Phobetor: Princeton University's Entry in the 2010 Intelligent Ground Vehicle Competition

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

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

May 17, 2010

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* is a newly constructed vehicle with numerous design modifications, including a redesigned chassis with reduced volume, increased serviceability, and improved waterproofing.

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*, is an all-new robot and represents a significant evolutionary step in hardware design and software innovation. We are confident that *Phobetor* and ***PAVE*** will be a competitive force in the 2010 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.

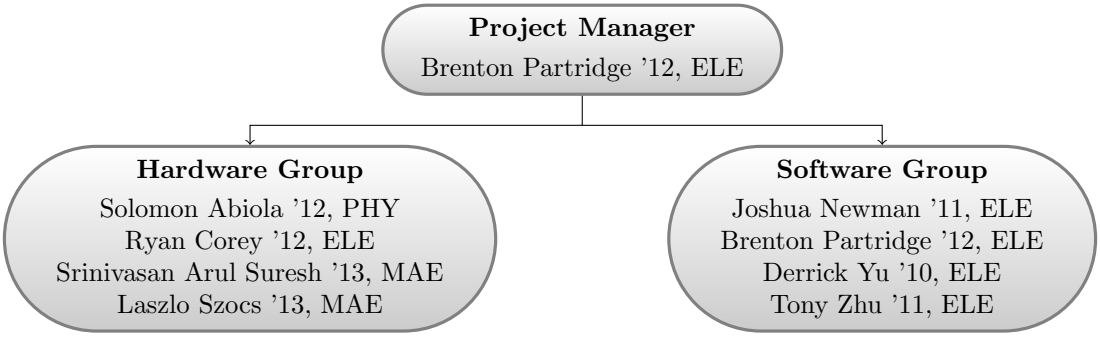


Figure 1: Team Organization Diagram

2 Design Process

As the 2010 IGVC competition presents similar challenges to those in previous years, ***PAVE***'s basic design objectives have remained approximately the same. Effort this year was primarily focused on improving upon the successful elements of previous designs. Utilizing the design process illustrated in Figure 2, the team began with a critical assessment and evaluation of last year's robot, determining the strengths and shortcomings of that design. The resulting requirements, enumerated in Table 1, were prioritized and compared

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

against the ability and knowledge of the current year's team, motivating a specific set of design features for *Phobetor*. Specific hardware components and software modules were modeled independently and integrated into higher-level models of hardware layout and software communications protocols. In the acquisition and implementation stages, separate subsystems were developed in parallel as much as possible to minimize development time and maximize testing. If any subsystem failed tests done on it independently, it was redesigned accordingly and brought through the process again before integration testing was attempted.

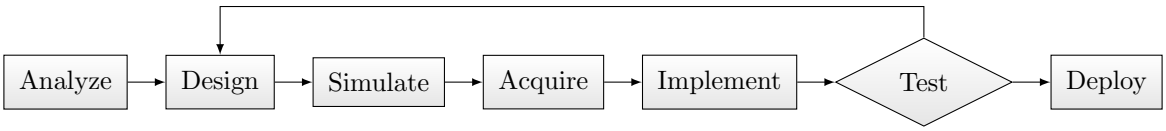


Figure 2: Flowchart of Design Process

Hardware	**Software**
1. Drivetrain suffered from noise from loss of chain tension	1. Performance of obstacle detection limited image resolution that could be processed in real-time
2. Inefficient use of space in *Argos'* base	2. Incorrect fusing of detected lane markings into continuous boundaries for navigation
3. Lack of effective waterproofing made *Argos* a less robust robotic platform	3. Failure to reject spurious lanes
4. Paneling material had high coefficient of thermal expansion, allowing some water to enter tower	4. Performance of path planning limited system response time
5. Structure of midsection prevented easy access for servicing	

Table 1: Areas of *Argos* Targeted for Improvement

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

To address the shortcomings in the mechanical design of our 2009 entry, we decided to build a completely new robot from the ground up. In doing so, we were able to address all of the hardware design flaws listed in Section 2 and make several other innovations. We examine in detail the redesign of the robot's drivetrain and chassis in Section 3.1, and changes to our electrical system are discussed in Section 3.2. We discuss our sensors and electronics in Sections 3.3 and 3.4. 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	$1,660	$1,660
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:	**$9,532**	**$6,256**

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

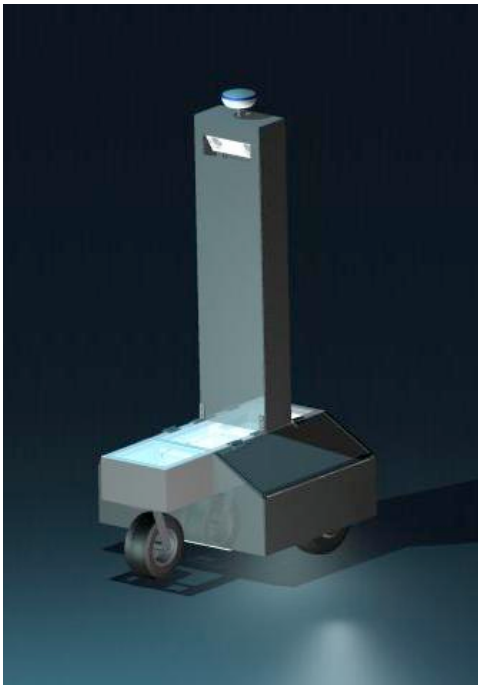
Table 2: List of Costs for Building *Phobetor* (all trademarks property of their respective owners)

3.1 Mechanical Design

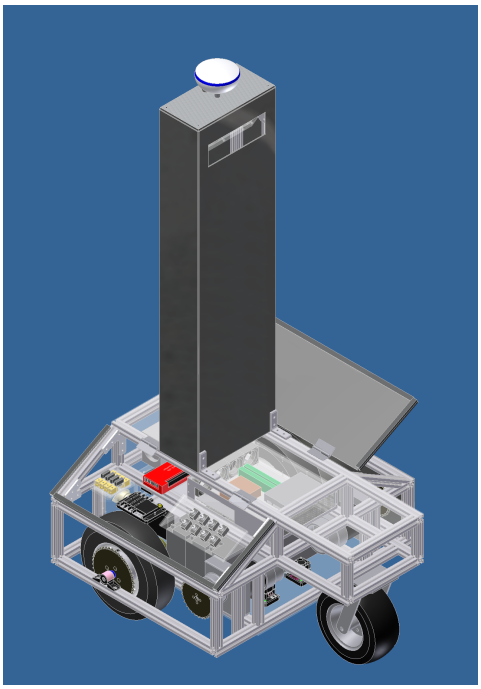
Phobetor, like last year's *Argos*, measures 31" wide, 37" long, and 69.5" tall; it weighs approximately 270 pounds excluding payload. The robot features a tricycle wheelbase with two powered rear wheels and a leading caster; this design ensures contact with the ground at all times, regardless of terrain. 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. As with all ***PAVE*** robots, *Phobetor* was designed entirely in CAD before any material was cut. Making note of the electronics used last year, we were able to model all of the components that would be placed in the robot's interior and choose spatially efficient layouts. Using Autodesk Inventor 2010 in the design process accelerated the redesign from last year's platform, 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.

3.1.1 Drivetrain

This year's drivetrain represents the latest in a series of revisions beginning with *Kratos*. *Kratos'* system experienced problems accelerating to speed, climbing hills, and reaching maximum allowed speed. Last year's drivetrain introduced a 24V electrical system to address these problems: *Argos'* drivetrain provided the required performance in various driving modes, including full speed in a straight line, zero-radius turning, and driving up a 10° incline, all on grass. On *Phobetor*, after verification that the wheelbase and weight distribution would be approximately the same, we used identical NPC T64 motors to those in *Argos*.



(a) CAD Render of *Phobetor*



(b) CAD render showing internal components

Figure 3: Visualizations of *Phobetor*

Argos had a few issues to correct, however, as it suffered from occasional loss of chain tension after strenuous driving, as well as the chain impinging on a motor support. On *Phobetor*, this was corrected by first redesigning the motor mount so that there are no structural or support members in the plane of the sprockets. Second, the wheel bearings have tensioners on them, using a bolt to ensure that the bearings cannot slip forward under tension as they did on *Argos*.

3.1.2 Chassis Innovations

Phobetor has an entirely new chassis built from 80/20 structural framing aluminum. *Phobetor* shares much of *Argos'* chassis layout; the two-layer organization allows all heavy components to be placed near the center of rotation, lowering *Phobetor's* moment of inertia. Both horizontal dimensions were preserved, allowing *Phobetor* to fit through a standard door. *Phobetor's* chassis redesign was driven primarily by a need to increase the volume efficiency of the chassis, as well as to facilitate maintenance of the robot's systems. To this end, the second layer was shortened by approximately three inches in height. In order to preserve the camera's viewpoint, the height of the tower was increased accordingly. The second level also features two large doors, one on each side. When open, these allow for easy access to every component on the second level.

In designing for serviceability as well as full waterproofing (according to the design considerations in Table 1), we were forced to sacrifice some accessibility of the lower level; however, as opposed to *Argos*, the joints in *Phobetor's* frame were oriented to make exposing that level a simple and quickly reversible operation. *Phobetor* was designed to improve on *Argos'* waterproofing, and this played a key role in the chassis redesign. The wheel wells have been sealed better, and the doors are angled to allow water to run off. The doors are

edged with rubber-bulb seals, and are mounted on a continuous hinge, thereby sealing the full perimeter of these sections.

Our innovative design feature of waterproofing also required careful consideration of materials for doors and panels. During outdoor testing, the paneling material on *Argos'* tower began to expand in high temperatures, developing a wave shape that could not be adequately sealed. The material previously used, polyethylene, had a linear expansion coefficient of $111 \times 10^{-6} \frac{\text{in}}{\text{in} \circ \text{F}}$. This year, the material used for the tower and exterior paneling was 1/8" MDS Filed Nylon 6/6, which boasted 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.

3.2 Electrical Design

Phobetor's drive motors, computer, and electronics are powered by a 24-volt electrical system. Although the robot is required to run for only six minutes during the competition, our 132 amp-hour battery bank provides power for 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.

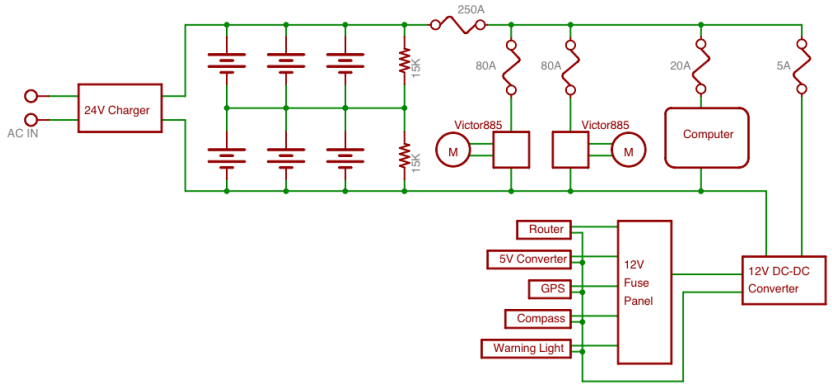


Figure 4: Electrical System

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 3.4) 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)

3.3 Sensors

Environmental sensing in *Phobetor* relies on a single Videre Design stereo-on-chip (STOC) color camera with a 15 cm baseline. This is the same camera that was used in the 2009 entry *Argos* and has proven to be a reliable and efficient stereo optical system that fulfills all of our requirements. By detecting both lanes and obstacles on this device, the camera avoids the usual, error-laden frame-of-reference transformations that come with using a different camera for each task. Additionally, an onboard Field Programmable Gate Array (FPGA) computes the three dimensional point cloud using highly parallelizable algorithms, reducing the computational load placed on our computer systems and increasing both resolution and refresh rate. The camera is mounted such that three-dimensional features can be detected between 20 centimeters and 20 meters from the front caster, sufficient for obstacle and lane detection at competition speeds [2].

Besides incorporating advanced stereo vision systems, *Phobetor* also boasts additional 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, and an OceanServer OS500-US digital compass for accurate heading information. Any limitations on the compass' preciseness at high speeds are corrected by GPS and encoder data, as described in Section 4.3. In order to provide yaw rate data without depending on differential wheel speeds, which would introduce error due to slippage, a Gladiator G50 MEMS gyroscope is mounted to *Phobetor's* midsection.

3.4 Electronics and Computer Hardware

By utilizing a LabJack UE9, *Phobetor's* computer is able to access all low-level electronic sensors and outputs through a single interface, greatly simplifying design and limiting the amount of custom circuitry. The LabJack provides an array of features including pulse width modulation (PWM) to control the Victor 885 motor controllers. Using on-board 4x quadrature counting, position and speed are measured using wheel encoders on the drive wheels; a 14-bit analog input interfaces with the Gladiator G50 gyroscope.

For safety, *Phobetor* can be instantly stopped by a large red button on the center of the tower, or by a handheld wireless transmitter. A pair of mechanical relays in series comprise a reliable control switching

and emergency stop circuit. The first relay selects between PWM signals from the computer and the radio controller. The 2.4 GHz spread-spectrum wireless radio system allows manual control of the motors with minimal noise and interference. The second relay cuts the signal to the motor when the E-Stop system is activated. These relays isolate and protect the control circuits while providing a reliable emergency stop mechanism at a low level.

Like its predecessor *Argos*, *Phobetor* operates with one on-board computer, offloading stereo image processing to Videre's on-board camera FPGA (see Section 3.3). To minimize computer volume, *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.

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 5.

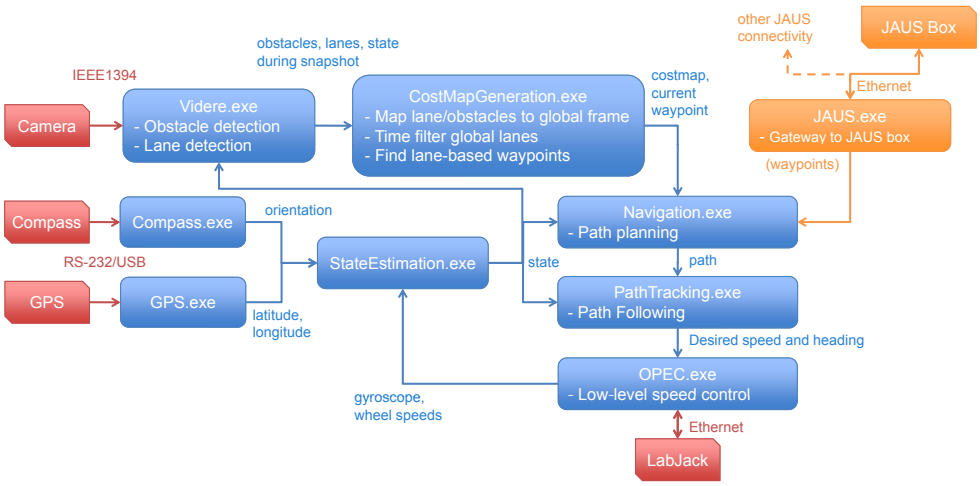


Figure 5: Diagram of Software Architecture

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 [12], 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 uses a state-of-the-art square root central difference Kalman filter (SRCDKF) [13] 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 utilizing a deterministic set of points to represent a multivariate Gaussian distribution over possible states and measurements. Parameters for Gaussian noise variables in our 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 was developed for *Kratos* [6] and updated to include gyroscope measurements for *Argos* [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

Using the point cloud provided by the Videre camera (described in Section 3.3), we detect obstacles using an algorithm developed by Manduchi et. al. [11], in which we search for pairs of points that are approximately vertical to each other [6]. While this performed sufficiently for *Argos* and *Kratos*, *Phobetor* implements further optimizations from Andersen et. al. [3], in which the point cloud (compensated for robot pitch and roll) is mapped into 1cm-square "bins" in the local ground plane. These improvements allow us to increase the resolution of input images without dropping frames.

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 [14], 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 6. Response time is less than one second, which has proven sufficient in the past two competitions.

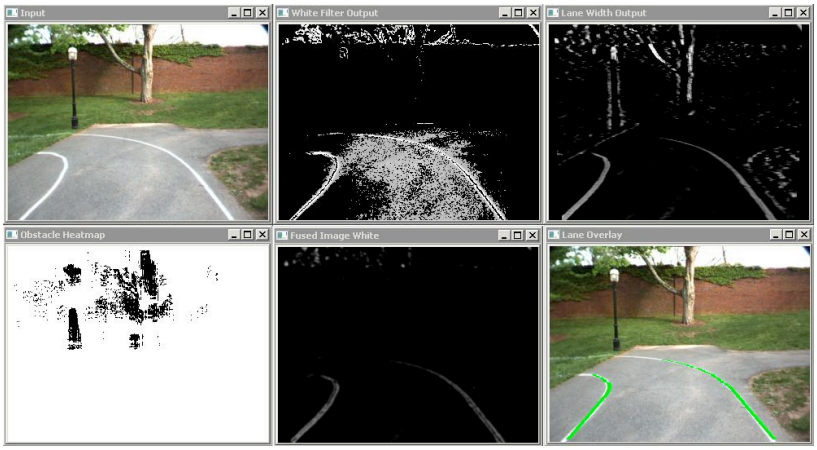


Figure 6: Stages of Lane Detection [6]

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 [15]. 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

Last year, *Argos* used a crosstrack error navigation law to follow paths, sequences of points $\vec{r}(n)$, generated by navigation algorithms [8]. This algorithm minimizes the crosstrack error, $e(t)$, shown in Figure 7.

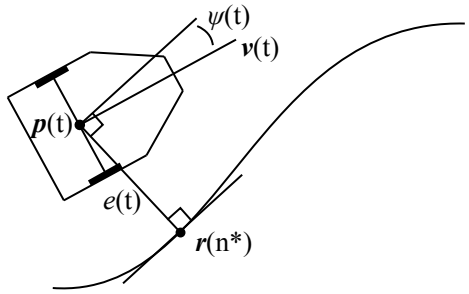


Figure 7: Crosstrack Error Law

The error $e(t)$ 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.

This controller works well for following smooth trajectories that the robot can easily follow. However, 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. To generate smooth trajectories that follow our planned paths, *Phobetor* will use B-spline paths, which are piecewise polynomial curves that connect a given set of points. We globally limit the degree of component polynomials and require the derivatives of the polynomials to be equal where adjacent segments meet, so the entire path is smooth.

4.7 Speed Control

Phobetor and *Argos* have similar size and weight and use the same motors, so the speed control system we used last year [1] is highly applicable to *Phobetor*. The control system must accept a desired speed $v_d(t)$ in m/s and a signed error $e(t)$ between desired and actual speed, also in m/s. The controller's output is motor voltage $u(t)$. The controller implementation is based on a proportional-integral control with a feed-forward term, and is 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 8 shows a block diagram of the controller from a Simulink model.

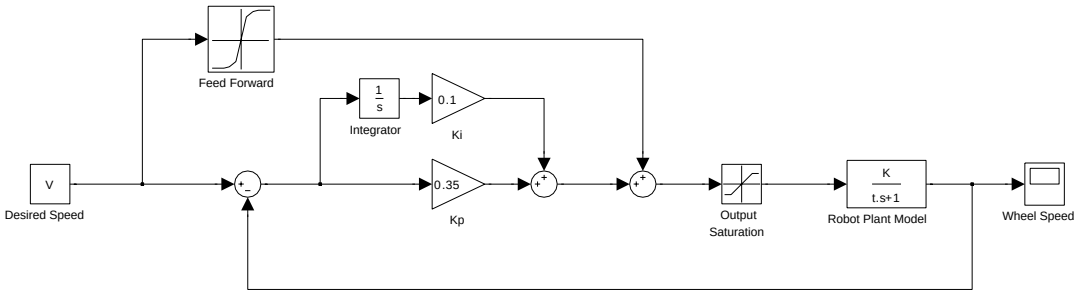


Figure 8: Block Diagram of Speed Controller

The proportional and integral terms are familiar from standard PID controllers, where the integral term eliminates steady-state error. The rotational speed of the motor is not a linear function of voltage, though, so we need a feed-forward term based on a model of the motor to compensate for the non-linearity and allow the PI controller to work well.

Specifically, from data collected on these motors last year we fit a function to map desired speeds to motor voltages:

$$V(y) = -\tau \ln\left(1 - \frac{y}{K}\right),$$

where y is the speed in m/s, V is the motor voltage, and K and τ are fit parameters. By adding the feed-forward term given by $V(y)$ to the control law, we linearize the control problem so the plant can be controlled by PID. Note that the motor's response is different when rotating in forward and reverse directions, and since the motors are mounted with mirrored orientation the feed-forward terms for *Phobetor's* left and right motors are different.

Lastly, we made additional 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, Tony H Zhu, and Applied Science. Argos: Princeton University's Entry in the 2009 Intelligent Ground Vehicle Competition. In *Intelligent Robots and Computer Vision XXVII*, volume 2, 2010.

[3] Hans Jø rgen Andersen, T. Bak, Kristian Kirk, T. L. Dideriksen, C. Madsen, and M. B. Holte. Obstacle detection by stereo vision, introducing the PQ method. In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics, ICINCO 2005*, Barcelona, Spain, 2005.

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

[13] 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.

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

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

Special Thanks

The 2010 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.