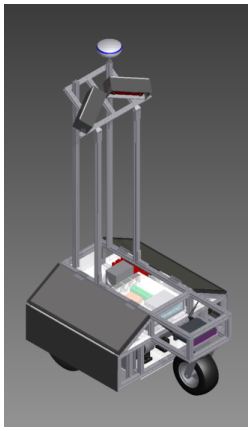
Phobetor: Princeton University's Entry in the 2011 Intelligent Ground Vehicle Competition

Solomon O. Abiola, Ryan M. Corey, Joshua P. Newman,
Laszlo J. Szocs, Brenton A. Partridge, and Tony H. Zhu

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

May 10, 2011



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 a redesigned tower, new sensors, and novel software algorithms and optimizations.

Clarence Rowley

PRINCETON
School of Engineering and Applied Science

PAVE PRINCETON AUTONOMOUS VEHICLE ENGINEERING

Contents

1 Team Overview

Princeton University's IGVC team consists of members of Princeton Autonomous Vehicle Engineering (***PAVE***), Princeton University's undergraduate student-led robotics research group. Our team builds upon ***PAVE***'s experience in robotics competitions, including participation in the 2005 DARPA Grand Challenge [4], the 2007 DARPA Urban Challenge [9], the 2008 Intelligent Ground Vehicle Competition (IGVC) [5], and the 2009 IGVC [1]. In the 2008 IGVC, our team placed third overall and won rookie-of-the-year, placing 1st, 4th and 6th in the Design, Navigation and Autonomous challenges, respectively. *Argos*, our entry in the 2009 IGVC, placed fourth overall, winning the Navigation challenge and successfully completing the JAUS challenge. Our entry for the 2010 IGVC, *Phobetor*, took second place in the Design competition, but due to last minute malfunctions did not place in any challenges. We are confident the updated *Phobetor* and will be a competitive force once again in the 2011 IGVC.

Our team is composed of undergraduate students from several engineering and non-engineering departments.[1] As in previous years, we maintained a straightforward and relatively flat organizational structure, as shown in Figure 1. Team members were grouped into hardware or software teams according to their area of expertise. The hardware group is responsible for all physical aspects of the robot, including design, fabrication, sensor selection, electronics, electrical wiring and computer hardware. The software group is responsible for algorithm design and programming implementation. Primary tasks include sensor processing, intelligent navigation schemes and robust feedback control systems. To oversee these groups, we dedicated a student team leader to specialize in project management. Overall, over 600 person-hours this academic year have been spent working on *Phobetor* and its specific software improvements.

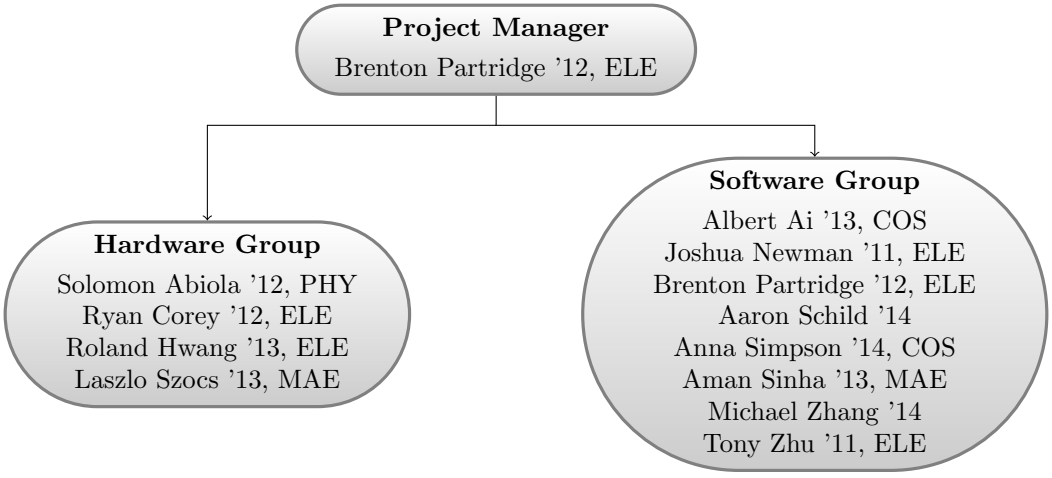


Figure 1: Team Organization Diagram

2 Design Process

After the 2010 IGVC competition, ***PAVE*** made an effort to improve our design process to ensure that we meet our objectives on time. Our process is focused on collaborative design, communication, and docu-

[1] Departments represented include COS (Computer Science), ELE (Electrical Engineering), MAE (Mechanical and Aerospace Engineering), and PHY (Physics).

mentation. The team uses an internal wiki, Dropbox, subversion server, and other collaborative tools to ensure that all group members, especially newcomers, have access to information, code, and data. We hold regular in-person meetings to discuss ideas and make sure that every team member understands the larger picture. A particular concern is coordination between our hardware and software teams, which operate fairly independently. To enable rapid testing of new code while the robot is serviced, the software team can use *Argos*, our old robot, which has the same basic interface and physical dimensions.

In a meeting after the 2010 competition, our hardware team focused on ways to make the robot more robust, reliable, and easy to maintain. The consensus was that the least polished part of *Phobetor* in 2010 was the tower, which took a great deal of time and effort to take apart, was difficult to waterproof, and was not aesthetically pleasing. We decided to completely restructure the tower, as described in Section 3.1.3. To reduce cable clutter, the electronic control circuits were redone, and can be found in Section 3.2.2. The new tower also allows the robot to support a second stereo-vision camera, which provides our software designers with better visual data to work with; the two-camera approach is described in Section 3.2.3.

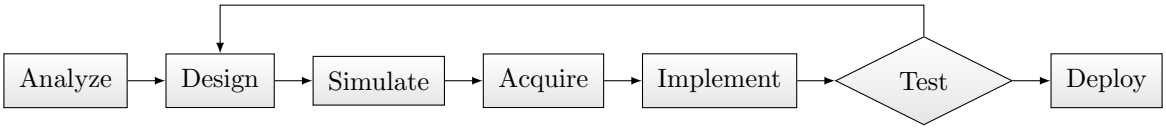


Figure 2: Flowchart of Design Process

Hardware	**Software**
1. Tower difficult to service and disassemble for transportation	1. Performance of obstacle detection limited image resolution that could be processed in real-time
2. Inefficient wiring, especially between base and tower	2. Incorrect fusing of detected lane markings into continuous boundaries for navigation
3. Awkward and bulky camera housing	3. Performance of path planning limited system response time
4. E-Stop indicator not visible enough	

Table 1: Areas of *Phobetor* Targeted for Improvement

Our goals for the 2011 iteration of *Phobetor* are enumerated in Table 1. Once we had established 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. The software group created

original simulation tools that allow new vision alogirthms to be tested in a virtual environment, without the time and hassle of taking the robot outside.

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. 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 redesigned and built from the ground up in 2010. This year, we refined the existing design to make the robot more robust and easier to maintain. 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)	$215	$430
NPC T64 Motor, .7 HP (2x)	$286	$572
IFI Robotics Victor 885 Motor Controller (2x)	$440	$440
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:	**$11,192**	**$7,916**

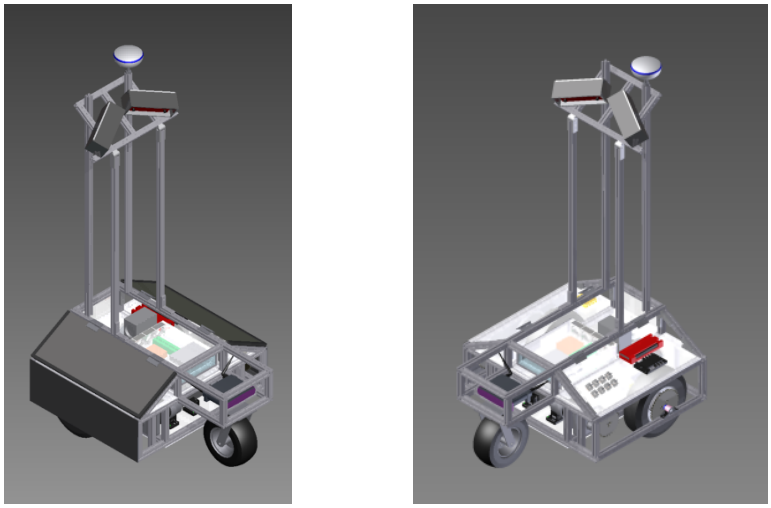
*Denotes donated item. All other zero-cost items were borrowed from *Argos*, the 2009 robot.

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 newly built sensor tower, which is designed to provide *Phobetor* with a vantage point equivalent to that of

an average human. 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. A CAD visualization of the robot can be seen in Figure 3a, with a detailed interior image shown in Figure 3b.



(a) CAD Render of *Phobetor*

(b) CAD render showing internal components

Figure 3: Visualizations of *Phobetor*

3.1.1 Drivetrain

Phobetor's drivetrain is the result of years of iteration from previous IGVC robots. Whereas lower-voltage motors struggled to accelerate, climb hills, and reach the maximum allowed speed, 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 NPC T64 motors with Victor 885 motor controllers. To prevent the loss of chain tension after strenuous driving, we added tensioners to the wheel bearings, using a bolt to hold them in place and prevent them from slipping. The motor mount was redesigned to separate structural supports from the sprockets, so that the chain would not rub against other components.

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. Compared to our previous robots, *Phobetor* makes more efficient use of chassis space - the base is three inches shorter - and is far easier to maintain. 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. The electronic components were laid out to minimize wire clutter, and are attached with screws, adhesives, and cable ties.

Phobetor was designed to be 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. To maintain a waterproof seal on the paneling in changing temperatures, we had to account for the materials' different expansion coefficients. The previous material, polyethylene, has a linear expansion coefficient of $111 \times 10^{-6} \frac{\text{in}}{\text{in} \circ \text{F}}$ and would cause the panels to bend and warp. *Phobetor's* exterior paneling uses 1/8" MDS Filed Nylon 6/6, which boasts a coefficient of only $44.4 \times 10^{-6} \frac{\text{in}}{\text{in} \circ \text{F}}$. For the moveable doors, we decided to use 1/8" Garolite, a sturdier and more rigid material with an impressive expansion coefficient of $6.6 \times 10^{-6} \frac{\text{in}}{\text{in} \circ \text{F}}$. This combination of materials ensures that our robot will resist temperature changes and that the paneling will adequately protect the electronics.

3.1.3 Tower Design

While the base of *Phobetor's* chassis was only slightly revised from last year, the tower has been entirely rebuilt. To accomodate a second stereo-vision camera (discussed in Section 3.2.3), the camera housing was rebuilt 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.

To make the tower easier to maintain and transport, as well as reduce its bulk, all paneling has been removed. The electronic components in the tower, including the gyroscope, 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. The wires from the GPS unit and the base connect to the box with sealed strain reliefs and sockets. A new plug panel contains watertight connectors for the 5V and 12V power supplies, control signals, and USB and Firewire connections. Whereas before we had to disconnect each individual wire to remove the tower, now we can simply uplug these cables from the panel and fold the tower down for transport.

3.2 Electrical and Electronic Systems

Phobetor's electrical and electronic systems performed well in last year's competition. For 2011, we've made minor updates, replaced some failure-prone systems with more robust components, and simplified connections between the base and the tower.

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.

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

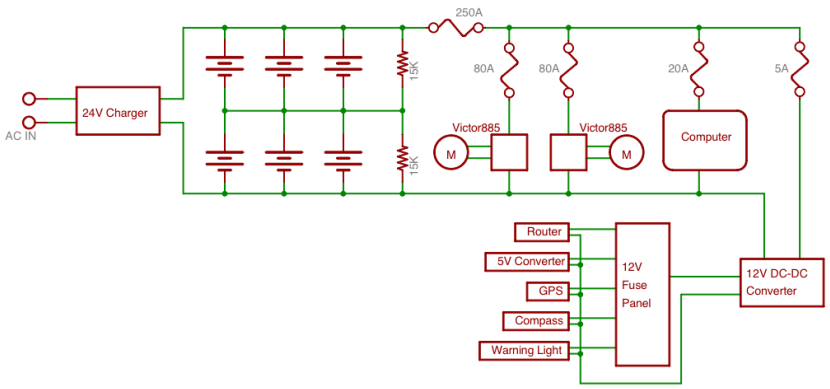


Figure 4: Electrical System

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.

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)

3.2.2 Electronic Control

Phobetor has a new, simplified control system for 2011. The computer connects with most of its sensors and other electronics through a LabJack UE9 interface device. The computer controls the robot's motion using simple pulse width modulation (PWM) signals, one for each wheel. The PWM signals from the LabJack drives a pair of Victor 885 motor controllers. The robot steers by altering the speed and direction of each wheel.

The main custom-built circuit, which was redesigned for this year, is the motor control switching system. This circuit, shown in Figure 5, provides emergency stop functionality, allows the user to remotely switch between RC and autonomous control, and drives the new indicator 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.

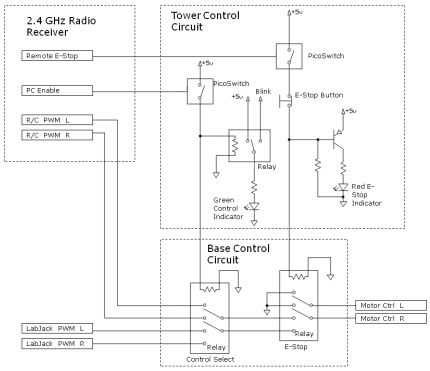


Figure 5: *Phobetor's* electronic control circuit

Both relays are wired in parallel with red and green indicator lights on the tower, which show the state of the robot. When the robot is powered on and the 5V system is active, the green indicator is steadily lit. When the robot enters autonomous mode, the light flashes using a simple timer-based flashing circuit. The red light comes on whenever the E-Stop is active. In the event of a power failure in the 5V system, the red light will be off, but the E-Stop will still be active. The lights are controlled by circuits in the tower's control box, reducing the number of connections to the base.

Phobetor operates with one on-board computer, offloading stereo image processing to Videre's on-board camera FPGA (see Section 3.2.3). To minimize the computer's footprint, *Phobetor* utilizes a Micro ATX motherboard with onboard firewire ports, eliminating the usage of a firewire card. *Phobetor's* computer consists of a quad-core Intel Core i7 CPU at 2.66 GHz, 6 GB of RAM, and an 500 GB hard drive. Internet access is provided via dual 802.11g Linksys® wireless routers; this also allows interfacing with JAUS systems and development machines using a WiFi or Ethernet connection.

3.2.3 Sensors

Environmental sensing in *Phobetor* uses two Videre Design stereo-on-chip (STOC) color cameras with a 15 cm baseline. Compared to other sensors such as laser range-finders, stereo vision offers a strategic advantage as a passive sensor; additionally, it 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.

Besides incorporating these advanced stereo vision systems, *Phobetor* also uses absolute and differential sensors for awareness of robot state. The robot is equipped with a Hemisphere GPS unit, US Digital wheel encoders to generate high-precision and low-drift data for wheel angular velocity, an OceanServer OS500-US

digital compass for accurate heading information, and a Gladiator G50 MEMS gyroscope to provide accurate yaw rate data even in the event of wheel slippage. Most of the state variables are redundantly measured by multiple of these sensors, and their readings are combined and filtered over time to give an accurate, corrected estimate of the robot's state, as described in Section 4.3.

4 Software

Phobetor's software employs the same paradigm as *Argos* and *Kratos*, consisting of independent processes that communicate asynchronously over our publish/subscribe messaging framework, IPC++. Individual modules provide JAUS compliance (Section 4.2), capture and process sensor input (Section 4.4), estimate the state (position and direction) of the robot (Section 4.3), plan a desired path (Section 4.5), and determine the motor speeds necessary to follow the path (Sections 4.6 and 4.7). A holistic diagram showing module roles and message contents is displayed in Figure 6.

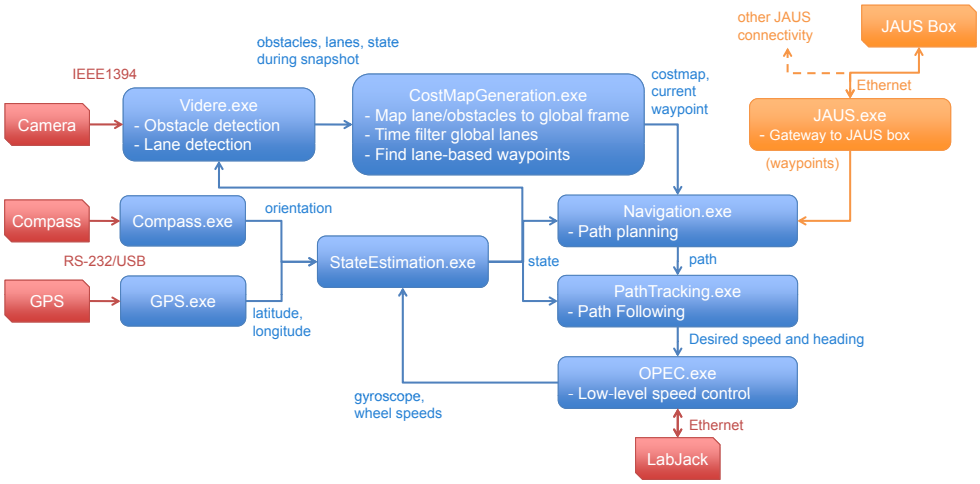


Figure 6: Diagram of Software Architecture

4.1 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 4.3), algorithms for lane detection and validation developed entirely by ***PAVE*** members (Section 4.4), and highly optimized implementations of graph search algorithms for path planning (Section 4.5.3).

4.2 Computing Platform

Phobetor's computer runs the Windows Server 2008 operating system, which improves upon Windows Server 2003 (used on *Kratos'* computers) by incorporating support for the High Precision Event Timer (HPET) included on modern motherboards. This allows for timers with millisecond accuracy and sub-millisecond jitter. All software is written in platform-independent C++ using Visual Studio IDE for ease of development.

We continue to employ IPC++, an object-oriented wrapper of Carnegie Mellon's Inter-Process Communication (IPC) platform [13], as our underlying robotics framework. Each software component runs as a discrete program communicating over TCP with a central server; message publishing and subscription, serialization, and timing are all abstracted with a custom-developed C++ API [2].

Because IPC++ is a similar paradigm to JAUS, implementing the communication protocol is rather simple, with an additional process translating between external JAUS messaging and internal IPC++ messaging. The process replies to queries from the COP with the latest update of the robot's state. For control messages sent from the COP, such as start/stop and set waypoints, the translator simply sends the appropriate equivalent message over IPC++ [2].

4.3 State Estimation

Phobetor's state estimation module implements a square root central difference Kalman filter (SRCDKF) [14] 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, and it ensures numerical stability and efficiency by maintaining symmetric and positive definite error covariance matrices.

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 [6, 2], gives *Phobetor* a robust method of determining its state and accurately arriving at waypoints, even under conditions of wheel slippage or GPS outages.

4.4 Vision

4.4.1 Obstacle Detection

Obstacle detection processes the point clouds generated by the cameras, in order to distinguish obstacle points from traversable locations. In previous competitions, we have used an algorithm developed by Manduchi et. al. [11], in which we search through all pairs of points for ones that are nearly vertical relative to each other. However, we found that when increasing the image resolution to 640 by 480 pixels, this implementation was

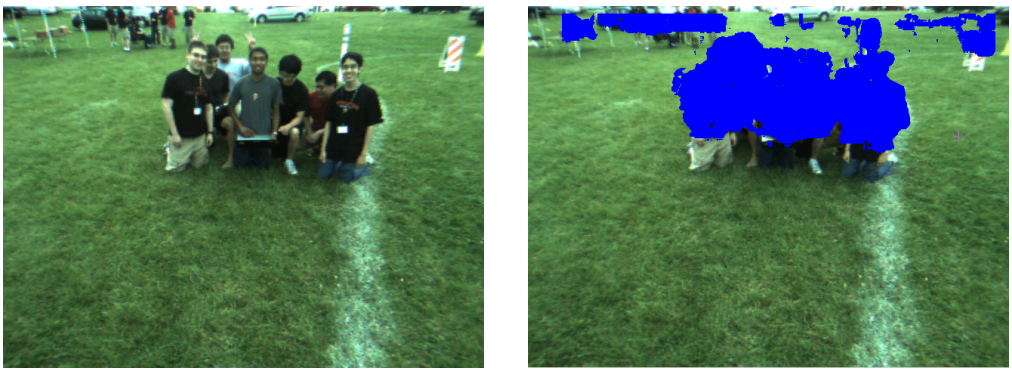
not fast enough to keep our robot responsive. We implemented two alternatives, which were described in detail in a paper presented at the 2011 IS&T/SPIE Electronic Imaging Conference [12], and which will be reviewed here.

Firstly, we implemented 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. The number of iterations of the algorithm can be tuned to improve accuracy or improve response time, and the computational efficiency depends on the least squares implementation.

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. Although this algorithm is very performant when this is the case, we found in testing that the ground plane often varies in the field of view.

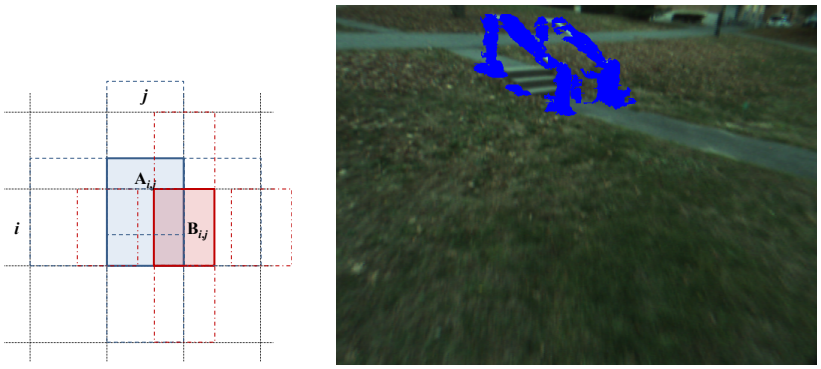


(a) Input image from stereo camera, which has a corresponding depth map.

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

Figure 7: Results of ground-plane obstacle detection.

Therefore, we chose to create a parallelized implementation of Manduchi's algorithm, 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 [12] 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. When we timed our single-threaded implementation and the parallel one running on 640×480 test images collected from our camera, the parallel implementation took on average 150 milliseconds per frame when approximately half the frame contains obstacles, while the single-threaded implementation was 3x slower.



(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.

4.4.2 Lane Detection

The lane detection algorithms used in *Phobetor* represent significant innovations in image processing. Our basic algorithm, developed for the DARPA Urban Challenge in 2007, applies filters for yellow content, white content, pulse width, and obstacles detected in the same frame, then fuses the results into a heat map which is searched for lines [6]. To ensure that lane markings within shadows are not ignored, *Phobetor* and *Argos* use a novel white filter operating in hue-saturation-value space [15], which utilizes separate brightness thresholds based on local saturation. The RANSAC algorithm then searches for the parabola that passes near the most on-pixels in the heat map [7, 6]. This algorithm can tolerate gaps in the heat map lane markings, making it ideal for handling the dashed lines in the autonomous challenge. Various steps and results of lane detection are shown in Figure 9. Response time is less than one second, which has proven sufficient in the past two competitions.

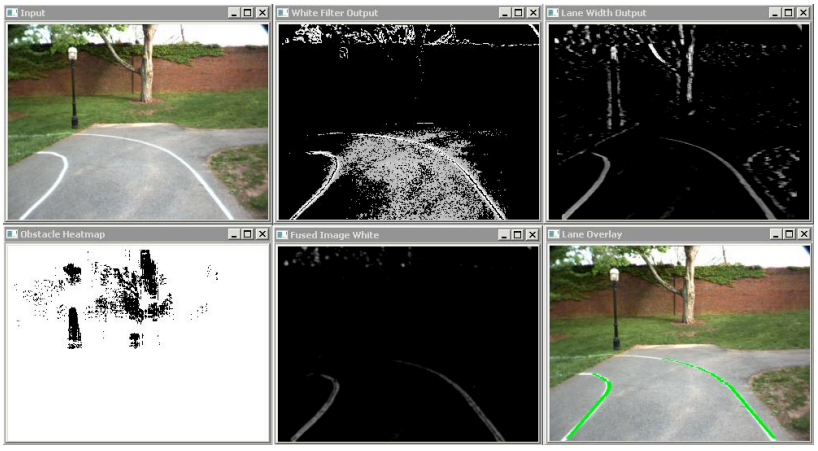


Figure 9: Stages of Lane Detection [6]

This year, the majority of our vision development efforts have focused on developing robust methods of 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 [16]. The result is a pair of continuous boundaries that allows for effective autonomous navigation.

4.5 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.

4.5.1 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 [6, 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].

4.5.2 Waypoint Selection

Phobetor's approach to waypoint selection is very similar to that of *Argos* [6]. For the navigation challenge, *Phobetor* iterates through a presorted list of waypoints. The presorting can then be done in non-real-time, finding the shortest overall path by checking all possible permutations. In the autonomous 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, a new waypoint is only submitted to path planning once the robot has approached the current waypoint.

4.5.3 Path Planning

For both the autonomous and navigation challenges, provided that the lane markings in the former are represented as well-defined continuous boundaries (see Section 4.4), a variant of an A* graph search algorithm is ideal to handle arbitrarily complex obstacle collections such as switchbacks. Last year, *Argos* used the Dynamic A* Lite algorithm (D* Lite) which guarantees optimality and completeness of the path it finds through the search space like the a simple A* planner, but also uses knowledge from previous searches to reduce computation for each search. This provided an order of magnitude speed improvement over the basic algorithm, which restarts the search with each iteration. However, we determined that further optimization was necessary to improve reaction time.

There is a different class of search algorithms, anytime algorithms, that produce suboptimal paths but can complete the search in some given time. These start by producing a highly suboptimal path with bounded

cost and then improving the path in any remaining available time. This year on *Phobetor*, we implemented the Anytime D* replanning algorithm which combines the time-efficiency of anytime algorithms with the D* search [10]. 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 rather than the continuous search used in *Argos*, but more discrete levels can be considered if necessary.

4.6 Path Following

Phobetor uses a crosstrack error navigation law to follow paths, sequences of points $\vec{r}(n)$, generated by navigation algorithms [1, 8]. This algorithm minimizes the crosstrack error, $e(t)$, shown in Figure 10.

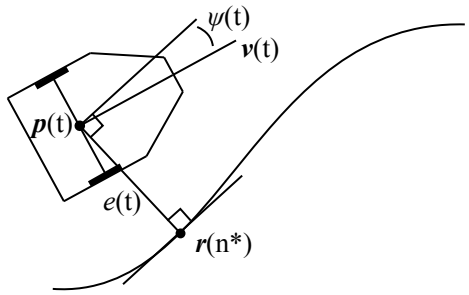


Figure 10: Crosstrack Error Law

$e(t)$ 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.

4.7 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] 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. Figure 11 shows a block diagram of the controller from a Simulink model.

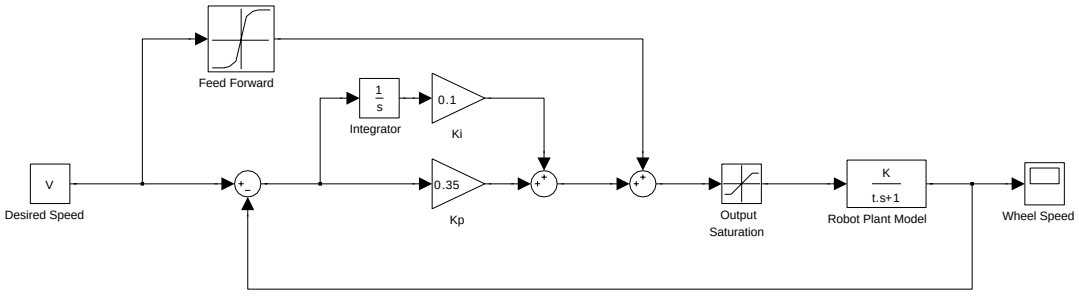


Figure 11: Block Diagram of Speed Controller

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. Additionally, there are slight modifications to the controller to compensate for nonlinearities in the plant. 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 success of our team's two previous entries. Enhancing *Phobetor*'s capabilities, creating a more streamlined and low profile chassis, and augmenting the vision and path planning software algorithms for *Phobetor* took priority as we approached the design process for this year's entry. 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] Anand R. Atreya, Bryan C. Cattle, Brendan M. Collins, Benjamin Essenburg, Gordon H. Franken, Andrew M. Saxe, Scott N. Schiffres, and Alain L. Kornhauser. Prospect Eleven: Princeton University's Entry in the 2005 DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):745–753, 2006.

[5] Christopher Baldassano, David Benjamin, Benjamin Chen, Gordon Franken, Will Hu, Jonathan Mayer, Andrew Saxe, Tom Yeung, and Derrick Yu. Kratos: Princeton University's Entry in the 2008 Intelligent Ground Vehicle Competition. IGVC Technical Paper 2008, May 2008.

[6] 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.

[7] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[8] 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.

[9] Alain Kornhauser, Issa Ashwash, Christopher Baldassano, Lindsay Gorman, Jonathan Mayer, Andrew Saxe, and Derrick Yu. Prospect Twelve: Princeton University's Entry in the 2007 DARPA Urban Challenge. Submitted to *IEEE Transactions on Intelligent Transportation Systems*, 2008.

[10] 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.

[11] 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.

[12] 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.

[13] Reid Simmons and Dale James. *Inter-Process Communication: A Reference Manual*, August 2001.

[14] 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.

[15] Derrick D. Yu. Building A Robust Real-Time Lane Detection Algorithm. 2008.

[16] 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 Clarence W. Rowley, for his continued 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.