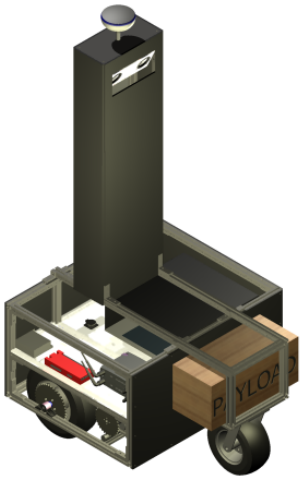


Argos: Princeton University's Entry in the 2009 Intelligent Ground Vehicle Competition

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

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

July 20, 2009

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.

PRINCETON
School of Engineering and Applied Science

PAVE PRINCETON AUTONOMOUS VEHICLE ENGINEERING

Contents

1 Team Overview

The Princeton 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 [1], the 2007 DARPA Urban Challenge [9] and the 2008 Intelligent Ground Vehicle Competition (IGVC) [2]. Our team placed third overall and won rookie-of-the-year in the 2008 IGVC, placing 1st, 4th and 6th in the Design, Navigation and Autonomous challenges, respectively. *Argos*, our entry in the 2009 IGVC, is an all-new robot based on improvements from 2008. We believe that *Argos* and the Princeton Team will once again be competitive in the 2009 IGVC.

Our team is made up of undergraduate students from several engineering and non-engineering departments[1]. We maintained a simple 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 two student team leaders to specialize in project management and logistics.

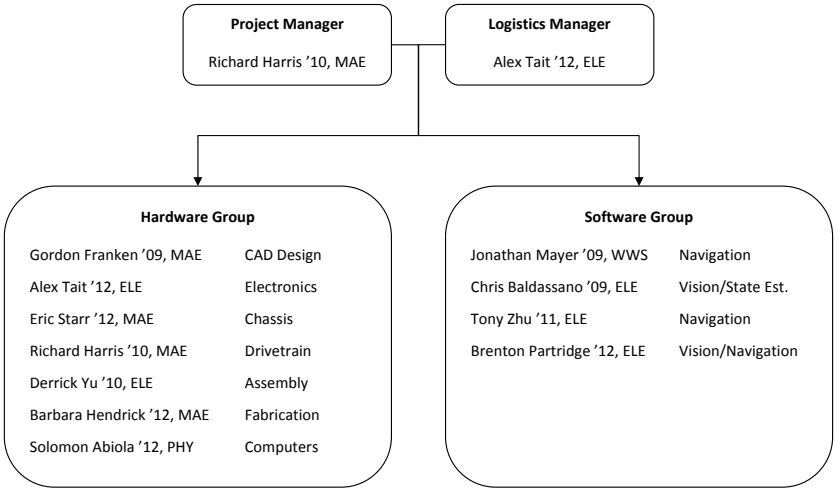


Figure 1: Team Organization Diagram

2 Design Process

Our primary goal in designing our robot was to improve on the largely successful design of our 2008 IGVC entry, *Kratos*. Our overall design process followed the steps outlined in Figure 2. In the initial phase, we analyzed the requirements for the project along with the strengths and shortcomings from last year's design to create a set of desired design objectives. Given these design objectives, we determined a feature set for *Argos* based on our knowledge and expertise. These systems were then modeled or simulated using computer software tools. To minimize wasteful spending and fruitless development time, the required components were

[1]ELE = Electrical Engineering, MAE = Mechanical and Aerospace Engineering, PHY = Physics, WWS = Woodrow Wilson School of Public Policy and International Affairs

acquired only after successful computer simulations. Each system was tested individually, then integrated and tested as a whole. If a system's actual performance did not adequately meet its design objective, redesign was necessary. Systems that met their objectives were deployed, though their performance is continuously monitored.

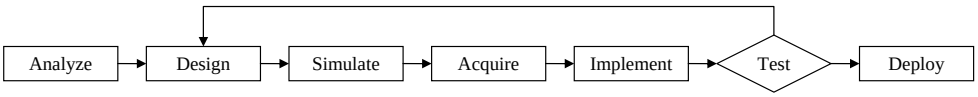


Figure 2: Flowchart of Design Process

The rules and requirements for the 2009 IGVC are very similar to those of the 2008 competition, so most of our overall design objectives remain the same as enumerated in last year's tech paper [2]. In analyzing our performance during 2008, we identified the following elements of our system that needed improvement:

Hardware

1. Drivetrain was underpowered and noisy
2. Wheels had tendency to slip on the grass
3. Physical layout of the robot had large moment of inertia
4. Robot was not water-resistant and offered little protection for internal components
5. Overheating inside the robot led to component failure

Software

1. Camera field of view was insufficient for lane and obstacle detection
2. Wheel slip contributed to significant errors in heading estimate
3. Path planning failed when lanes were not sufficiently detected
4. Path planner generated non-optimal/ unnecessarily tight paths around obstacles
5. Path following algorithm cut corners, causing the robot to hit objects

This technical paper divides the discussion of our robot design into two parts: Hardware (Section 3) and Software (Section 4). In each section we discuss the improvements we made to address the shortcomings of our 2008 entry. *Argos* combines the features that worked well in 2008 with improvements made to our previous design, resulting in a robotic platform that outperforms its predecessor in every respect.

3 Hardware

To address the shortcomings in the mechanical design of our 2008 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 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 *Argos* and their costs is shown in Table 1.

3.1 Mechanical Design

Argos has a width of 31", a length of 37", a height of 69.5", and a weight of 280 pounds without payload. It is based on a three wheel chassis with two drive wheels and a leading caster. This setup keeps both drive wheels in contact with the ground on sloped or uneven terrain, does not cause the drive wheels to slip when turning, and is always stable. The two fixed, powered wheels provide a zero turning radius enabling

Item	Actual Cost	Team Cost
Gladiator Technologies G50 gyroscope*	$1,000	$0
HemisphereGPS A100 GPS receiver	$1,500	$0
Honeywell HMR3000 digital compass	$850	$0
US Digital E6 rotary encoder (2x)	$500	$0
Videre Design STOC-15 stereo camera	$1,660	$1,660
NPC T64 motor (2x)	$256	$256
IFI Robotics Victor 885 motor controller (2x)	$440	$440
Drivetrain components	$525	$525
Raw material for chassis	$1,388	$1,388
Miscellaneous hardware	$340	$340
Labjack UE9 data acquisition card	$500	$0
Computer components	$1,082	$1,082
Tempest TD-22 battery (6x)	$388	$388
IOTA DLS-27-25 battery charger	$295	$295
Samlex SDC-08 DC-DC Converter	$75	$75
Linksys WRT54G wireless router (2x)	$75	$0
R/C radio system	$350	$350
Total:	**$11,224**	**$6,799**

*Denotes donated item. All other zero-cost items were reused from *Kratos* in 2008.

Table 1: List of Costs for Building *Argos*

maximum mobility and holonomic path planning. The drive wheels have a diameter of 12"; along with the 10" caster, *Argos* has roughly 4" of ground clearance. On top of the chassis is the sensor tower, which holds various sensors as well as the required emergency-stop button. A region above the caster was designed specifically to hold and secure the payload. In keeping with our design process, the entire robot was modeled with Autodesk Inventor 2008 CAD software prior to any construction. Inventor allows for rapid component modeling and assembly. It also tracks mass data for each component so the overall weight, center of mass and moments of inertia of the robot were known before fabrication. As can be seen from Figure 3, the final robot bears close resemblance to the CAD model.

(a) CAD Model **(b)** Actual Robot

Figure 3: Visualizations of *Argos*

3.1.1 Drivetrain

Last year's drivetrain was underpowered, so *Kratos* accelerated slowly and could not reach the maximum allowed speed. We made a decision to switch to a 24-volt electrical system to accommodate a new set of motors that could deliver more power at lower electrical currents. Following our design process from last

year [2], we started by estimating the mechanical power requirements of three driving scenarios: full speed forward on grass, turning in place on grass, and driving up a 10° incline. We selected a pair of NPC T64 motors rated at 1.6 horsepower each, which is more than enough to meet our needs. The T64's have the following specifications:

$$\begin{aligned} V_n &= 24 \text{ V} \quad \textit{(nominal voltage)} \\ I_s &= 110 \text{ A} \quad \textit{(stall current)} \\ \tau_s &= 93.21 \text{ N-m} \quad \textit{(stall torque)} \\ \omega_f &= 25.475 \text{ rad/s} \quad \textit{(no-load speed)} \\ I_f &= 4.707 \text{ A} \quad \textit{(no-load current)} \end{aligned}$$

Using these parameters, we ran numerical simulations using the standard DC motor equations to calculate speed, power, and efficiency as functions of output torque. Based on the calculated angular speeds at which the motors achieve peak efficiency and our desired speed of 5 mph, we determined we needed a gear ratio of 3:4 for the sprocket and chain drive connecting the motors and drive wheels. To alleviate the problem of our wheels slipping on grass we optimized the layout of components (discussed below in Section 3.1.2) and upgraded the tires to ones with knobby treads that would better grip the thick grass. After fabrication, we applied an open loop step function to the motors and verified that their experimental response agreed with our simulated predictions (Figure 4).

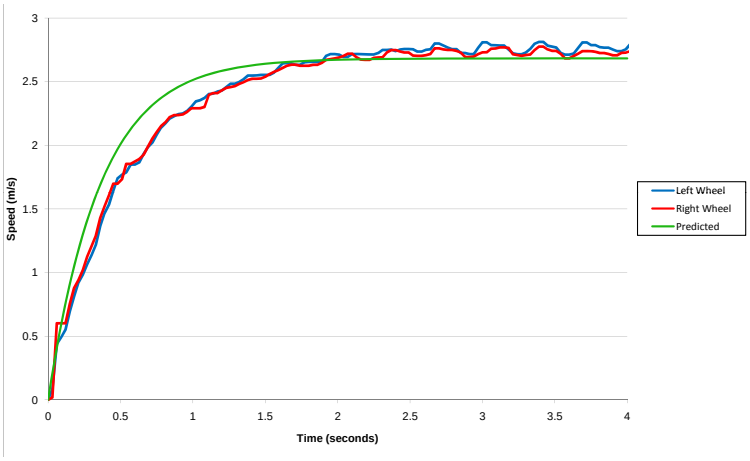


Figure 4: Open Loop Step Response

3.1.2 Chassis

The chassis consists almost entirely of 80-20 extruded aluminum bars and 1/4" clear polycarbonate sheets. Though we recycled much of the raw material for the frame from *Kratos*, the entire chassis itself was completely redesigned to address the design flaws in the 2008 robot. In an effort to reduce the footprint of the robot to make it more compact and maneuverable, we included two levels in the chassis design. This bilevel frame allows more components to be located closer to the drive wheels, allowing *Argos* to have a lower moment of inertia. Furthermore, we increased the height of the sensor tower, providing the camera a better perspective to detect lanes. All of our sensors, discussed in Section 3.3, are vertically aligned over the drive wheel axes so that they share the same reference frame on the robot. *Argos'* frame contains only square vertices, which are easily sealed with exterior panels. To protect the internal components from the elements the joints on the outer panels are sealed. However, to prevent overheating issues encountered at last years' competition, *Argos* is equipped with ventilation ports and exhaust fans.

3.2 Electrical Design

To support *Argos'* new drive motors, we designed a 24-volt electrical system to efficiently provide adequate power and enough runtime for extended testing. Though the robot is only required to operate for 6 minutes during the competition, we decided to store enough energy to operate for at least 45 minutes. A summary of power usage is shown in Table 2. Factoring in the nominal power draw, desired running time, and battery inefficiencies at high current draw, we determined that we needed a battery bank with a capacity of 132 amp-hours. Few 24-volt batteries are available with sufficient capacity, so our design uses two 12-volt battery banks in series to generate a 24-volt source.

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 2: Power Requirements for *Argos* (all power listed in Watts)

Our initial electrical design aimed to reuse the three 12-volt batteries and 12-volt battery charger from 2008. We designed a switching circuit employing two high-current relays that would put all three of our batteries in parallel for charging, and switch one of the batteries into series for 24-volt operation (Figure 5). 12-volt electronics were always run off of the center terminal, even when charging.

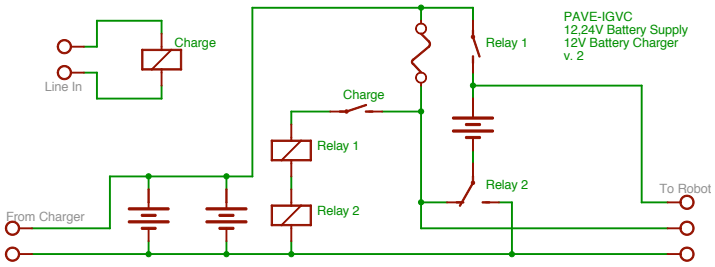


Figure 5: Switching Power Circuit

During early testing, we observed that *Argos* was experiencing sporadic power failures. We determined that the problem originated from uneven discharging of the batteries while connected in series. When switched into parallel, even a 1 volt difference in battery voltages caused excessive current to flow through the system, ultimately damaging our charger. The electrical system was redesigned so that no switching is needed. *Argos* now has six 12-volt batteries in a series-parallel configuration and one 24-volt charger. To ensure that all 12-volt banks charge evenly, the center terminal is equilibrated at exactly half of total voltage by a resistive voltage divider. This divider maintains the long-term health of our battery system and contributes

negligible parasitic power loss. An 8-amp DC-DC converter was added to provide 12-volt power for most of the components; a 5-volt, 5-amp DC-DC converter provides regulated power for the electronics. Figure 6 shows the final power storage and distribution setup of *Argos'* electrical system.

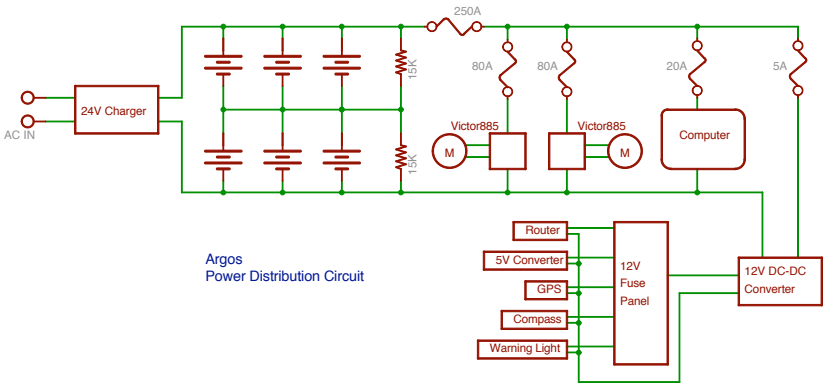


Figure 6: Electrical System

3.3 Sensors

Argos uses a single environmental sensor, a Videre stereo-on-chip (STOC) color camera with a baseline of 15 cm. Experience with *Kratos* informed our decision to use a relatively inexpensive stereo optical system. Its ability to detect both obstacles and lanes avoids error-prone frame of reference transformations for multiple sensors. The Videre camera improves upon Point Grey's Bumblebee2 (used on *Kratos*) with an embedded Field Programmable Gate Array (FPGA) that performs the highly parallelizable algorithms necessary to compute a three-dimensional point cloud. In addition to significantly decreasing general computing requirements, STOC raises limits on resolution and refresh rate, improving the effective detection range and reaction time of the robot, respectively. Although processing requires additional data be transferred over the camera's IEEE1394 interface, the query for a stereo map, computation on the FPGA, and response occurs on the order of one hundredth of a second. The camera's wide baseline allows for an even greater detection range, and 2.5 mm lenses widen the field of view, enabling *Argos* to maneuver more accurately around obstacles.

To sense *Argos'* state we employ an array of absolute and differential sensors. *Argos* is equipped with the same HemisphereGPS receiver, US Digital wheel encoders, and Honeywell digital compass used by its predecessor and described in last year's technical paper [2]. The combination of GPS and wheel encoder data provide high-precision, low-drift position and speed data. The compass provides noisy heading data; at high speed it is corrected by GPS data. In last year's competition we computed the yaw rate of the robot using differential wheel speeds; we found this approach produced significant error under wheel slip conditions, however. To correct for slippage, *Argos* incorporates a Gladiator G50 MEMS gyroscope to directly measure the instantaneous yaw rate. The gyroscope has a range of $\pm 350°/s$, and outputs an analog voltage.

3.4 Electronics and Computer Hardware

In the interest of developing a simple and robust system, we aimed to minimize the use of custom electronics and circuitry. A LabJack UE9 provides *Argos'* computer systems with a single interface to all low-level electronic sensors and outputs; speed and position of the drive wheels are measured from the wheel encoders

using on-board 4x quadrature counting, and the Gladiator G50 gyroscope is connected via 14-bit analog inputs. The UE9 also produces pulse width modulation (PWM) signals to control the Victor 885 motor controllers, while digital I/O lines control the warning siren and read in run/pause/disable commands from the external emergency stop (E-Stop) radio.

The only custom circuitry on *Argos* (not including power distribution) are the electronics in the emergency stop (E-Stop) system, which assure safety of operation through parallel manual and wireless pathways. Pushing a red button mounted on the back of the tower or flicking a switch on the independent, handheld wireless transmitter will activate the E-Stop system. When the E-Stop is activated the control signals to both motors are disconnected, causing the robot to come to an immediate stop. A new 2.4 GHz spread-spectrum wireless radio generates significantly less noise and encounters less interference than the 75 MHz analog hobby radio used on *Kratos*, and can be directly connected to the motor controllers to allow human operability even with computer failure.

By offloading a portion of the stereo image processing to the Videre camera's dedicated FPGA, *Argos* is able to operate with a single on-board computer, resulting in significant weight, cost, and energy savings over *Kratos*. We built a new quad-core computer with an Intel Core i7 CPU at 2.66 GHz, 6 GB of RAM, and an 80 GB hard drive. We are able to capture the performance benefits of additional processor cores through our multiple-process software architecture, discussed in Section 4.1. The computer draws DC power directly from the 24-volt system with a 450W ATX power supply, eliminating the need for an AC inverter and thereby saving weight and increasing energy efficiency. Shock-isolating feet beneath the computer enclosure prevent damage to the sensitive components within. A pair of 802.11g wireless routers simultaneously allow clients to access *Argos'* systems wirelessly and, when in range of an accessible wireless network, provide Internet access to the onboard computer and clients. This combination also allows *Argos'* computer to be remotely accessed over the Internet whenever in range of a wireless network (including the majority of the Princeton campus), and supports interfacing with JAUS systems via Ethernet or WiFi.

4 Software

Argos' software employs the same paradigm as *Kratos*, consisting of independent asynchronous processes linked through our publish/subscribe messaging framework, IPC++. Individual software modules implement JAUS compliance (Section 4.1), provide sensor input and processing (Section 4.3), estimate pose (Section 4.2), plan a desired path (Section 4.4), and accomplish path following and speed control (Sections 4.5 and 4.6). A holistic diagram displaying module roles and message contents is displayed in Figure 7. Substantial innovations were made in all software aspects, particularly in our approach to navigation.

4.1 Computing Platform

Argos' 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 as opposed to Server 2003's 10-15 ms accuracy and 2-3 ms jitter. Driver incompatibilities require a 32-bit version of the operating system, but our software is implemented to take advantage of 64-bit processing in the future. All software is written in platform-independent C++ using the Visual Studio IDE

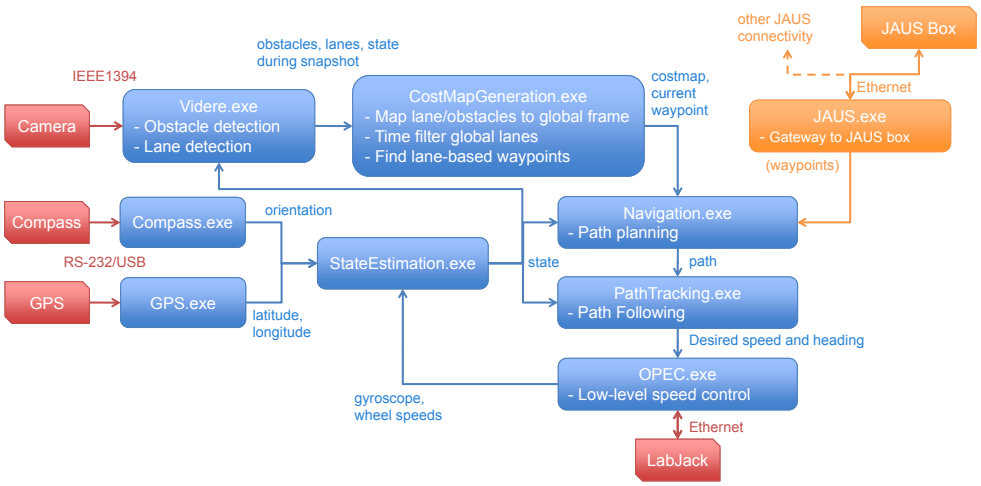


Figure 7: Diagram of Software Architecture

for ease of development. The Newmat and wykobi libraries provide linear algebra and geometry functionality respectively, and the Qt library and Designer allow for rapid multi-platform GUI development.

We continue to employ IPC++, an object-oriented wrapper of Carnegie Mellon's Inter-Process Communication (IPC) platform [10], as our underlying robotics framework. Each software component runs as an independent program connected to a central server; message publishing and subscription, serialization, and timing are all accomplished transparently.

Implementing JAUS interoperability is straightforward with IPC++; an additional process translates between external JAUS messaging and internal IPC++ messaging. For JAUS query messages, the process replies with the latest update of the robot's state. For JAUS control messages, such as start/stop and set waypoints, the translator simply sends the appropriate equivalent message over IPC++.

4.2 State Estimation

Argos' state estimation module uses a square root central difference Kalman filter (SRCDKF) [11] to fuse all of the state sensor (compass, GPS, wheel encoders, gyroscope) data and maintain an optimal estimate of several key variables that define the state of the robot. The SRCDKF, also used last year, is a sigma point filter that uses a deterministic set of points to represent a multivariate Gaussian distribution over states and measurements. This year's major modification is the addition of an explicit yaw rate estimate, ω, which is used during the state update step of the filter to help predict the vehicle's heading. Last year, we simply estimated this term as

$$\omega(t) = \frac{v_l(t) - v_r(t)}{b},$$

where $v_l(t)$ is the left wheel speed in m/s, $v_r(t)$ is the right wheel speed in m/s, and b is the wheel base in meters. This estimate was often poor, however, since the wheels would slip frequently on the field, especially when turning.

We appended our previous state vector to account for gyroscope data, and arrived at a seven variable state vector that explicitly includes the yaw rate:

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

where x is the vehicle x coordinate in meters in a Cartesian local frame, y is the vehicle y coordinate in meters, $\theta \in [0, 2\pi)$ is heading, $\delta \in [0, 2\pi)$ is the bias between magnetic compass heading and true GPS heading, ω is the yaw rate of the vehicle, v_r and v_l are the right and left wheel ground speeds in m/s. The measurement model for the gyroscope, used in the prediction step of the filter, is given by

$$\mathbf{y}_{\text{gyroscope}}(t) = \begin{pmatrix} \omega(t) + \gamma \end{pmatrix},$$

where γ is a Gaussian random variable corrupting the gyroscope measurement.

The Gaussian noise variables were estimated by analyzing the long-term behavior of the sensors' signals while at rest. In all cases except the wheel encoders, the Gaussian random variable is an accurate representation of the sensor noise; for the encoders it serves as an approximation to the finite, discrete noise corrupting the digital pulse train. The addition of the gyroscope into state estimation is intended to guard against severe errors in heading estimate resulting from wheel slip that we observed during 2008. We anticipate the improved state estimation will demonstrate accuracy and robustness to unreliable data better than in the previous year.

4.3 Vision

Aside from slight modifications to accommodate the Videre stereo camera and its STOC capabilities, our obstacle detection algorithm (which performed flawlessly in 2008) is unchanged: we examine a point cloud for protrusions from the ground plane, and report them as obstacles [3].

Our lane detection system, on the other hand, inconsistently detected markings at last year's competition. We kept the basic algorithm structure, originally developed for the DARPA Urban Challenge, in which filtered images representing white content, yellow content, pulse width, and stereo obstacles are fused into a heat map which is then searched for lines [3]. However, both the filter and fusion steps have been updated.

First, to counteract sensitivity to shadows and ambient light conditions, new white and yellow filters were developed that operate in hue-saturation-value space [12]. The new white filter calculates separate brightness thresholds for high and low saturation image components to assure shadowed markings will still be detected and to avoid false positives in bright conditions. The new yellow filter, meanwhile, thresholds the image relative to its maximum yellow hue to similarly provide robustness against lighting conditions.

Second, we implemented a localist perceptron neural network to replace the "binary and" fusion step previously employed [6]. The network architecture consists of a single pixel output unit, with sigmoid activation, connected to a 5x5 input grid of local pixels from the filtered images. We trained the network for 10,000 iterations on randomly selected pixels from logged images with the delta rule, then compared performance to "binary and" fusion on a test set of images. While daytime performance was relatively unchanged, we found an over 30% increase in detection rate for low-light images. *Argos'* consequent ability to operate in the early morning and late in the day increases its window of operational effectiveness and provides additional time for testing.

4.4 Navigation

Argos maintains the three-pronged approach to navigation employed by its predecessor: the robot's environment is represented by a cost map incorporating obstacle and lane data, a desired waypoint is selected, and a path planner calculates an optimal trajectory for achieving the waypoint.

4.4.1 Cost Map Generation

The goal of cost map generation is to synthesize noisy state, obstacle, and lane marking sensory inputs into a reliable, high resolution two-dimensional grid of the world. We retain the basic architecture from last year by assigning continuous costs, calculated with a moving average, to a global map of 10 cm x 10 cm grid cells [3]. As did its predecessor, with every vision update *Argos* generates a local cost map of obstacles and lanes immediately in front of the robot, then merges this local map into the global map. A "footprint" cost map is employed for path planning, in which each cell evaluates to the sum costs in its surrounding region, thereby accounting for the physical geometry of the robot. To compensate for detection error on lane markings we are experimenting with filtering through time, which provides hysteresis for rejecting outlying false-positives and interpolation to fill in false-negatives.

4.4.2 Waypoint Selection

As with *Kratos*, waypoints for the navigation challenge are presorted into the shortest overall path by enumerating all possible permutations. In the autonomous challenge, where the course structure is unknown, the desired waypoint is generated dynamically; a target point is inferred by finding the midpoint of the two furthest lane markings seen. In both cases, a new waypoint is only calculated when the robot approaches the current waypoint.

4.4.3 Path Planning

Argos significantly improves upon *Kratos'* simple A* path planner by incorporating incremental replanning and continuous headings from cell to cell. We accomplish the former by replacing A* with the Dynamic A* Lite algorithm (D* Lite), which retains the optimality and completeness guaranteed by A* but leverages knowledge derived from prior search iterations to avoid searching when unnecessary [8]. By planning backwards from the waypoint, search results remain relevant even as the robot moves. In our experience the move to D* Lite provides a speed-up of over an order of magnitude relative to repeated A* searches.

Our second innovation is the implementation of the Field D* algorithm to formulate the path search problem [5]. Field search is premised on the recognition that restricting the search to cost cell centroids discounts feasible robot positions and constrains path heading changes to 45° increments, thereby disallowing preferred paths that do not neatly move from centroid to centroid (including most straight-line paths). Moreover, these artificial constraints can result in an arbitration crisis between equal cost paths, leading to erratic robot motion. We implemented a rudimentary solution to these problems in our previous planner with a tie-breaking "path straightness" heuristic to select paths that, when naturally interpolated by robot motion, would approach optimality, but in the presence of obstacles paths quickly degenerated. Field search provides a more flexible solution to these problems by treating the search nodes samples from a continuous cost space, and applying linear interpolation to approximate an optimal solution with arbitrary headings and positions.

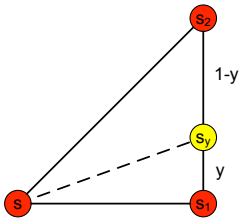


Figure 8: A Trivial Example of Field Search

When determining the depth g of node s, field search takes the minimum of a path to s from each of its neighbors s' directly as usual, but also considers a path to s from any point on the boundary of a neighboring cell (between connected neighboring nodes). Figure 8 shows a canonical example of expanding node s with neighbors $(s_1, s_2) \in \{connected\, neighbors\, of\, s\}$. Traditional search only considers paths directly from neighbors, such that

$$\mathrm{g}(s) = \min\left[\mathrm{g}(s_1) + \mathrm{c}(s, s_1), \mathrm{g}(s_2) + \mathrm{c}(s, s_2)\right],$$

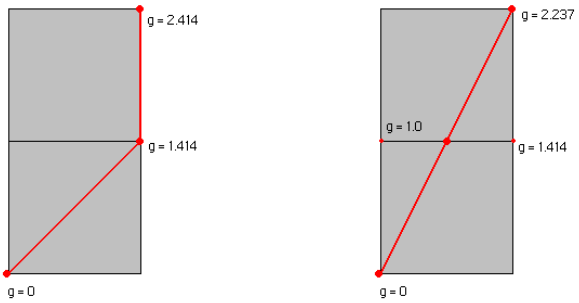
where $\mathrm{c}(x, y)$ is the cost of traversing from node y to node x. Field search expands upon this formulation by allowing for a continuous range of predecessors between s_1 and s_2,

$$\mathrm{g}(s) = \min_{0 \le y \le L}\left[\mathrm{g}(s_y) + \mathrm{c}(s, s_y)\right],$$

in which the depth of novel predecessors s_y are linearly interpolated according to

$$\mathrm{g}(s_y) = \frac{y}{L}\mathrm{g}(s_1) + \left(1 - \frac{y}{L}\right)\mathrm{g}(s_2)$$

where y is the distance from node s_y and L is the length of a cell side. We use Euclidean distance multiplied by the cost of the cell (given by the cost map) as the cost function for admissibility and consistency [4].

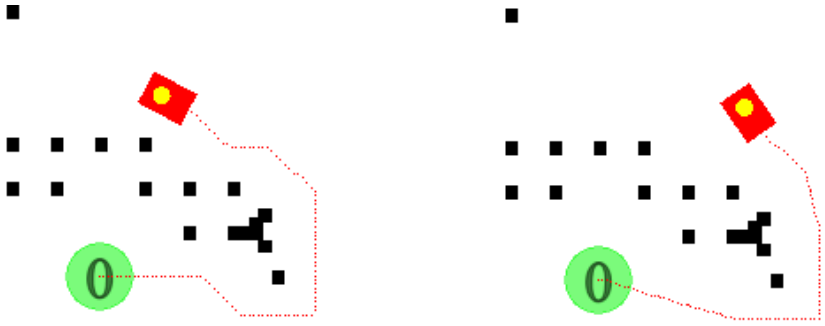


(a) No Interpolation **(b)** Linear Interpolation

Figure 9: Example Determination of Node Depth

Figure 9 gives a full-scale example of field search. Searching from the bottom-left to the top-right, without interpolation the middle-right and top-right nodes are expanded and concatenated to give a highly undesirable path. Field search, on the other hand, expands the middle-right and middle-left nodes in the regular manner, but when expanding the top-right will interpolate between the middle-right and middle-left nodes for a shorter path to the source. Paths are extracted through gradient descent backwards in the space of interpolated search nodes, giving the near-optimal result shown in Figure 9b.

Field search comes at the cost of further node expansion (in the trivial example above, an extra node was required to allow interpolation), which offsets the speed improvements from our D* Lite implementation. However, with optimizations such as a binary heap, cached costs, and compiling with the SSE3 instruction set extensions, our Field D* planner re-plans at roughly 10 Hz the same speed as its predecessor A* planner, which we found sufficient for autonomous operation. Figure 10 shows a simulated comparison of the two planners.



(a) A*, Headings Constrained to 45°

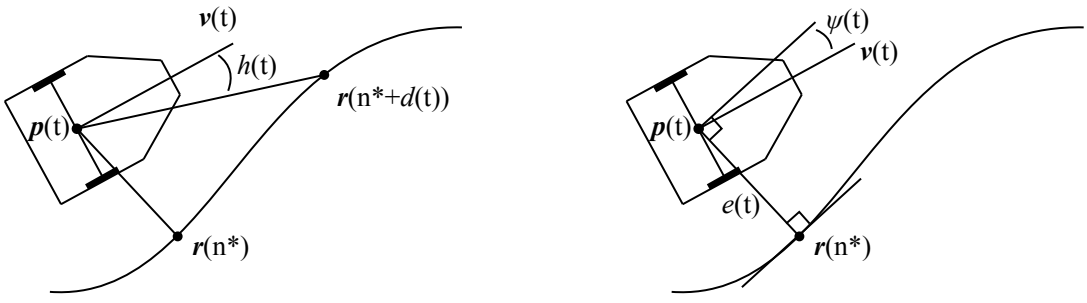
(b) Field D*, Continuous Path Headings

Figure 10: Quality of Planned Paths

As a final step, the planner computes a desired speed profile along the path based on the maximum allowed acceleration, a weighted average of nearby costs (to ensure slow, precise movement around obstacles), and path curvature.

4.5 Path Following

Paths are specified as a desired series of positions, $\vec{r}(n)$. Last year's technical paper describes two path following schemes based on the proportional and crosstrack navigation laws [2]. *Kratos* only used a proportional navigation law, which sets the desired yaw rate proportional to the difference between the current heading and the angle to a lookahead point (Figure 11a). This scheme is easier to integrate with our actuation model, but its tendency to cut corners caused *Kratos* to collide with several obstacles during the 2008 IGVC. *Argos* eliminates this shortcoming by successfully implementing a crosstrack error navigation law [7].



(a) Proportional Navigation Law

(b) Crosstrack Error Law

Figure 11: Path Following Schemes

Our path tracking controller minimizes the crosstrack error $e(t)$, defined as the signed distance from the closest point on the path to the point $\vec{p}(t)$ between the robot's wheels (Figure 11b),

$$|e(t)| = \min_n \|\vec{r}(n) - \vec{p}(t)\|.$$

In a point model of the robot, the crosstrack error evolves according to

$$\dot{e}(t) = v(t)\sin(\psi(t)),$$

where $v(t)$ is the robot speed and $\psi(t)$ is its heading relative to the trajectory. The heading control law is

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

where $\omega_d(t)$ is the desired yaw rate for the robot, and k_h and k_e and tunable constants. Note that when the crosstrack error is zero, this controller degrades to a pure-pursuit controller in which the desired heading is set to instantaneous path heading. The inclusion of the crosstrack error term explicitly avoids the understeer problem we observed with the proportional navigation controller, so we expect significantly better path following with *Argos*.

4.6 Speed Control

We took a theoretical approach towards obtaining optimal control of the robot's wheel speeds. Our goal was to be able to track a step input with zero steady state error, low overshoot, and minimal sensitivity to external disturbances such as slopes and other varying outdoor terrain, these being the most desirable control behaviors for this competition. With these parameters in mind, we developed a feed-forward proportional integral design with the control equation

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

where $u(t)$ is the controller output to the motor in volts, $f(v_d(t))$ is a nonlinear feed-forward function of the desired speed $v_d(t)$, $e(t)$ is the signed error between desired and actual speed in m/s, and k_p and k_i are tunable control constants. Figure 12 shows a block diagram of this controller. The controller constants k_p and k_i are kept small enough so that high frequency noise is not amplified. Their values were manually tuned to obtain empirically ideal responses.

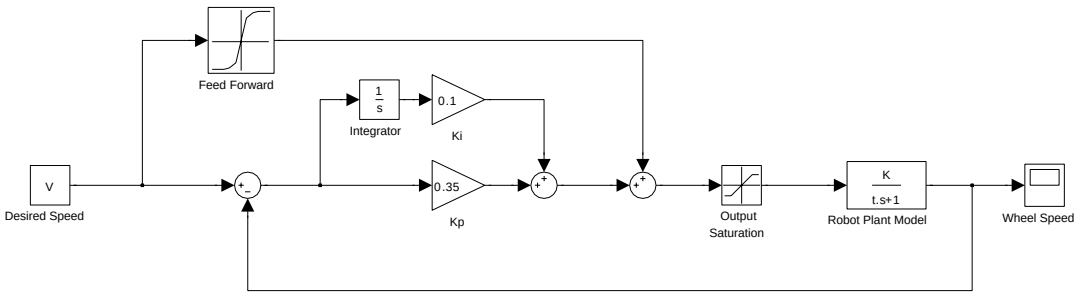


Figure 12: Block Diagram of Speed Controller

The feed-forward branch will produce zero steady-state error given a perfect plant model and zero external disturbances. The proportional term then compensates for the majority of the error that results from not actual operation diverging from these unrealistic idealizations. Finally, the integral term guarantees zero steady-state error. To implement this control scheme, we first had to obtain an open-loop plant model of *Argos*. We fit a mathematical model to sampled data and found that the plant model approximates the exponential function

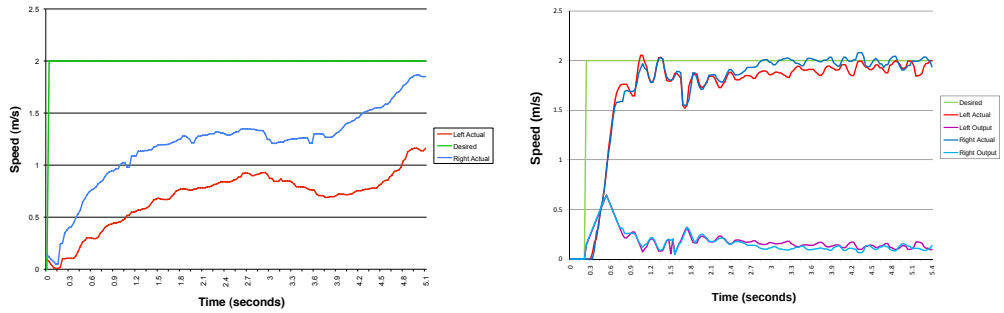
$$y(V) = K\left(1 - e^{-\frac{V}{\tau}}\right),$$

where y is the speed in m/s, V is the motor voltage, and K and τ are tunable parameters set by the computerized fit. The inverse of this function is

$$V(y) = -\tau * \ln(1 - y/K),$$

which maps desired speed y to motor voltage V. This became the feed-forward function used in our controller. Though PID control is invalid for a nonlinear plant, the inclusion of the feed-forward term compensates for nonlinearities, resulting in a close realization of a linear system for which PID control is applicable.

We noticed that the forward and reverse responses of the motors were different due to motor winding bias and the opposite orientation of the motors on the chassis. The forward response of the left motor matches the reverse response of the right motor, and vice versa. We made sure to fit the data obtained to two different models, one for the forward case and one for the reverse case, and to code the controller such that each case undergoes the appropriate speed to voltage conversion.



(a) Feed-Forward Only **(b)** Feed-Forward + Proportional

Figure 13: Stages of Speed Control Development

A series of plots in Figures 13 and 14 demonstrate the response of each part of our controller to a step input of 2 m/s. With only the feed-forward branch engaged there is significant error (Figure 13a), indicating that the model of the system is not perfect. The addition of a proportional term (Figure 13b) results in a significant drop in error and decrease in rise time, though there is still room for improvement. Including the integral term (Figure 14) eliminates steady state error while producing no overshoot.

Our controller incorporates several minor alterations to account for nonlinearities of the robot. The motors are powerful enough such that under high acceleration, either the front of the robot may lift off the ground or the wheels may spin in place. To compensate for this, we limit the incremental change in controller output. This effect on the output voltage is clearly seen during the first second of Figure 14. In addition, we prevent

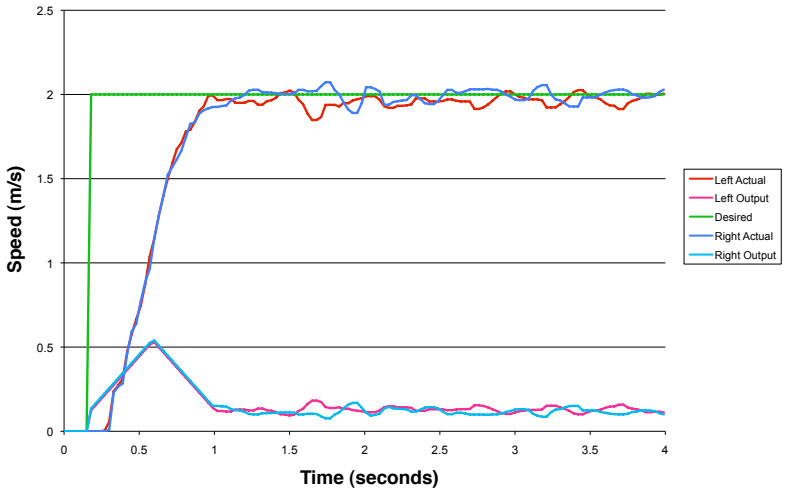


Figure 14: Feed-Forward + Proportion + Integral

the integral term from winding up when the output saturates at battery voltage. We have not observed these modifications adversely affecting the controller performance.

5 Conclusion

Argos is a robust and reliable robotic system that will be competitive at the 2009 Intelligent Ground Vehicle Competition. Its design and implementation, as discussed in this paper, represent significant innovations over our entry in the 2008 IGVC. *Argos* is built on an entirely new hardware platform, including a drivetrain that is more powerful and quieter, and a chassis that is small enough to fit through a standard doorway. Improved navigation and path planning algorithms combine the latest cutting-edge research with the versatility and maneuverability of *Argos*. Early simulations and testing have already shown significant improvements in both software and hardware performance when compared to *Kratos*. We look forward to demonstrating *Argos* at the 2009 IGVC.

References

[1] 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.

[2] 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.

[3] 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. *Proceedings of IS&T/SPIE Electronic Imaging Conference*, 7252, 2009.

[4] Rina Dechter and Judea Pearl. Generalized Best-First Search Strategies and the Optimality of A*. *Journal of Association of Computing Machinery*, 32(3):505–536, February 1985.

[5] David Ferguson and Anthony Stentz. Using Interpolation to Improve Path Planning: The Field D* Algorithm. 23(1):79–101, February 2006.

[6] Gordon H. Franken and Jonathan R. Mayer. A Neural Network for Fusing Lane Marking Detectors. 2009.

[7] 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.

[8] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the Eighteenth National Conference on Artifical Intelligence (AAAI)*, pages 476–483, 2002.

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

[11] 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.

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

Special Thanks

The 2009 IGVC team thanks Professor Schapire for his continued involvement with the project. In addition, we could not have accomplished so much in one academic year without the resources of the School of Engineering and Applied Science, Department of Computer Science, and Department of Mechanical and Aerospace Engineering. We would like to especially thank the Keller Center for Innovation in Engineering Education and the Norman D. Kurtz '58 fund for their gracious support of our project.