates input from camera, 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

New 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, and the bottom-right panel shows what the camera sees.

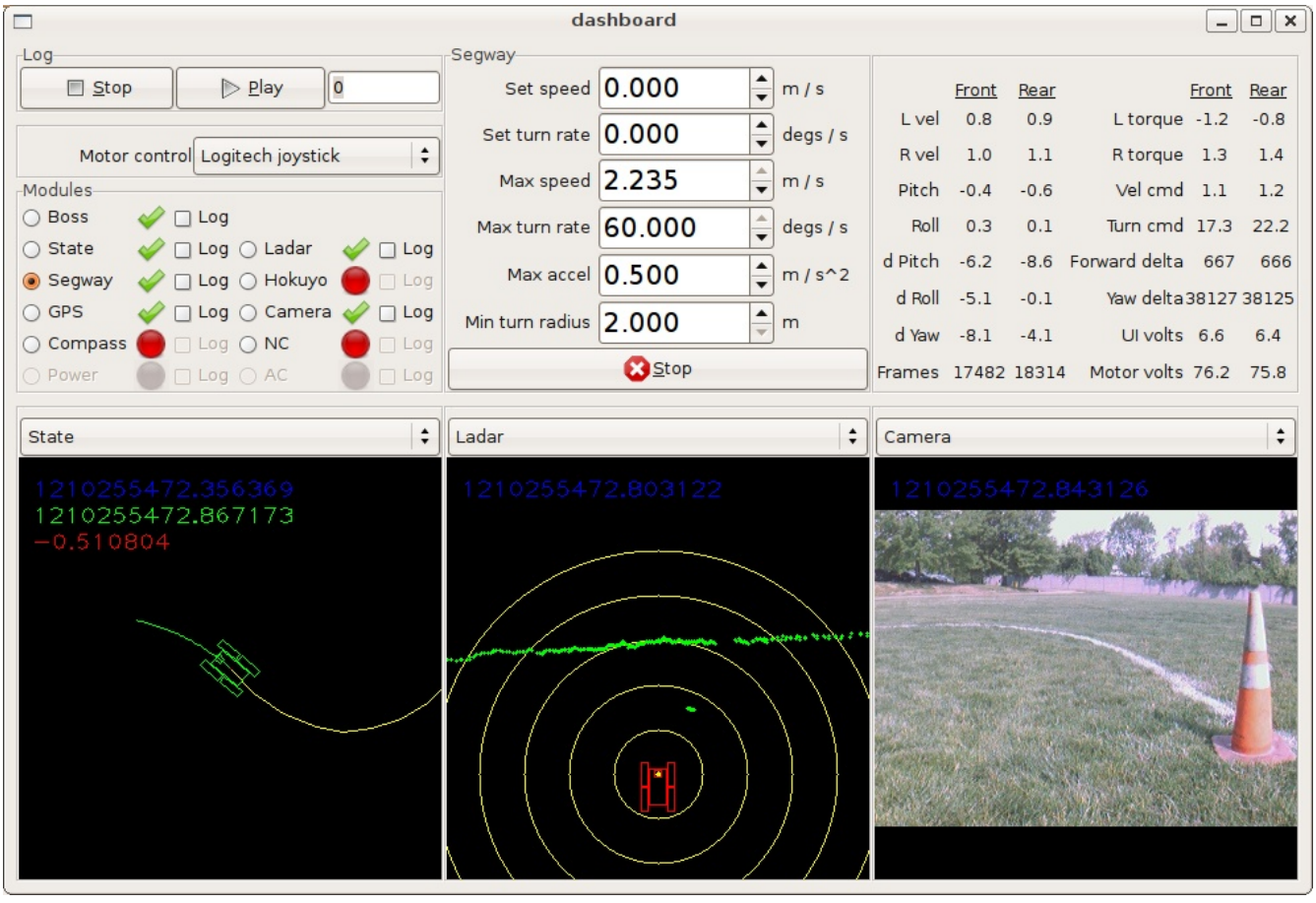

Figure 3: dashboard module screenshot

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. Thus, camera image-based obstacle detection algorithms are used as a supplement to ladar sensing of obstacles to fill in blind spots. First, obstacles can be detected in images beyond the 8 m ladar range in order to start avoidance maneuvers even earlier. Second, nearby obstacles invisible to the ladar because of their height and position (e.g., the crossbar of a barricade or a very close 5-gallon pail which may be out of the ladar scan plane) are still visible to the camera. We use a single basic strategy: detection of obstacles as "non-ground" through modeling of the ground color distribution. The latter approach is especially important for barricades, whose middle sections may entirely elude ladar detection. Without explicit depth information, image-detected obstacles can be transferred to a vehicle-coordinate map of obstacles with appropriate internal and external calibration of the camera and a planar ground assumption via their bottoms (i.e., where they touch the ground).

A new feature of the motion planner this year is the ability to plan complex trajectories, including 3-point turns, when necessary. A modified rapidly-exploring random tree (RRT) [LaValle2006] is used which generates random collision-free, short-term plans from the current location. A “best” plan is selected and executed which gets the robot closest to the ideal obstacle-free path to the next waypoint (for the Navigation Challenge) or which follows the vector field generated from the detected lines (Autonomous Challenge; see below). Hysteresis is used to maintain plan continuity in the face of replanning as the robot moves.

Ground color modeling

Rather than looking for specific obstacle types like barrels or cones, we simply assume that the ground, typically grass for IGVC, represents the dominant color in the image. If this color or mixture of colors can be reliably determined as the robot moves, then pixels which do not match it may be assumed to be obstacles. In the paper [UN2000] (see References section), the ground colors are obtained from a trapezoidal reference area at the bottom of the image that is assumed free of

obstacles. These colors are modeled with a histogram or mixture of Gaussians that is adapted as new images are captured. The reference area need not be assumed to be free of obstacles. In recent work [DKSTB2006], an approach to dynamically resizing the reference area to exclude ladar-detected obstacles was shown to work quite well, and we use this technique. An advantage of this technique is that it is useful for both the Navigation Challenge and the Autonomous Challenge (an example figure is shown in the next subsection).

3.2 pilot_auto (Autonomous Challenge)

The are two major phases to how lines are handled for the Autonomous Challenge. First, lines must be detected in images and transformed to vehicle coordinates. Second is the question of how to steer the robot to continue along the course without a definite goal point given the sometimes conflicting demands of avoiding physical obstacles and not crossing the lines.

Our line detection algorithm is based on the non-ground segmentation method described above. This technique outputs candidate obstacle pixels including painted lines, but also other above-ground obstacles. We want to isolate lines in order to extract directional information from them, so we perform further testing on obstacle pixels. The whiteness of line pixels is not explicitly used, since this may vary with illumination conditions and ground material. Instead, several geometric filters are first run to remove stray pixels as well as regions larger than the expected with of the lines given the camera calibration. The remaining pixels are searched iteratively with an approximate continuity criterion to find curve segments. The Hough transform is also used for verification. Any curve segments over a threshold length are output explicitly. Example output is shown in the figure below.

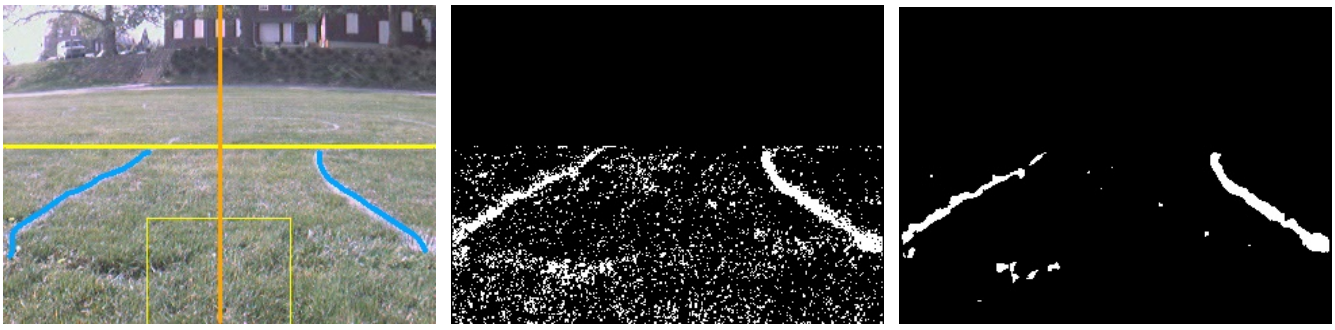

Figure 4: Line tracking. (Left) Captured image with final traced lines in cyan. The yellow box shows where pixels for modeling the grass were drawn from; (Center) Initial non-ground pixels; (Left) Filtered line pixel used for tracing

Given the curve segments detected, we infer a vector field of directions via anisotropic diffusion from them and the curves themselves are put into a vehicle-centric obstacle map as solid obstacles. The vector field gives a rough guide to which direction the robot should prefer to move in the absence of obstacles, while the lines in the vehicle map prevent crossing them accidentally. 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 global information on where the robot has already been to "push" it toward novel regions.

TangentBug

The Bug family of motion planning algorithms navigate a 2-DOF mobile robot in a completely unknown environment with global position knowledge and local obstacle knowledge. This perfectly describes the situation at the outset of the Navigation Challenge. TangentBug [KRR1998] is an algorithm in this family which uses range data (e.g., from a SICK ladar) to compute a locally shortest path, based on a structure termed the local tangent graph (LTG). The LTG describes the extents of visible objects in polar coordinates, and the robot uses it to choose the locally optimal direction while moving toward the target (*motion-to-goal*). Local minima in obstacle configurations which might otherwise trap the robot are explicitly detected and overcome through a *wall-following* behavior. The transition between these two modes of motion is governed by a globally convergent criterion which is based on the distance of the robot from the target.

We made several modifications to the standard TangentBug algorithm to implement it on a real robot. First, the limited field of view of our ladar (180 degrees rather than the full 360 assumed) required changes to the definition of a local minimum and to the basic characteristics of the wall-following mode. Second, the abstract algorithm treats the robot as a point, so the reactive obstacle avoidance provided by the base motion planner is critical to avoid brushing against obstacles as they are passed. Some concepts from TangentBug are combined with our RRT planner to assure that the robot does not get stuck and that it follows a relatively short path.

4 References

[DKSTB2006] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski. "Self-supervised monocular road detection in desert terrain." Robotics Science and Systems Conference (RSS), Philadelphia, PA, 2006.

[KRR1998] I. Kamon, E. Rimon, and E. Rivlin. "Tangentbug: A Range-Sensor-Based Navigation Algorithm." International Journal of Robotics Research, 17(9), 1998.

[Lavalle2006] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[TBF2005] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics. MIT Press, Cambridge, MA, 2005.

[UN2000] I. Ulrich and I. Nourbakhsh. "Appearance-Based Obstacle Detection with Monocular Color Vision", National Conference on Artificial Intelligence (AAAI), 2000.