Oakland University Proudly Presents:

BOTZILLA

IGVC 2011

Student Members:

Micho Radovnikovich – PhD, Systems Engineering

Lincoln Lorenz – PhD, Electrical Engineering

Kirk McGuire –Senior, Electrical Engineering

Kevin Hallenbeck – Sophomore, Computer Engineering

Scott Marginet – Senior, Mechanical Engineering

Mike Truitt – Junior, Electrical Engineering

Paul Abdo – Senior, Electrical Engineering

Todd Perkins – M.S., Electrical Engineering

Pranab Bade – Senior, Electrical Engineering

Varun Vummaneni – M.S., Electrical Engineering

Renee' Kolleth – Senior, Biology

Kiran Iyengar – M.S., Systems Engineering

Walid Elsady – M.S., Electrical Engineering

Honorary Alumni Members:

Steve Grzebyk, Pavan Vempaty

BATTERY GIANT

Oakland UNIVERSITY

Dataspeed Inc.
Making Mobile Easier™

SCT
Sankuer Composite Technologies Inc.
A Fusion of Craftsmanship and Technology

I certify that the engineering design present in this vehicle is significant and equivalent to work that would satisfy the requirements of a senior design or graduate project course.

Signed,

_____, **Dr. Ka C. Cheok,** Faculty Advisor

1 Introduction

Oakland University is proud to enter Botzilla into the 19th annual Intelligent Ground Vehicle Competition! Botzilla is a very rugged platform, featuring four-wheel drive and double Ackermann steering control (Section 3.2). The four-wheel drive allows it to handle any terrain encountered at IGVC with ease, and the double Ackermann steering control allows for strafing maneuvers impossible to achieve with conventional skid steer systems, while also being able to make tighter turns than a single Ackermann setup would be capable of.

In the same way Botzilla's physical platform is more robust and reliable than previous years' entries, the software platform running on Botzilla is also much faster and more reliable. This year, all high-level software has shifted from a Windows computer running Matlab to an Ubuntu computer running Robot Operating System (Section 5.3), and all low-level software and sensor interfacing has shifted from microcontrollers to FPGAs (Section 5.1). In addition, the obstacle detection system solely uses a monocular camera system, without the need of a lidar.

1.1 Team Organization

Oakland University has seen an increase in membership since last year, thanks to the efforts of the team to increase its visibility on campus. The team consists of 15 members, with a composition of about 50% graduate students and 50% undergraduate students, majoring in mechanical, electrical and computer engineering. Figure 1 shows the organization of the team and how the responsibility is distributed. Each major sub-system of Botzilla has its own captain responsible for taking the lead role in its development. The captains were in charge of managing the rest of the group and guiding them towards completion of their respective components. It is estimated that about 1500 person-hours were invested in the development of Botzilla.

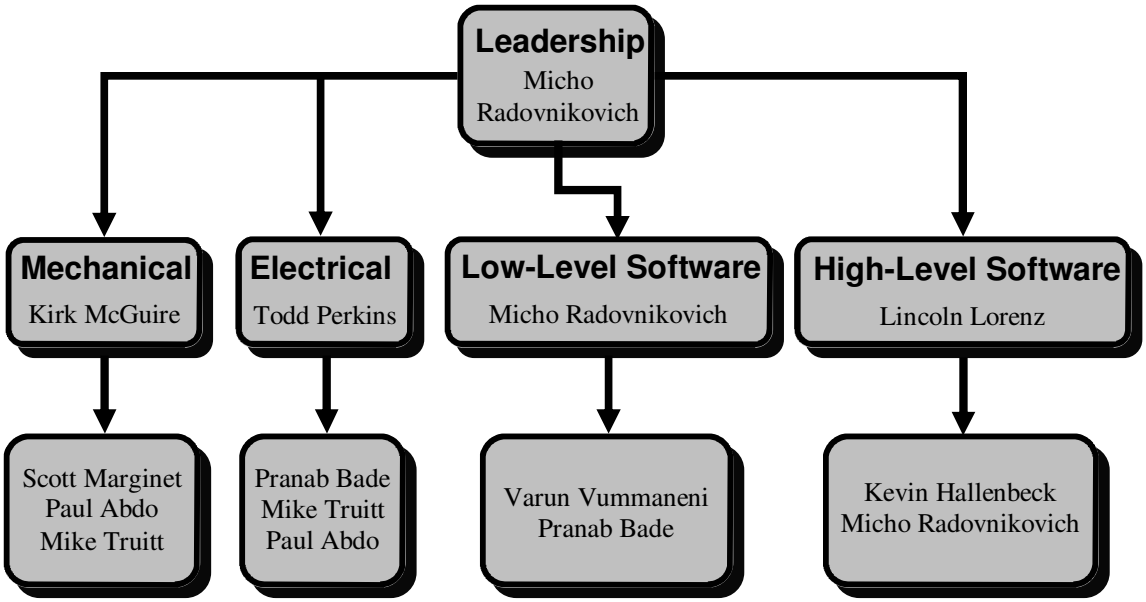

Figure 1: Chart outlining the team's organization.

1.2 Design Process

A classic 'V-Model' design process was followed to develop Botzilla, shown in Figure 2. After deciding on the concept of Botzilla, the requirements to achieve it were defined and a design was formed. After implementing the design and integrating the various components, a rigorous test cycle began, where consistent failure points were identified and rectified through minor adjustments or larger design modifications.

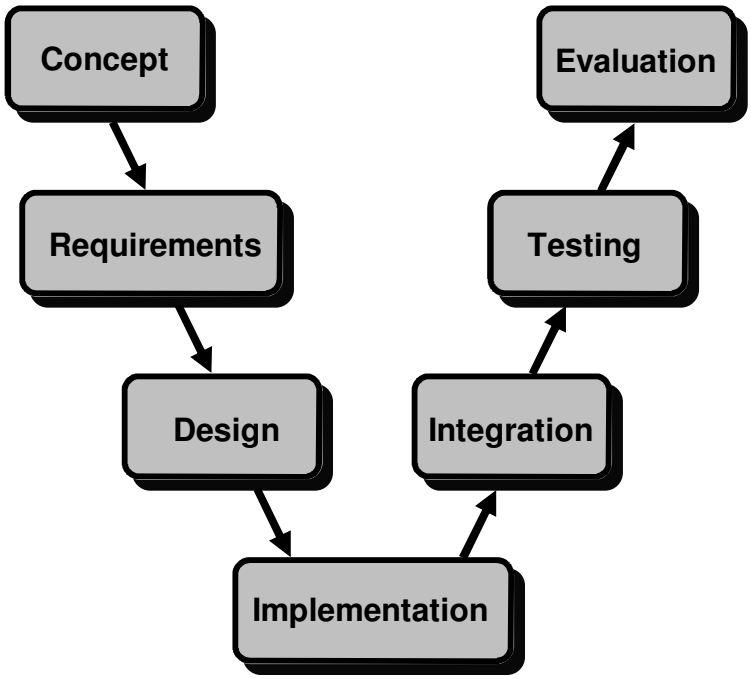

Figure 2: V-Model design process.

2 Innovations

Below is a list of the main innovative aspects of Botzilla's design. They are listed in brief here, and discussed in more detail in their respective sections.

- Stand-alone monocular vision-based obstacle detection (Section 5.2)

- Robot Operating System (ROS) software integration (Section 5.3)
- Mapping of the robot's environment (Section 6.1)
- D* Map-based path planning algorithm (Section 6.2)
- Custom H-Bridge design and fabrication (Section 4.1)
- Ackermann steering actuation on both front and rear wheels (Section 3.2)
- FPGA sensor data gathering and drive control system (Section 5.1)

3 Mechanical Design

3.1 Chassis

The chassis of the robot is a simple ladder style design. This allowed for easy assembly and a rugged structure. The payload carrier is under the main chassis, allowing the quick installation and removal of the competition payload. A significant portion of Botzilla is built from aluminum in order provide weight reductions. The use of steel was kept to a minimum; used only when strength or ease of fabrication dictated the need for it. The use of polycarbonate Plexiglas and hollowing of the driveshafts is another attempt to reduce the overall weight of Botzilla. The use of lightweight tubing for the camera mount also helped to reduce the mass of the robot. Further weight reduction is planned for the near future in the area of hubs and further thinning of the driveshafts. All of this was done with minimal weight and dependability as the main design criteria.

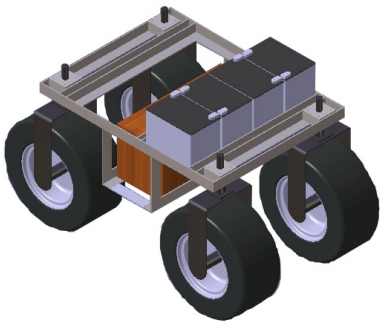

Figure 3: Botzilla's chassis

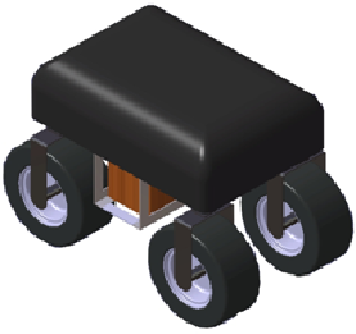

Figure 4: Chassis with cover

3.2 Drive Train

There are two significant design features implemented in the chassis of Botzilla. The robot features all-wheel drive and double Ackermann steering, where the front and rear wheels are independently rotated by a linear actuator. The use of four-wheel drive allows for consistent locomotion while traversing irregular terrain. The four-wheel drive also allows the robot to ascend steep inclines with relative ease. This design allows the use of a relatively closed pattern tire while still providing excellent traction under adverse conditions, thereby minimizing the impact on the environment.

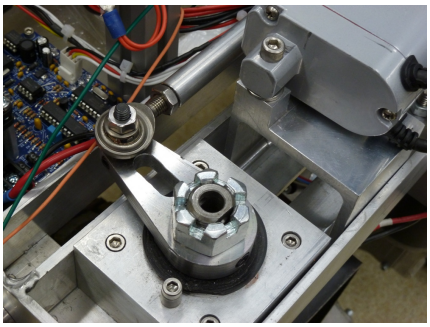

Figure 5: Steering mechanism

Figure 6: Drive axle

The double Ackermann steering is similar to that of a monster truck. It allows both ends to be turned either independently or simultaneously. A linear actuator drives the left side which is connected to the opposite side through a tie rod link. The link has opposing threads on the ends allowing toe adjustments to be made. The double Ackermann steering allows for decent mobility in confined quarters while maintaining control of Botzilla under extreme conditions.

4 Electrical Design

4.1 H-Bridges

Botzilla's H-bridges are completely custom-designed PCBs. Based on past experience with other H-bridges such as IFI's Victor series, it was desired to use an H-bridge that is more flexible, robust, and capable of chopping the motor power at a much higher frequency. Key features of the H-bridges are:

- Reverse battery protection
- On-board fuses
- Fan control
- Current sense
- Temperature monitoring
- Serviceable components

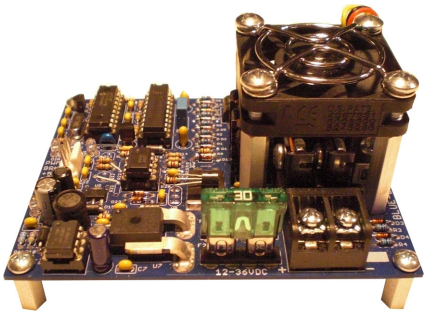

Figure 7: Custom H-bridge

The reverse battery protection utilizes a P-channel 4 mΩ low R_{ds}-on power MOSFET. The current sense uses a Hall-effect sensor to monitor current without having the resistance drop of a typical current sense resistor arrangement. Temperature monitoring ensures that the H-bridge is kept within a safe thermal window, and allows dynamic fan control to save power. The H-bridge itself is composed of N-channel MOSFETs, with each leg paralleled to further reduce R_{ds}-on. A conventional single-channel PWM signal controls the speed and direction of the H-bridge output.

4.2 Sensors

Botzilla is equipped with an array of sensors that allow it to detect obstacles around it, compute its location, heading and speed, and be operated in a safe and reliable manner. The sensor array consists of:

- NovaTel FlexG2-Star GPS receiver, which provides accurate GPS fixes within 1 meter
- Honeywell HMC5843 digital compass, which provides raw three-axis magnetometer data from which tilt-corrected heading measurements can be made
- Analog Devices ADIS16350 six-axis IMU
- U.S. Digital E3 optical wheel encoders, providing wheel speed data for closed loop drive control, as well as odometry for the mapping algorithm
- DX5E wireless aircraft joystick, whose standard RC signals are used for embedded manual control, as well as a wireless E-Stop.

4.3 Computing Hardware

Botzilla's computing system is distributed between a Dell Latitude E5410 laptop running Ubuntu and ROS (Section 5.3), and a Xilinx Spartan3E FPGA, employing a MicroBlaze soft processor and custom hardware written in VHDL (Section 5.1). The two communicate with each other over a single RS232 link, through which the FPGA transmits sensor data to the computer, and the computer transmits motor control messages to the FPGA.

For the JAUS challenge, an external computer communicates with the Ubuntu laptop over Wi-Fi, and relays the processed JAUS commands. The Ubuntu laptop also responds back with the data reports necessary to satisfy the requirements of the JAUS challenge.

4.4 Power Distribution

Botzilla's 24V power source is derived from four 12V AGM lead acid batteries, arranged with two parallel sets in series, with a total charge capacity of about 60 AH. The 24 volts are wired directly to

the H-bridges, which then pulse power to the motors according to the PWM control signals coming from the FPGA. The FPGA board takes a 12 volt power input, coming from a switching DC-DC step down converter. The FPGA board has its own on-board switching regulator that powers the small sensors connected to it, as well as the Spartan3E itself.

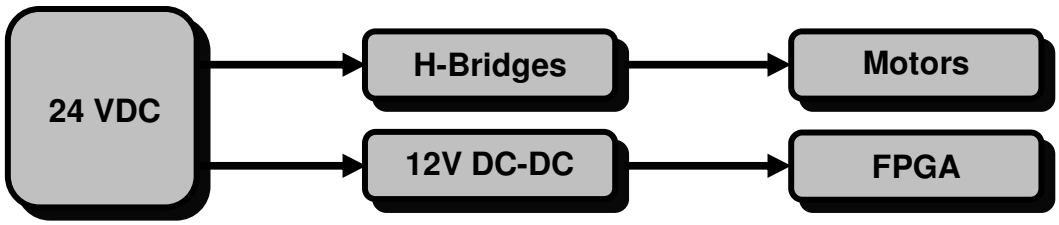

Figure 8: High-level diagram of how power is distributed to the built-in components

4.5 Safety Considerations

Botzilla is a very powerful vehicle with lots of torque and is capable of slightly over 10 mph. As such, it has potential to be dangerous. Therefore, many precautions were taken into account when designing the emergency stop system, where many layers of protection exist. Besides the conventional turn-to-release E-stop switch, the DX5E joystick is capable of disabling the motor output wirelessly from up to a couple hundred feet away. In addition, the drive control system automatically turns off the motors if it fails to receive commands from the computer after a short time, and a software-handled joystick functions as a dead-man switch for the vehicle when in testing mode.

5 Software Design

This section describes the computing system used on Botzilla. As described in Section 4.3, all computation is distributed between a laptop and a FPGA. A diagram of how the major software elements interact with each other is shown in Figure 10, and the subsequent sections discuss them in detail.

5.1 Low-Level Hardware/Software

All of Botzilla's low-level functionality is implemented on a single 1.2 million gate-equivalent Spartan 3E FPGA, utilizing custom hardware components developed in VHDL, and a Xilinx MicroBlaze soft processor running hand-written C code. The system was designed to process all the data from the sensors listed in Section 4.2, extract the measurements, assemble the data into a convenient packet and transmit it all to the computer. At the same time, the drive control algorithms interpret high-level vehicle motion commands from the computer and apply closed-loop steering control to the motors. Figure 9 shows a block diagram of the FPGA architecture.

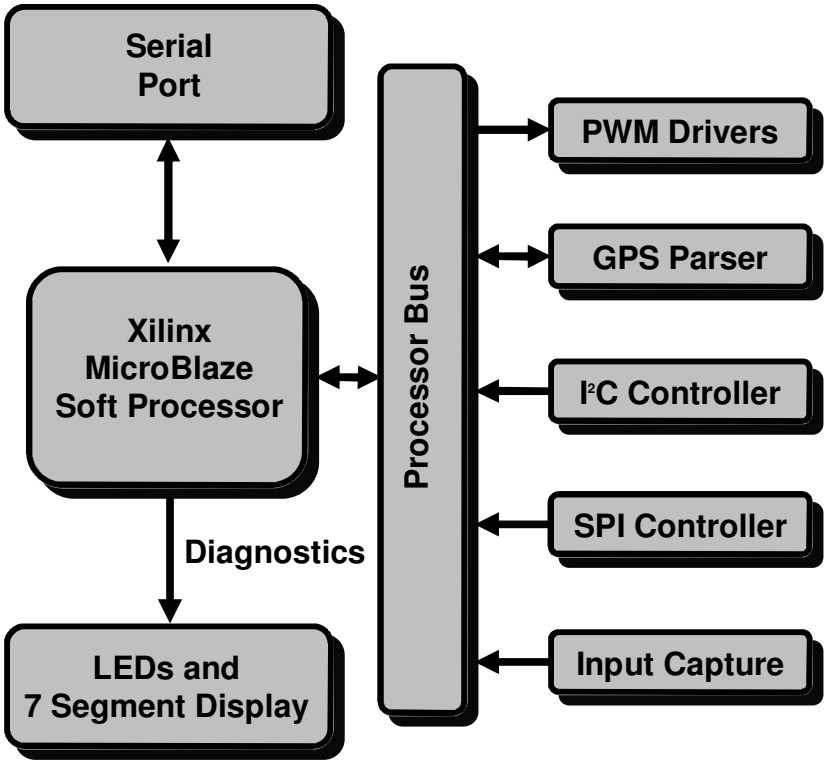

Figure 9: Custom FPGA Architecture

- Data Gathering

Custom hardware was developed in VHDL to gather the data from all the sensors. These hardware components consist of:

- Six parallel PWM channels, capable of independent control of each motor
- One NMEA GPGGA message parsing component that extracts the vehicle's current GPS coordinates from a continuous stream of ASCII data from the receiver
- One I^2C controller to request data from the compass and extract the heading data from the resulting transmissions
- Two SPI controllers used to interface to the IMU and external A/D hardware module

- ❖ Eight parallel input capture channels to grab quadrature data from the four optical wheel encoders

All the custom hardware is interfaced to the MicroBlaze processor by means of a 32-bit wide processor bus, clocked at 50 MHz. The processor addresses each hardware component in sequence and refreshes its measurement from each sensor. At a rate of 20 Hz, it packages all the data into a 22-byte packet and transmits it to the computer.

- ➢ Drive Control

Botzilla's drive system consists of four DC motors rotating the wheels of the vehicle, and two DC linear actuators controlling the steering mechanism on the front and rear wheels. Making use of the wheel encoder data being gathered by the FPGA, the MicroBlaze processor applies PI control to the wheel motors to match speed commands from the computer. Likewise, the potentiometers on the steering motors are used by the MicroBlaze processor to apply a lead-lag compensator to track angular position commands from the computer.

5.2 High-Level Software

- ➢ Lane and General Obstacle Detection

Detecting obstacles and lines in the robot's view requires a robust vision system able to differentiate lines and obstacles from the grass background. In direct contrast to making a robust system, the robot's vision system also needs to be responsive enough to react to the changing scenery as the robot navigates through the course. Any algorithms used will have to satisfy both constraints.

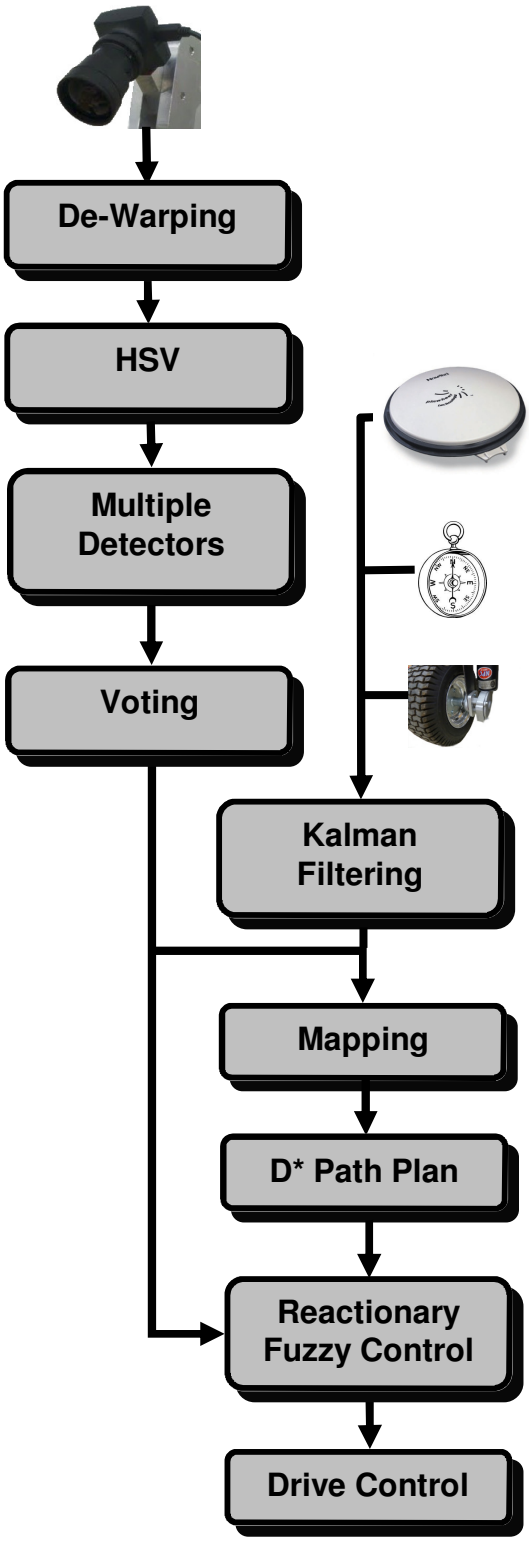

Figure 10: Block diagram of software component interaction

To satisfy the requirements of robust differentiation and good responsiveness a system was developed that uses multiple fast algorithms that vote on whether or not an obstacle is detected. The system consists of three stages: pre-processing, detection, and voting. Pre-processing smoothes the image and converts it to the HSV domain. In the HSV domain, shadows are present mainly in the value component, setting this to a fixed number allows object detection algorithms to be shadow agnostic. The second stage, detection, is performed using multiple simple algorithms which later are combined to vote on the presence of obstacles on a per-pixel basis.

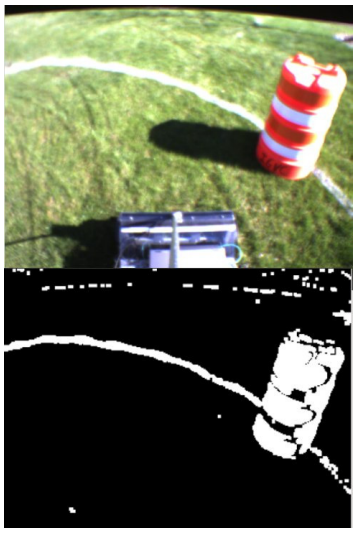

Figure 11: Example of voter-based obstacle detection

Each algorithm outputs a binary image that is its vote for each pixel on the pixel grid. Algorithms that are used consist of standard implementations of Canny edge detection, auto thresholding, frame differencing, and a custom developed median windowing algorithm. The final stage, voting, takes each voting image and determines the majority. During testing the vision proved reliable enough to replace the lidar functionality entirely.

- Image Transformation

In order to create a monocular vision system capable of acting as a suitable lidar replacement, the distance perception of such a system must be very accurate. A very general system to do these transformations was formed, and only assumes that the objects being detected are on a flat surface, which is a realistic assumption for situations like IGVC.

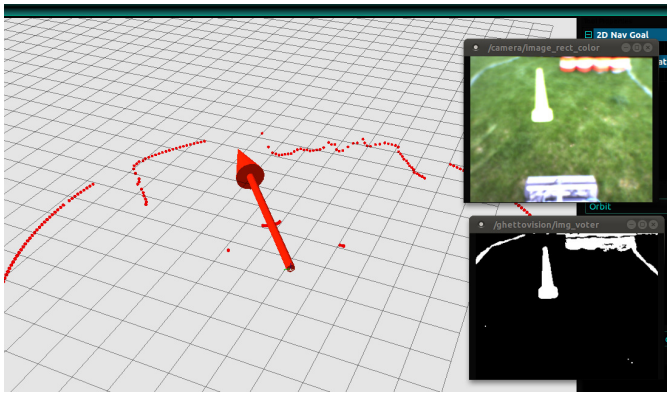

Figure 12: ROS visualization of the imitated lidar scan

By making some simple calibration measurements, such as the height and pitch of the camera, etc., a transformation is defined to project individual pixels into an azimuth and zenith rotational axis offset from the center of the camera. A non-linear trigonometric transformation function is then used to project the angle offsets down onto the ground plane. These points are then rotated and shifted into the vehicle's coordinate frame to simulate Cartesian obstacle location data obtained from a lidar scan. An illustration of this imitated lidar scan is shown in Figure 12. More details about the transformation can be found in [2].

- Kalman Filter Sensor Fusion

The role of the Kalman Filter is to input the raw GPS fix, compass and wheel encoder data and fuse the readings together to provide a better estimate of the vehicle's position and heading angle in the global frame. Sensor fusion with a Kalman Filter is a very powerful technique, since it is capable of modeling the variance of measurement signals and then deciding how much to "trust" each sensor to come up with a very nice estimate. Specifically, the Kalman Filter on Botzilla is designed to:

- Use GPS course-over-ground heading reading to correct the bias in the compass heading measurement
- Use compass heading measurement to eliminate the meandering GPS heading when traveling slowly or during a GPS outage
- Use wheel encoder measurements to provide odometry that helps filter wandering GPS data

5.3 ROS Integration

Botzilla's high level software system uses Robot Operating System (ROS) as its base interfacing layer. ROS provides many features that aided efficient development of an IGVC ready robot including: pre-built robotics libraries, standard sensor drivers, support for simple message passing between processes, debugging and system visualization facilities, and dynamic reconfiguration. Additionally, the wealth of documentation for different features in ROS allowed the team to understand and implement non-trivial concepts quickly.

5.4 Logging

One of the many features that ROS has built in software libraries for is a logging system. This allows the team to log any of the data being passed in the system, including camera, GPS, and other sensor data. In typical use, the robot is human driven along a course similar to that of IGVC while recording data. The "raw" data is later played back using another component of the logging system, which emulates the data stream of real sensors. This enables the software team to test new algorithms and gauge their effect on the full system while sitting at their desks.

5.5 JAUS

The JAUS Protocol was implemented in Microsoft Visual Basic .NET, where a class was written for each JAUS message. In each class, parameters can be set, an array of bytes can be parsed, and the message can be converted into a byte array. A class was written for each JAUS service as well. Services that require inherits are passed a handle to the required services on creation.

To test the JAUS implementation, a separate COP simulator was programmed using the same language and classes. The COP simulator can manually send all of the messages required for the competition and display the messages received. The COP simulator also has a map on which way-points can be set and graphed, and the position history of the robot can be plotted. All raw packets are saved to a debug window and a log file to help diagnose problems.

6 Artificial Intelligence

6.1 Mapping

Botzilla uses mapping to recall where it has been so that it can decide its route in the autonomous challenge. The mapping algorithm used is provided in the karto package from ROS. The package uses a technique called sparse bundle adjustment for fast and accurate determination of a map from scan and localization data [3]. The algorithm is capable of loop closure, which allows the repeated paths to be detected as the same path to improve path detection. This will be used to remember the obstacles encountered on previous runs for more accurate path planning on consequent runs.

6.2 Path Planning

The path planning system integrates a short-term reactionary obstacle avoidance system based on a fuzzy rule-based algorithm, and a long-term deliberate obstacle avoidance system based on the map output from Section 6.1.

- Reactionary Fuzzy Logic Controller

The role of the reactionary component of the path planning system is to provide effective first-time navigation of an unknown environment toward the goal point, and to augment the map-based path planning algorithm and adapt to uncertainties and error in the map.

This reactionary system is implemented using the very simple, yet incredibly powerful technique of fuzzy logic control. A set of qualitative rules is formed based on how the vehicle should respond in certain situations. By assigning quantitative values to the relative degree to which the qualitative statements describe the current situation, a real-world decision can be arrived at to best satisfy the rules.

The goal point, as well as the imitated lidar scan (Section 5.2) are input to the fuzzy system, where the location of the goal relative to the vehicle and the immediate occupancy of the vehicle's surroundings govern both the steering and velocity through the set of fuzzy rules.

- Deliberate Map Navigation

To compute an optimal path through the map of the environment, a D* algorithm ROS package is employed, developed by Michael Otte of the University of Colorado [1]. The package takes in a goal point, standard occupancy grid message, and a pose estimate of the robot, and computes an optimal path through the given map. This path is broadcasted to the rest of the system via a standard navigation stack message.

However, to account for error in the map and other uncertainties, proper combination with the reactionary system is necessary. To solve this, critical points of the D* path are chosen as intermediate waypoints to the reactionary system.

7 Performance Predictions

- Maximum Speed

Botzilla's motors spin at 157 RPM at nominal load, so combined with 15 inch diameter wheels, the resulting maximum speed is 10.3 mph. This estimate correlates with the observed performance.

- Ramp Climbing Ability

At nominal load, the drive motors provide 101 in-lbs of torque. Assuming a realistic vehicle weight of 175 lbs, this corresponds to a max slope of 18 degrees. However, experiments have shown that Botzilla can handle much steeper slopes, up to about 30 degrees, although not at the nominal load of the motors.

- Reaction Time

The ROS system running on the laptop gathers new sensor readings from the FPGA at 20Hz, and processes camera frames and extracts obstacle locations at 15 frames per second. The artificial intelligence systems were designed to be able to handle this frequency easily, thereby allowing the robot to make new decisions at the slowest sensor sampling rate 15 Hz = 66.7 ms.

- Battery Life

The AGM batteries on Botzilla provide a total of 60 AH. The sensors and FPGA consume 2 amps. Experiments have shown that the steering motor current draw averages 3 amps under normal operating conditions, and that the drive motors consume a total of 25 amps maximum in a grass environment typically encountered at IGVC. Based on these observations, total battery life is approximately 2 hours.

- Obstacle Detection Range

With the configuration of the camera, obstacles can be detected up to a maximum of 24 feet away, although it was experimentally determined that vision measurement data becomes most reliable within 17 feet. The camera configuration also makes the front of the vehicle visible. This allows for the vision system to detect lines and obstacles up to 3 feet to either side of the front wheels, thereby minimizing the size of critical blind spots.

- GPS Accuracy

Under normal conditions, the Novatel FlexPackG2-Star GPS receiver is accurate to within 1 meter, which is enough positional accuracy to reach the small waypoints on the GPS Challenge course. However, since the mapping algorithm (Section 6.1) is dependent on both the position and more noisy ground heading estimate of the GPS, the Kalman Filter (Section 5.2) is used to fuse the GPS with the other sensors.

8 Cost Analysis

A breakdown of the cost of the components on Botzilla is shown in Table 1.

Table 1: Cost breakdown of components

Item	Cost	Cost to Team
FlexG2-Star GPS Unit	$1,000	$1,000
SICK LMS-200 Lidar	$6,000	$0[1]
Dell Latitude E5410 Laptop	$800	$800
Analog Devices ADIS16350 IMU	$550	$0[2]
Other Sensors	$400	$400
(4) 12V AGM Batteries	$375	$0[3]
Power Electronics	$600	$600
Fiberglass Body	$2,000	$0[4]
Electromechanical Components	$1,000	$1,000
Frame Materials	$1,000	$1,000
Total	**$14,225**	**$5,300**

[1] Re-used from previous vehicle
[2] Donated from Dataspeed, Inc.
[3] Part of a larger donation from Battery Giant, Rochester Hills
[4] Custom-made and donated by Sankuer Composite Technologies

9 Conclusion

Botzilla has proven to be very rugged, efficient and reliable, performing well while driving on any kind of terrain. The new artificial intelligence design shows promising results, and the Oakland University team has great confidence going into this year's competition.

Acknowledgements

The Oakland Robotics Association would like to thank several people and organizations. The team thanks Pete Taylor and Derrick Hurley for their insights during the fabrication phase. The engineering faculty of Oakland University engineering was invaluable, specifically KC Cheok, Louay Chamra, Len Brown, Markhanna McBurrows, Brenda Bond, in the development of Botzilla. The robot would not have been possible without the gracious support of the following sponsors, Battery Giant, Sankuer Technologies, Dataspeed, the SAFB and the Cheerful Dollar.

References

[1] Otte, M.W.; Grudic, G.; , "Extracting paths from fields built with linear interpolation," *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on* , vol., no., pp.4406-4413, 10-15 Oct. 2009
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5354775&isnumber=5353884

[2] M. Radovnikovich, P. K. Vempaty, and K. C. Cheok, "Auto-preview camera orientation for environment perception on a mobile robot," in SPIE Conference on Intelligent Robots and Computer Vision, 2010.

[3] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent, "Sparse pose adjustment for 2d mapping," in Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2010.