Athena

Franklin W. Olin College of Engineering
2010 Intelligent Ground Vehicles Competition Entry

Nicholas Hobbs, Raphael Cherney, Daniel Grieneisen, Jacob Izraelevitz,
Gray Thomas, Jessica Noglows, Eric Kolker, Luas Hill

Faculty Advisor: Professor David Barrett

Required Faculty Statement: I certify that the engineering design of the new vehicle, Athena, described in this report, has been significant and equivalent to what might be awarded credit in a senior design course.

David Barrett

Professor David Barrett

1 Introduction

This report proudly presents the Franklin W. Olin College of Engineering 2010 Intelligent Ground Vehicle Competition entry: Athena. The vehicle attempts to increase cognition rates and utilize alternative computing fabrics while promoting systems level design thinking across the vehicle's many technical domains. We believe that these efforts have resulted in an innovate design that contains several elements novel to the IGVC. This design, as well as the methodologies used to develop it, will be described in the following report.

Figure 1: Photograph of Athena. Pictured is the Franklin W. Olin College of Engineering 2010 IGVC entry. Athena brings together several computing fabrics to implement a true subsumptive architecture.

2 Team Organization

This year the team had two organizational goals: improved project management and development of external relationships. These two initiatives were led by the team's Project Manager and the Director of New Technology Development. In order to track project progress, adapt to lags in scheduling, and manage team funds, the Project Manager developed a work breakdown structure based on the technical performance requirements. The work breakdown structure was then converted to a resource leveled critical path schedule. Each activity was given value based on work hours to completion as estimated by the team's subject matter expert. The project manager then set an earned value baseline and tracked the cost and schedule performance indices throughout the project – submitting biweekly earned value progress reports to the sponsoring faculty member. These reports helped reform activity duration estimates and will serve as a valuable data set for future project feasibility studies.

In addition to increasing the management and tracking of technical goals, the team looked to establish relationships with corporate partners while continuing to develop our strong relationship with National Instruments. NI has been a vital partner for our team's success and we look forward to working with them in the future. Without our NI sbRIO and the LabVIEW FPGA, none of our technical innovations would be possible. Additionally, the close support of their on site liaison and remote application engineers made for much shorter debugs and a streamlined development process. In addition to our renewed partnership, we are happy to begin a partnership with VectorNav Technologies. Their Attitude and Heading Reference System uses an Extended Kalman Filter to fuse data from a 3-axis accelerometer, 3-axis magnetometer, and 3-axis gyro at 200 Hz. This powerful sensor is a critical part of this year's Navigation strategy, and we are very grateful for VectorNav allowing us to showcase their impressive hardware. Additional relationships are currently being pursued with both Valde Systems and Segway Robotics.

3 Design Process

The team started this year with a review of the 2009 IGVC competition, including video dissection, failure analysis, and design process review. This led directly to the technical requirements that drove many of this year's innovations. Additionally, team members read robotic literature from both past IGVC competitors and the robotics community. These inputs led to a several high level goals for the season:

1. This year's vehicle platform should include truly original design decisions that have the potential to extend the knowledge base of the IGVC and general robotics communities.

2. Cognition speed should be increased to allow the average effective operation velocity to exceed two miles per hour.

3. Our design should extensively explore and utilize alternate computing fabrics.

4. Our design should integrate seamlessly and expand easily with extended vehicle capability.

Our team chose to explore and extend the subsumption architecture purposed by Rodney Brooks. As a result, our design process was a tightly coupled development and test cycle with little to no vehicle simulation [3]. Individual behavioral models were developed incrementally and fully debugged before moving onto creation of additional behavioral modules. This was enabled by an agile software development process that defined future behavior requirements on the behaviors observed during testing. The entire code base is hosted on a Google Code repository. In addition to helping our own development, we hope that publicly hosting the code will encourage community innovation and extension of our own ideas. Thus, our entire code base is shared publicly under then GNU General Public License v3 code license.

Throughout our design process, we prioritized decisions that created systems that would be easily integrated and extended in the future. Robotics is by its very nature a systems discipline. Successful robots must efficiently distribute shared resources – such as power and data – while also consolidating disparate inputs into a single, intelligent action. Only through careful consideration of the system design can a truly robust and expandable system be created.

This year began with a complete rewiring of our vehicle platform due in no small part to poor systems level decisions made in the past. Recognition of the importance of systems level design caused our team to reexamine our overall vehicle strategy. This understanding led to key redesigns that we will briefly enumerated here and discussed in greater detail later. These innovations include compartmentalization of similar electrical components into independently weatherproofed submodules and common Ethernet interfaces for signal lines passed between modules. Our computer architecture also integrated more seamlessly with other robotic systems by introducing a wider range of computing fabrics with more expansive I/O capacities. Finally, even our software strategy was targeted towards integration and future extension of Athena. Our implementation of a modified subsumption architecture creates highly abstract, fully debugged software modules that can be deployed by later developers who have no knowledge of the modules' original development. Furthermore, these modules' design for FPGA deployment make them expandable to a wide range of different FPGA targets.

In the following sections we will describe the implementation of these design principles: the vehicle Athena. Athena strives to be a robust, expandable platform that pushes the boundary of our own ability to create rapid cognitive process.

4 Electrical Systems

This section discusses Athena's electrical systems. The new capabilities of Athena's electrical systems create a more robust, intuitive hardware structure, while still managing to implement higher complexity level systems.

4.1 Actuation

Athena's primary actuator is a 24 V, 1.3 horsepower brushed DC drive motor, reduced by a 2.5:1 sprocket-chain drivetrain. Its power is delivered by a Curtis 1228 speed controller, which includes programmable on-board acceleration profiling. Both this drive motor and the motor controller are native to the base vehicle, a Doran Electric Chimp™ tricycle. This base platform was selected based on its ample cargo space and stable three wheel configuration. The base vehicle is designed for paved and grassy environments, so additional mechanical adaptations to the original vehicle were limited to automation of vehicle control. The location of a drive wheel in the front of the vehicle makes Athena inherently stable on uphill inclines. The center of mass of the vehicle, located behind the drive wheel, acts to correct the direction of motion. This model has been verified by a notable decrease in vehicle performance when running backwards uphill.

The steering actuator is a 24 V Pittman gear motor with an integrated 500 CPR encoder, reduced by an internal 65.5:1 gearbox and an external 7:85 sprocket-chain drive. This actuation scheme turns the front wheel, which is also the primary drive wheel. The steering system is therefore nonholonomic, unlike a vehicle with differential steering that can drive on any path. While this adds an additional burden to the software algorithms, path planning for nonholonomic vehicles is a primary research topic for intelligent vehicles because of the inherent hardware advantages of converting standard automobiles.

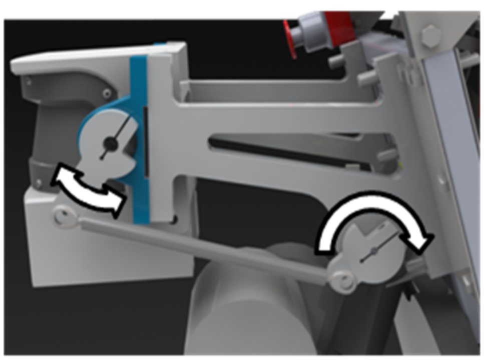

Figure 2: LIDAR Actuation Mechanism. The four bar linkage oscillates the LIDAR angle and acts as a mechanical safety mechanism. If the motor were connected directly to the LIDAR and began spinning uncontrollably, the LIDAR would damage itself and its cabling.

We have an additional sensor actuator to tilt the front LIDAR, which consists of another 24 V Pittman gear motor with an integrated 500 CPR encoder, reduced by an internal 65.5:1 gearbox. This actuator is connected to the LIDAR by a four bar linkage mechanism (Figure 2). The four bar linkage is optimally suited for actuation over a limited range of motion. Because the motor level arm is less than half the length of the other arm, continuous rotation of the motor will only bob the LIDAR head, rather than turning the sensor into a hard stop or tangling the cables. This mechanical safety mechanism ensures that bugs in software development run no risk of sensor damage. Both the steering and sensor actuators are powered by Galil PWM servo amplifiers. These motor controllers are internally current limited.

4.2 Computation

Athena makes use of multiple computing fabrics to create an architecture that maximizes the strengths of traditional CPUs, real time controllers, and field programmable gate arrays (FPGAs). The result is a highly parallelized, extendable, low cost robotic architecture. Athena's processing is divided across a Dell laptop and a National Instruments Single Board RIO (sbRIO) – which itself contains a real time controller and Xilinx FPGA. The laptop is a Dell Latitude E6400 with an Intel Core2 Duo 2.93GHz processor and an NVIDIA Quadro NVS 160M graphics processing unit. This PC runs LabVIEW 2009 for Robotics in a Window 7 environment. The NI sbRIO-9642 runs the LabVIEW real time operating system on a 400 MHz processor and contains a 2 million gate Xilinx Spartan III 2000 FPGA. In addition to these computing resources, the sbRIO has extensive I/O capabilities. The board has 110 TTL digital I/O channels, 32 24 V logic digital I/O channels, 32 single ended analog input channels, and 4 16-bit DACs for output. Additionally, the sbRIO provides four 5 V regulated power supplies. These systems are both powered off of a 24 V regulated DC supply. The Latitude and sbRIO draw 4.6 amps and 1.8 amps, respectively.

This computing architecture empowers three important innovations. First, the diverse range of computing environments allows algorithms to be implemented in the environment that creates optimal performance. Additionally, the extensive use of FPGA computing enables simultaneous data acquisition at rates many times faster than Olin's 2009 vehicle. Finally, the architectures are easily extended to accommodate future development.

4.2.1 Optimized Environment

Diverse computing fabrics allow algorithms and filters to be implemented in a performance optimized environment. This begins at each individual sensor input, where Athena takes advantage of the FPGA to create several dedicated, parallel processors which parse input streams, transform data, and then react to this new data by updating the set of possible future actions. FPGAs can run the finite automata used to recognize sensor output strings, calculate the repetitive math used for coordinate transforms, and control the finite state machines used for basic behaviors with no overhead. Because a traditional CPU runs an operating system whose scheduler must run these numerous parallel processes through two cores, its overhead is massive by comparison. On the other hand, many image algorithms - which require large matrix operations and a much higher degree of algorithmic complexity - perform much more effectively in a traditional PC. For this reason, all of our vision processing is done on the Dell laptop. Finally, the real time processor cannot compute the volumes of data that a PC CPU can, but it provides an environment where deterministic cognition can create predictable behavior.

4.2.2 Increased Processing Speed

In addition to providing better performance, use of multiple computing fabrics allows Athena to move the hardware abstraction boundary. Traditional robotic architectures have a hardware/software interface at the level of a string based API. In contrast, Athena's FPGA processing cores convert sensor output into meaningful representations before the data is transferred into software. The specifics of these interfaces and representations will be discussed later in the software section.

This higher level of hardware abstraction also leads to our second significant innovation. Hardware implementation of parsing, conversion, and data processing drastically increases the vehicle's overall processing rate. By removing lower level processing from software, the FPGA frees CPU cycles to be spent on higher level algorithms and more intensive image processing. The speed gains introduced by the FPGA can be seen in the handling of the SICK LIDAR. Athena is able process LIDAR data scans at 70 Hz while last year's vehicle, Brian, was only able process scans at 10 Hz. This is a direct result of implementing each sensor's processing in true parallel. The increased data rates give Athena a more accurate view of her environment and empowers faster decision making. These high data rates combined with the oscillation of the LIDAR also allow Athena to scan five different plains and create a three dimensional obstacle detection sensor with a 15Hz update rate. This is one example of how parallelizing data processing in hardware creates a richer environmental awareness. Alternatively, this increased speed could be used in a single plane to drastically reduce Athena's reaction time. Because the IGVC navigation and autonomous courses are static and well structured, the team has chosen increased spacial awareness provided by oscillating LIDAR. Even with this trade off, the three dimensional obstacle scans update a rate 50% faster than the single-plane LIDAR data processing from the 2009 vehicle.

4.2.3 Extendability

The final advantage of this architecture is that it is highly extendable. In order to build more complicated robotic systems, a code base must develop over time. To achieve this, software teams must be able to preserve abstraction barriers between code modules that have been previously developed. This becomes challenging, however, if computing occurs solely on CPUs. While software may not require interaction with other code modules, parallel processes must vie for the same, finite number of cores. This means that while you do not explicitly design interaction, parallel threads and processes are managed by the operating system's scheduler. As a result, they will frequently interact in ways that even developers who understand the entire system cannot predict or comprehend. This problem is greatly reduced in the Athena architecture. Software modules developed for the FPGA will function identically if they are placed in parallel with no processes or thirty processes. It is true that resources are still finite and that if they run out, the code will fail to compile. However, upon successful compilation, a developer will know that a FPGA module will not interact with other

Part	*Cost to Team*	*Estimated Cost*
Sony DFW-SX910	$0	$2090
SICK LMS291-S05 LIDAR	$0	$6000
Doran Electric Vehicles Chimp™ Human Transport*	$0	$3195
Dell Latitude D620	$0	$2000
National Instruments 9642 sbRIO	$0	$1200
u-blox Odometry Enabled GPS	$0	$600
Galil MSA-12-80 Motor Controller (quantity 2)	$0	$360
Pittman GM14904S016-R1Motor with Encoder	$350	$350
Pittman GM9236S027-R1	$260	$260
Mechanical Emergency Stop Button (quantity 2)	$240	$240
Autec Wireless Emergency Stop	$0	$700
Aluminum Stock	$180	$180
Encoder	$0	$225
Fujinon DV3.4X3.8SA-1 Megapixel Lens	$90	$90
Total	*$1,120*	*$17,490*

Table 1: Cost Breakdown of Athena

*Includes two Chairman AGM-12100T Deep Cycle Lead Acid Batteries and one Curtis 1228 controller

modules as the two different programs share no resources. As a result, the developer can preserve a perfect abstraction barrier.

This improvement is not limited, however, to code which is implemented on the FPGA. By removing processing to the appropriate computing fabric, there is less complexity in each environment. Domain specific experts can program vision algorithms on a PC without having to have to worry about preserving memory so that a LIDAR obstacle occupancy grid can maintain the proper update rate. Even if the system must be extended to include more computing on more processors, it can easily adapt. Because the architecture converts inputs into useful representations before passing them across computing fabrics, the communications between the various processors (FPGA to CPU or RT processor) does not require transfer of large data volumes. These lighter weight representations can thus transfer over relatively low rate serial communication lines. In addition to the lower volume of data that is transferred, each module's ability to process large volumes of data into lightweight representations at a high rate means that these representations can be updated at a high frequency. As a result, lossy communication is tolerable. An imperfect message can either be ignored or replaced quickly by new information.

By not limiting Athena's intelligence to a single computational domain, we have created an architecture that optimizes algorithm performance to appropriate hardware, processes more data, and can easily be extended in the future. Not only does this computer architecture create a competitive advantage, it does so without raising the vehicle price tag. The cost of the current system is less than deploying two parallel laptops. In the case of our system, conversion to the multifabric architecture saved $1100. Additionally, the system is not even near capacity. At the current stage of development, we have utilized only 38% of the FPGA's logic resources.

4.3 Sensing

Athena's sensor suite for the autonomous challenge consists of a forward-facing, actuated SICK LIDAR for obstacle detection and a statically mounted Sony DFW-SX910 color CCD camera with a wide-angle lens for line detection. Additionally, for the navigation challenge, we use a VectorNav VN-100 Attitude Heading Reference System and a u-blox GPS. The LIDAR is mounted as low as possible on the front of the vehicle for maximum effective range, while the camera is located on the sensor mast pointing down at 45°. Given the much faster data rates enabled by FPGA processing, we can run the LIDAR at 70Hz; fast enough to enable three-dimensional LIDAR scanning. Scanning multiple planes at high speeds allows Athena to scan the entire driveable area for obstacles of all heights. This decreases the need for a global memory representation. In our implementation, we select a range of scanable angles between 6° and 33° from horizontal. This allows us to see both tall and short obstacles, including the lowest obstacle at the 2009 competition. The faster Athena can sweep this 27 degree range, the faster she can travel without hitting obstacles.

In order to oscillate the LIDAR, we implemented a classical feedback controller. This task was complicated by the fact that the four bar linkage is a nonlinear system. There is an angle dependent mechanical advantage between the two shafts. In addition, the LIDAR head has a center of gravity above its point of rotation, so it generates a torque on the LIDAR shaft which is also angle dependent. Using physics and geometry, we were able to design two non-linear compensators to make the LIDAR head system appear linear to its classical controller. The entire controller, with both non-linear compensators and the classical linear compensator, can run seamlessly on the FPGA due to a custom multiplier sharing architecture.

To explain the computed torque compensator, the force of gravity acting on the LIDAR's center of mass exerts a torque on the LIDAR head based on the sine of the angle the center of mass makes with the vertical at the LIDAR shaft. The computed torque compensator estimates the motor current needed to counteract gravity and maintain the current position. Using trigonometry we can derive the relationship between angles ϕ_A and ϕ_B, which correspond to the LIDAR and motor cranks respectively, and thus correlate motor encoder readings to the angle of the LIDAR center of mass, Figure 3 and (1) demonstrate this relationship.

$$\begin{cases} \phi_A = \frac{\pi}{2} - \arcsin\left(\frac{\bar{BC}\sin(\phi_B + \frac{\pi}{2})}{\bar{CA}}\right) - \arccos\left(\frac{\bar{CD}^2 - \bar{AC}^2 - \bar{AD}^2}{2\bar{AC}\bar{AD}}\right) \\ \bar{CA} = \sqrt{\bar{BC}^2 + \bar{AB}^2 + 2\bar{AB}\bar{BC}\sin(\phi_B)} \end{cases} \tag{1}$$

This relationship assumes we start at a known motor position. We handle dependence by setting $\phi_B = \frac{-\pi}{2}$ before the robot turns on. This position was chosen because it is both easy to find by the robot operator and because

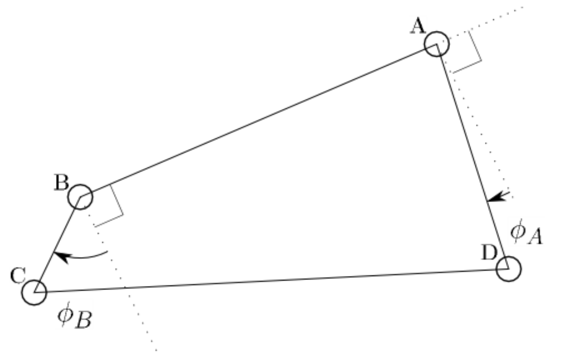

Figure 3: LIDAR Control Diagram. This figure shows the basic setup for the LIDAR control problem. A is the LIDAR shaft, B is the motor shaft, C and D are the linkage joints.

the friction makes this a very stable position.

Our second non-linear compensator linearizes the mechanical advantage, the ratio of the marginal ϕ_b to ϕ_a in terms of the point of interest Φ_a and Φ_b, where Φ_a is calculated from Φ_b by way of (1).

$$\frac{\phi_b}{\phi_a} = \frac{\bar{AD}\bar{AB}\cos(\Phi_a) + \bar{AD}\bar{BC}\sin(\Phi_a + \Phi_b)}{\bar{AB}\bar{BC}\cos(\Phi_b) - \bar{AD}\bar{BC}\sin(\Phi_a + \Phi_b)} \tag{2}$$

We can thus write the linear controller acting on an error defined as the desired LIDAR position less the computed position, and with an additional nonlinear gain equal to the linearized ratio of $\frac{\phi_a}{\phi_b}$. This gain is zero at the starting point, so we add a small additional non-linearity to allow the crank to return to an operating region.

With these two compensators in place, the plant will appear to the classical controller as though it were just a moment of inertia with friction, and the results of a sine wave input will be very close to a sine wave output. We decided on a lead compensator network maintaining a phase margin of 45°to limit overshoot, and maximizing bandwidth. A quick tuning phase allows the operator to oscillate the motor current at an adjustable frequency and measure the magnitude and phase margin of the LIDAR angle with the classical compensator in place. This is done while live updating the classical controller's gains so that it can be tuned to its maximum performance while fulfilling its phase margin requirements.

The two non-linear compensators inevitably misestimate to some degree and introduce noise into the system, but once a feedback control is applied, this noise will becomes negligible. Since this year we are only oscillating the LIDAR head, it may not have been necessary to compensate the non-linearities in this manner, but this control system will allow future versions of Athena to intelligently move the head with predictable results.

4.4 Major Electrical Innovations

Whenever one designs an electrical system, some degree of compromise must be made between efficient use of space and ease of serviceability. This compromise is strongly biased toward serviceability for prototype vehicles, as opposed to commercial vehicles where material optimization directly affects performance. Given the added software complexity of programming a FPGA and limited manpower, we chose not to modify the basic structure of the Chimp™ vehicle platform. The Chimp™ chassis has proven to have ample space to allow for an electrical design optimized for organization, modularity, expandability, and weatherproofing.

4.5 Power Distribution

Actuators generally have much higher current requirements than computing or sensing systems, meaning that rapid changes in actuator motion can cause unwanted transients in the power lines. This problem is solved in Athena's power distribution system through the use of additional power regulation systems for computing and sensing electrical components. In addition, kill switches on the front and back of the robot only cut power to actuators, allowing Athena's computation systems and more sensitive components to continue normal operation. These constraints are traditionally difficult to wire efficiently in a small, mobile system given the spatial requirements of different components. Last year's electrical system encountered this problem; the slowly expanding wiring scheme eventually became an unserviceable tangle. Athena's power distribution system is both organizationally advantageous and expandable.

In Athena's electrical system, all actuators and their associated controllers are powered through a contained, kill switch enabled electrical system, while the computational and sensing systems are housed in an isolated, regulated electrical system with a direct connection to Athena's batteries through the main on/off switch. A simplified wiring diagram is shown in Figure 4.

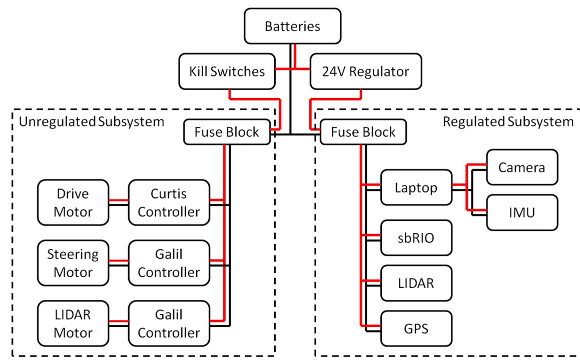

Figure 4: Simplified Wiring Diagram. The power is distributed from the batteries to two separate subsystems, one with multiple kill switches and one with a regulator. These two systems house the actuation and cognition/sensing systems respectively.

The primary advantage for such a system is its inherent modularity. Each subsystem can be powered off a single fused power distribution block, making the system as a whole highly expandable and compartmentalized, shortening the debugging cycle. The two power subsystems are also highly analogous, allowing for recurring use of parallel components. The spatial layout of such a subdivided power system has additional advantages.

4.6 Electrical Layout and Waterproofing

The power distribution strategy allowed us to spatially divide the electrical system into two waterproof enclosures, one for regulated power components and one for unregulated and E-stop controlled components. The unregulated box contains all motor controllers, powered directly from the batteries via the E-stops (Figure 5). The regulated box contains the computational and sensing components, such as the sbRIO and u-blox GPS. The power distribution systems for each box are completely isolated. Signal lines from each individual sensor are combined into Ethernet cables to provide a standard interface, decrease the number of wires to be routed, and provide easy connection at module boundaries. As a result, all cross-communication takes place over waterproof Ethernet, FireWire, and serial cables, so no terminals are exposed to the elements. The weather sensitive components of the electrical system are thereby individually waterproof, thereby avoiding the much more complex problem of creating an entirely sealed vehicle chassis. The other sensors not in the waterproof enclosures are spatially located for optimal sensor range. These sensors are enclosed in their own custom waterproofing enclosures.

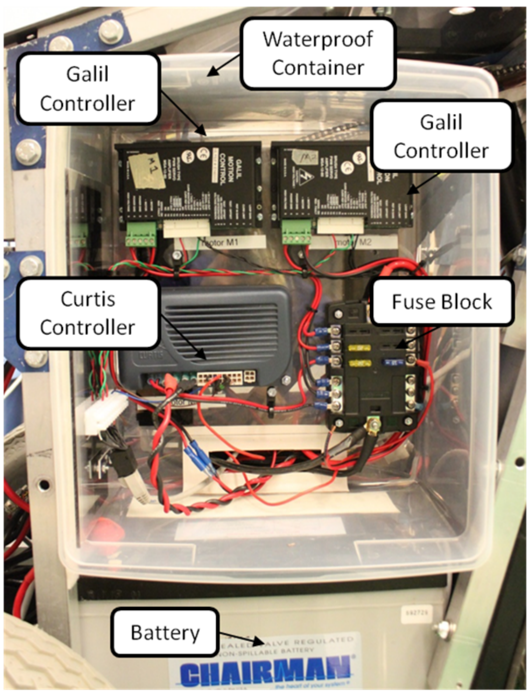

Figure 5: Unregulated Power Electronics Box. This waterproof box, one of two boxes on the vehicle, houses the actuation electronics. Note the ease of service and room for expansion made possible by the organizational technique.

5 Software Systems

5.1 Software Architecture

As discussed, Athena's computation is spread across three different processors: an FPGA, a real time processor, and a PC. Each of these processing units performs a unique role. Figure 6 shows a high level breakdown of software modules. On the FPGA, there are several truly parallel processes that parse and interpret the input data from various high data rate sensors. These inputs are then passed through a series of finite state machines to create a set of possible vehicle behaviors. The PC is used primarily for image processing. Vision algorithms identify lines for the purpose of avoidance and for destination generation. The PC also hosts the GPS because this low data rate sensor has a National Instruments PC-hosted code module. Finally, the RT processor selects the behavior which optimally achieves the goal generated by the vision processing. These operations will now be described in greater detail.

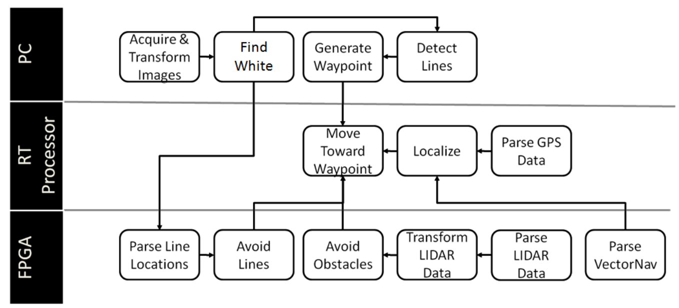

Figure 6: Athena's Software Architecture. The diagram shows which behaviors and algorithms are hosted on what processors.

5.2 FPGA-Hosted Computation

The FPGA hosted 'software' performs three important operations: sensor parsing, data transformation, and behavior generation.

5.2.1 Sensor Parsing

Each sensor's input stream is parsed with either a discrete finite automata (DFA) or a push down automata (PDA), depending on the demands of the parser. Each sensor input is not buffered. Instead, DFAs and PDAs parse input strings byte by byte as they are acquired. An example of a PDA modeled parser can be seen in Figure 7.

This PDA is used to recognize the two state strings output by the VectorNav: a reset acknowledge and a data stream. These strings take the form of "$VNRRG,<x quaternion>,<y quaternion>,<z quaternion>,<quaternion scalar>,<x acceleration>,<y acceleration>,<z acceleration>,<x magnetic force>,<y

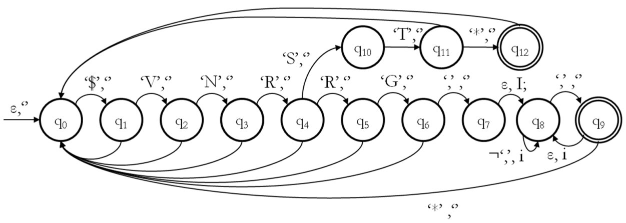

Figure 7: VectorNav Push Down Automata. Each transition is labeled with a 2-tuple, where the first element is the input that generates that transition and the second is the character written to the stack (‘’ denotes no character being written). I is the input character and E is used to denote a transition which is taken regardless of input. Unlabeled transition are taken if the input character does not match and labeled transitions.

magnetic force>,<z magnetic force>*<checksum>” or “$VNRST*<checksum>” where each bracketed element represents fixed point numeric data. Each time the PDA enters the q_{12} accept state, a reset flag is set acknowledging the VectorNav's state. In order to read the data passed to the FPGA in the VNRRG command once the PDA enters the q_8 state, the string that is received is pushed onto the stack until that data element is done being input (as is indicated by the comma delimiter). Then, when the PDA enters the q_9 accept state, the stack is flushed and its data is written to a global variable which corresponds to that data element. The correct data element to write to is tracked by another DFA, which monitors the parsing PDA's state. Each input is tracked with a similar custom core that implements sensor output stream parsing in hardware. Many of the resulting PDA and DFA recognizers are much more complicated, but the same basic approach is applied.

While string parsing of sensor data streams is not usually detailed in robotic design reports, the approach detailed above allows a few key advantages, and is thus important. The use of PDA and DFA recognizers allow unbuffered input to be parsed. As with later stages of FPGA processing, Athena's sensor data acquisition is done without the formation of intermediate representations. Instead, each character is immediately translated into a meaningful representation. This speeds data processing by eliminating overhead. These parsers are also robust in the face of data loss.

5.2.2 Data Transformation

After the parsers recognize strings and turn them into measurements, these measurements are then immediately placed into Athena's local coordinate frame. This converts LIDAR data into Cartesian inches centered about the vehicle's LIDAR head, LIDAR tilt angle and steering wheel angle into degrees, VectorNav heading into compass degrees, and robot velocity in inches per second. Again, this processing is done without buffering or intermediate representations. Transformed data is then immediately passed to the relevant behavior generation module.

5.2.3 Behavior Generation

Once data is parsed and transformed it is immediately passed to parallel behavior generation modules. The primary behavior system is based on an obstacle and line avoidance algorithm successfully implemented by Virgina Tech's 2007 IGVC vehicle, Johnny-5 [5]. The algorithm, summarized in Figure 8, examines 18 possible circular paths corresponding to different front wheel angle positions.

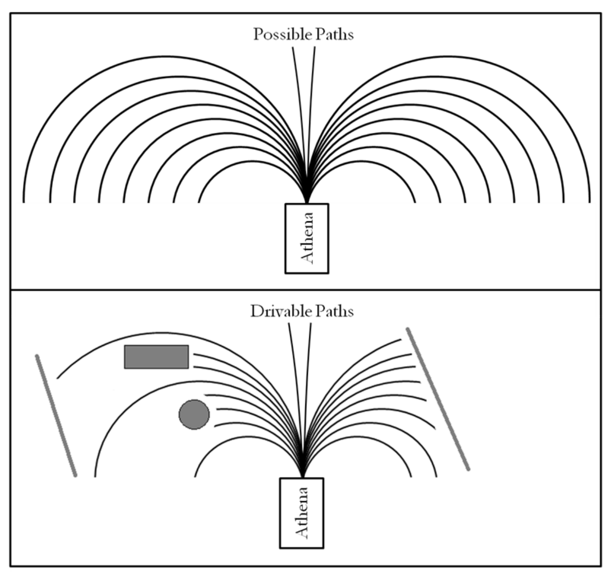

Figure 8: Drivable Paths Algorithm. The algorithm generates a set of 18 possible behaviors in the form of a distance driven along 18 arcs, corresponding to 18 possible wheel angles.

Each path is then shortened to the length that can be driven before an obstacle is encountered. In our implementation of this algorithm, each time an obstacle is identified, its location is checked relative to each path. If at any point along that path, the obstacle is closer than a tunable passing distance, then the path is shortened to length where the path is no longer passable. This algorithm is implemented on the FPGA using the state machine shown in 9.

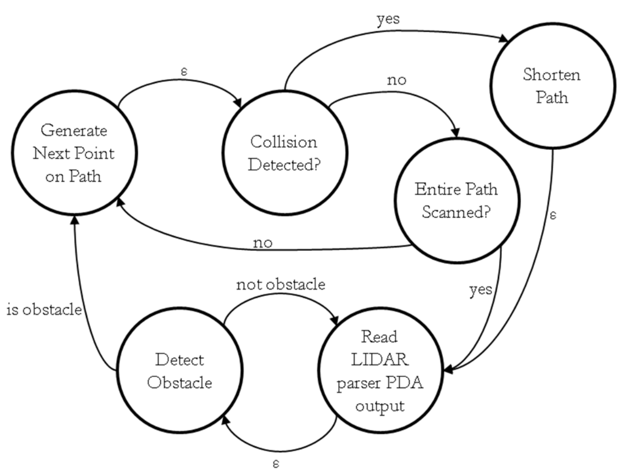

Figure 9: Behavior State Machine. The state machine which is used to generate a set of possible drive behaviors (ϵ transitions are taken regardless of sensor state decision).

Data to update path lengths based on detected lines is provided by a serial connection which directly connects the FPGA to the PC. This connection transmits the location of points which represent 3 inch squares deemed to be the color of lines. Data for the obstacle path lengths is provided by the output of the LIDAR data stream and transform.

In order to update LIDAR scans at 70Hz, the FPGA must be able to process the effect of a data point on each of our 18 parallel paths 12,600 times per second. Each path, in turn requires 112 inches of travel to be examined for collisions. The necessary data rate is achieved by creating custom parallel processors to evaluate each path. Both the line path evaluation and obstacle path evaluation have their own cores, each of which have two parallel path evaluation processors – one for left paths and one for right paths.

Other behavior modules create robot motivation. An "avoid explored areas" behavior tracks a trailing average of vehicle position based of integration of VectorNav output. This in turn gives a heading which corresponds to the least explored area of the vehicle environment. Another useful motivating behavior is the "prefer constant heading" behavior. This creates a trailing average of vehicle heading. Both of these behaviors provide Athena with a useful, lightweight representation of past vehicle action which can be an accurate predictor of appropriate future action.

All of these behavior modules are then fused using a modified subsumption architecture. Subsumption architectures combine behavior modules like the ones described above by layering behaviors into a hierarchy [1]. These behaviors are then blended using subsumption functions. In our architecture, these functions are hosted on the real time processor, and thus, will be discussed in the real time processor section.

5.3 PC

The previous section discussed how data received from sensors and PC based image processing is turned into a set of potential robot behaviors. This section outlines how image processing is implemented so that data can be turned into behaviors. Additionally, this section outlines how vision is used to create a goal for Athena. This goal will later be used to judge the relative merit of the many available behavior sets and combinations.

5.3.1 Vision Overview

The vision software on Athena follows the theme of modular design, adaptability, and optimization present in other subsystems and software branches. Athena captures live images using a Sony DFW-SX910 equipped with a Fujinon 3.4mm wide-angle lens mounted on the static sensor mount of the vehicle. The live images are then transformed onto a 2D plane in front of the robot and corrected for nonlinear distortion (as defined by a previously acquired calibration template). This transformed image – ideally, an "overhead" view – is then made available to a series of filters which extract lines and/or obstacles. The filters include standard and adaptive RGB and HSL

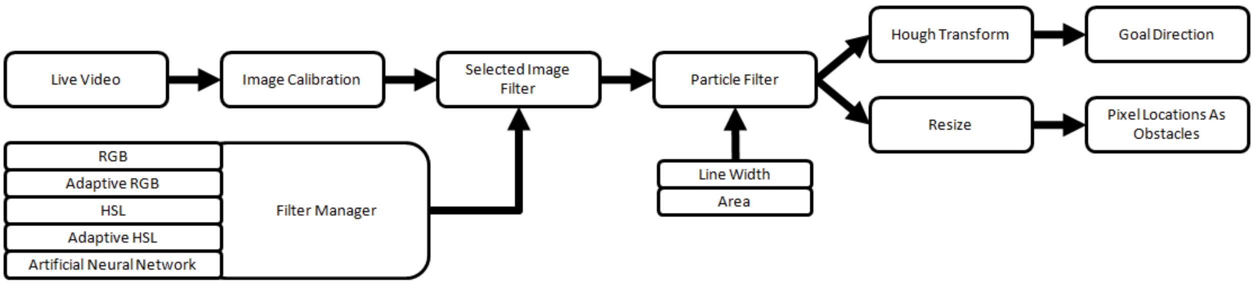

Figure 10: Vision Processing Flowchart. A flowchart showing the vision processing algorithm from the image capture to goal extraction and obstacle identification.

filters, along with a pattern-matching filter based on an artificial neural network. The binary image results can then put through a series of particle filters to filter for object size and width. The final line-filtered image is then downsized, and the x and y coordinates of lines are passed to the same FPGA-based obstacle avoidance and path-planning algorithms used with LIDAR data. Finally, a Hough transform is taken of the image data which allows the robot to detect the presence and consistency of lines and extract a goal from this. Figure 10 diagrams the overall process.

5.3.2 Image Calibration

Image calibration allows us to remap pixels to real-world coordinates. The calibration process analyzes a prerecorded image of a grid of dots with a known spacing and resizes the image such that the grid is spatially correct. Given the information from this transformation, other images can also be corrected. We use this method to transform the camera image to a top-down view, compensate for the fisheye effects of wide-angle lenses, and find the location of lines in real-world units. Figure 11 corrections on the calibration template.

5.3.3 Filters

A wide array of image filters have been created and implemented on Athena. Because lighting conditions can vary greatly between environments, each filter can be selected and tuned for a given test. Filters include basic RGB and HSL filters, adaptive versions of these filters that account for varying lighting conditions, and an artificial neural network pattern-matching filter. Particle filters are then applied to the binary images to reduce noise and extract articles of interest. Line width and area filters are particularly effective. The final filtered image is then resized to reduce the obstacle density, and the x, y coordinates of the lines are passed to the FPGA using RS232. The drivable paths algorithm is then run on this data to avoid lines.

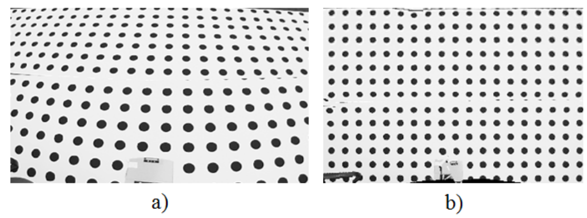

Figure 11: Correcting For Nonlinear Distortion. Image a) shows a calibration image from the wide angle lens before correction and b) shows the same image after transformation.

5.3.4 Goal Orientation

Once the solid lines have been filtered out of an image, a Hough transform is taken of the data to identify the strongest lines. The slope of these lines, if deemed good enough fits, are used to determine the desired heading. Because we are using a wide-angle lens, we can guarantee that we will see past small gaps lines and will often see both sides of the course. An effective Hough transform can ensure that the desired heading remains correct even when there are gaps in lines.

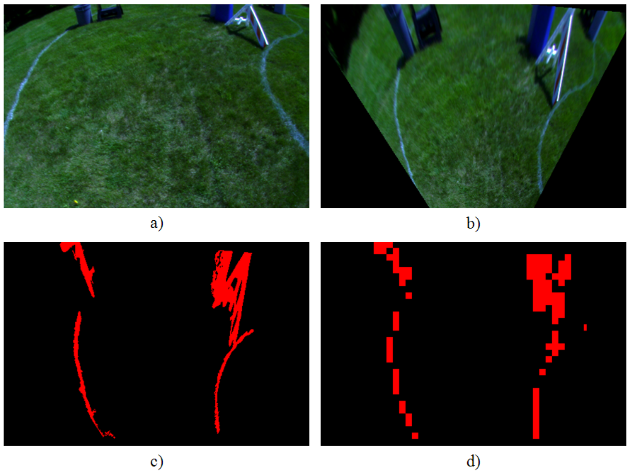

Figure 12: Vision Processing Example. Image a) shows the raw camera output, b) shows the same image after calibration, c) shows the lines extracted with an adaptive RGB and particle filters, and d) shows the resized image passed to the FPGA to calculate drivable paths.

5.3.5 Navigation

Athena's localization and global navigation system is composed of an u-blox series 5 GPS and a VectorNav-100T Attitude and Heading Reference System (AHRS). A low-accuracy localization unit, the u-blox GPS provides readings of a global position to within three meters of the true location, at a rate of one reading per second. At one hertz, the data rate is low enough to negate simple averaging as a method of increasing the accuracy of global position measurements and improving global waypoint finding and navigation.

Our solution to combating this deficiency in localization takes a macro to micro approach. We use the u-blox GPS to guide Athena to the area roughly around a designated GPS target coordinate. Combining the current GPS coordinates with a compass heading from the VectorNav, our navigation algorithm provides Athena with a preferable path upon which to drive to the target location. Once in range, the control system ignores further information from the GPS, and executes a circular search pattern, reaching the target position at some point in its path. To ensure that Athena does not wander from the specified search pattern, IMU measurements from the VectorNav AHRS keep Athena aware of her position relative to the start of the search pattern. This allows us to fully cover the area in which we know the target location to be. Athena's navigation system is simple in structure and implementation, but it provides a solution to the problem of global waypoint finding and navigation using only low-cost sensors and simple algorithms.

5.4 Real Time Processor

As discussed in the previous section, data from the sensors and the PC image processing is converted into a set of possible behaviors by the FPGA. These behaviors are then organized into a subsumptive architecture. The subsumption functions which allow these behaviors to interact are hosted on the real time processor. In the original subsumptive architecture proposed by Rodney Brooks layers the set of behavior generating modules into a hierarchy. The architecture then allows ranking behaviors to modify the inputs and outputs of lower level behaviors. Traditionally, this override behavior is managed by a series of static subsumption functions, which take in inputs from ranking and subservient behaviors and allow the output of the dominate behavior to choose how the inputs are blended or overridden. This enforces a rigid hierarchy in which each behavior is always either subservient or dominate over another behavior. Athena uses the real time processor to create a more flexible hierarchy.

Athena's subsumption functions blend five different behaviors: avoid obstacles, avoid lines, move towards goal point, prefer similar heading, and avoid explored areas. In different contexts, Athena's architecture will allow a behavior to override the same behavior it submitted to just moments ago. This is based off of an assertiveness asserted by each behavior. If the Hough transform of the image processing has strong correlation, it will assert its goal behavior more. Similarly, if there is a high standard deviation to the prefer similar heading's trailing average, it is less likely to assert dominance over other behaviors. This flexible hierarchy allows us to leverage the advantages of subsumption while also empowering very different behavior sets for the courses many contexts (solid versus dashed lines or sparse versus dense obstacle environments).

5.5 Software Innovation

We believe that the team's software strategy is novel to the Intelligent Ground Vehicle's Competition. While in the past other successful teams have used subsumption to moderate between a smaller set of behaviors, [5] we believe that no one has been able to implement behavior generation modules as truly parallel, independent processes as we have on Athena's FPGA. In fact, with the exception of an implementation of a three behavior FPGA simulation done by researchers at Chulalongkorn University [6] and a RC car controlled by a 256 gate programmable logic chip created by Rodney Brooks [4] it appears as though this approach is relatively novel in the greater robotics community.

However, innovation requires more than novelty; an innovative approach must be selected on a belief that it will yield better outcomes. We believe that a reactive subsumption behavior is in its own right well suited to the IGVC. The inability to store prior knowledge means that a robot's world models must be constructed while the vehicle explores its environment. While not an intractably hard problem, implementing SLAM on an IGVC vehicle would prove very challenging. The high hardware cost (both in sensors and computation) and extensive development time to create a vehicle capable of constructing such a world model again seems to indicate that an abstract model based approach may not be appropriate. This is particularly true given the relatively short distances traveled by IGVC vehicles. This short distance means that optimal path solutions yield little improvement from simpler reactive navigation behaviors. However, just as we must select the proper computational environment to run algorithms, we also must select algorithms that are well suited to our robot's environment.

This is the basis upon which our team was drawn to subsumption. However, our software strategy does not simply implement a previously postulated architecture. It solves two key problems of subsumption by creating truly parallel behavior modules and allowing for more flexible blending of these modules. Even though subsumption architectures were theorized as a composition of parallel, finite state machine modeled behaviors, this was not how they were initially implemented. In practice, Rodney Brooks robots were frequently controlled by single core Lisp machines which simulated finite state machines [2]. Having to be tethered to a computer presents obvious complications for the practical operation of such robots. Not only does our software strategy break that cord, it also enables behaviors to be implemented in parallel as true state machines. In addition to helping advance the theorized parallelism of subsumptive architectures, our software allows for flexible adaptation to the IGVC courses differing environments, rather than following Brooks rigid architecture.

6 Performance

Athena is based on the Chimp™ platform by Doran Electric Vehicles. This personal transportation vehicle (PTV) is designed to carry a 300 pound rider at 11 mph on a flat surface. The additional electric and mechanical systems on Athena total less than 100 pounds. Therefore, we expect performance well within the base vehicle's specifications. In our testing, we have never been limited by the overall power and speed of the base vehicle.

Our 2009 competition entry, Brian, used the same base platform and fared quite well in the Autonomous Challenge. It was, however, limited by its effective speed and ability to distinguish gaps in lines. Through a better motor control algorithm and enhanced machine vision capabilities, we were able to eliminate some of these downfalls. At competition, Brian showed choppy motion and uneven travel speeds that resulted in an average rate of 0.5 mph. It also tended to oscillate between lines, a known limitation of potential field algorithms in narrow corridors [7]. Athena, on the other hand, has a more consistent and fluid motion as defined by our drivable paths algorithm. Initial tests show an average operating speed of 3 mph. We have also made several improvements to the machine vision capabilities of our robot. Athena uses a wide angle lens that provides a visible range of 25 ft and a usable horizontal range of 30 ft. This allows us to always have lines in view from which we can extract a desired travel direction. This increased viewable area along with our advanced filters should allow us to quickly and easily identify lines and desired paths through the course.

Metric	Brian (2009 entry) Competition Performance	Athena (2010 entry) Predicted	Actual
Cognition Performance Metrics			
Line Avoidance Reaction Time	0.1 s	0.067 s	0.067 s
Obstacle Avoidance Reaction Time	0.1 s	0.014 s	0.014 s
Localization Update Rate	2 Hz	50 Hz	50 Hz
Sensor Performance Metrics			
LIDAR 2D Update Rate	10 Hz	70 Hz	70 Hz
LIDAR 3D Update Rate	N/A	15 Hz	15 Hz
Obstacle Detection Range	9 ft	22 ft	22 ft
Line Detection Range	12 ft	23 ft	25 ft
Heading Update Rate	2 Hz	50 Hz	50 Hz
GPS Watch Circle	3 m	3 m	3 m
Vehicle Performance Metrics			
Top Speed	4.5 mph	4.5 mph	4.5 mph
Average Operating Speed	0.5 mph	3 mph	3 mph
Ramp Climbing Ability	15% grade	15% grade	15% grade
Running Battery Life	1.8 h	1.8 h	1.9 h
Standby Battery Life	10 h	14 h	16 h

Table 2: Performance Metrics

We also expect major improvements in the Navigation Challenge. In 2009, we only reached a single waypoint due to our inability to establish a reliable heading. Athena incorporates a new VectorNav Attitude Heading Reference System that provides filtered heading data for the robot. With this information, we can now accurately calculate our desired path to the waypoints.

Athena is capable of real-time obstacle avoidance, meaning that it can avoid obstacles even in a dynamic environment. The LIDAR can detect obstacles at a distance of 22 ft at a rate of 70 Hz. Because the course only includes static obstacles, we have opted to take a 3D scan of the environment by oscillating the LIDAR head. These scans are accomplished at 15 Hz, which is still faster than the update rate on our 2009 vehicle.

7 Conclusion

Athena represents a clear and already proven improvement over the team's 2009 entry. Athena's novel use of FPGAs and hardware abstraction has allowed the Olin team to reduce Athena's line avoidance reaction time by a third and the obstacle avoidance reaction time by an order of magnitude. Higher processing speeds coupled with LIDAR pitch control has more than doubled Athena's effective obstacle avoidance range to twenty two feet. Similar improvements in lens selection and image processing have increased the line detection range to twenty feet. The result is a vehicle whose 3 mile per hour average speed has far surpassed our original goal. We believe that Athena's architectures also represent careful systems design, a innovative use of alternative computing fabrics, and novel approach to the IGVC. However, the ultimate proof of these methods will be her performance at competition. The Olin College Ground Robotics Engineering group proudly presents the college's 2010 IGVC entry, Athena.

References

[1] Rodney A. Brooks. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, RA-2, 1986.

[2] Rodney A. Brooks. A Hardware Retargetable Distributed Layered Architecture for Mobile Robot Control. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages pp. 106–110, Raleigh, NC, 1987.

[3] Rodney A Brooks. Planning Is Just A Way Of Avoiding Figuring Out What To Do Next. Cambridge, MA, September 1987. MIT Artificial Intelligence Laboratory.

[4] Rodney A Brooks. Elephants Don't Play Chess. Cambridge, MA, 1990. MIT Artificial Intelligence Laboratory.

[5] Andrew Fife, Jason Fountain, Andrew Livick, and Peter King. Virginia Tech Autonomous Vehicle Team Presents: Johnny-5, 2007.

[6] Prabhas Kongmunvattana, Angkul; Chongstitvatana. A FPGA-based Behavioral Control System for a Mobile Robot. In *IEEE Asia-Pacific Conference on Circuits and Systems*, Chiangmai, Thailand, 1998.

[7] Y. Koren and J. Borenstein. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 1398–1404, Sacramento, California, 1991.