Robo Goat 2012

United States Naval Academy, Systems Engineering Department

By Midshipman:

Adam Albrecht (Sr)

Cory Oberst (Jr)

Nicholas Mehalic (Ens)

Adviser Statement: I certify that the design and engineering of the vehicle by the current student team has been significant and was awarded credit for a capstone senior design course.

Adviser: Assoc. Prof. Joel M. Esposito _____

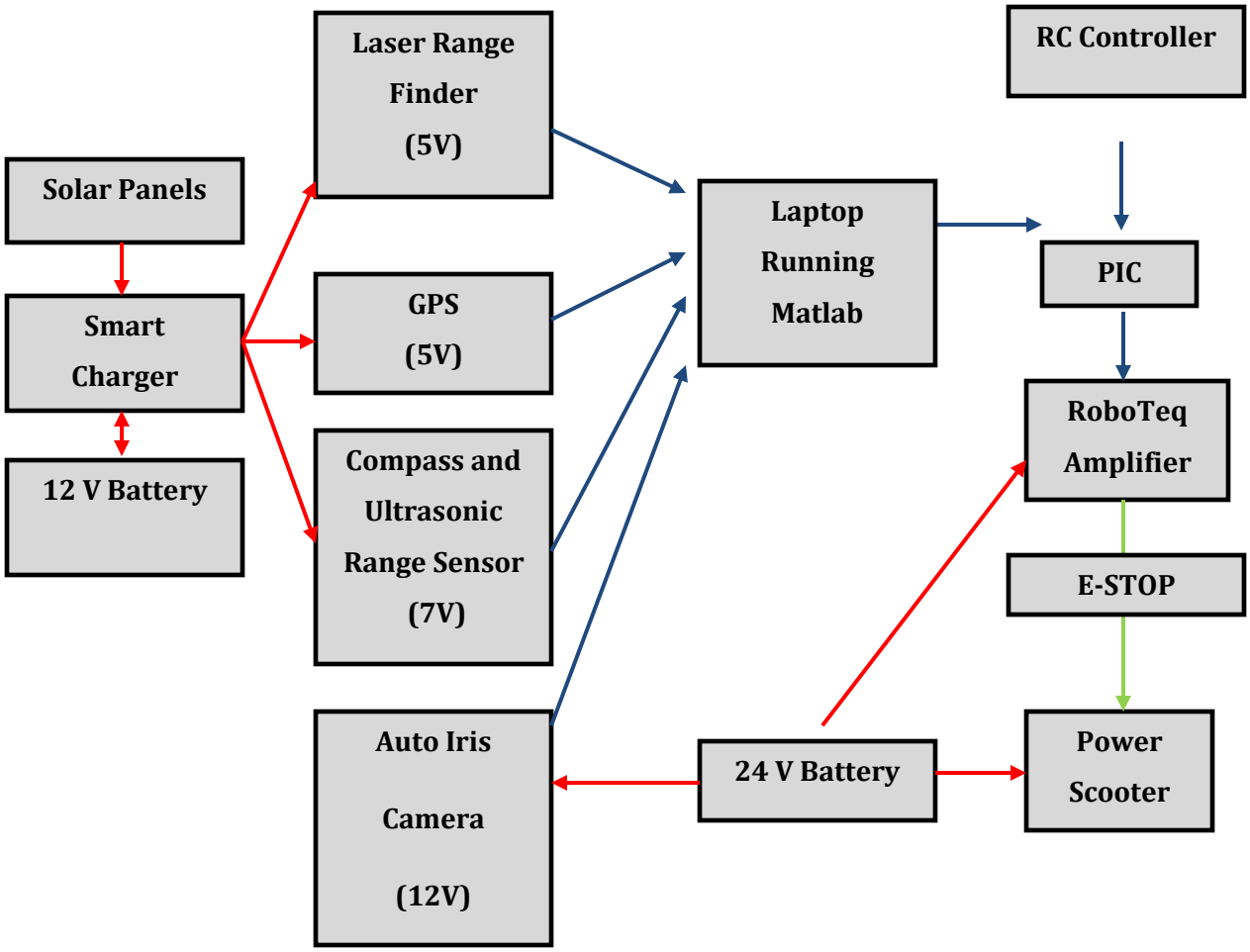
New For 2012 Entry:

Differential GPS	Reworked Solar panel and charging system
Adaptive Vision algorithm	Shadow detection algorithm
Completely redesigned body	Redesign electrical system

Overview

The Robot-Goat is designed around four principles:

1. Maximize off the shelf hardware use.
2. All processing done by one laptop running Matlab
3. A system whose state and world view are easily visualized and controlled by the developers, can be debugged more efficiently.
4. Mapping is not necessary to complete the competition tasks and introduces unnecessary complexity.



Innovation: We believe these are the most innovative features of the Robo-Goat.

- Frame Design: As compared with our previous entries, this version is much closer to ideal. The robot has a nearly minimal 2' 4" by 3' 2" footprint and maximal height of 6' -- making vision and obstacle avoidance easier.
- Solar Power: it uses solar power to charge a battery while runing the onboard electronics, including the laptop, eliminating battery life as one potential limiting factor when testing.

- Robust Vision System: in our experience, this is the most challenging component of the competition. Our robot uses a combination of alternate color spaces, automatic background detection and s new shadow compensation algorithm to challenges arising from variable lighting conditions.
- User Interfaces: We subscribe to the adage: if you can't see what the system is thinking, you will not be able to debug or improve it. To that end we have implemented a variety of user interfaces: a robot centered world view, a remote control model that can be switched to on the fly, a integrated power system displayand a camera threshold selection tool.
- Maximize Off the Shelf Components: When possible we try to focus on the planning and perception algorithms rather than building custom hardware.
- Runs Entirely in Matlab: We exploit Matlab's extensive library of image processing routines, statistics toolbox, GUI and visualization tools. Programming in Matlab enables rapid prototyping of code, and makes visualization easy. Despite Matlab's reputation for being slow, with proper coding technique and a new laptop we are updating at 10 Hz. We believe we are the only entry running entirely in Matlab.

New Design Decision Process

Background: The Robo-Goat capstone project is a legacy project that began in 2009. Every year the vehicle has competed in the IGVC (Intelligent Ground Vehicle Competition). In 2009 the Goat placed 20th, and in 2010 and 2011 it placed 10th. This year, we hope to be within the top five. Last year, the majority of issues with the Goat came from the path detection capabilities of the camera. Also, robot handled poorly on ramps and speed bumps and our GPS was susceptible to systemic errors based on satellite geometry. Our strengths include obstacle avoidance, and reliability

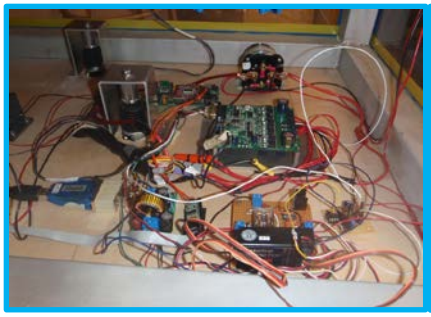
Characteristic	Option 1	Option 2	Option 3	Option 4
Vision Hardware	Auto Iris Camera	USB Camera	Stereo Camera System	-
Vision thresholds	Fixed	Global Adaptive	Locally Adaptive	-
Obstacle Avoidance	Ultra-Sonic Sensors	Laser Range Finders	Camera System	-
Body	Existing	New		
Power	Battery	Solar	Wind	Bio-Fuel
GPS	Non-differntial	Differential		

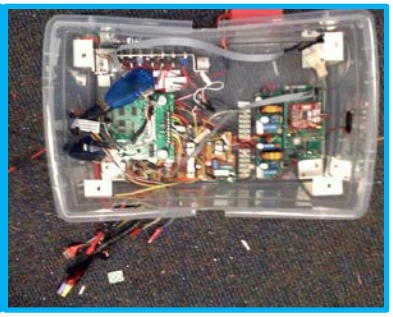
After identifying these weaknesses, our team developed the following morphological chart to help identify the solutions. **The hi-lighted sections of this chart represent the major additions made to our vehicle this year.**

Construction

One of the main problems that we noticed in the 2011 design was the stability of the vehicle itself with the likeliness of tipping and shaking. The 2011 vehicle design weighed in at 230 lbs with a center of gravity of 24" off the ground. The center of gravity was also not directly over the center of the wheel base which posed a problem in stability. In order to improve this we completely overhauled the body of the Robo-Goat seeking a lighter and more stable design. We realized that by taking off much of the excess weight in the upper frame of the vehicle that we would inherently lower the center of gravity of the vehicle at the same time. To do this we took off the old box frame and decided to use a simple water proof box made of plastic that would mount just above the wheel chair base of the previous design. In doing this we eliminated the metal and plywood frame and replaced it with lightweight plastic. We also eliminated all of the excess space that was available in the old design but still left room to add more equipment if desired. In the end our new design weighed in at 170 lbs and had a center of gravity of 12" that was right over the center of the wheel base. Overall we dropped 60 lbs and lowered the center of gravity by half of its original height to just a foot off the ground. After testing the new vehicle design by manually driving it we found that our turns were much more fluent and when we stopped the vehicle there was no shaking as there was in the 2011 design due to the weight being so high on the vehicle. The changes we implemented will allow us to drive and turn at higher speeds without the instability of the previous design while also nearly eliminating the probability of tipping on slopes or while going over bumps in the course.

One of the primary flaws of the 2011 design was the disorganized wiring harness and difficult maintenance and trouble shooting. The primary causes of these flaws were that the entire electronics package was mounted in a box with access from only one side and the wires were run without prior planning or color coding. To fix these issues we mounted the entire electronics package into a smaller clear container with full accessibility from the top. Using a smaller container we were able to better map out the wiring harness and use an intuitive color coding system. The improved wiring layout and color coding makes trouble shooting and adjustments faster and more efficient. To further improve trouble shooting and maintenance we installed an integrated voltage meter that is user selectable to each power source (5 v, 6 v, 12 v, or 24 v). To increase the life-span of electronics a central cooling system was installed in the primary electronics container. To aid in on-the-fly adjustment of the vehicle we installed an adjustable height mast with a fully adjustable mount for the forward and side facing ultrasonic range sensors. Also to increase maintenance efficiency an externally accessible switch plate was installed which gives the user full access to all internal components' power supplies.





Old Wiring **New wiring** **New Switches**

	Before	After	Difference
Total Weight	230 lb	170 lb	Lost 60 lb
Center of Gravity	24”	12”	Lowered 12”

(Left) Old Body: Heavy and High CG

(Right) New Body: Lighter and Lower CG, capable of pitching 60 degrees without toppling

Power System

Figure 1

Solar Power System: The next improvement we sought to implement was to design a central power system to allow us to be away from a power source for an extended period of time while testing outside. We ran into some problems with the 2011 design due to our limited testing time because either the laptops or the chair battery would become low, forcing us to return to a power source indoors. To fix this we integrated solar panels along with a solar charge controller to charge the chair battery while outdoors. The controller also displays what the chair battery level is so we know if we do need to charge with a regular power source. Then we placed a power inverter in the base of the chair to allow us to charge our laptops off of the chair battery while we are testing. These two improvements will allow us to stay out in the field for extended periods of time while not worrying about any of the batteries dying.

The battery (Fig 1) that runs the electronics on the Robo-Goat is a 12V sealed lead acid battery. The electronics run by the 12V battery are: GPS, Laser, Nav-Board, Ultrasonic range sensors, and RC boards. The total draw on the battery when idling is .64A. There are two Siemens Solar Industries SM20 solar panels (Fig 4). They are individually rated at 20W and designed to support a load of up to 1.38A at 14.5V. These panels were wired in parallel to double the charging current supplied to the 12V battery.

Figure 2

The solar charge controller (Fig 2) is a Morningstar Corp. SunSaver-10. The important feature of the controller is the circuitry that keeps the panels from draining the battery in poor lighting conditions. The controller is rated to handle 10A of input from the solar panels and a 10A load. The controller is designed to run a 12V battery and load and a comparable voltage solar panel. We chose a controller without a LVD (Low Voltage Disconnect) feature. Since all the electronics attached to the battery require between 5V and 6V, there was no danger of equipment damage if the battery voltage dropped below the 11.5V factory set disconnect voltage. In fact, we hypothesized that this feature would prematurely force us to replace the 12V battery, thereby defeating the purpose of the solar recharging system. The capabilities of this controller significantly exceed the requirements of our current system.

Figure 4

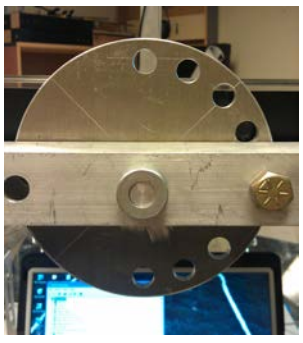
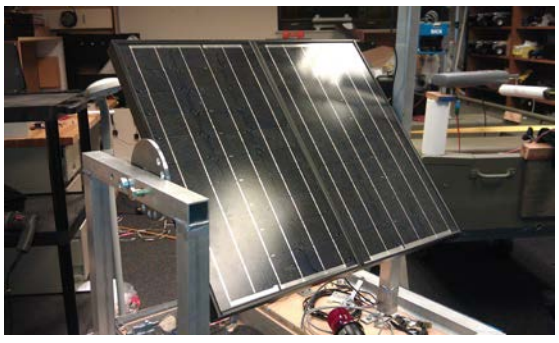


Figure 3



Testing Solar System: Using a 12V motor that draws .4A we tested the solar system to see how well it worked. A baseline test with the motor attached to a fully charged battery caused the battery

voltage to drop below 11V in approximately 45 minutes. Note that this would imply that the 12V battery has a rating of .3Ah, when it is rated for 7Ah. The batteries are 5 or 6 years old and have been used year after year in capstone projects. When the solar panels were attached and tested on a partly cloudy day, the battery voltage had yet to drop below 11V after 2 hrs. Our true test occurred during the DC demonstration of our project in April. On both days we started with a fully charged battery and did not need to change batteries at any time during the day. In fact, the second day was sunny and the 12V battery was still fully charged at the end of the day.

Lane Following System

Auto Iris: We've chosen to use an auto-iris camera, purchased from The Imaging Source. It uses a motorized iris to control the amount of light entering the camera. This ability should not be confused with changing the exposure, which alters the frame rate. The reason that we have chosen an auto-iris camera is to compensate for dynamic changes in outdoor lighting. In previous years, when an auto-iris camera was not used, small changes in daylight would negatively affect the program's ability to track selected colors.

Hood, mount and filter, field of view: The camera is mounted underneath a hood to reduce glare from the sun overhead and uses a circular polarizing filter. It is mounted at about 70 inches above the ground, so that it has the largest view of the ground as possible. It is angled downwards, such that the bottom of the viewing window sees 6-inches from the front of the robot. This to reduce the size of a blind spot directly in front of the robot. At this angle, the camera sees 24-inches wider than the robot on each side.

Why YCbCr: The color space that has been chosen for this project is YCbCr. The Y stands for brightness, while Cb and Cr refer to color values (Fig 5). If the user only looked at the Y values of an image, they would see a grayscale image with the black areas having a Y-value of 0 and the whitest areas having a Y-value of 1. This color space is handy for the RoboGoat because it distinguishes color from brightness. This means that the thresholds selected with Cb-Cr are more stable in differing lighting conditions, because Cb-Cr values refer to "pure" colors and are not affected by shades or shadows. Look back at the thresholded red buoy (Fig 1), notice how all of the red paint has been selected despite shadowing.

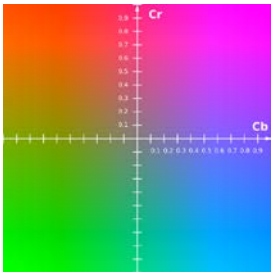


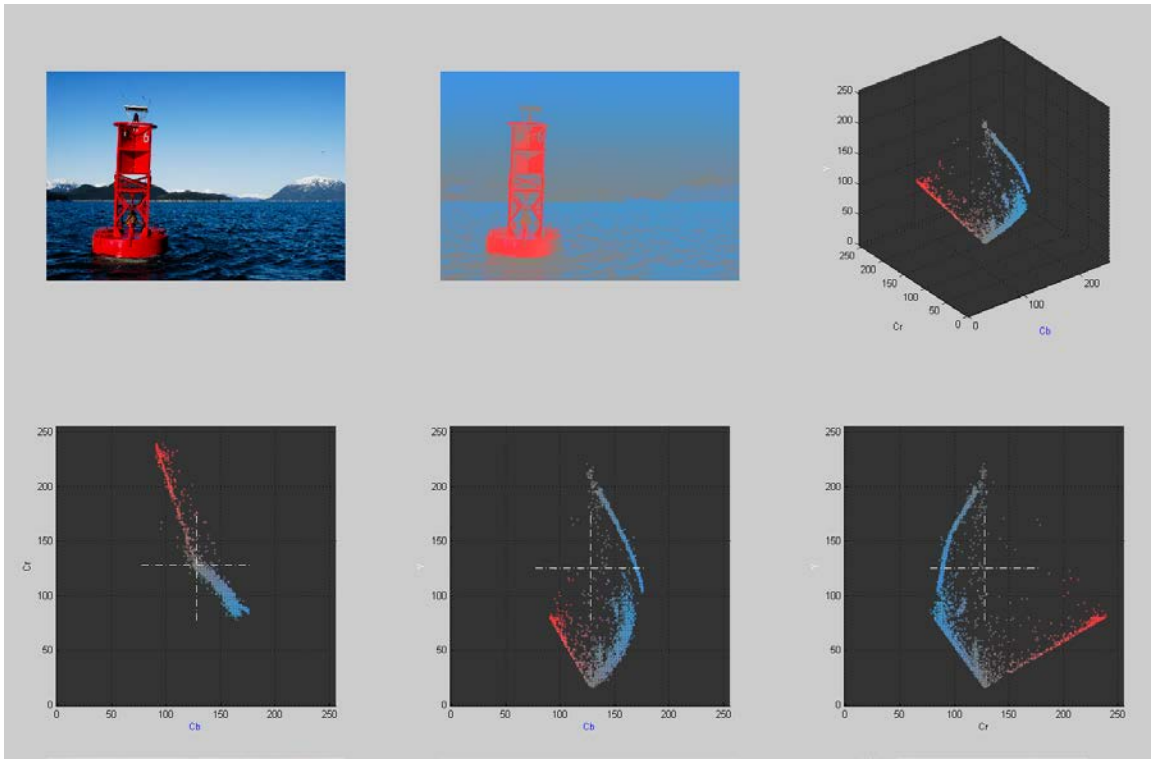
Figure 5

Cb-Cr colorplane, Y set to 0.5

The 6-Plot Figure is shown below (Fig 7), and is the most powerful source of image information in the program . The top three plots, moving left-to-right, display: the original image (top-left), the image shown in YCbCr with a uniform/normalized brightness level (top-middle), and a rotatable 3D scatter plot of the image's pixels within the YCbCr color space (top-right). The colors of the dots in this 3D scatter plot match the colors in the YCbCr image, so

the user can see what part of the image they refer to. The bottom three plots, moving left-to-right, display the three 2D perspectives of the 3D scatter plot: Cb-Cr (bottom-left), Y-Cb (bottom-middle), and Y-Cr (bottom-right).

Figure 6



The three 2D graphs are very handy. In the example of using the picture of the red buoy, all three 2D plots show the dots for the red buoy from various angles of the 3D graph. The dots in first 2D graph (Cb-Cr), however, include every value of Y (brightness). This means that if the red dots are selected from the Cb-Cr graph, every shade of the red buoy will be included. The other 2D graphs have Y in the vertical axis, so the red shade-ranges are spread vertically up and down the graph. But if the user wanted to specifically choose the lightest or darkest red pixels, they could choose the Y-Cb or Y-Cr graphs. Once the user chooses the 2D graph that they would like to use for threshold selection, the program will display a new window. This window displays three items: (left-to-right) the original image, the image shown in YCbCr with uniform/normalized brightness level, and the selected 2D graph.

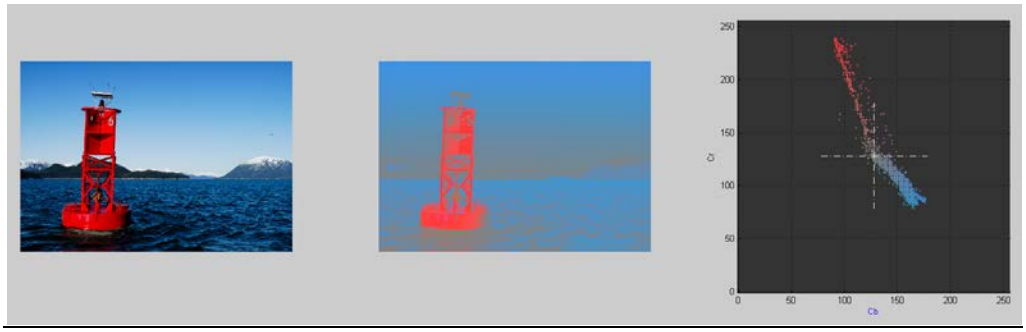


Figure 7_*Original Image* *YCbCr view, uniform brightness* *Distribution of pixels in color space*

In the 2D graph, the user will freehand-select (draw) around the pixels they want to threshold. This is done by left-clicking in the graph, holding down the mouse button, and dragging the cursor around the desired dots. Figure 9 shows what the freehand selection looks like, in-progress:

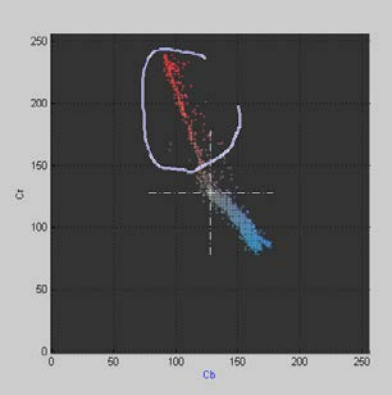


Figure 8

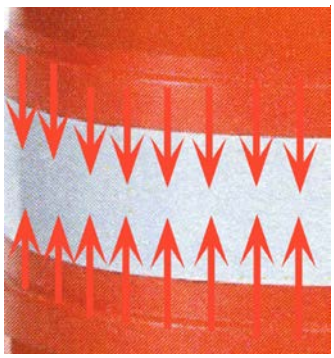
The program reads the coordinate values of every selected pixel. In this example, the coordinates are Cb-Cr values (just like the x-axis and y-axis). The program then takes the maximum and minimum values from both axes - these four values become the thresholds.

Solving the "Barrel Problem": Certain objects on the IGVC course pose a problem for the robot's vision system. The robot is designed to use its camera to look at chalked/painted white lane lines on the ground, using them as a visual reference to stay within the lane. There are many obstacles to avoid along the route, the most common of which is an orange construction barrel. Figure 10 shows a picture of several such barrels on the IGVC course - the lane line is visible in this figure too

Figure 9

When pixels in each frame of the streaming camera fit the thresholded criteria of a "lane line" - when our robot sees what it believes to be a white lane line - it assigns a trend line to those pixels. This is a simple concept, the same process occurs in the brain when one sees a less-than-perfectly painted line on a field. However, the robot's line-fitting ability cannot distinguish the difference between white paint in the grass and the white stripes on the barrels. If they both fit the color-criteria for a lane line (white), then the program marks those pixels as "lane line" pixels. Then, when it calculates the trend line for the "lane line", the result is completely incorrect. The robot thinks that the lane line is pointed/angled in a certain direction, when any person looking at the field knows otherwise. Fortunately, there is a way around this problem through the use of structuring elements. A structuring element is a shape created around a pixel. For this particular task, we use a rectangular-shaped structuring element.

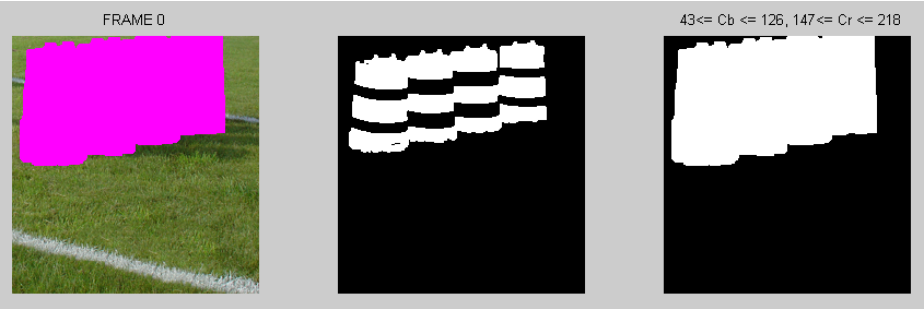
Figure 10



Orange pixels are dilated to overlap and conceal the white stripes

The result of this method is that the program is free to threshold and track the "actual" lane lines [in the grass], using the trend line calculations, since it is no longer confused by the white regions on the barrels. Figure 12 shows this dilation executed on the image of the course (Fig 10):

Figure 11



Calibration to ground plan with CalTech Toolbox: Using the CalTech Toolbox camera calibration software, we are able to determine the intrinsic and extrinsic parameters of the camera. This software allows the robot to know where the camera is, in reference to the ground. Since the camera is not stereoscopic, and without true depth perception, it calculates "depth" by assuming that all objects are flat on the ground.

Shadow Correction: Changing light conditions, both over time and even within the same image, are a constant challenge for outdoor vision. While the use of the YCbCr colorspace can help mitigate this, one defining feature of the while lines is that they have a higher Y compoenent than the grass. We developed a new vision algorithm this year to address these issues.

1. Find all “green” pixels in image based on CbCrthresholds for grass.
2. Compute the average brightness (Y) of the grass. Rescale the image to make this brightness level the new median brightness (128) (Figure 14 Top Center)
3. Regions that are two standard deviations darker than this are considered shadows.
4. Apply a shadow correction technique inspired by Mark Ollis’s 1995 Master’s Thesis at Carnagie Mellon University. (Figure 14 Top Right)
    a. Rescale shadow regions to median 128 brightness.
    b. Apply non-uniform color correction to compensate for sunlight vs skylight illumination.
    c. Clean up halos and artifacts using morphology
5. Regions that are 2 standard deviations higher then background brightness, become candidate lane markings an are they subject to color, shape and size filters.

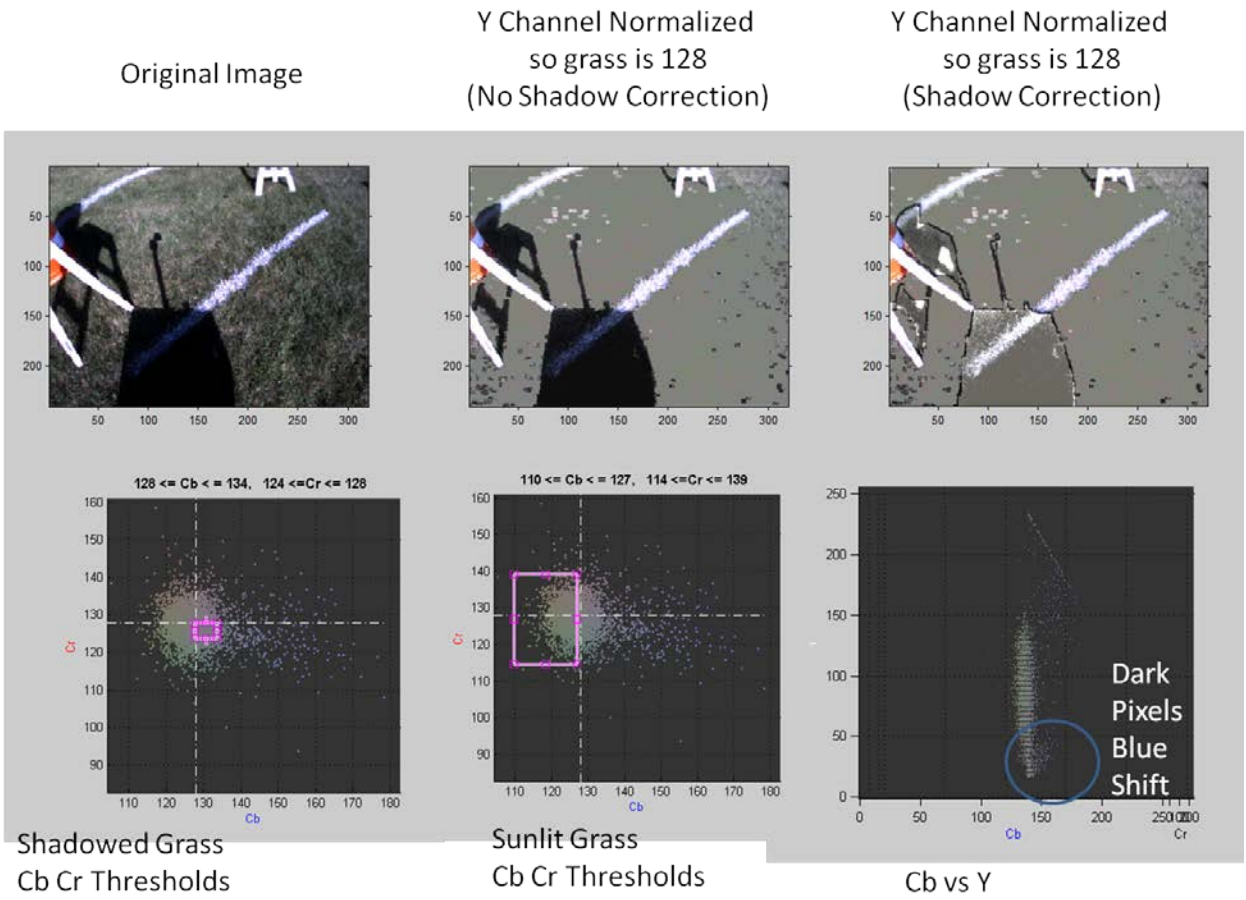


Figure 14: Shadow compensation

Obstacle Avoidance System

Laser: The primary obstacle avoidance system is a laser scanner -- the Hokuyo URG-04LX. The laser is configured to scan every 2 degrees in a 240 degree arc. The laser measures phase difference and time of flight to determine the range to an object.

Our algorithm is patterned off of the Dynamic Window Obstacle Avoidance Method. After accounting for the robot's size and speed limitations, it finds the best "gap" or heading direction. "Best" is defined along the following criteria (1) closest to desired heading (specified by camera or GPS); (2) closest to current heading; and (3) maximum clearance.

The laser is capable of seeing up to 4 meters out; however, we set the maximum range that our navigation algorithm will recognize lower so we can navigate switchbacks. A switchback can be seen as three layers of obstacles. When the distance threshold of the laser is set properly, only one layer of obstacles can be seen at a time. This allows the Robo-Goat to enter the first gap before trying to avoid the second layer of obstacles. If the distance threshold is set too high, the Robo-Goat will see both the first and second layer at time, which will look like a solid wall, making the Robo-Goat perform a u-turn.

The layering of the switchback obstacles is important because our algorithm that analyses the laser, camera, and ultrasonic data finds gaps in the obstacles. Lane data from the camera and returns from the ultrasonic sensors are combined with the laser data to form a complete picture of the obstacles around the Robo-Goat. After finding the gaps, it compares the width of the gaps to the parameter that defines the width of the Robo-Goat and determines if a gap is large enough. The code then finds the gap that is closest to the desired heading, which was defined using either GPS waypoints or lane information from the camera.

GPS Navigation System

GPS: The GPS receiver we use is a Trimble Ag 7500. The receiver sensitivity is -185dBW and is accurate to within 1m using WAAS differential GPS. The receiver is attached to our laptop via a serial to USB adapter. The GPS receiver is mounted atop our mast to have a clear view of the satellites and is powered by the 12V battery through a 5V regulator. The receiver sends a NMEA (National Marine Electronics Association) sentence every .2 seconds to the serial object in MATLAB.

We display our position on a google maps overlay written in Matlab. This visualization tool greatly improves our way points etting and debugging ability.

The Robo-Goat does not use any form of mapping. In fact, the Robo-Goat has no memory. In an attempt to achieve a 10Hz speed on our navigation loop and the reaction benefits that that brings, we decided to not map our environment. This, however, is making it difficult to keep ourselves from performing a u-turn when we encounter an obstacle or from crossing a line that has disappeared beyond our field of view

Nav-Board: The Nav-Board used on the Robo-Goat (Fig 14) is designed by the technical staff in the department for use on numerous different projects. The board is based on a Rabbit 3000 microprocessor and hosts a XBee module, 3-axis accelerometer, 3-axis magnetometer, a GPS receiver, and numerous I/O ports (including analog, serial, and interrupts). We use the Nav-Board's magnetometers as a compass and the analog I/O to read data from our ultrasonic sensors. Data is collected by MATLAB from the Nav-Board via serial port.

Figure 12

The compass on the Nav-Board is calibrated by recording the raw numbers from the magnetometers while spinning the Robo-Goat in a circle. The result when the x and y axis are graphed simultaneously is a circle. The values needed to zero the compass are the x and y coordinates of the center of the circle, taken by averaging all the x and y values separately. These numbers are then written into the MATLAB code that reads the data from the serial object through which the laptop and the Nav-Board communicate.

These numbers are added to the raw data so that heading can be accurately determined. If improperly calibrated, typical errors include all headings being in the same quadrant or headings appearing to skip a quadrant when the Robo-Goat is rotated.

Drive System

Speed/Performance: The Robo-Goat is hardware limited by its capabilities to 4.77 MPH. We also discovered that in order to maintain our minimum speed we have to, on average, send at least 6V to our motors.

Voltage (V)	RPM	MPH (10" wheel)
5.93	37	1.1
23.2	160.3	4.77

Roboteq: The Robo-Goat is propelled by two 24V DC motors rated at 4.5A each. We control the motors with the Roboteq motor control board via commands from the laptop through a serial connection. We used the supplied RoboRun utility to measure the limits of our system by using a tachometer to measure our rotational velocity at varying voltage outputs. Since the motors are powered using two 12V lead acid batteries in series, we have an available voltage range of 24V. The RoboRun utility has a power setting that ranges from 0 to 127 and is proportional to the voltage range. This was the range used to test the speeds of the motors. The MATLAB function that sends commands to the Robo-Goat, however, has translational and rotational inputs that range from 0 to 1. These are combined to form individual commands for each motor that correspond to the desired voltages. Forward and reverse changes are handled by changing which pin the Roboteq board uses to output a positive voltage to the motor with.

Wheel Chair: The chassis for the Robo-Goat is a Sunfire Plus SP3C Power Mobility Chair. It weighs 82 lbs. The chair has a range of up to 20 miles, making it well suited to the competition. The chair is powered by two 12V U1 34 Ah batteries in series. The ground clearance on the chair is only 2.5 inches, meaning we can navigate small pot holes, but not large ones.

Safety Systems

Safety Light: A new requirement for the IGVC this year is a safety light. The light must be solid when the vehicle is in ROV/manual mode and be flashing when it is in autonomous mode. The safety light we chose to use is a Federal Signal LP3TL-024R. We chose it for two reasons. First, it has an LED bulb that has a low current draw, .08A. The second reason we chose the light was its 4X weather rating, which means it is able to withstand dust, hose directed water, and corrosion. With a 100,000 hour life on the LED bulb, this light should remain useful to the project for many years.

Figure 13

Emergency Stop: The Robo-Goat is paired with a 6CH Futaba transmitter and receiver. The Futaba transmitter manually controls the Robo-Goat, has an Emergency Stop, and has the ability to switch the input of the Roboteq motor control board. A PIC12f675 is used to monitor CH5 which is the Emergency Stop channel. If tripped, the PIC disengages a relay that cuts power to a solenoid, which in turn cuts power to the motors running the Robo-Goat. Another PIC12f675 monitors the other channels to manually control the Robo-Goat and to change the input to the Roboteq board. The input for the Roboteq board can be set to the RS-232 input from the laptop running MATLAB or it can be set to receive input from the second PIC trough a MAX 323 which converts the signal from the transmitter to RS-232 levels.[1]

Integration Testing

We participated in the AUVSI demonstration on DC's National Mall in April. There the system performed well, traveling over 500ft autonomously in the two days we were there. Specific testing points:

1. GPS Navigation: Based on 10 way points tested, the goat got within 2 meters of 9 of the 10. The final way point was declared "reached" once the goat was within 3 meters of the actual point. This can be difficult to test without surveying equipment.
2. Speed: The vehicle's max speed is 5 mph. However, at this time the autonomous navigation has only been tested at speeds up to 2 mph. Initial experiments with our new laptop suggest an update rate of 10 hz (vs the current rate of 5 hz). We believe this will permit running the course at max speed.
3. Battery Life: At a recent test evolution the power systems was initiallyfully charged using AC power. Over the next 48 hours, we ran for about 4 hours using only the preexisting charge, and the energy contributed by the solar cells.
4. Obstacle Avoidance: Our obstacle avoidance algorithm is nearly flawless. It is perhaps the strongest feature of robo goat. At this time we intentionally use a detection distance of 1.5 meters, even though our hardware is capable of up to 4 meters.
5. Lane following: Over the course of 48 hours, we observe the vision system work nearly 100% of the time. More importantly we did not adjust the color thresholds despite the fact that the light conditions changed. This is extremely promising. Note it is very difficult for us to replicate the lane markings on campus because we cannot paint the grass.
6. The states "…expect natural or artificial inclines with gradients not to exceed 15% and IGVC randomly placed obstacles along the course." We ae confident the robot can power itself up these inclines based on our inhouse ramp tests. More importantly we tipped the robot to its edge of stability and concluded that 15 degress does not pose a problem. (Fig 16-17)

Figure 14

Figure 15

[1] Information taken from Adam Albrecht's report "Robo-Goat: Integrated Radio Control System" written 09DEC10.

Cost Estimate

Cost:

	Number	hr/wk	$/hr	# of weeks	Total # of Hours	Total Cost
Students:	5	12	26	16	192	$24,960.00
Shop	1	-	25	-	25	$625.00
Advisor	1	2	50	16	32	$1,600.00

Parts		
Body Work:	Chassis	$1,741
	Weather Proofing	$80
Total Body Parts:		$1,821
Hokoyo Laser (1):		$2,500
Roboteq Amp		$700
GPS:		$199
Accelerometer:		$10
Compass:		$40
Rabbit Board:		$99
Ultrasound Range Finder (4):		$80
Emergency Light:		$100
Solar Panel (2):		$280
Charge Controller:		$60
Camera Lens (1):		$113
Laptop and software:		$2,200
Camera (1):		$670
	Total Parts:	$8,912

Direct Labor Cost:	$ 27,185.00
Indirect Labor Cost:	$ 27,185.00
Overhead:	$ 27,185.00
Total Cost:	$ 89,367.00