Kratos:

Princeton University's Entry in the 2008 Intelligent Ground Vehicle Competition

Christopher Baldassano, David Benjamin, Benjamin Chen,
Gordon Franken, Will Hu, Jonathan Mayer,
Andrew Saxe, Tom Yeung, and Derrick Yu

Faculty Adviser:
Professor Robert Schapire, Department of Computer Science

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 Autonomous Vehicle Engineering (PAVE) IGVC team consists of members of Princeton University's all-undergraduate semi-finalist DARPA Urban Challenge team and brings substantial experience with computer vision and autonomous navigation. Our entry into the 2008 IGVC, Kratos, builds upon the systems in our prior robots, Prospect Eleven [1] and Prospect Twelve [5]. Despite a one semester development cycle, Kratos possesses the complete set of capabilities required to be competitive at the 2008 IGVC.

Our team consists of nine undergraduate students from several engineering and non-engineering departments.[1] Because of our small team size and familiarity we felt that a flat organizational structure, as shown in Figure 1, would be most effective. Team members were grouped by their area of expertise as hardware or software. The hardware group is responsible for all physical aspects of the robot, including design, fabrication, sensor selection, electronics, electrical wiring and computers. The design and implementation of these systems is discussed in Section 3. The software group is responsible for algorithmic design and programming implementation. Primary tasks include sensor processing, intelligent navigation schemes and robust feedback control systems. The software aspects of Kratos are discussed in Section 4. To oversee these groups, one student was designated as team leader and one handled accounting and logistics. Weekly team-wide meetings were held to set goals, establish tasks and track progress.

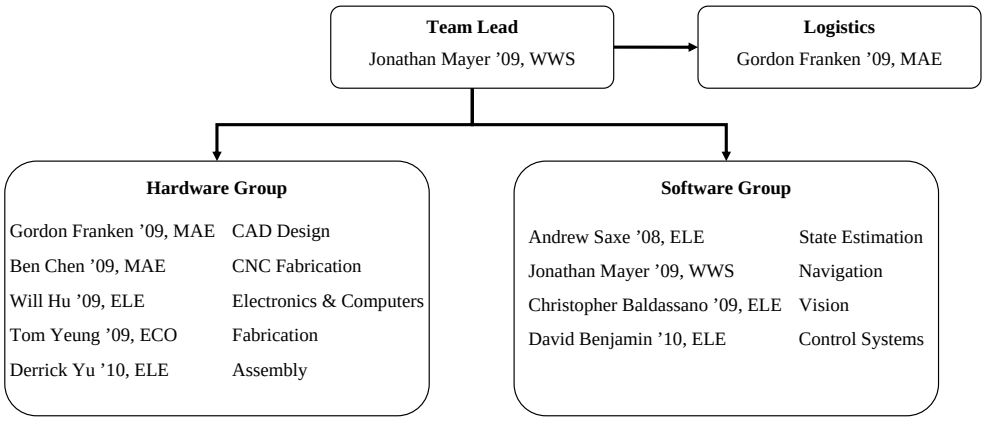


Figure 1: Team Organization Diagram

2 Design Process

Our overall design process followed the steps outlined in Figure 2. In the initial phase, we analyzed the requirements and restrictions for the project to create a set of desired design objectives. Given these design objectives, we determined specific systems and features for Kratos, based on our own knowledge and expertise. These systems were then modeled or simulated to the fullest extent possible using computer software. To minimize wasteful spending and fruitless development time, the required hardware and software components were acquired only after successful computer simulations. We could then fabricate and implement our design. Each system was tested individually, then integrated and tested as a whole. If, at any stage, a system's actual performance did not adequately meet its design objective, a re-design was necessary. Systems that did meet their objectives were deployed, though we continue to monitor them for deteriorating conditions.

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

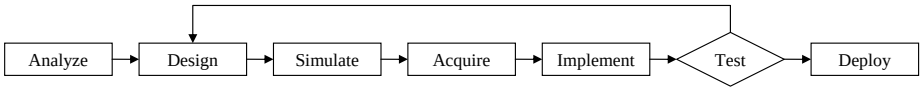


Figure 2: Flowchart of Design Process

We analyzed the rules and requirements of the 2008 IGVC, as well as restrictions unique to our team, and identified the following design objectives for Kratos.

1. Complete design, fabrication and testing phases during one academic semester
2. $6000 budget for materials and components
3. Meet size requirements as specified in competition rules
4. Carry 20 lb payload
5. Observe maximum speed of 5 mph over flat grass
6. Scale a 15% incline
7. Detect and follow white and yellow lane markings on grass
8. Detect and avoid trash-can sized obstacles of various colors
9. Know global and local position to <1m accuracy at all times
10. Travel through environment to quickly achieve mission objectives
11. Dynamically re-plan based on changing information
12. Contain on-board power for at least 6 minutes of continuous operation
13. RJ-45 interface for JAUS interoperability
14. Substantial computing power to process all sensor data and perform decision making
15. Programming interface that supports expedited deployment, data collection and implementation

Initial examination of objectives 1 and 2 led us to our primary guiding principle: simplicity. Our previous experience with robotics competitions has taught us that reliability and testing are vital for success. The primary way to achieve this is through simple designs: complex mechanisms contain more failure points and complex algorithms are hard to debug. Throughout this project, we sought to keep our designs "As simple as possible, but no simpler".[2] In doing so, we not only decrease our implementation time, but expedite debugging and testing. All of the design decisions discussed herein are assumed to have considered design objectives 1 and 2 foremost, under the premise of simplicity.

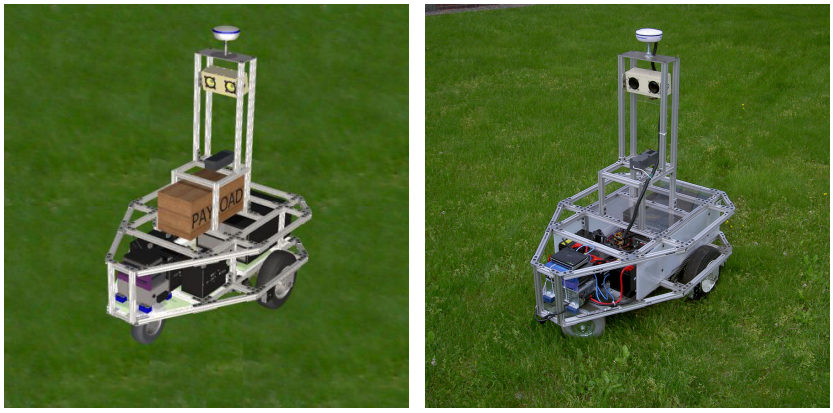
Objectives 3 - 6 are mechanical design requirements and are addressed in Section 3.2. Objectives 12 - 15 are electrical and computing requirements and are addressed in Sections 3.5 and 3.6. Objectives 7 - 9 are sensing requirements and are addressed in Sections 3.7 and 4.3. Objectives 10 - 11 are software requirements and are addressed in Section 4.

3 Hardware

Kratos has a width of 32 inches, a length of 47 inches, a height of 58 and weighs 205 pounds without payload. Kratos is based on a three wheel chassis, which is stable in any orientation on sloped or uneven terrain. Two fixed, powered wheels result in a zero turning radius enabling maximum mobility and holonomic path planning. The 12.5" drive wheels and an 8" caster provide roughly 5" of ground clearance. On top of the chassis is the sensor tower, which holds navigation and obstacle detection sensors as well as the required emergency-stop button. A cavity beneath the sensor tower supports and secures the payload, fulfilling design objective 4.

In keeping with our design process, the entire robot was modeled with *Autodesk Inventor 2008* CAD software before any construction began. *Inventor* is a powerful CAD tool that 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 fabricated robot bears a close

[2]quote by Albert Einstein



(a) CAD Model (b) Real-World

Figure 3: Visualizations of Kratos

resemblance with the original CAD design. An overall parts list of components used on Kratos is shown in Table 1. The total team cost stayed within budget, thereby meeting design objective 2.

Item	**Actual Cost**	**Team Cost**
Hemisphere GPS	$1500	$0
Point Grey Bumblebee2 stereo camera	$1895	$1895
Honeywell HMR3000 digital compass	$850	$0
US Digital E6 rotary encoders	$500	$500
Drive motors	$170	$170
Motor controllers	$460	$460
BaneBots gearboxes	$300	$300
Drivetrain components	$300	$300
Raw material for chassis	$600	$600
Misc. hardware	$450	$450
Labjack Data acquisition card	$500	$0
2x Computers	$1800	$400
3x Batteries	$215	$215
Battery charger	$185	$185
Ethernet routers and switch	$75	$45
Total:	**$9800**	**$5520**

Table 1: List of costs for building Kratos

3.1 Hardware Innovation

Our time and budget constraints prevented us from exploring innovation in mechanical design. Our guiding principle of simplicity further discouraged us from attempting to fabricate complex mechanisms. We chose to maintain simple mechanics so that we could focus on our software development. That said, there are several innovative aspects of our hardware implementation that are worth noting.

We chose to use the 80/20© framing system to construct the entirety of Kratos's chassis (Section 3.3), which allows for modularity and adaptivity in our design. For example, at one point during our testing phase we decided to switch to larger wheels. Rather than fabricate an entirely new chassis, we were able to extend the existing one in a single day. In the future, Kratos will remain on-campus as a functional robot test platform and it will be very easy for other students to mount additional sensors and mechanisms for their own research purposes.

We believe that our sensor suite (Section 3.7) is innovative and possibly unique among IGVC participants. Our use of a single stereo camera for detecting both lanes and obstacles is simple solution that is also extremely advantageous.

In this configuration, lane and obstacle detection are both performed in identical reference frames and require no transformations in software to merge data from them. Furthermore, our detection routines are set up so as to assist each other. For example, if a pixel is identified as an obstacle, it can be ignored during the search for lane markings.

Because stereo vision is computer processor intensive, Kratos requires two computers (Section 3.6). Our computers were custom built in an innovative way so as to take full advantage of their capabilities. Both machines have identical internal hardware and share spare parts. Both computers are built from the same drive image and data from each computer is backed up on the other. We can also easily perform hardware upgrades, such as faster processors or more RAM at any point, should we find it necessary. Each of our code modules is deployed as an individual program, which allows the operating system to perform process-level switching, taking the best advantage of the multi-core processors.

3.2 Mechanical Design

To meet design objectives 5 and 6 we performed a comprehensive analysis of power requirements for motor selection, carefully factoring real-world conditions. Gearbox efficiency was assumed to be 90% in each of three reduction stages. Coefficients of rolling friction over grass and wood were taken to be .2 and .1, repsectively, as estimated from published data and confirmed experimentally. In addition, motor power curves were adjusted for increased resistance. Three situations were identified as distinctive cases of robot motion. Their power requirements are analyzed in Table 2. Case 3 was found to be the most demanding and was considered first during motor analysis.

Case	**Net Force**	**Power Required**	**Motor Power**	**Power per side**
1: Full speed forward on grass	196.2	438.5	601.5	300.8
2: Turning in place on grass	130.8	292.3	401.0	401.1
3: Driving up a 10° incline	268.4	600.0	823.0	411.5

Table 2: Power requirements for robot motion. Speed in (m/s), force in (N), power in (W). Cases 1 and 3 assume power is split evenly between the each side of the robot, whereas case 2 assumes only one motor is operating.

Motor choices were limited by our decision to use only 12V power. After examining several options, we decided to use two CIM motors on each side of the robot, controlled by Victor884 speed controllers made by IFIRobotics. These motors were selected due to their popularity in the FIRST Robotics Competition as well as the availability of many aftermarket gearboxes and parts for them. This configuration produces a maximum of 440W per side, which is enough power to meet all of the cases outlined in Table 2.

To determine final gearbox ratios and wheel diameters, a full analysis of the motor power curve was undertaken. Key values for the 2-motor setup were taken from published data[2] and are shown below:

$$\begin{aligned} V_n &= 12 \text{ V} \quad \textit{(nominal voltage)} \\ I_s &= 266 \text{ A} \quad \textit{(stall current)} \\ \tau_s &= 4.84 \textit{ N-m} \quad \textit{(stall torque)} \\ \omega_f &= 556.7 \text{ rad/s} \quad \textit{(no-load speed)} \\ I_f &= 5.4 \text{ A} \quad \textit{(no-load current)} \end{aligned}$$

Additional motor parameters were calculated and a numerical simulation of the motor power curve was conducted using the following formulas; the results are shown in Figure 4.

We narrowed the design region on the motor power curve by considering two key points, peak efficiency and peak available power. The Victor884 speed controllers are only rated to 40A, limiting our power to 80A per side. This reduced the peak available motor power from 440W to 433W. As can be seen in Figure 4, the motors produce 212W at their peak efficiency of 60%. We predicted that we will be operating at or near case 1 about 50% of the time, at or near case 2 45% of the time, and in case 3 only 5%. We therefore aimed to have cases 1 and 2 fall near the peak efficiency of the motors, allowing case 3 to fall closer to peak allowable power.

$$R = \frac{V_n}{I_s} \quad (motor\ resistance)$$

$$K_t = \frac{\tau_s}{I_s - I_f} \quad (torque\ constant)$$

$$K_v = \frac{V_n - I_f \cdot R}{\omega_f} \quad (speed\ constant)$$

$$\tau_l = I_f \cdot K_t \quad (loss\ torque)$$

$$I = \frac{\tau + \tau_l}{K_t} \quad (current)$$

$$\omega = \frac{V_n - I \cdot R}{K_v} \quad (speed)$$

$$P_{in} = V_n \cdot I \quad (power\ in)$$

$$P_{out} = \tau \cdot \omega \quad (power\ out)$$

$$\eta = \frac{P_{out}}{P_{in}} \quad (efficiency)$$

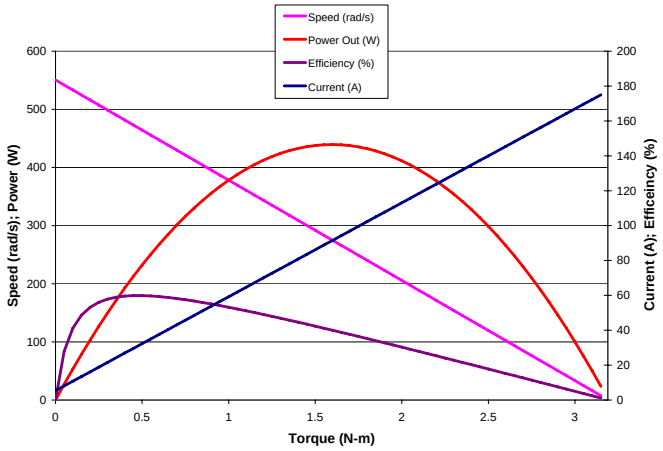


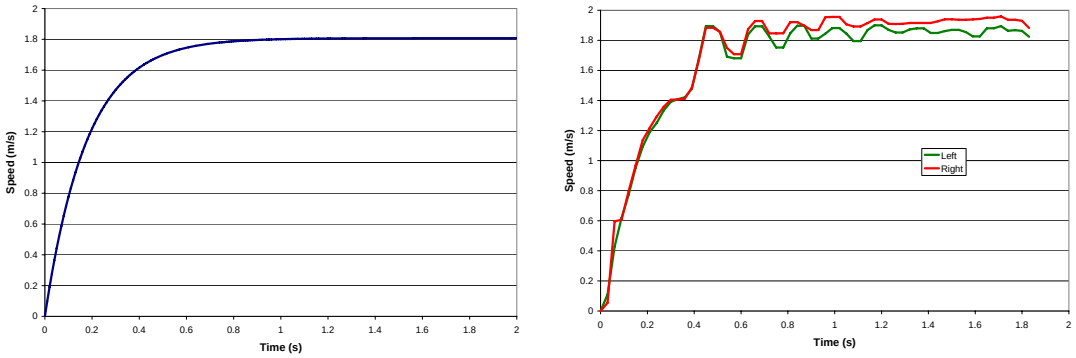
Figure 4: Motor Power Curve for 2x CIM Motor

We selected 12.5" diameter wheels and a 40:1 gear ratio, consisting of a 12:1 planetary gearbox and a 10:3 chain drive. This configuration provides ample power and quick acceleration in all of the design cases, with substantial safety margins for known failure points such as gearbox torque limitations, working chain load and current limits. Since transient current values can be much higher, we wanted to have a large safety margin with respect to the current limit, to ensure that the circuit breakers would not trip. The drawback of this setup is that our max speed in all cases was reduced from 5 mph to at or below 4 mph. However, we considered it better to relax our original design requirements than operate with minimal safety margins.

Case	Speed	Total Power	Motor Power	Current	Efficiency
1:	1.81	177.3	243.2	33.9	59.8
2:	1.69	220.8	302.8	43.4	58.1
3:	1.67	224.9	308.4	44.4	57.8

Table 3: Predicted top speed and power usage for robot motion. Power and current values are for one side only. Speed in (m/s), Power in (W), Current in (A), Efficiency in (%)

A numerical simulation was used to predict our top speed and acceleration. A summary of our predicted performance is shown in Table 3, while a predicted speed response is graphed shown in Figure 5a. These results are very closely aligned with actual speed data taken from the robot, shown in Figure 5b. Both of these graphs demonstrate that our robot is hardware-limited to adhere to the 5mph speed limit set forth in the competition rules and design objective 5.



(a) Numerical Simulation (b) Data From Test Run

Figure 5: Predicted and Actual Robot Speeds

3.3 Fabrication and Assembly

Since Kratos was fully designed using *Autodesk Inventor* CAD software, it could be constructed easily and quickly. Parts were either converted to 2D engineering drawings to aid manual fabrication or exported for CNC manufacturing.

We chose to use the 80/20© framing system as opposed to aluminum or steel bars to construct the entirety of Kratos' chassis. These custom-extruded aluminum bars are stronger than square-tube extruded aluminum of the same size and cost less, although 80/20© requires proprietary fasteners which add to the total cost. Our cost-benefit analysis showed that, due to its strength and ease of assembly, 80/20© best met design objectives 1 and 2.

The primary chassis members are held together by custom machined aluminum plates, while 2 inch aluminum L-brackets provide corner reinforcement in areas of high stress. A platform of 1/4" polycarbonate rests on the bottom of the chassis and supports all the components in Kratos. Polyethylene sheets line the perimeter of the robot, protecting sensitive internal equipment from external hazards.

3.4 Testing and Redesign

We encountered several problems during our initial testing phase that required us to redesign parts of the robot. Our initial design had the two drive wheels in the front with a trailing caster. Early driving tests of Kratos revealed undesirable oversteer properties of this orientation, which caused the robot to spin out in a turn. To fix this, we re-oriented the forward direction of the robot. The tower was reversed so the camera was aimed correctly and wheel rotation inversions were handled in software.

We purchased off-the-shelf gearboxes from a third-party supplier, BaneBots Inc. These gearboxes were designed to interface easily with the motors we selected and expedited the fabrication and assembly process of our robot drivetrain. However, these gearboxes had a maximum torque limitation, beyond which the carrier plates for the planetary geartrains would deform or break. This failure point necessitated a major re-design of our drivetrain. We switched to a smaller diameter wheel to minimize torque applied to the gearbox from the wheels by Kratos' inertial motion. In addition we abandoned our initial decision to directly drive the wheels and switched to a chain drive system. Having the final gear reduction stage be an external chain drive significantly reduced the torque applied to the gearbox and successfully avoided the failure point.

3.5 Electrical Design

Design objective 12 represents the longest running time of the challenges at the IGVC. However, we felt it was extremely important to have longer running time to facilitate extended testing. We aimed to have enough on-board

power for one hour of continuous operation with at least three hours on standby. Power requirements for all electrical components were analyzed as shown in Table 4. With our nominal 12V electrical system, this translates to 82A under

Average Component Power Draw (W)	**Peak**	**Idle**
Driving Straight (50% duty cycle)	243W	0W
Turning (45% duty cycle)	273W	0W
Climbing (5% duty cycle)	31W	0W
Drivetrain Weighted Average	547W	0W
GPS	2W	2W
Compass	0.5W	0.5W
Bumblebee2 Camera	w/computer	
Labjack DAQ	2.5W	2.5W
Encoders	w/Labjack	
Sensors Total	5W	5W
Computers	400W	200W
Routers	15W	15W
Gigabit Switch	6W	6W
Misc. Electronics	10W	10W
Total:	**983W**	**236W**

Table 4: Analysis of Power Requirements

full load and 20A at idle. We selected three Tempest TD33-12 Deep-cycle batteries with a rating of 33A-hr, giving us a total of 99A-hr at full charge. Kratos can therefore last under its own power for over one hour of continuous operation. Furthermore, Kratos can remain powered at idle for over 5 hours. A 12V battery charger is located on-board, so Kratos can easily be tested for extended lengths of time in any location with access to a 120V AC power source.

3.6 Electronics and Computing

A Labjack UE9 data acquisition unit allows for digital and analog input/output between our electronics and computers. Though the UE9 can communicate over both USB and Ethernet, Kratos utilizes the Ethernet connection so that either computer can access data at any time. The UE9 is used by Kratos to generate PWM signals that command the Victor884 motor controllers. It also performs 4x quadrature counting to track the position of the incremental encoders on each axle. The UE9 has several digital I/O, which are used on Kratos to control the warning lights and detect the E-Stop status.

Given the computational intensity of some of the algorithms discussed in Section 4, we determined that Kratos needed two computers to meet design objective 14. One computer is devoted to vision processing, while the other handles navigation and low-level control. Both computes are configured identically, with Intel Core2Duo processors, 2GB of RAM and 200GB hard drives. Each computer is mounted in a low-profile case. The cases are sandwiched together and secured to the chassis with shock-isolating feet. The computers were switched over to 12VDC power supplies rated at 200W, increasing power efficiency. Kratos' network consists of a gigabit Ethernet switch, 802.11g wireless client and a 802.11g wireless access point. Collectively, these devices enable high speed data transfer between the two computers and the Labjack, Internet access through the campus network to ease field testing, and wireless access to Kratos' computers for debugging purposes. The internal network also fulfills design objective 13 by providing and Ethernet connection between the computers and the external JAUS box.

3.7 Sensors

The sole environmental sensor on Kratos is a Point Grey Bumblebee2 color stereo camera. Though design objective 7 necessitates the use of a color camera, design objective 8 can be met with many different sensor types. We chose

to use the same sensor to meet both of these objectives due to our team's previous experience working with stereo vision. The Bumblebee2 is mounted near the top of the sensor tower, with a 12° downward pitch. The Bumblebee2 has a horizontal FOV of 70° and a effective range of 18 meters. The minimum detection distance is 1m, however the Bumblebee2's location on Kratos (elevated and set back from the front) means that this undetectable zone is contained entirely within the robot's physical envelope. A detailed description of our sensing algorithms is found in Section 4.3.

To meet design objective 9, Kratos is equipped with a Hemisphere A100 GPS receiver, located at the top of the sensor tower. Using WAAS differential corrections, it provides global position data to 0.6 m accuracy, as well as heading and velocity at 10Hz via an RS232 serial connection. A Honeywell HMR3000 digital compass outputs magnetic bearing with 1° accuracy at 20Hz, also over RS232. The compass is used for heading information at low speeds, when GPS heading is unreliable, or during a GPS outage. Finally, for precise local motion estimates including individual wheel speeds, Kratos uses two US Digital HB6M rotary incremental encoders, one mounted on each wheel axle. Each encoder has a resolution of 2048 counts per revolution; the 4x quadrature input on the Labjack allows Kratos to detect is own motion with a resolution of .1 mm.

The GPS receiver, camera and compass are all vertically aligned within the sensor tower, which is positioned above the midpoint between the two wheels. In this way, all of Kratos' sensors share the same reference location, thus eliminating the need to perform frame transformations in software.

3.8 Safety

Kratos is equipped with a hardware emergency stop system ("E-Stop") to ensure safe testing conditions. The E-Stop can be manually activated via a red button mounted on the rear of the sensor tower. When enabled, the E-stop disables the input signal to the Victor884 motor controllers, causing the Kratos to come to an immediate stop. A independent wireless E-Stop system with a range of 75 feet has been designed to operate in parallel with the manual E-Stop. Because Kratos' maximum speed is below 5 mph, no hardware speed cutoff is necessary.

4 Software

4.1 Software Innovation

Because our hardware efforts were hindered by time and budget limitations, we aimed to solve the challenges of this competition by combining proven methods with our own ingenuity to form elegant and innovative software solutions.

Our stereo obstacle detection algorithm (Section 4.3.1) is based on an algorithm proposed by [7] in 2005. A main advantage of this algorithm over other stereo processing techniques is that it does not rely on detecting a ground plane for reference, which makes high-accuracy depth information for grass or asphalt unnecessary. We made several simplifications to the original algorithm to increase performance: rather than searching inverted cones above each point, we search inverted triangles and constrain the difference in depth between the points. In addition, only every other point in each triangle is examined.

Our lane detection system (Section 4.3.2) uses a custom set of filters (color, width, and obstacle) to produce a heatmap representing the probability of a yellow or white lane passing through a given location. We then fit parabolas to the map using RANSAC, allowing for better performance around turns, and further improve these results by extending the ends of the lines using a greedy search. The combination of these methods detects a variety of lane shapes with better real-time performance than traditional Hough transforms or Canny detectors.

Our state estimation system (Section 4.4) uses a square root formulation of the central difference Kalman filter. This represents an important innovation over previous positioning approaches deployed in the IGVC. Relative to the

popular extended Kalman Filter, the SRCDKF achieves higher positioning accuracy, and the square root formulation completely eliminates susceptibility to numerical error, because only valid covariance matrices can be represented. Additionally, in recognition of the fact that magnetic north varies with location, we estimate a bias between the compass heading and GPS heading, further improving the accuracy of our position estimates.

Another software innovation is our use of a closed-loop path tracking system (Section 4.6). We adapted a control law for a car-like robot for use on our differential drive robot. Closed-loop path following has a number of advantages over open-loop systems: Paths may be planned at a slower rate than the path tracking controller is updated; paths may be planned with a crude vehicle dynamics model to increase planning speed, and the path tracking controller will still stably track the desired path; and finally the closed-loop feedback will allow the system to track the desired path more accurately through disturbances such as hills, potholes, and sand traps. The specific control law we implemented provably converges to the desired path.

4.2 Platform

Kratos' two computers run the Windows Server operating system for ease of use and maximal hardware compatibility. All software is written in platform-independent C++ using the Visual Studio IDE. 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. Each software component runs as an independent program connected to a central server using Carnegie Mellon's Inter-Process Communication (IPC) platform [9], which provides an API for message publishing and subscription, serialization, and timing.

4.3 Vision

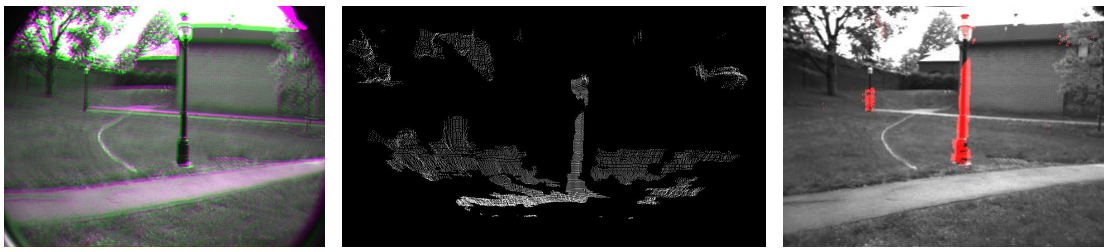
4.3.1 Obstacle Detection

The first step in stereo obstacle detection is the generation of a depth map. Point Grey's stereo vision library computes a disparity map by matching similar pixels between the two stereo images. Leveraging advanced knowledge of the camera's intrinsic characteristics as well as the inverse relationship between disparity and distance, the disparity map is converted into a depth map of each matched point in the image.

Stereo obstacle detection's primary shortcoming is the imperfection inherent to disparity maps; pixel matching is difficult in areas with low texture, poor lighting, or uneven contrast. To improve the quality of its disparity map generation Point Grey's library imposes several validation checks on points in the disparity map, including rejecting pixels in low texture areas or that match to a large number of other pixels. Another significant problem with stereo vision is its inherent minimum and maximum sensing distances. Though the maximum sensing range is limited by the stereo pair's physical baseline separation, the 18 meter effective detection range provided by the Bumblebee2 is more than adequate for the IGVC. Similarly, close objects are not in the field of view of both cameras, resulting in a minimum sensing range. We designed around this constraint by recessing the sensor tower, resulting in a minimum detection distance that does not extend beyond the front of the robot. After computing a depth map we apply a simplified version of the obstacle detection algorithm proposed by Manduchi et al. [7], which searches for pairs of points that are roughly vertical (Algorithm 1). The results of our stereo obstacle detection are shown in Figure 6.

4.3.2 Lane Detection

Our lane detection algorithm functions in three phases. First, a series of filters are applied to each pixel in the image to determine whether it might fall on a lane line. Two color filters respond to pixels that correspond to the yellow and white colors of lane markings, while a rectangular pulse width filter responds to edges of the correct width for a lane marking (adjusted by vertical location in the image). Finally, an obstacle filter raises the score of pixels that do not



(a) Camera Image (b) 3-D Map (c) Obstacle Points

Figure 6: Stages of Obstacle Point Detection

Input: Depth Map, $MaxDepthDiff$ = .3 meters, $MinHeightDiff$ = .3meters, $MaxHeightDiff$ = .5 meters, $MinAngle$ = 80 degrees
Output: Obstacle Points
foreach *Point P* **do**
  T = the inverted triangle of pixels above P;
  foreach *Point Q in T* **do**
    $HeightDiff$ = abs(Height(Q) - Height(P));
    $DepthDiff$ = abs(Depth(Q) - Depth(P));
    $XDiff$ = abs(LateralPosition(Q) - LateralPosition(P));
    $GroundPlaneAngle$ = AngleBetweenGroundPlaneAndPoints(P, Q);
    if $DepthDiff < MaxDepthDiff$ *and* $MinHeightDiff < HeightDiff < MaxHeightDepth$ *and* $HeighDiff/XDiff > tan(MinAngle)$ **then**
      FlagAsObstacle(P);
      FlagAsObstacle(Q);
    end
  end
end

Algorithm 1: The depth map to obstacle points algorithm.

fall on an obstacle. The results of all three filters are fused into a "heat map" indicating the likelihood that each pixel falls on a lane marking. The RANSAC algorithm then attempts to find the parabolic fit that passes near the most lane pixels [3]. Because a parabola does not perfectly describe the geometry of a lane, a greedy pixel-by-pixel search is performed from the furthest pixel in the lane marking. Figure 7 demonstrates the stages of the lane detection algorithm.

4.3.3 Vision Results

Our stereo obstacle detection scheme is able to reliably detect obstacles up to 18 meters (60 feet) away, with the number of obstacle points detected increasing exponentially as the distance decreases (Figure 8), thus meeting design objective 8. After testing, we determined that the depth calculations performed by Point Grey's libraries showed a slight bias at long distance. We accounted for the bias by empirically fitting a correction function, yielding

$$d_a = \frac{2}{100000}(d_m)^4 + d_m,$$

where d_m is the measured depth and d_a is the actual depth.

Our lane detection algorithm was tested extensively during the DARPA Urban Challenge. We evaluated algorithmic performance on 200 images along a highway, with one solid lane marking and several dashed lane markings. Between two and three lanes are visible in each image. In the 200 images, we find that one lane is found in 7% of images, two lanes are found in 88% of images and three are found in 5% of images. In all 200 images, only 5 false positives were observed. We therefore believe that our lane detection algorithm successfully fulfills design objective

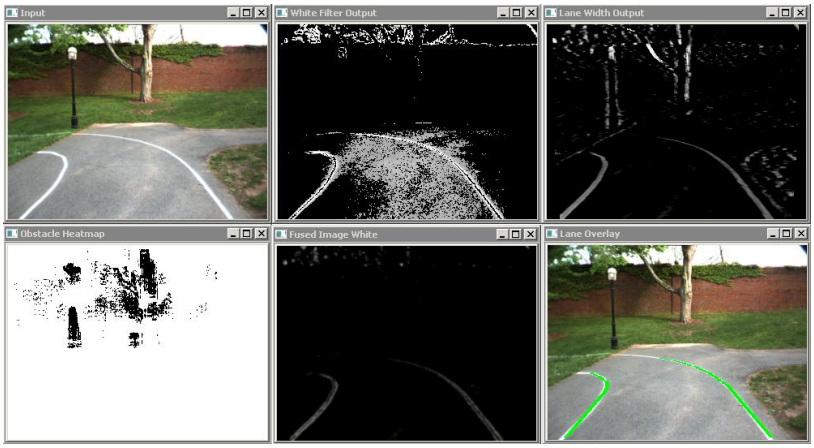


Figure 7: Stages of Lane Detection

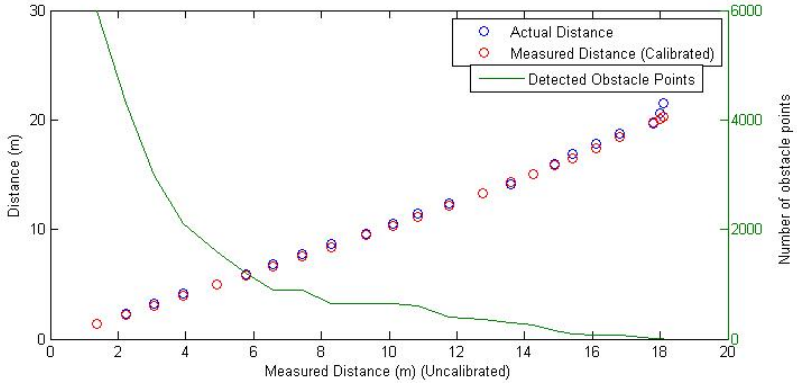


Figure 8: Stereo Vision Results

7.

4.4 State Estimation

We use a square root central difference Kalman filter (SRCDKF) to fuse GPS, wheel encoder, and compass data into a near-optimal estimate of the vehicle's position[10]. The central difference Kalman filter (CDKF) is a type of sigma point filter like the unscented Kalman filter (UKF) which uses a deterministic set of points to represent a multivariate Gaussian distribution over states and measurements. When propagated through nonlinear process and measurement models, these points represent the resulting posterior Gaussian random variable accurately to the second order Taylor series expansion of the nonlinear system. In contrast, the popular extended Kalman filter (EKF) is only accurate up to the first order.

To obtain numerical stability and efficiency, the square root formulation of the CDKF manipulates the lower triangular Cholesky factor of the error covariance matrices. In this way the represented covariance matrices are necessarily symmetric and positive definite. The computational complexity is $O(L^3)$ where L is the length of the

state vector. We represent the state of the robot with a six variable state vector,

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ \theta \\ \delta \\ v_r \\ v_l \end{pmatrix}, \tag{1}$$

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 a bias between compass heading and GPS heading, v_r is right wheel tread speed in m/s, and v_l is left wheel tread speed in m/s.

4.4.1 Process Model

We use a discrete time model of a differential drive robot given by Larsen et al. [6]

$$\mathbf{x}(t+dt) = \begin{pmatrix} x(t) + \left(\frac{v_r(t)+v_l(t)}{2}\right) \sin\left(\theta(t) + \frac{\theta(t)dt}{2}\right) dt \\ y(t) + \left(\frac{v_r(t)+v_l(t)}{2}\right) \cos\left(\theta(t) + \frac{\theta(t)dt}{2}\right) dt \\ \theta(t) + \left(\frac{v_l(t)-v_r(t)}{2}\right) dt \\ \delta(t) + \eta_1 dt \\ v_r(t) + \eta_2 dt \\ v_l(t) + \eta_3 dt \end{pmatrix},$$

where $\eta = [\eta_1 \quad \eta_2 \quad \eta_3]^T$ is a Gaussian random variable and dt is the integration time step of the system (approximately 1/20 s in our implementation).

4.4.2 Measurement Models

The filter incorporates sensor measurements using nonlinear measurement models that predict the value of a given sensor measurement from the current estimate of the state variables. GPS position measurements are predicted to be

$$\mathbf{y}_{\text{GPS Position}}(t) = \begin{pmatrix} x(t) + \alpha_1 \\ y(t) + \alpha_2 \end{pmatrix},$$

where $\alpha = [\alpha_1 \quad \alpha_2]^T$ is a Gaussian random variable corrupting the x and y GPS measurements. The GPS heading measurement is predicted to be

$$y_{\text{GPS Heading}}(t) = (\theta(t) - \pi u(v_r(t) + v_l(t)) + \zeta) \,\%2\pi,$$

where $u(x)$ is the Heaviside step function, ζ is Gaussian noise corrupting the GPS heading measurement, and % is the modulo operator. When the robot is traveling in reverse, the GPS measured direction is opposite the robot heading. The $\pi u(v_r(t) + v_l(t))$ term incorporates this effect by subtracting π from the predicted heading value when the robot is traveling backwards. The GPS speed measurement model is

$$y_{\text{GPS Speed}}(t) = \left| \frac{v_r(t) + v_l(t)}{2} + \sigma \right|$$

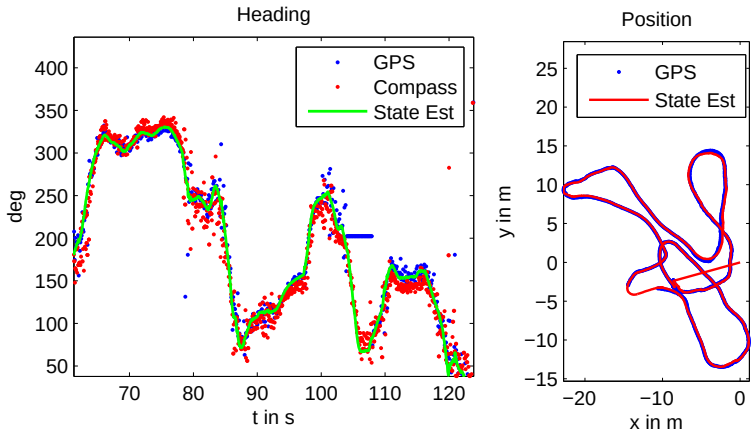


Figure 9: Sample input measurements and filter output.

where σ is Gaussian noise in the GPS speed measurement. The absolute value reflects the fact that the GPS speed is always a positive reading. Measurements of the vehicle's wheel speed are provided by encoders, which use the following measurement model,

$$\mathbf{y}_{\text{Wheel Encoders}}(t) = \begin{pmatrix} v_r(t) + \xi_1 \\ v_l(t) + \xi_2 \end{pmatrix},$$

where $\xi = [\xi_1 \quad \xi_2]^T$ is a Gaussian random variable. Finally, we take the compass measurement model to be

$$y_{\text{Compass Heading}}(t) = (\theta(t) + \delta(t) + \beta)\,\%2\pi,$$

where β is Gaussian noise and $\delta(t)$ is the estimated bias between the compass heading and GPS heading. This bias term allows the filter to automatically correct for magnetic variation by learning a correction factor based on GPS heading.

4.4.3 Testing and Results

The SRCDKF was implemented first in Matlab and checked against an implementation of a linear Kalman filter before being transcribed into C++. The C++ implementation was verified against the Matlab code and tested in a simulator of the process and measurement dynamics before being deployed on the robot. Fig. 9 shows sample heading and position estimates from the filter. We expect the improved accuracy and stability of the SRCDKF to increase the reliability and safety of our robot.

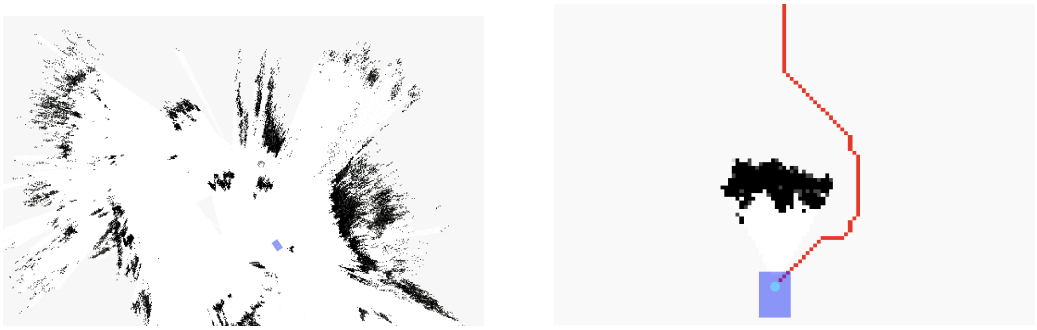
4.5 Navigation

Kratos' navigation scheme follows a cost map approach: the environment is gridded into rectangular cells, which each have an associated numerical cost. With this environmental representation, path planning can be reduced to the well-understood graph search problem.

4.5.1 Cost Map Generation

Stereo points are stored in a dedicated cost map representing the camera's field of view in front of the robot. Each cell in the stereo cost map is assigned a numerical cost according to

$$C(P) = \max(k_{\text{clear}}, \min(1, N(P))k_{\text{first}} + \max(0, N(P) - 1)k_{\text{additional}}), \tag{2}$$



(a) A cost map generated from stereo vision data.

(b) A path planned through a cost map.

Figure 10

where $C(P)$ is the cost associated with the cell containing the point P, $N(P)$ is the number of obstacle, lane, or pothole points in the cell, k_{clear} is the cost associated with a clear cost cell, k_{first} is the cost associated with the first stereo point in a cell, and $k_{\text{additional}}$ is the added cost for each additional stereo point.

The camera's field of view is divided into discrete angular regions. Clear cells closer than the nearest non-clear cell in each region as well as all occupied cells are transformed into global coordinates according to the state estimation data associated with the stereo capture and averaged into the global cost map according to a set learning rate. By initializing the cost map's cells with a cost greater than k_{clear}, open cells' costs are lowered as they are viewed to be clear. Figure 10a demonstrates a cost map generated from stereo data.

4.5.2 Waypoint Selection

Waypoints are presorted into the shortest overall path by enumerating all possible permutations. When lane following, the target waypoint is inferred by extending forwards the center of the furthest lane markings seen.

4.5.3 Path Planning

Path planning is accomplished via a 2D A* graph search from the robot's current position to the robot's destination waypoint. Search nodes correspond to the sum of all cost cells within the robot's footprint (a square with side length equaling the robot's longest protrusion from the state estimation point) at a given point, and nodes are expanded by allowing transitions to the eight neighboring cost cells. Path cost is determined by the sum of footprint costs at each node in the path added to the path's length. The heuristic function determines the minimum possible path cost to the goal from a node, assuming all cost cells are unseen. Because the heuristic may overestimate the path cost from a node it is neither admissible nor consistent, leading to a non-optimal though complete search [8]. While removing the assumption that all costs to the goal are the unseen terrain cost returns the heuristic to optimality, the associated performance decrease necessitates its use. By utilizing a binary heap and minimizing memory allocation requests, the planner is able to operate at roughly 5Hz. Figure 10b shows a path planned through a cost map. The path planner also determines the cost per unit distance of the path it recommends. If the cost per unit distance exceeds a threshold value, the path is assumed to lead to a collision and a low speed is commanded.

4.6 Path Tracking

Paths are specified as a desired series of positions, $\vec{r}(t)$. 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,

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

Hoffmann et al. [4] show that the crosstrack error metric in a point model of the robot evolves according to

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

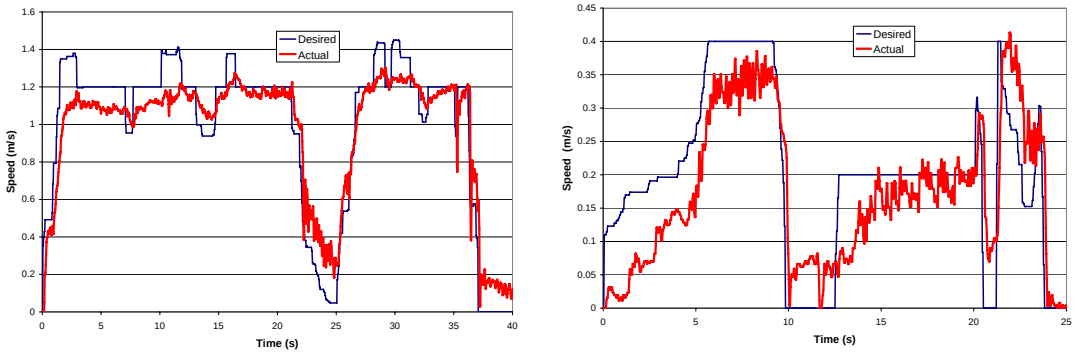
where $v(t)$ is the robot speed and $\psi(t)$ is its heading relative to the trajectory. The heading control law is as given in Hoffmann,

$$\delta(t) = \psi(t) + \arctan\frac{k_e e(t)}{v(t)}.$$

If the heading is tracked perfectly, the cross-track error will provably converge to zero. We use proportional control on the vehicle's angular velocity to track the desired heading. Finally, the path tracker recommends individual wheel speeds based on max robot speed and path curvature.

4.7 Speed Control

Speed control is accomplished using proportional-integral controllers. The controller design is based on a first order exponential approach model of the motors, $y = V \cdot K \cdot (1 - e^{(\frac{-t}{\tau})})$, where y is motor speed, V is input voltage and t is time. The constants of K and τ were found experimentally by curve fitting to motor step response data. The transfer function is of the form $\frac{V(s)}{U(s)} = \frac{a}{s+b}$, which is always stable with PI control if delay is neglected. Preliminary gain selection was performed using the root-locus method. The control gains were then fine-tuned by hand. Figure 11 shows the speed control response at low and high speeds. The response shows good tracking with zero steady state error. Further tuning may necessary to achieve better performance.



(a) High Speed (b) Low Speed

Figure 11: Robot Speed Control Response

5 Conclusion

We believe that the design and implementation of Kratos, as presented in this paper, represents a robust and reliable robotic system that will be competitive at the 2008 Intelligent Ground Vehicle Competition. We hope that this paper demonstrates the innovative problem solving performed by our team members. We would like to further acknowledge the exceptional education benefits that arise from bringing a complex, interdisciplinary project like this through the entire design process. Thanks to the IGVC we have gained greater respect and understanding of the engineering process and look forward to competing this year and in years to come.

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] Inc. BaneBots. Two 62mm cim motor specifications, 2007.

[3] 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.

[4] 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.

[5] 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.

[6] Thomas Dall Larsen, Karsten Lentfer Hansen, Nils A. Andersen, and Ole Ravn. Design of kalman filters for mobile robots; evaluation of the kinematic and odometric approach. In *Proceedings of the 1999 IEEE International Conference on Control Applications*, volume 2, pages 1021–1026, 1999.

[7] 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.

[8] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition, 2003.

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

[10] Rudolph van der Merwe and Eric Wan. Sigma-point kalman filters for integrated navigation. 2004.

Special Thanks

We are deeply indebted to Professor Schapire for assisting us throughout this project. Thanks are also due to the PAVE Prospect 12 team, including adviser Professor Alain Kornhauser and team leader Issa Ashwash, who always seemed to have a replacement ready when our equipment failed. Finally, we couldn't have accomplished so much in one semester without the resources of the School of Engineering and Applied Science, Department of Computer Science, and Department of Mechanical and Aerospace Engineering.