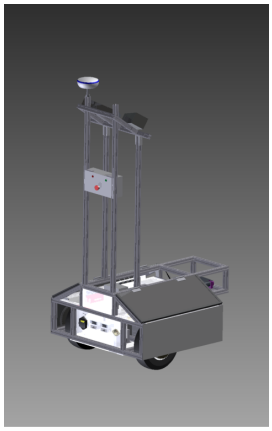
Phobetor: Princeton University's Entry in the 2013 Intelligent Ground Vehicle Competition

Joseph C. Bolling, Michael Zhang, Derrick Dominic,
Elaine S. Chou, Nicole M. Gonzalez

Princeton University School of Engineering and Applied Science
Princeton, NJ, USA

May 8, 2013



Faculty Statement:

I hereby certify that the design and development of the robot discussed in this technical report has involved significant contributions by the aforementioned team members, consistent with the effort required in a senior design course. *Phobetor* has undergone significant design modifications, including completely new computing hardware, a new operating system and an improved computing environment.

PRINCETON
School of Engineering and Applied Science

PAVE PRINCETON AUTONOMOUS VEHICLE ENGINEERING

Contents

1 Team Overview **1**

2 Design Process **1**

3 Hardware **2**

3.1 Mechanical Design . . . 3

3.1.1 Drivetrain . . . 3

3.1.2 Chassis Design . . . 4

3.1.3 Tower Design . . . 4

3.2 Electrical and Electronic Systems . . . 4

3.2.1 Power System . . . 5

3.2.2 Electronic Control . . . 5

3.2.3 Sensors . . . 7

4 Software **8**

4.1 Code overview . . . 8

4.2 Key Software Innovations . . . 9

4.3 Computing Platform . . . 9

4.4 State Estimation . . . 9

4.5 Vision . . . 10

4.5.1 Obstacle Detection . . . 10

4.5.2 Lane Detection . . . 11

4.6 Navigation . . . 12

4.6.1 Cost Map Generation . . . 12

4.6.2 Waypoint Selection . . . 12

4.6.3 Path Planning . . . 12

4.7 Path Following . . . 13

4.8 Speed Control . . . 13

5 Conclusion **14**

1 Team Overview

Princeton Autonomous Vehicle Engineering (***PAVE***) is proud to submit its second revision of *Phobetor* to the 2013 Intelligent Ground Vehicle Competition. The updated *Phobetor* boasts a completely revamped computing framework, from hardware to operating system to software, as well as improved electrical wiring, which we are confident will perform well in this year's IGVC.

PAVEis Princeton University's undergraduate student-led robotics research group. As a completely undergraduate effort, our team builds on previous competition experience in the 2005 and 2007 DARPA Challenges, as well as four years of previous IGVC submissions. Student members of the 2013 IGVC team hail from Princeton's Electrical Engineering, Mechanical Engineering, Computer Science, and Physics departments, providing a range of expertise useful in *Phobetor's* redesign. We employ a simple, flat organizational structure that allows our small team to allocate manpower as design challenges arise. We divide our team into the broad headings of "hardware" (wiring, sensors, power distribution, and mechanical design) and "software" (computing environment, state estimation, mapping, and navigation). Each subteam has a designated leader to ensure goals and benchmarks are met in a timely manner. An overall team captain ensures synchronization between the two teams and handles logistical issues. We estimate that our team has collectively put in 600 hours of work on Phobetor for the 2013 IGVC.

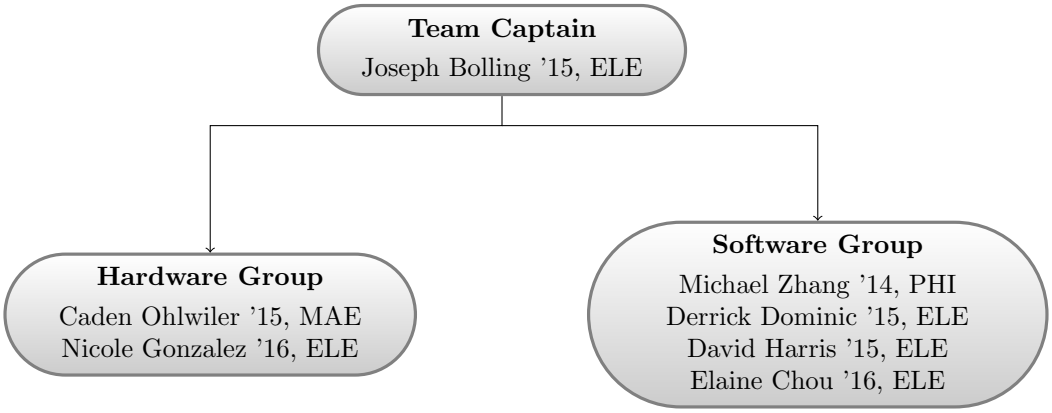


Figure 1: Team Organization Diagram

2 Design Process

Our design process focuses on collaboration and communication. With this year's IGVC submission, we experimented with a modification of ***PAVE***'s traditional approach to collaborative design, combining our hardware and software work sessions into group work sessions attended by all members. This allows a higher degree of collaboration between the two groups on design sections at the hardware/software interface, and allows different teams to work simultaneously on different sections of the robot without setting back each other's work. We use git and ROS's built in development features to synchronize our software versions between different coders. We document our work using our website, a wiki database, and Dropbox, allowing members easy access to information about our projects.

After the 2011 IGVC, we identified a number of issues with *Phobetor's* design. While *Phobetor's* mechanical

systems performed largely as expected, the robot was plagued by electrical issues, including a failed motherboard and speed controller. Software bugs also limited the machine's performance, and much of the 2011 IGVC team's time was spent fixing code errors. We based our design calendar on these issues, undertaking a fundamental redesign of our software framework with a transition to Willow Garage's Robot Operating System (ROS), and leaving extra time to debug and test *Phobetor* before the 2013 IGVC.

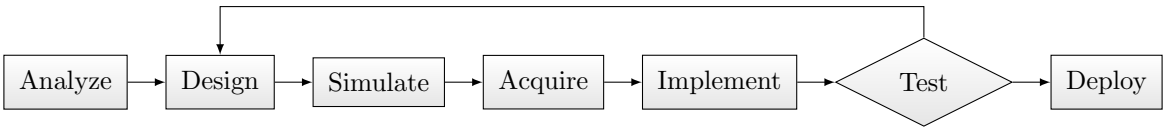


Figure 2: Flowchart of Design Process

Hardware	**Software**
1. Poorly protected and ventilated computer housing	1. Debugging difficult and time consuming
2. Faulty motherboard	2. Insufficiently tested vision processing code
3. Faulty speed controller	3. Inaccurate costmap generation

Table 1: Areas of *Phobetor* Targeted for Improvement

Table 1 displays our improvement goals for the 2013 iteration of *Phobetor* . After establishing a cohesive set of goals, we followed the design process shown in Figure 2. Because of the modularity of both our hardware and software designs, the process can be applied in parallel to different subsystems. This strategy allows us to focus on particular components that require improvement without worrying that other systems will break. Many of the best-designed core systems, such as the wheels, have remained the same for years, while others, like the sensors, are frequently iterated.

Simulations and models are a crucial part of the design process. Our hardware designers use CAD software to plan all components, down to the smallest details. The computer-aided process lets us see the finished product before we build it, and ensures that our parts fit together properly the first time. We simulate

The discussion of the robot design has been divided into two parts within this paper: Hardware (Section 3) and Software (Section 4). In each section, we outline the specific improvements implemented as well as the specific design process behind them, as well as the sections of the robot that remain unchanged from the 2011 design and the reasoning for doing so. Based on our designs, our entry into this year's IGVC, *Phobetor*, is a robust and advanced autonomous system that we expect to perform competitively.

3 Hardware

Phobetor was designed and built from the ground up in 2010[3]. In 2011, the robot was improved with a full redesign of the sensor tower that facilitated easier transport, maintenance, and testing[4]. After the 2011 IGVC, we targeted several electrical sytems for improvement, including *Phobetor's* primary computer and

its speed control system. We examine in detail the design of the robot's drivetrain and chassis in Section 3.1, and the electrical and electronic system is discussed in Section 3.2. An overall list of components used on *Phobetor* and their costs is shown in Table 2.

Item	**Actual Cost**	**Team Cost**
Gladiator Technologies G50 gyroscope*	$1,000	$0
Wheels, Tires, & Drivetrain Components	$567	$567
Videre Design STOC-15 Stereo Camera (2x)	$3,320	$3,320
HemisphereGPS A100 GPS Unit*	$1,500	$0
OceanServer OS5000-US Digital Compass*	$299	$0
Labjack UE9 Data Acquisition Card	$500	$0
US Digital HB6M Rotary Encoder (2x)	$430	$430
NPC T64 Motor, .7 HP (2x)	$572	$572
Roboteq LDC1430C Speed Controller (x2)	$290	$290
Raw Material for Chassis	$1425	$1425
Miscellaneous Hardware	$340	$340
Computer Components	$844	$844
Tempest TD-22 12V Battery (6x)	$390	$390
IOTA DLS-27-25 Battery Charger	$295	$295
Samlex SDC-08 24V/12V DC-DC Converter	$60	$60
Linksys WRT54G Wireless router (2x)	$75	$0
R/C Radio System	$350	$350
Total:	**$12,257**	**$8,883**

*Denotes donated item. All other zero-cost items were borrowed from *Argos*, ***PAVE***'s 2009 robot.

Table 2: List of Costs for Building *Phobetor* (all trademarks property of their respective owners)

3.1 Mechanical Design

Phobetor measures 31" wide, 37" long, and 69.5" tall; it weighs approximately 270 pounds excluding payload. The robot features a tricycle wheelbase with two powered rear wheels and a leading caster; this design ensures contact with the ground at all times, regardless of terrain. Along with the low center of gravity, the wheelbase makes *Phobetor* highly stable and holonomic. The drive wheels are fitted with snowblower tires, which provide superior traction on a variety of surfaces, especially off-road. On top of this base is the sensor tower, which is designed to provide *Phobetor* with a vantage point equivalent to that of an average human. At the 2011 IGVC, *Phobetor's* mechanical systems performed largely as expected, and were left unchanged for the 2013 IGVC, save for minor repairs. As with all ***PAVE*** robots, *Phobetor* was designed entirely in CAD before any material was cut. Using Autodesk Inventor accelerated the updated design process, and allowed for estimation of weights and moments before construction. CAD visualizations of the robot can be seen in Figure 3.

3.1.1 Drivetrain

Phobetor's drivetrain is the result of years of iteration from previous IGVC robots. The 24V electrical system described in Section 3.2.1 gives *Phobetor* enough power respond nimbly in different driving modes. The drivetrain allows zero-radius turning, driving up a 10°incline, and navigating on grass. *Phobetor* uses

(a) Top **(b)** Front **(c)** Side

Figure 3: Visualizations of *Phobetor*

NPC T64 motors. After a speed controller failure, both of *Phobetor's* speed controllers have been replaced with Roboteq LDC1430C speed controllers, which are more electrically robust and can interface directly with our digital wheel encoders.

3.1.2 Chassis Design

Phobetor's lightweight chassis is built from 80/20 structural framing aluminum. The two-layer organization allows all heavy components to be placed near the center of rotation, lowering *Phobetor's* moment of inertia. The horizontal dimensions were chosen to meet competition specifications and allow *Phobetor* to fit through a standard door. The second level features two large doors, one on each side. When open, these allow for easy access to every component on the second level. *Phobetor* is waterproof, allowing us to test and operate the vehicle in New Jersey's rainy spring weather. The angled doors allow water to run off, and are mounted on continuous hinges with rubber-bulb seals.

3.1.3 Tower Design

Phobetor features a sensor tower that provides mounting locations for its stereo vision cameras and other sensors(discussed in Section 3.2.3). The stereo vision camera housings are built on movable 80/20 aluminum racks. The camera enclosures were custom-built to provide a waterproof seal and better secure the polarizers. The cameras can be moved and calibrated for testing. The tower also provides mounts for *Phobetor's* GPS and compass.

The electronic components in the tower, including Phobetor's compass, radio receiver, and control circuit, are encased in a waterproof plastic box halfway up the tower. Mounted on the box are the E-Stop button and new indicator lights. Signal lines from the GPS unit and the base connect to the box with sealed strain relief sockets. A plug panel contains watertight connectors for the 5V and 12V power supplies, control signals, and USB and Firewire connections.

3.2 Electrical and Electronic Systems

Phobetor's electrical systems have undergone a comprehensive renovation to increase their durability for the 2013 IGVC. Electrical failures proved to be the limiting factor in Phobetor's 2011 performance, so we have given extensive consideration to areas that can be made more robust. For 2013, we've replaced *Phobetor's*

motherboard, selected better speed controllers, installed new batteries, and have examined and organized much of the robot's wiring.

3.2.1 Power System

Phobetor's drive motors, computer, and electronics are powered by a 24-volt electrical system which allows almost one hour of uninterrupted testing. Figure 4 shows the power storage and distribution layout of *Phobetor's* electrical system. The six 12-volt lead-acid batteries are arranged as two banks in series, and a voltage divider with precision 15kΩ resistors ensures that the banks charge and discharge evenly. This divider prevents power failures and protects the long-term health of the batteries while drawing negligible power. For the 2013 IGVC, we have replaced *Phobetor's* aging batteries with newer models, to ensure full power capacity.

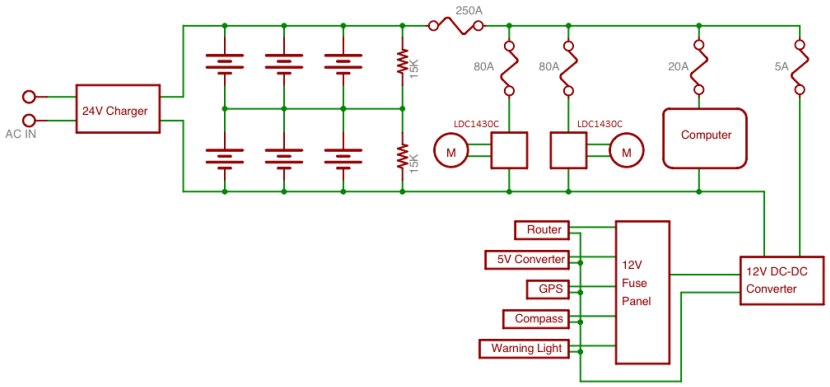


Figure 4: Electrical System

The computer draws DC power directly from the 24-volt system with a 450W ATX power supply, eliminating the need for a heavy and power-inefficient AC inverter. The 12-volt electronics, including router and GPS, are run from an 8-amp DC-DC converter and a six-fuse distribution block. The remaining electronics, including the LabJack computer interface (described in Section 3.2.2) and encoders, are powered by a separate 5-volt converter. All components are protected by water-resistant breakers and replaceable fuses. A summary of power usage is shown in Table 3. This power distribution system performed well in the 2010 and 2011 IGVCs, and has been left largely unchanged, save for the aforementioned battery replacement.

3.2.2 Electronic Control

Phobetor's computer connects with most of its sensors and other electronics through a LabJack UE9 interface device. The LabJack facilitates communication with most lower-level electronics on the robot, including the two LDC1430C speed controllers used to power the drive motors. The robot steers by altering the speed and direction of each wheel.

The custom built motor control switching system, shown in Figure 5, provides emergency stop functionality, allows the user to remotely switch between RC and autonomous control, and drives *Phobetor's* indicator

Voltage	Device	Peak Power	Nominal Power	Idle Power
24	Drive Motors	5,280	1,180	0
24	Motor controllers	7.04	2.48	2.48
24	Computer	450	300	60
12	Router	6	4.8	3
12	Access Point	6	4.8	3
12	GPS	2	1.8	1.8
12	Compass	0.6	0.42	0.16
12	Warning Light	15	10	0
5	LabJack	0.8	0.58	0.43
5	Encoders	0.86	0.58	0.58
5	Gyroscope	0.33	0.25	0.25
5	E-Stop R/C Link	1	0.5	0.5
	Total	**5,755**	**1,396**	**72**

Table 3: Power Requirements for *Phobetor* (all power listed in Watts)

lights. PWM signals from the LabJack are routed through a pair of mechanical relays, which isolate and protect the control circuits while providing a reliable emergency stop mechanism at a low level. The relay inputs are connected to both the physical E-stop button and the 2.4 GHz spread-spectrum wireless radio system, which allows manual control of the motors with minimal noise and interference. The emergency stopping circuit is fail-safe, meaning that if a wire comes disconnected or the radio fails, the robot's motors will be disabled by default.

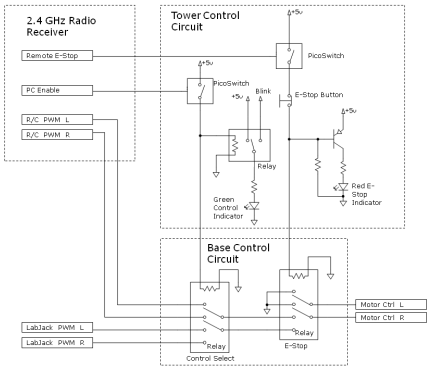


Figure 5: *Phobetor's* electronic control circuit

Phobetor operates with one on-board computer, offloading stereo image processing to Videre's on-board camera FPGA (see Section 3.2.3). *Phobetor's* main computer has been completely rebuilt for the 2013 IGVC. To minimize the computer's footprint, *Phobetor* utilizes a Micro ATX motherboard. *Phobetor's* computer consists of a latest-generation quad-core Intel Core i7 CPU at 3.5 GHz, 6 GB of RAM, and an 128 GB solid-state hard drive. Internet access is provided via a 802.11g Linksys® wireless router and a Cisco Wireless Access Point, which allows interfacing with JAUS systems and development machines using a WiFi or Ethernet connection.

3.2.3 Sensors

Environmental sensing in *Phobetor* is done primarily using stereo vision analysis but is augmented with absolute and differential sensors for awareness of local robot state. Most of the state variables are redundantly measured by multiple sensors, and their readings are combined and filtered over time to give an accurate corrected estimate of the robot's state.

Gyroscope
Phobetor uses a Gladiator Technologies G50 gyroscope for accurate 3 dimensional yaw rate data. Because our gyroscope operates independently of the wheel rotary encoders, we can reliably use gyroscope data even in the case of wheel slippage on low friction terrain. In most cases, though, the acceleration information from the gyroscope gets cross-checked by the rotary encoder for redundancy and to improve expected accuracy. This model of gyroscope offers a high resistance to noise (on the order of 0.0015/sec/Hz) and a G-sensitivity of less than 0.01/sec/g.

Camera
We use a Videre Design STOC-15 Stereo Camera with a 15cm baseline for stereo vision processing. Compared to other sensors such as laser range-finders, stereo vision offers a strategic advantage as a passive sensor; additionally, in can acquire additional scene information including colors and textures, and it is more economical and commercially scalable. However, it can be difficult to generate depth maps from stereo images at a high enough frame rate for real-time autonomous navigation. The STOC system incorporates a Field Programmable Gate Array (FPGA) into the camera to compute the three-dimensional point cloud using a highly parallelizable algorithm alleviating this problem.

GPS
Phobetor gets its location information from a Hemisphere GPS A100 GPS unit. This particular GPS unit offers accurate data even during periods of weak signals or signal outages by integrating COAST, software which allows the receiver to use old differential GPS correction data without affecting positioning quality. The A100 also integrates e-Dif software which enables receives to achieve differential GPS quality accuracies without a differential signal for regions where differential signals are not freely available. We use the GPS device for location, speed, and heading, the last two being cross checked by the rotary encoder and compass respectively.

Compass
We determine our heading for mapping and state estimation from an OceanServer OS5000-US Digital Compass. This device typically offers a 1 RMS accuracy with electronically gimbaled compensation for tilt giving us reliable heading information even on rough terrain when the robot may yaw unexpectedly. The heading information from our compass also gets cross checked by our GPS device.

Encoder
We have installed two US Digital HB6M hollow bore rotary encoders, one on each driving wheel of Phobetor. These devices are high resolution optical encoders with industrial strength bearings and housing to withstand the irregular vibrations of uneven terrain. The encoders provide wheel speeds and wheel accelerations which we send through a median filter to remove noise and cross check with our GPS and gyroscope.

4 Software

Phobetor's software employs a similar paradigm as *Argos* and *Kratos*, but on a completely different robot framework and operating system that has been newly implemented on ***PAVE***ś 2013 submission. It consists of independent processes that communicate asynchronously over the framework provided by Willow Garage's Robot Operating System (ROS). Individual modules capture and process sensor input (Section 4.5), estimate the state (position and direction) of the robot (Section 4.4), plan a desired path (Section 4.6), and determine the motor speeds necessary to follow the path (Sections 4.7 and 4.8). A holistic diagram showing module roles and message contents is displayed in Figure 6.

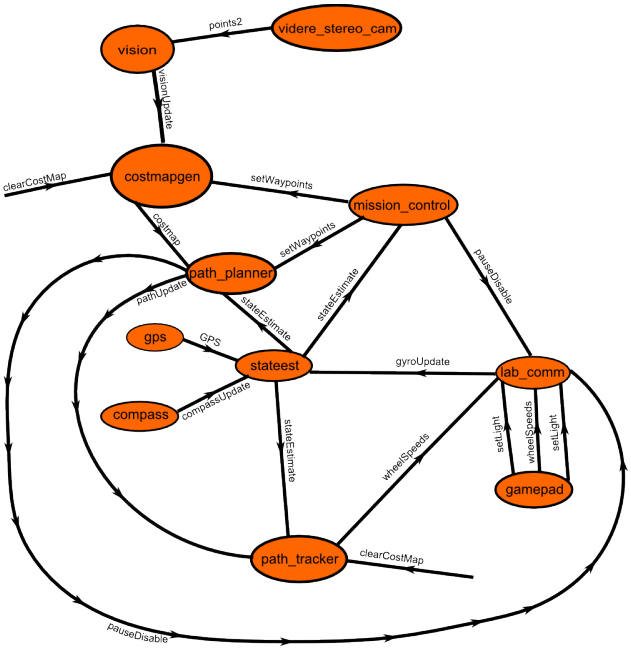


Figure 6: All packages and intercommunications between them

4.1 Code overview

In our code, 3 nodes are solely responsible for broadcasting sensor data: videre_stereo_cam, compass, and gps. All hardware control is done through lab_comm, which communicates with the LabJack to set the wheel speed, disable or enable the robot, and get gyro information, which is subsequently published. Stateest estimates the robot's position and heading by reading the compass, GPS, and gyro information, and publishes its state estimate. The vision node detects obstacles and lanes; costmapgen receives this information and generates a cost map, indicating the undesirability of driving to a certain point on the ground. path_planner takes this cost map, along with waypoint information from mission_control, to plan a path using Anytime D*. Path_tracker receives the path, and using the latest stateestimate, decides how to manipulate motor speeds to follow the path. It then tells lab_comm to set the motor speeds to the appropriate values.

4.2 Key Software Innovations

Phobetor's software includes a number of unique and innovative approaches to address the various design challenges. Notable among these are the implementation of a state-of-the-art variant of Kalman filter to fuse sensor data for state estimation (Section 4.4), algorithms for lane detection and validation developed entirely by ***PAVE*** members (Section 4.5), and highly optimized implementations of graph search algorithms for path planning (Section 4.6.3). We have maintained the core structure of these algorithms, but have continued to develop and optimize them for use with our new computing framework for the 2013 IGVC.

4.3 Computing Platform

Phobetor's computer runs 64-bit Ubuntu 12.04. All software is written in C++, with the Robot Operating System (ROS) as our underlying framework. Each software component, called a package, runs as a discrete program, called a node. Nodes communicate with each other using messages. To send or receive messages, nodes publish or subscribe to topics.

Besides facilitating communication, ROS offers a variety of convenient libraries. catkin, the latest ROS build tool, automatically resolves package dependencies in order to include and link to the appropriate files during compilation. The high-level Videre camera driver takes advantage of ROS's stereo image processing pipeline. tf handles transformations between different reference frames. The Time class is an easy way of getting the system time, and Timer provides an easy way to call a function periodically.

4.4 State Estimation

Phobetor's state estimation module implements a square root central difference Kalman filter (SRCDKF) [10] to combine data from all state sensors (compass, GPS, wheel encoders, and gyroscope) and maintain an optimal estimate of a vector that defines the state of the robot:

$$\mathbf{x} = [x, y, \theta, \delta, \omega, v_r, v_l]^T ,$$

where x is *Phobetor's* x coordinate in meters in a Cartesian local frame relative to its startup location, y is the vehicle's y coordinate in meters, $\theta \in [0, 2\pi)$ is heading, $\delta \in [0, 2\pi)$ is the bias between true GPS heading and magnetic compass heading, ω is the yaw rate of the vehicle, and v_r and v_l are the right and left wheel ground speeds in m/s, respectively.

The SRCDKF is a sigma point filter, similar to the unscented Kalman filter (UKF), utilizing a deterministic set of points to represent a multivariate Gaussian distribution over possible states and measurements. As opposed to other formulations, the SRCDKF is accurate up to the second order Taylor series expansion of the process and measurement models.

Parameters for Gaussian noise variables in the model were estimated by analyzing the long-term at-rest behavior of the sensors' signals. In all cases except the wheel encoders, the Gaussian random variable is an accurate representation of the sensor noise; for the encoders, it approximates the finite, discrete noise corrupting the train of digital pulses. The filter, which has been under development since 2009 [5, 2],

gives *Phobetor* a robust method of determining its state and accurately arriving at waypoints, even under conditions of wheel slippage or GPS outages.

4.5 Vision

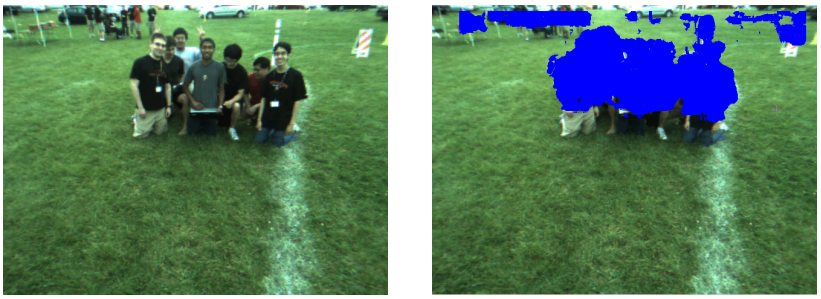
4.5.1 Obstacle Detection

Obstacle detection processes the point clouds generated by the cameras, in order to distinguish obstacle points from traversable locations. We implemente two a two-phased process, which is described in detail in a paper presented at the 2011 IS&T/SPIE Electronic Imaging Conference [9], and which will be reviewed here.

Firstly, we use a robust approach to fitting a ground plane, then thresholding points as obstacles based on their distance from the plane. Our ground plane fitting algorithm uses RANSAC (Random Sample Consensus), which involves alternately selecting random sets of inliers to a model, and fitting a better model to the inliers. Each model m_j is evaluated by the scoring function

$$s(m_j) = \sum_{p \in I_j} \frac{1}{1 + \left[d_{m_j}(p)\right]^2}$$

where $d(p)$ is the distance from the point to the ground plane m_j, and I_j is the inlier set at that time. Sample results of the ground plane fitting are shown in Figure 8. Note that this image was taken on a relatively flat surface, where the ground plane is approximately constant in the field of view.

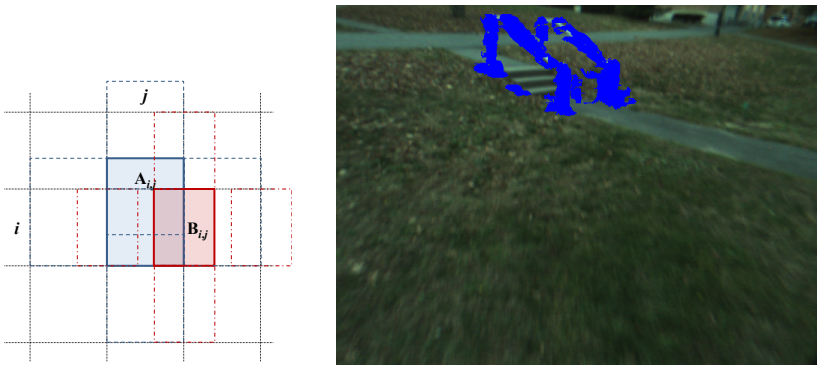


(a) Input image from stereo camera, which has a corresponding depth map.

(b) Output of obstacle detection. Pixels detected as obstacles are overwritten in blue.

Figure 7: Results of ground-plane obstacle detection.

To compensate for variations in the angle of the ground plane in the field of view, we use a parallelized implementation of Manduchi's algorithm [8], to take advantage of the multiple cores on *Phobetor's* computer. First, we ensure each pair of points is checked for compatibility only once. To parallelize the computations, we divide the image into a series of overlapping tiles. We show in [9] that a time-consuming computation can be computed independently for each tile using this formulation, and we ensure that many of the tiles fit in the L2 cache for lower access latency. This yeilds runtimes approximately 3x faster than the single-threaded implementation of Manduchi's algorithm.



(a) Overlapping tiles for parallel obstacle detection

(b) Output of obstacle detection. Pixels detected as obstacles are overwritten in blue.

Figure 8: Results of parallelized Manduchi et. al. obstacle detection.

4.5.2 Lane Detection

The lane detection algorithm used in *Phobetor* are a substantial improvement to our algorithms in previous years. Our algorithm has 3 phases: filtering, finding clusters, and processing clusters.

In the filtering phase, we try to identify lane pixels by applying a threshold in HSV space. Specifically, we look for pixels with high brightness and low saturation. Most lanes are identified, along with many false positives. Fortunately, false positives usually do not form long, thin clusters like lane points do.

In the cluster-finding phase, we take the binary image consisting of candidate lane points, and smooth it. Clusters are found by finding contours at some high value. We use the contour algorithm from OpenCV, which in turn uses the border-following algorithm from Suzuki 1985. An alternative is to flood fill the image to locate regions of high value, but this is slower than finding contours.

In the cluster-processing phase, we try to decide which clusters represent lanes and which are spurious. For each cluster, we start at one end and (metaphorically) draw several circles of decreasing radius around that end. If the cluster is part of a lane, the circles should intersect it along only 1 or 2 runs, and those runs should not be too thick. If a circle ever intersects the cluster at more than 2 runs, the cluster is divergent and therefore not a lane. Otherwise, we find an intermediate-sized circle that intersects the cluster at 1 or 2 runs, and repeat the cluster-checking process for that point. This process continues until we reach the other end of the cluster.

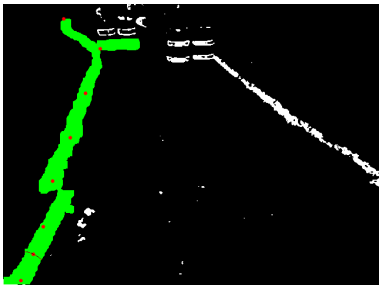


Figure 9: Our 'walk-the-line-filter' identifying lanes

After all lane clusters have been identified, cubic spline fitting is used to get a mathematical model of the lane. It does not matter if multiple overlapping lanes are identified, because all of these are eventually drawn on the cost map and treated as obstacles.

Our code contains robust methods for fusing and filtering lanes over time, which allows us to build a continuous piecewise model of the lane and to reject spuriously detected lanes. Because turns are guaranteed to have a minimum turning radius, we can use a road model which, for each of the left and right lane boundaries, rejects possible lane markings that disagree with an extrapolation of the history of detected lane markings. We approximate the "error" between any two lane markings in Cartesian global space to be the normalized sum of exponentially scaled differences between sampled sets of points, then threshold on this value [11]. The result is a pair of continuous boundaries that allows for effective autonomous navigation.

4.6 Navigation

Phobetor's environment is represented by a cost map incorporating lane and obstacle data. A desired waypoint is selected, and the path planning algorithm then calculates the most efficient trajectory to that waypoint given the constraints of the cost map.

4.6.1 Cost Map Generation

The purpose of cost map generation is to build a noise-tolerant model of the environment from obstacle, lane marking, and state data. *Phobetor* uses a similar architecture to that of previous years, assigning continuous costs (calculated with a moving average) to each point in a 10cm x 10cm grid of cells within our global map [5, 2]. With every vision update, the robot generates a local cost map of lanes and obstacles directly in front of it. To account for the physical geometry of the robot during path planning, a "footprint" cost map is employed, in which the cost of each cell is set to the sum of the costs of its neighbors [2].

4.6.2 Waypoint Selection

Phobetor's approach to waypoint selection is very similar to that of its previous iterations, which in turn built on the methods used in *Argos* [5]. In the auto-nav challenge, the desired waypoint is generated dynamically. The midpoint between the furthest points seen in the left and right lanes is found, and the cell with the lowest cost within a certain radius is chosen as the waypoint. In both cases, there is a periodic check on whether or not all waypoints have been reached. If they have not, the waypoint s that have been reached are published and the distance between the current position and the current waypoint is computed. If this distance is within a threshold, the waypoint is marked as reached and the robot proceeds to the next waypoint.

4.6.3 Path Planning

Phobetor employs the Anytime D* replanning algorithm which combines the time-efficiency of anytime algorithms with the D* search [7]. Anytime D* maintains a search history to reduce computation by updating paths each time new information is incorporated, but does so by finding an initial suboptimal

path and improving it. Currently the search considers 16 discrete headings, but more discrete levels can be considered if necessary.

4.7 Path Following

Phobetor uses a crosstrack error navigation law to follow paths, sequences of points $\vec{r}(n)$, generated by navigation algorithms [1, 6]. This algorithm minimizes the crosstrack error, $e(t)$, shown in Figure 10.

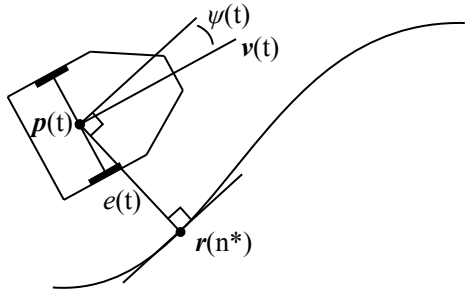


Figure 10: Crosstrack Error Law

$e(t)$ is defined as the signed, perpendicular distance from the center of the robot (midpoint between the wheels) to the closest point on the planned path. $e(t)$ and its time derivative are given by:

$$|e(t)| = \min_n \|\vec{r}(n) - \vec{p}(t)\|, \quad \dot{e}(t) = v(t)\sin(\psi(t)),$$

where $v(t)$ is the robot speed and $\psi(t)$ is *Phobetor's* heading with respect to the planned path. From the model above, the control law specifies a desired yaw rate

$$\omega_d(t) = k_h\psi(t) + \arctan\frac{k_e e(t)}{v(t)},$$

where k_h and k_e are tunable constants.

Because the path generator in our navigation algorithm generates paths choosing from 16 discrete directions, connecting adjacent and nearby cost map cells, the resulting paths are often not smooth. Therefore, we smooth the paths by fitting their component points with B-splines, piecewise polynomial curves that are at least C^1 continuous. Once this is done, the crosstrack error navigation law is suitable for path following.

4.8 Speed Control

To control the motor voltage $u(t)$, given a desired speed $v_d(t)$ (m/s) and the signed error $e(t)$ (m/s) between desired and actual speed, our IGVC entries since 2009 [1, 3, 4] have implemented a speed controller based

on proportional-integral control with a feed-forward term, given by

$$u(t) = f(v_d(t)) + k_p e(t) + k_i \int e(t)dt,$$

where k_p and k_i are constants tuned to minimize overshoot and are kept small so high-frequency noise from speed measurement input is not amplified in controller output.

The proportional and integral terms are familiar from standard PID controllers, where the integral term eliminates steady-state error. Because the rotational speed of the motor is not a linear function of voltage, a feed-forward term based on a model of the motor is necessary to compensate. To prevent *Phobetor* from accelerating too quickly, which might lift the front wheel off the ground or cause the drive wheels to slip excessively, we limit the magnitude of changes in controller output. Also, when the motor output voltage saturates, we prevent integral windup to keep the controller responsive to later setpoint changes.

5 Conclusion

Phobetor is a reliable, robust, and innovative autonomous platform that builds off of the experience gained through ***PAVE***ś previous IGVC performances. A more robust and durable electrical system, a faster, more advanced computing environment, and well-tuned and tested software took priority in this year's design process. By refining the system with a long life span and low maintenance in mind, we believe that we were able to make significant improvements to our robot. We are proud of our final product and eager to demonstrate its capabilities in the upcoming competition.

References

[1] Solomon O Abiola, Christopher A Baldassano, Gordon H Franken, Richard J Harris, Barbara A Hendrick, Jonathan R Mayer, Brenton A Partridge, Eric W Starr, Alexander N Tait, Derrick D Yu, and Tony H Zhu. Argos: Princeton University's Entry in the 2009 Intelligent Ground Vehicle Competition. 2009.

[2] Solomon O Abiola, Christopher A Baldassano, Gordon H Franken, Richard J Harris, Barbara A Hendrick, Jonathan R Mayer, Brenton A Partridge, Eric W Starr, Alexander N Tait, Derrick D Yu, and Tony H Zhu. Argos: Princeton University's Entry in the 2009 Intelligent Ground Vehicle Competition. In *Intelligent Robots and Computer Vision XXVII*, volume 2, 2010.

[3] Solomon O Abiola, Ryan M Corey, Joshua P Newman, Srinivasan A Suresh, Laszlo J Szocs, Brenton A Partridge, Derrick D Yu, and Tony H Zhu. Phobetor: Princeton University's Entry in the 2010 Intelligent Ground Vehicle Competition. 2010.

[4] Solomon O. Abiola, Ryan M. Corey, Joshua P. Newman, Laszlo J. Szocs, Brenton A. Partridge, and Tony H. Zhu. Phobetor: Princeton University's Entry in the 2011 Intelligent Ground Vehicle Competition. 2011.

[5] Christopher A. Baldassano, Gordon H. Franken, Jonathan R. Mayer, Andrew M. Saxe, and Derrick D. Yu. Kratos: Princeton University's Entry in the 2008 Intelligent Ground Vehicle Competition. In *Proceedings of IS&T/SPIE Electronic Imaging Conference*, volume 7252, 2009.

[6] Gabriel M. Hoffmann, Claire J. Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In *Proceedings of the 26th American Control Conference*, pages 2296–2301, 2007.

[7] Maxim Likhachev, David Ferguson, Geoffrey Gordon, Anthony (Tony) Stentz, and Sebastian Thrun. Anytime dynamic a*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, June 2005.

[8] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle Detection and Terrain Classification for Autonomous Off-Road Navigation. *Autonomous Robots*, 18(1):81–102, 2005.

[9] Joshua Newman, Han Zhu, Brenton A. Partridge, Laszlo J. Szocs, Solomon O. Abiola, Ryan M. Corey, Srinivasan A. Sureh, and Derrick D. Yu. Phobetor: Princeton University's Entry in the 2011 Intelligent Ground Vehicle Competition. In *Proceedings of IS&T/SPIE Electronic Imaging Conference*, 2011.

[10] Rudolph van der Merwe and Eric A. Wan. Sigma-Point Kalman Filters For Integrated Navigation. In *Proceedings of the 60th Annual Meeting of The Institute of Navigation (ION)*, pages 641–654, 2004.

[11] Derrick D. Yu. A Robust Method of Validation and Estimation of Road Lanes in Real Time for Autonomous Vehicles. 2009.

Special Thanks

The 2011 IGVC team thanks our team advisor, Professor Alain Kornhauser, for his support of this project. We would also like to express our gratitude to the Princeton School of Engineering and Applied Sciences, the Keller Center for Innovation in Engineering Education, and the Norman D. Kurtz '58 fund for their gracious donation of resources. Also, we thank Stephanie Landers of the Keller Center and Tara Zigler of the Department of Operations Research & Finance Engineering for their tremendous logistical assistance. Our team could not have gone far without the encouragement and assistance of the numerous professors, students, and faculty members of Princeton University, and we would like to thank all who have continued to provide us with the feedback and help that has allowed us to come this far.