UNIVERSITY of TEXAS at AUSTIN

RAS

IEEE ROBOTICS and AUTOMATION SOCIETY

PRESENTS:

BlastosauRAS

Collaborative effort by

Richard McClellan · Nicolae Stiurca · Josh James

Alexander Böhm · Erica Taylor · Charlie Manion · Emily Chen · Kyle Miller

Faculty Advisor Statement
I certify that the engineering design of the new vehicle, BlastasauRAS, has been significant and each team member has earned or could have earned at least two semester hour credits for their work on this project.
Signed,
[illegible]

1.0 INTRODUCTION

This paper describes the University of Texas at Austin’s (UT-Austin) design of BlastasauRAS for the 18th annual Intelligent Ground Vehicle Competition (IGVC). This vehicle is a redesigned version of UT-Austin's 2009 robot, the RASmanian Devil. At the end of last year's competition, our team did a thorough evaluation of the things that RASmanian Devil did and didn't do well. The mechanical chassis and electrical system both exhibited a high degree of robustness throughout the entire competition, despite the pouring rain and muddy field, not causing any issues throughout the entire weekend. On the other hand, our custom software platform worked to an extent, but was not flexible or versatile enough for our needs. This year's IGVC team formulated a plan to use RASmanian Devil as a starting point to build BlastasauRAS. This year's software architecture is completely new and a lot of the electrical system is brand new, but we chose to stick with the same robust mechanical system as last year. BlastasauRAS was built by voluntary student members as a robotic platform with portable software and integrated commercial off-the-shelf (COTS) hardware to compete in the three competitive events at IGVC.

2.0 DESIGN PLANNING PROCESS

The process of designing a robotics system involves a careful balance of trade-offs with complexity and hardware. Our team set many deadlines and deliverables to schedule time for testing and re-evaluation. We set a timeline to achieve hardware, portable software drivers, functional electronics and sensor integration. After multiple progress reviews of the design at set time periods, we made appropriate modifications based on available monetary resources and testing time.

2.1 Sub-Teams

The team divided into mechanical, electrical and software section sub-teams. The mechanical group focused on drive dynamics and manufacturing simplicity. The electrical group focused on power management and interconnection between all electrical systems. The software team integrated together sensor data for path planning and navigation.

2.2 Design Sequence

Our design sequence begins with high level block diagram of sensors, electronics, software and mechanical ideas. Many design decisions involved choosing commercially-off-the-shelf (COTS) hardware to achieve a layer of abstraction from low level devices. After establishing a sufficient roadmap of the various sub-systems, the sequence begins an iterative loop of evaluating an integrated sub-section and redesigning when necessary.

Our approach differs from a more traditional planning which requires more simulation and analysis of each decision. Our organizational structure and time budget required us to accelerate the analysis phase. To demonstrate proof-of-concept, we create a series of rapid prototypes to eventually achieve a functional design to utilize in our final system. Our quick design strategy allows us to have a working robot throughout many design stages. A functional prototype improved our time in the overall system evaluation.

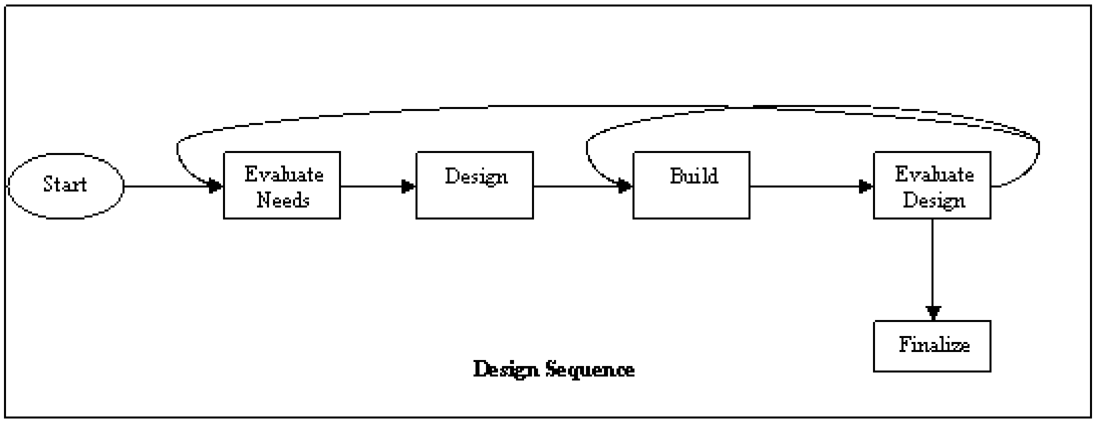


Figure 1: Process Flowchart

The UT-Austin team is composed of voluntary members of the IEEE Robotics & Automation Society called "RAS" in abbreviation. There are eight undergraduates participating in the contest. Overall work hours were approximated into the following breakdown below:

Student's Name	Academic Background	Level of Education	Focus of Contribution	Extent of Contribution
Richard McClellan	Mechanical Engr.	5th Year	Team Lead	250Hrs.
Nicu Stiurca	Computer Science	3rd Year	Software	300Hrs.
Josh James	Electrical Engr.	1st Year	Software	250Hrs.
Alex Böhm	Mechanical Engr.	3rd Year	Mechanical	50 Hrs.
Erica Taylor	Mechanical Engr.	3rd Year	Mechanical/ Report	50 Hrs.
Charlie Manion	Mechanical Engr.	2nd Year	Mechanical	20 Hrs.
Kyle Miller	Electrical Engr.	4th Year	JAUS	50 Hrs.
Emily Chen	Biomedical Engr.	4th Year	JAUS	20 Hrs.

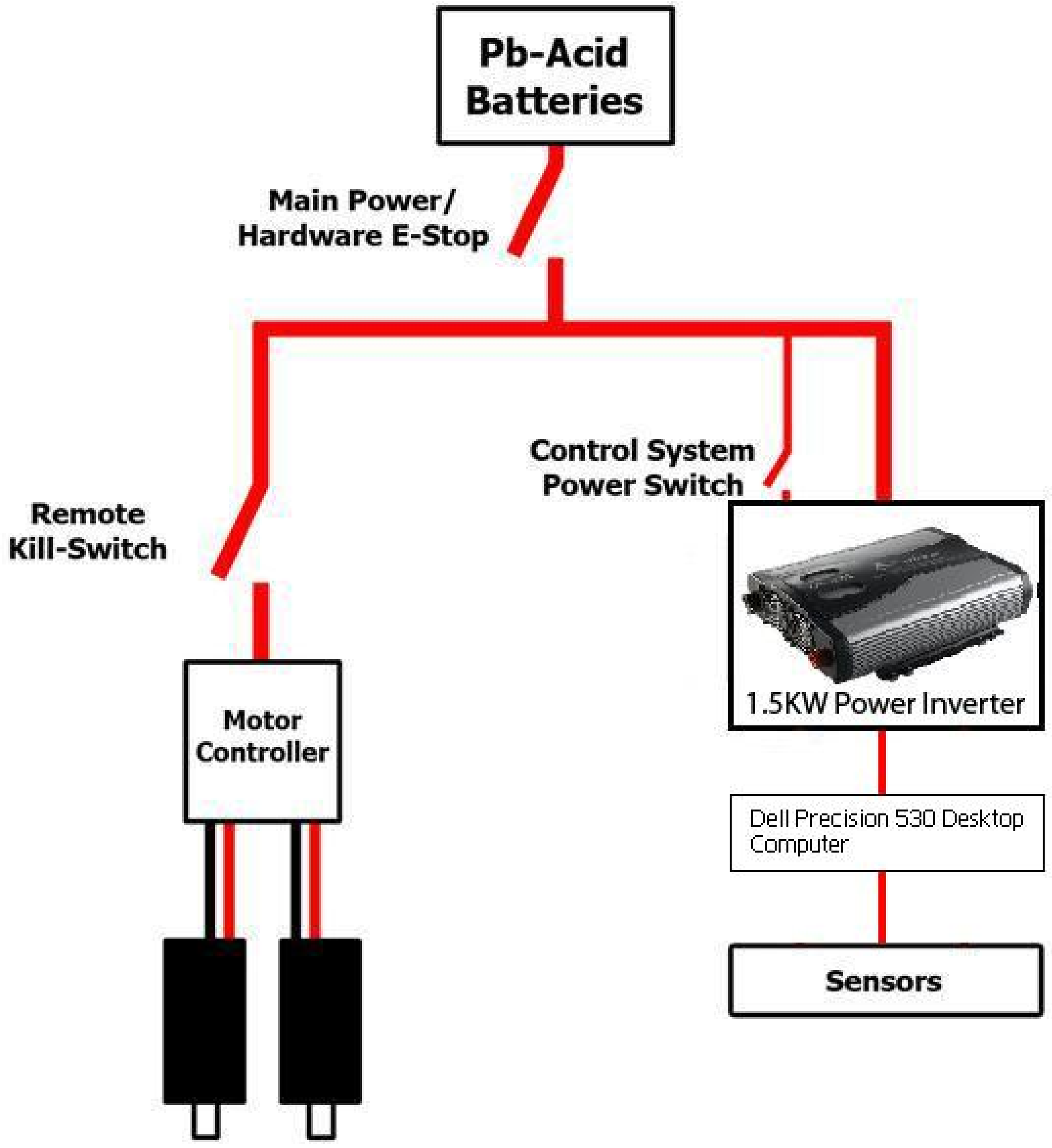
Table 1: Work Division Breakdown

3.0 ELECTRICAL SYSTEM

BlastasauRAS's electrical system can be broken up into four major systems: power, control, sensor, and communication systems. Each system is closely tied to the others so that we can have full control over how our robot performs.

3.1 Power System

Power management is a crucial element for a successful robot, and definitely not something that we want to fail at competition. BlastasauRAS is powered by single 12 V 17Ah lead acid battery, which has the advantage over Lithium batteries of being able to provide several hundred amps of current when needed. We have learned from past competitions that without some sort of surge protection or isolation between the control computer and the high current drive motors, the voltage to the controller can drop during current spikes and result in a reboot. We have solved this issue on several robots by simply using two batteries - one for the drive system and one for the controller. However, this is an inconvenience solution because we had to worry about charging two batteries. Last year, our controller was a small picoITX VIA C7 1GHz motherboard, which we were able to to power this controller as well as all the sensors using a 125W picoPSU power with built-in protection for high-current surges. This year, we made the decision to get more processing power by using a 700W desktop computer. While this has been a great asset from a software perspective, it has been a difficult challenge from the power side. We ended up using a 1.5KW power inverter with built-in UPS, so that if the supply from the primary robot battery drops out, the controller remains on. The UPS provides power for the on board computer, the laser rangefinder, digital compass, GPS, and other sensors. This actually provides a significant advantage over last year in that we can hot-swap robot batteries without shutting down our main control computer.   Figure 2 shows our power system block diagram.



Pb-Acid Batteries
Main Power/ Hardware E-Stop
Control System Power Switch
Remote Kill-Switch
1.5KW Power Inverter
Motor Controller
Dell Precision 530 Desktop Computer
Sensors

Figure 2: BlastasauRAS power diagram. Ground connections not shown.

The batteries directly power the motor controller and the power inverter. When the control system power switch is closed, the power inverter powers the on-board computer . All the robot's sensors draw their power from the on-board computer. The safety benefits of the three switches in the power system will further be discussed in section 7.0, Safety. We are using two Jaguar motor controllers to control our four DC drive motors.

Overall our power system is very simple. In earlier years we have had very complicated wiring, multiple separate battery sources, and many extra and unnecessary power switches. Having such a complicated power system lead to failures in our previous robots that prevented the robots from operating correctly, even before the code could be tested and debugged. Last year, we decided to simplify the power system, hoping to improve the reliability of our robot's power system. Due to the success of last year's power system, we uphold the same design principles in the current year, which will also improve our overall robot's success.

3.2 Controller

Last year's IGVC robot used VIA C7 1GHz processor. While lightweight and compact, our ultimate failure in competition was due to the fact that this controller was not powerful enough to keep up with the incoming sensor data from our USB webcam, and process everything. This year we have taken an opposite approach and are using a 2.2GHz Intel Xeon quad-core based desktop with 1GB of RAM. We use the 30GB solid state SATA drive from last year which is much more robust than a conventional hard drive as it is not subject to mechanical failure. This drastic processing horsepower upgrade is further justified in section 5.3, Performance Analysis.

Two Texas Instruments MDL-BDC24 Black Jaguar motor controllers are used for controlling the motors and reading the encoders [6]. The motor controllers are pre-programmed with velocity control capabilities and they can report wheel position thanks to the encoders. One of the motor controllers is connected to the desktop via the RS232 port on the motor controller/serial port on the desktop motherboard. The second motor controller is interconnected with the first via the CAN interface. In this configuration, the first motor controller relays messages to/from the second across the RS232 interface back to the motherboard. Commuciation with the microcontroller is handled according to the TI Jaguar communication spec.

Figure 3: Black Jaguar

3.3 Wireless Communication

To make our system easier to develop from remote workstations, and eliminate the need for a keyboard, mouse, and monitor connected to the robot, we wanted to have wireless communication. Initially, an EDIMAX EW-7711UAn Wireless USB Adapter was used to connect the desktop computer to a Linksys WRT54G router so that we could remotely connect to the robot from another computer connected to the same router. This solution worked, but was not ideal because the Wireless Adapter had a very limited range and when it disconnected from the router, it would not automatically reconnect. This required hooking up a monitor, mouse, and keyboard to the robot every time we needed to reconnect to the wireless network. To fix this issue, the robot now utilizes a Linksys WGA600N Wireless Gaming Adapter, connected directly to the computer's ethernet port. The gaming adapter is setup to automatically connect to the main router, and also provides a much greater range for the robot to travel which is very helpful.

3.4 Sensors

Choosing the right sensors was key to making a system that worked well as a whole. Significant effort was spent examining each datasheet of each sensor before its purchase to ensure that it would work the way we desired.

3.4.1 Laser Rangefinder

Last year our team chose to use a Hokuyo UHG-08LX scanning laser rangefinder for its 8m range, and relatively low price tag compared to the more common SICK rangefinders. We definitely had success with the rangefinder With 1mm trace resolution, a 270 degree field of view, 0.36 degree angular resolution, and 15Hz scan rate, it was determined to be suitable for obstacle avoidance in this competition [5].

3.4.2 Camera

Line detection is done with a 2MP Logitech Quickcam Pro 9000, capable of up to 1600x1200 pixels and up to 30 frames per second. It also features an autofocus system which eliminates all manual tuning, which was one issue we had in previous years [4].

Figure 4: Logitech Quickcam Pro 9000

3.4.3 Quadrature Encoders

Our instantaneous localization algorithm is done using odometry, which integrates values from two quadrature encoders (Grayhill 63R256), which are directly coupled to the left and right wheel shafts. The casing of the encoder is completed enclosed allowing it to work in any lighting without calibration, and also increasing durability. By looking at both the rising and falling edges of each signal, we can obtain 512 increments per revolution. With 16in diameter wheels, movement of the robot is measured with 2.5mm resolution [3].

3.4.4 Digital Compass

BlastasauRAS uses an OS5000-S Compass Module with Tilt Compensation which consists of a 3-axis magnetometer and a 3-axis accelerometer to accurately measure its heading angle. It features 24 bit A/D conversion on board, and an easy to use RS-232 communication interface, which allows us to easily connect it to the robot's computer. The compass provides data at rate of 75 Hz with a nominal heading accuracy of 0.5 degrees and a resolution of 0.1 degrees [2].

3.4.5 GPS

We have tested two different GPS receivers on the robot. Eventually we decided on the Garmin 72 GPS, which gives WAAS capability and a 3 meter accuracy [1]. We communicate with the GPS using a RS-232 serial link. The GPS has handheld features such as embedded button controls allowing the user to view all information on the GPS for quick debugging.

4.0 MECHANICAL SYSTEM

After competing in IGVC in 2009, our team was very proud of the reliability and robustness of our mechanical design and made the decision that it would be best to stick with what worked. Having a two wheel drive system with a caster helped minimize resistance to turning, thus making it easy to control and maneuver around obstacles. The one issue with last year's robot was that we did not have a very linear response from the Victor motor controllers that we were using, so this year we decided to use some more advanced Jaguar speed controllers which was discussed in Section 3.

Since the motors were much faster than we wanted the wheels to spin, gearboxes were required. We chose to use the gearboxes out of a Dewalt drill for their reliability, and coupled them with some 2.5" CIM motors, which we had available from a previous competition. Two Dewalt/CIM assemblies were made for each side, and were linked to the main driveshaft using sprocket and chain. Our motors free spin at 5310 rpm with 343 oz-in torque. This is reduced by 60:1 via the 12:1 gearbox and the 5:1 sprocket reduction. With two 16" driven trailer tires, we should move approximately 4.3 mph with a pushing force forward of 400 lbs. With these specifications we will meet the max speed limit of the vehicle with an ample amount of torque for climbing the fifteen percent grade ramp. In addition to the torque, the two 16" wheels have high traction to make the climb on the ramp easier.

The electronics mounting system was created to keep the center of gravity relatively low. As a result, the payload, speed controllers, breaker panel, microcontroller, and scanning laser range inders all had to be mounted relatively close to the bottom of the chassis to keep it from tipping over. The laptop was also mounted above the payload to allow for easy access while testing the system. From the base of the robot, two vertical aluminum square extrusions were mounted. The IMU, Camera, and GPS had to be mounted high for optimum sensor performance, so we put a tower above the two front drive wheels for the mounting of these three components.

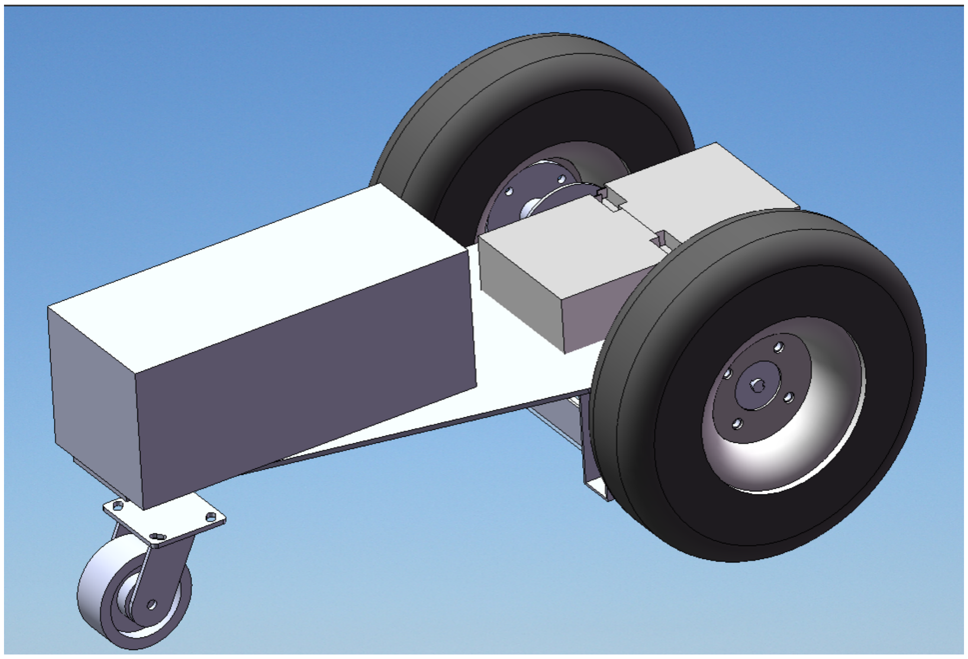


Figure 5: Solidworks System Level Model

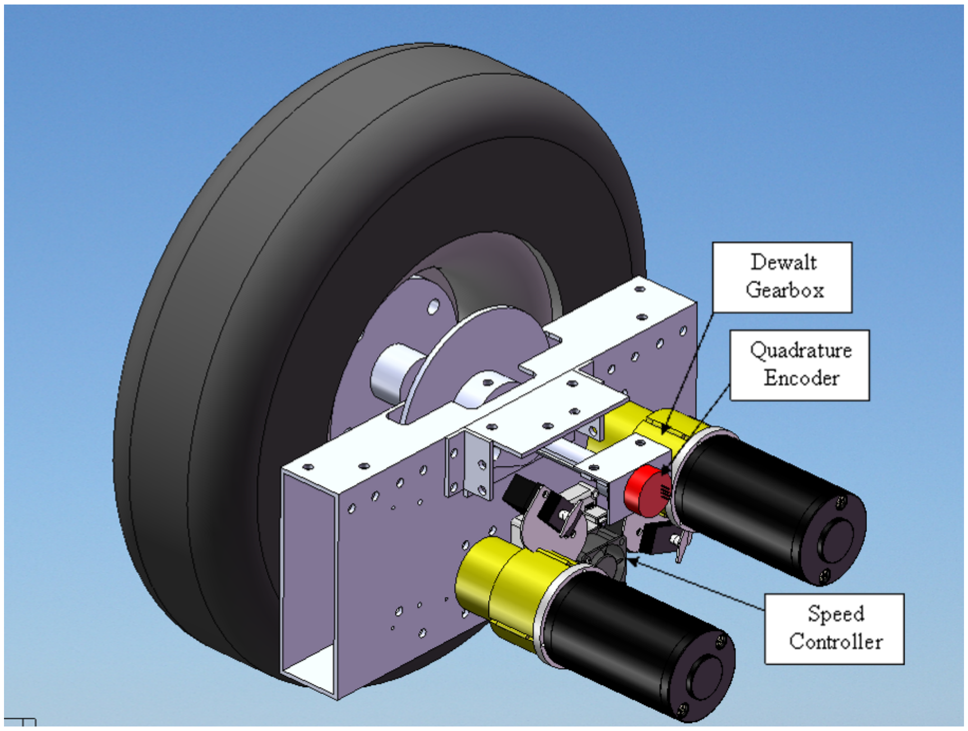


Figure 6: Solidworks Drivetrain System

5.0 SOFTWARE

Software is by far the area that has been most improved over last year's IGVC robot. Last year, we built the main software architecture around the OpenGL Utility Toolkit, or GLUT. This API was chosen because it fuses computer industry standard OpenGL with a mature, flexible, and simple framework for designing event-driven programs. However, this choice in architecture resulted in enormous amounts of time being spent on writing hardware drivers for all of our sensor drivers and custom development of a multi-threaded system, which ended up being very convoluted and difficult to modify. This year we made the decision to use an already existing software platform, Player/Stage. Player/Stage is an open source, language independent robotics platform, available for download by anyone through their website. It has been developed over the last decade by multiple contributors from the Stanford AI Lab, and the University of Southern California Robotics Research Lab. It also provides support for a lot of the hardware that we already had including our laser rangefinder and digital compass. It can also be programmed in several different languages including C++, Python, or Java. Lastly, it has the ability to create a simulated robot in a simulated environment, which has been very helpful for the development to develop while the mechanical team makes modifications to the robot.

5.1 Navigation and Obstacle Avoidance

Since the map of the course and placement of obstacles is not available ahead of time, it is impossible for the robot to plan its path from start to goal. Taking advantage of the fact that the road does not fork, the robot's goal is simply "go forward". As the robot advances through the course and obstacles are discovered and marked on the occupancy grid, the navigation code uses a simple bugging algorithm to circumvent cells that are impassable. For the navigation challenge, keeping track of which way is "forward" can be tricky as the road turns or if the robot is forced to backtrack after encountering a dead end because of obstacle placement. The main heuristic for determining which way is "forward" is to remember the traversed path and to avoid going towards regions that have already been explored. In the aforementioned case when the robot is forced to backtrack even as the heuristic "forward" into the dead end, the backtracking marks the path to the dead end as having been explored a second time, so the robot becomes more reluctant to fall into the same trap again since it avoids paths it has already taken. Eventually after sufficient backtracking, the robot comes to a point where it can go forward again without being forced into the dead end a second time, so it gets back on track towards the goal by exploring new regions of the course.

5.2 Vision

The vision algorithm uses an open source computer vision library developed by Intel called OpenCV (Open Computer Vision). The OpenCV process is for acquiring and processing the image from the webcam. We used C++ and the OpenCV library to process the image and extract information about hazards. We tried to remove as many user defined parameters as possible and eliminate false positives due to noise while at the same time keeping our algorithm as robust as possible.

The vision algorithm first gets the raw image from the camera and rescales the image size to 320x240 pixels. This resolution was empirically chosen because the processing time and information losses were deemed acceptable. The algorithm also converts the image to grayscale for computational efficiency (3 color channels are more difficult to work with, and they don't add much value).

Next, the obstacle extraction algorithm then converts the occupancy grid into robot centric coordinates using an inverse perspective transformation. This transformation allows us to represent the detected hazards in a robot centric frame of reference, which is of more value for control purposes than a camera centric frame of reference. By robot centric coordinates, we mean a top-down, bird's eye view of the robot.

The last preprocessing step is to smooth the image with a Gaussian kernel in order to filter out noise, primarily in the form of textured grass. The robot need not identify edges between blades/patches of grass as they do not pose a hazard to the robot.

Finally, the key to our hazard detection is the OpenCV implementation of the Canny edge detection algorithm to find boundaries between areas of the image. These boundaries correspond to white lines painted on the grass and the edges of barrels (and other obstacles). Therefore, we can treat the output of the Canny edge detection as an occupancy grid wherein edges (denoted by white pixels) are deemed impassable.

Shown below in Figure 7 is a sample video frame in each of the four main processing stages. The (roughly) V shaped outside edges are an artifact of the perspective transform which could be easily removed, but we keep them there as a deterrent to the obstacle avoidance algorithm against planning movements outside of the robot's current field of view since it is safer to assume there are hazards in the robot's blind spots.

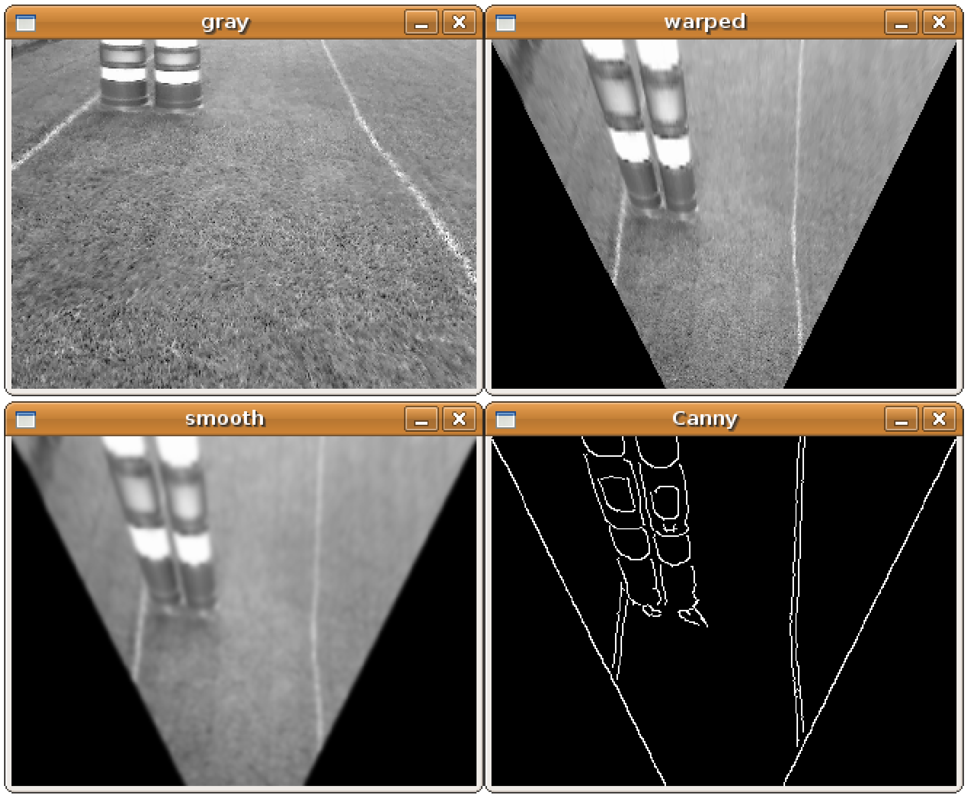


Figure 7: Vision Processing Stages

5.3 Performance Analysis

Last year, the RASmanian Devil's 1GHz CPU was overwhelmed by the flood of sensory data, and we were trying to do too much with it. The two main pitfalls were trying to map the entire course and not having enough processing power for real-time vision processing. We address both of these issues in this year's design.

Last year, we attempted to implement a SLAM (Simultaneous Localization And Mapping) algorithm so that our path planner can plan around obstacles. This proved to be very expensive computationally, and we scrapped the idea this year as unnecessary. Instead of trying to achieve optimal global navigation by mapping the course, we simply do local obstacle avoidance based primarily on lidar scans while relying on dead-reckoning for global navigation. This is a reasonable approach since optimal global navigation is impossible anyway without knowing the *full* map a priori, and building a full map of the course doesn't help since by the time the robot has seen (mapped) all of course, the robot has finished the course and no longer needs the map.

Last year, the vision processing loop could only process one video frame ever one or two seconds. That means that by the time the RASmadian Devil detected that it was approaching a white lane boundary line, up to two seconds had elapsed since the image was acquired. Clearly, this accounted for the Devil's difficulty staying within bounds. This year, the BlastasauRAS's quad core Xeon processor can process 15 frames per second by running the vision processing loop on a dedicated core. This is the same as the lidar's scanning rate and equates to a response time of 67 ms. A second core can be used to take advantage of the webcam's maximum 30 fps while leaving the remaining two cores free to acquire and process other sensory inputs (eg, lidar, compass, GPS) and to navigate. Clearly, BlastasauRAS is much better equipped than the RASmadian Devil to acquire and process (in real time) the large amounts sensory data.

5.4 JAUS

In order to participate in the JAUS challenge, the JAUS protocol was implemented using Jr Middleware 3.0, a SAE AS-4 (JAUS) AS-5669 compliant software toolset. This open source solution includes a Run-Time Engine which runs on the on-board computer and manages and routes message traffic between the robot and a base station. This Run-Time Engine receives JAUS formatted UDP packets on the incoming network port, then parses and stores the data. A higher level interface periodically polls the Run-Time Engine to see if any JAUS data requests have occurred. Responses to the data requests are handled through the higher level interface, which returns requested data to the Run-Time Engine, which subsequently sends the data to the base station.

6.0 PREDICTIONS

Our low-cost drive train and configurable electronics will provide a reasonable opportunity for success at the 2010 competition. The motors and gear box in this drive train allow a maximum speed of just under 5 MPH. However, as the robot was designed for outdoor use, we predict that we should be able to carry the payload up 15% grades with no problem. We predict that the 12V lead acid battery will provide six to eight hours of battery life.
Our camera can see objects in excess of 20 feet depending on the ambient brightness and the contrast, while our lidars have a maximum range of 8 meters. Our GPS has a best accuracy of 2 meters, but in practice on a clear day, far from buildings, we achieve an accuracy of 3-5 meters. Our waypoint navigation is bounded by our GPS navigation.

7.0 SAFETY

Safety has been a primary concern in many decisions throughout the design process of the robot. Human interaction with the robot is the main focus of robot safety. A testing operator will be present with an electrical safety stop (E-stop) whenever the robot is in operation. This hardware E-Stop is located directly after the batteries, see figure #2. It is meant to cut power to every system of the robot in the event of a serious problem or emergency. A less extreme option for disabling the robot is in our remote kill-switch. The remote kill-switch is wired so that it only cuts power to the motor controllers, disabling the robot's movement. This allows the user to disable the robot's motors while keeping the control system running, which is very useful for testing and debugging. In addition to the hardware E-stop and the remote kill-switch, we designed the software to immediately stop robot operation if our testing communication link is lost. In the early construction phase, we incorporated safety rules such as eliminating sharp edges, adding bumpers and covering exposed wires. All the high-current power wires use Anderson PowerPole connectors to avoid short circuiting batteries. In general, the robot has been designed for human interaction to ensure public safety around the BlastasauRAS.

8.0 COST

Quantity	Part	Retail Price	Our Price
1	Dell Precision 530 Desktop Computer	$350	$0
1	1500W 12VDC to 120VAC Power Inverter	$95	$95
1	OS5000-S Digital Compass	$270	$270
1	Logitech QuickCam Pro 9000	$90	$90
1	Garmin GPS 72 Unit	$110	$110
1	Hokuyo UHG-08LX Laser Rangefinder	$3950	$0
2	GrayHill 63R Encoders	$60	$60
2	TI BDC-MDL24 Black Jaguar Motor Controller	$220	$0
1	Linksys WGA600N Wireless Gaming Adapter	$80	$80
1	Linksys WRT54G Wireless Router	$70	$70
4	CIM Motors	$120	$120
4	Dewalt Hand Drills (for gearboxes)	$220	$220
2	16” Trailer Tires	$70	$70
1	Aluminum for frame	$200	$200
1	Sprockets and Chain	$100	$100
Total		$5,515	$1,485

Table 2: System Level Budget

9.0 CONCLUSION

BlastasauRAS is a culmination of effort of the RAS team from the University of Texas at Austin. As a fourth year team into the competition, our contributions to software infrastructure and low-cost electronic design are quite portable. In addition, our versatility and flexibility in software design and robust mechanical platform will be a defining aspect of our presence at the IGVC competition.

10.0 REFERENCES

[1] Garmin 72 GPS datasheet, https://buy.garmin.com/shop/shop.do?pID=214.
[2] OS5000-S Compass Module with Tilt Compensation datasheet, http://www.sparkfun.com/commerce/product_info.php?products_id=8507.
[3] Grayhill Encoders, http://lgrws01.grayhill.com/web/images/ProductImages/I-37-41.pdf.
[4] Logitech QuickCam Orbit Spec Sheet, http://www.logitech.com/index.cfm/products/details/US/EN,CRID=2204,CONTENTID=10628.
[5] Hokuyo UHG-08LX datasheet, http://www.acroname.com/robotics/parts/R311-HOKUYO-LASER2.html
[6] Jaguar Brushed DC motor controller datasheet: http://www.luminarymicro.com/products/mdl-bdc24.html
[7] G. Bradski, A. Kaehler, *Learning OpenCV*, O'Reilly Media, Inc., 2008.
[8] Canny, J. (1986), 'A computational approach to edge detection', *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**(6), 679-698.