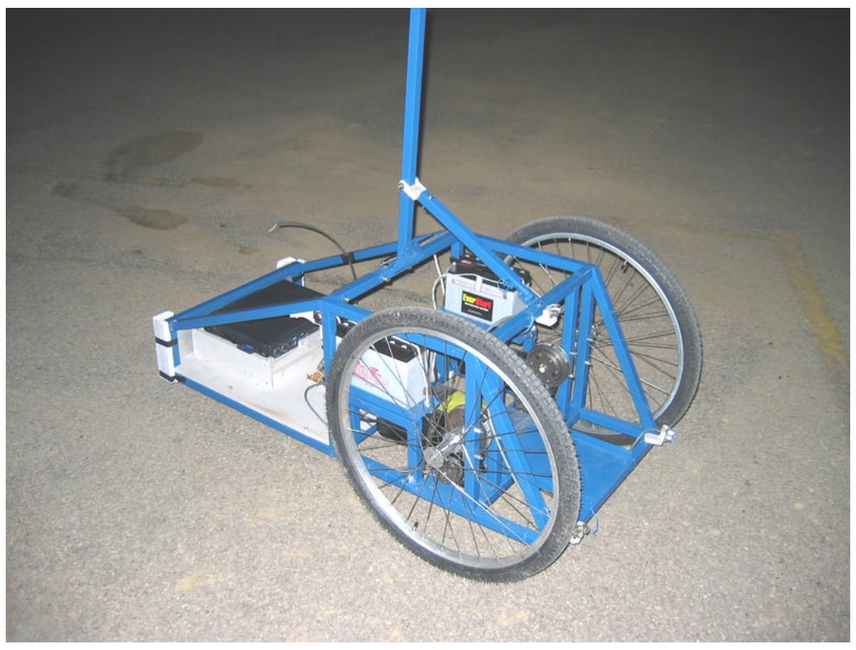
Cedarville University

Little Blue

IGVC Robot Design Report

June 2004



Team Members:

- Silas Gibbs
- Kenny Keslar
- Tim Linden
- Jonathan Struebel

Faculty Advisor:

- Dr. Clint Kohl

Table of Contents

1. Introduction

Imagine a robot that can drive down the highway without running into the other cars, or a robot that can transport hazardous material between two locations without human intervention. While these scenarios are more advanced than this autonomous vehicle competition, it is not difficult to envision them in the near future with the technology and designs that come out of this competition. With the increases in portable computing power and the quality of cheap digital cameras, it is possible to build a robot that can autonomously navigate a course that is similar to a road, including obstacles that must be avoided. Our design is a quick, rugged vehicle that was kept simple in order to increase its reliability.

2. Team Organization

The Cedarville University Little Blue robot team was organized as follows:

Member Name	Responsibilities	Major and Class	Total Hours
Silas Gibbs	Software Design, Mechanical Design	Electrical Engineering, Senior	250
Kenny Keslar	Software Design	Computer Science, Senior	200
Tim Linden	Electrical Design, Mechanical Design	Computer Engineering, Sophomore	250
Jonathan Struebel	Electrical Design, Mechanical Design	Electrical Engineering, Senior	200

3. Hardware Design

3.1. Robot Chassis

The robot chassis was custom designed by our team for this competition and built by our shop technician, Dave Denlinger. It is 18 inches high, 24 inches wide, 48 inches long and was constructed using ½” square steel tubing. The camera is mounted on a post that extends from the top of the chassis so that it can see further than if it were mounted directly on top.

Since most of the electronics are mounted within the chassis, we can protect them from rain by simply mounting plastic panels on the sides top and front.

3.2. Drive Train

The drive train consists of two 26-inch bicycle wheels mounted near the front of the robot and a single 5-inch caster at the rear of the robot. Each of the bicycle wheels was custom made and mounted on a shaft with a toothed pulley which is connected to the motor via a toothed belt so that it is driven independently; allowing the robot to maneuver in tight spaces if it ever encounters a dead end.

3.3. Motors and Gearing

The motors that we chose are ¼ horsepower right angle gearmotors from Bodine Electric Company. These motors have 24VDC windings and draw a maximum of 8.8A at max load. These motors provide more than adequate power to ensure good ramp climbing ability. We have successfully driven up and down grades of at least 20% without difficulty. Since the maximum output speed of the gearmotors is 125 revolutions per minute we used a 2:1 gear reduction to couple the motors to the wheels so that the maximum speed of our robot is 4.83 miles per hour, within the requirements of 5 miles per hour yet not limiting the robot if the image processing and control can manage the robot at speeds close to 5 miles per hour.

3.4. Motor Controllers

The motor controllers we chose are rated for 20A at 50VDC, providing plenty of margin for our motors, and have built-in inrush current limiting. These controllers interface to the Z8 microcontroller and provide pulse width modulated speed control for maximum torque at all speeds. The drive train provides an excellent response time and is able to accelerate the vehicle to top speed in under 4 seconds and to stop from full speed in under 2 seconds.

4. Electronics System

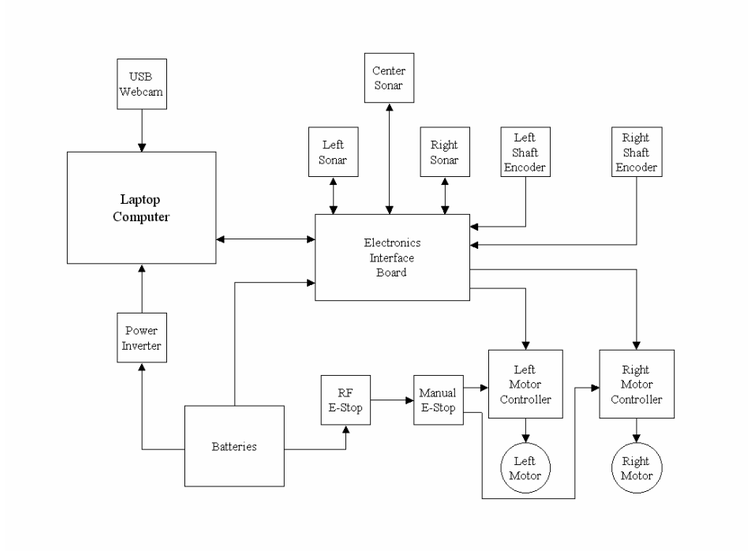


Figure 1. Robot Electronic System Diagram.

4.1. Vision System

To capture images, we were initially planning on using an inexpensive web camera; however, we had difficulty in finding one that would handle the outdoor light levels. We finally chose to use a USB frame-grabber; this configuration is still inexpensive, and allowed us more flexibility in choosing a camera. We chose a black and white security camera that has auto-iris and a wide-angle lens (about 80 degree field of view). Since we are not using a color camera, we have less information to use in the control algorithm; however we compensate for the lack of color information from the camera by placing a color filter over the camera lens. This filter was selected to increase the contrast between the lines and the grass, thus improving obstacle detection.

4.2. Sensors

In addition to our vision system we have two other sensing systems to aid in navigating the robot. The first is an array of three SRF08 sonar sensors from Robot Electronics that are

mounted on the front of the robot. Each sensor interfaces to the electronics interface board via the I^2C bus. These sensors are arranged to detect obstacles that are in front of and slightly to the sides of the robot. This system is primarily used as a worst case backup so that if the vision processing fails to detect an obstacle the sonar will detect it and stop the robot before it runs into the obstacle.

The second sensing system is a shaft encoder on each of the two drive wheels. These sensors consist of a cardboard disc with slots cut in it and mounted to the drive shaft and an infrared photo interrupter. The signal from the photo interrupter is fed into the electronics interface board which uses it to measure the speed of each wheel. This system is used in a feedback control loop to ensure that the robot moves as the control system intends it to move. It can also be used to measure the distance that the robot travels if that information is necessary.

4.3. Electronics Interface Board

The electronics interface board is used to interface with the sensors and motor controllers so that the laptop can be devoted to image processing and path planning. The electronics interface board uses a Z8 microcontroller from Zilog to communicate with and to process data from the sensors. The microcontroller has an onboard I2C bus controller that is used to interface to the motor controllers and the sonar. It also has onboard counters that are used to interface with the motor controllers and the shaft encoders. The laptop communicates with the Z8 microcontroller via an RS-232 serial port and is treated as a peripheral by the laptop.

4.4. Power

To power the robot we chose to use two 12VDC 22AH motorcycle batteries. These batteries are connected in series to provide 24VDC for the motors and will provide approximately 2 hours of running time under full load. The lower battery is used to power the electronics

interface board and the camera, and we can connect a power inverter to the upper battery to power the laptop computer when its battery runs low.

4.5. Emergency Stop

Our RF and manual emergency stops are connected in series with the battery power to the motors. This arrangement provides a safe way of stopping since the geared motors will not allow the vehicle to move when the power is removed. The RF emergency stop is an RC transmitter and receiver connected to a small servo that mechanically actuates a switch to stop the power to the motors. The manual emergency stop is a pushbutton toggle switch with a red disc mounted on it and located at the rear of the robot. All of these systems enhance the safe operation of our vehicle and have proven effective in testing.

4.6. Navigation and Control System

The navigation and control system is a Dell Latitude C800 laptop computer with a 1GHz Pentium III mobile processor and 512MB of RAM running a Linux OS. It interfaces directly to the USB frame-grabber and the electronics interface board. The primary function of the computer is to process the images from the camera and find the best path for the robot.

5. Software Design

5.1. Overview

Our primary concept for this robot was to use a laptop computer with a USB image capture device and an inexpensive security camera. Our main goals were to have a low-cost, consumer product based, efficient and robust control system. By limiting the number and complexity of our components we also hoped to improve reliability. One of our earliest design decisions was to run the Linux operating system on the laptop. Our reasons for this included the ability to turn off all unnecessary operating system functions so that as much

processing power as possible could be used for the robot control. Also to maintain efficiency, we decided to write all control software in C++ from scratch. These large decisions guided the remainder of the design stage and have resulted in a powerful controller.

5.2. Image acquisition

In the initial stages of the design, we wanted to use a USB web camera for the video feed to meet our goals of cost and simplicity. These cameras, however, are often not designed for use in the bright lighting conditions of the outdoors. An attempt to improve the web camera for use outdoors by fitting a standard CS lens mount to the web camera did not produce satisfactory results because the iris was almost impossible to adjust correctly.

We finally settled on using a USB video capture device and an external camera. Though this does raise the cost slightly, it allows more flexibility because there are more choices for cameras and lenses that can be used. The frame grabber that we chose is the Dazzle DVC 80 that provides a capture rate of 30fps at 320x240 24-bit resolution and is supported under Linux. Rather than purchasing a camera for the robot, we chose to use a black and white security camera that we already had from a previous competition. This camera has auto-iris and a wide-angle lens that provides a field of view greater than 80 degrees. Also, the small size and ability to use the 12V straight from the battery made it very easy to incorporate into the chassis.

In software, the image acquisition is done using the Video for Linux (V4L) API. This is a kernel-level set of tools that unifies the capturing routines for all supported devices. By using the V4L API we did not need to change any of our capture code when we changed our input device; we loaded a new driver, and the capturing worked correctly.

5.3. Processing

We had decided early on that we would minimize the amount of processing done on each image. We researched a number of methods for object detection and mapping to see the strengths and weaknesses of the different methods. We ultimately decided against edge detection because it is very calculation intensive, and we have developed another method that produces satisfactory results as described below.

To process a color image, the first step is normalizing, taking each channel and dividing by the sum of the red, green, and blue channels. This results in losing the intensity information (light and dark), but it virtually eliminates shadows and glare, and it makes the frames more uniform from one to the next. Figures 2 and 3 show a test image before and after normalization (in the images each channel was multiplied by 255 to make it visible). Obviously, this step cannot be done for a grayscale image.

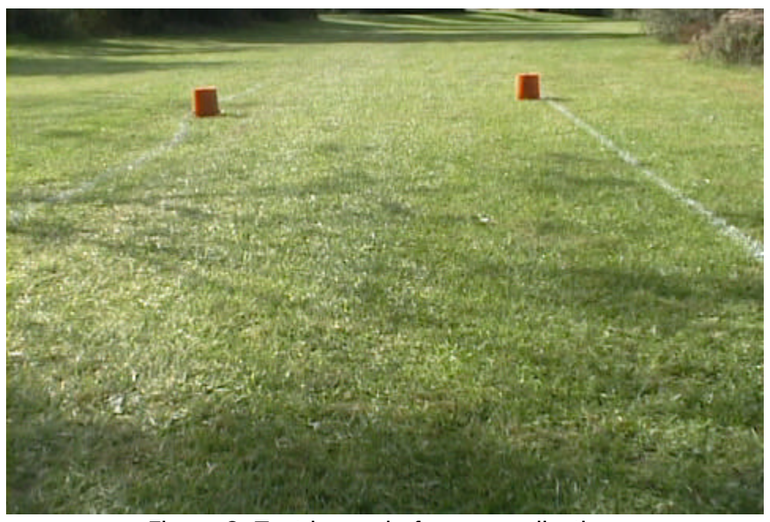


Figure 2. Test image before normalization.

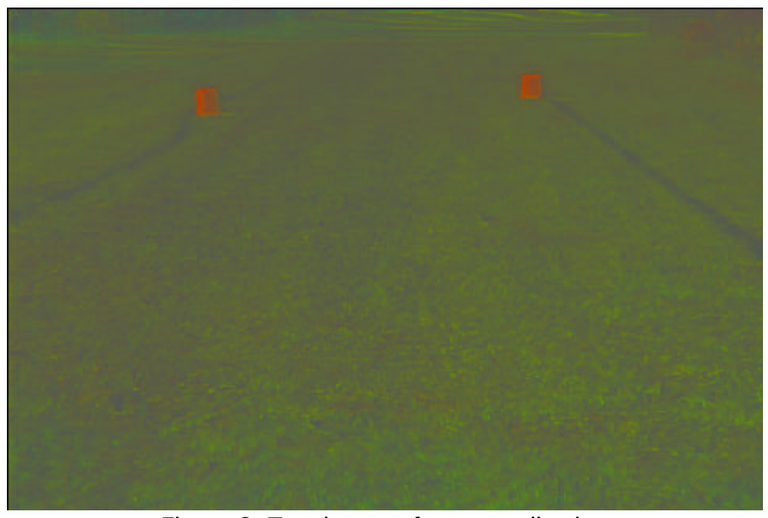


Figure 3. Test image after normalization

The next step in processing the images is image segmentation, the process of dividing the image into regions. Since we are assuming that all the robot will see is grass, barrels, and lines, image segmentation can be easily accomplished by dividing the image into grass, barrels, and lines through a simple process. The orange in the barrels is quite easy to distinguish from the white paint and the grass because it has a very high percentage of red. Through analysis of numerous test images, we discovered that the grass has a high percentage of green and that red is usually the lowest concentration. The lines have generally equal parts of each, though usually, the blue channel is the largest. This suggested a couple of different ways of segmenting, but we settled on a probability model that used the difference between the green and red channels. Pixels centered around 0 were classified as line, pixels with negative values below a threshold (green less than red) were classified as barrel, and pixels with positive values above a threshold (red less than green) were classified as grass. In each pixel, the information stored was a certainty value that reflected the

probability that the information was correct. The number stored was an intensity, and the closer it was to 0, the more probable that it was grass, and the closer to 255, the more probable that it was a line or barrel. A flag for each pixel was used to mark that pixel as a barrel. Figure 4 shows a segmented test image.

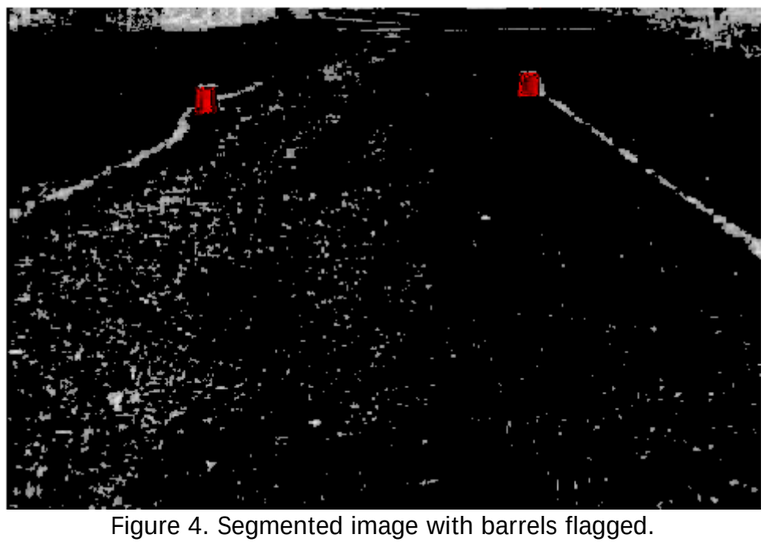


Figure 4. Segmented image with barrels flagged.

With the transition to a grayscale image, the method described above could no longer be used, but we were still able to apply the information that we gathered in developing it. Since the red component of grass is the lowest, we placed a red filter over the lens of the camera. This reduces the amount of light detected from the grass and heightens the contrast between the grass and the obstacles, both lines and barrels, which have higher red light components. With only the single color channel, all that can be done to segment the image is to compare each pixel value to a threshold to determine the probability that it is an obstacle.

5.4. Mapping

Once we have the image, we still have to plot a course through the obstacles. Since we wanted to keep processing to a minimum, we divided the image into 6-inch sections. Each section was evaluated to find the probability that there was an obstacle in that section, and then a spatial map was made. See figures 5 and 6 for examples of the section overlay and resulting map. This is a very low-resolution map, but still detailed enough for our purposes. The advantage of this method is that when plotting prospective courses, there are fewer pixels to analyze. This means that more courses can be tried in less time than if we attempted to plot courses through the full resolution image.

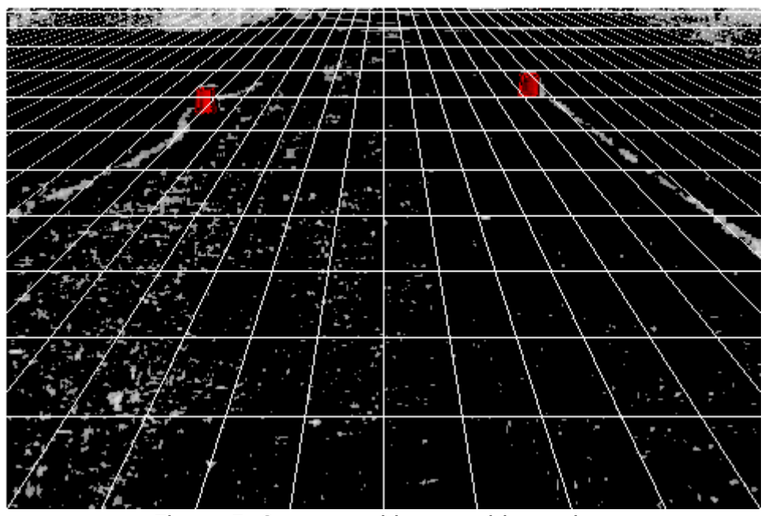


Figure 5. Segmented image with overlay.

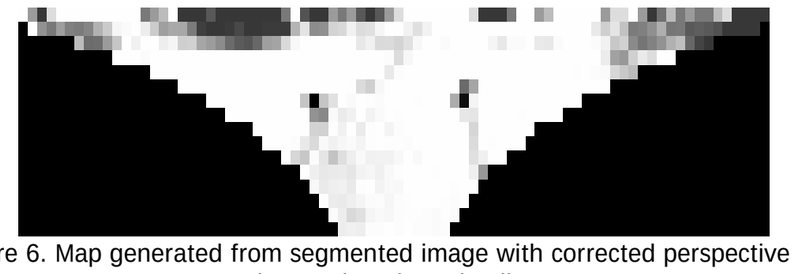


Figure 6. Map generated from segmented image with corrected perspective and inverted to show details.

To determine the best path for the robot, curved paths are plotted through the map and the path with the lowest obstacle total and which is closest to the current course is chosen as the direction to drive the robot. Though very simple, this method works well because we are able to analyze several frames each second, so course corrections happen very quickly. For example, at the robot's top speed of 4.8 mph, and the lowest frame rate of 3 fps, we travel less than 1 ft. before the next frame is completed. With the current camera, we can see over 15 ft. away, so adequate future planning can be accomplished. Dealing with dead ends, traps and potholes has proven to be a major challenge. Forward path planning is hoped to deal effectively with dead ends and traps. Potholes and sand traps require exception type processing and continue to pose a challenge on how best to handle these difficulties.

5.5. Control

Once the course is chosen, we still have to ensure that the robot follows the intended path. The laptop communicates with the electronics interface board over the serial port using a set of custom commands. Data can be sent or requested so that, if desired, the laptop control algorithm can look at the wheel counters, sonar, or other status information. We are using a simple proportional-integral (PI) controller that uses the previous three direction commands to correct the robot. Since the image processing is the feedback part of the loop, the speed of the image processing is crucial to ensuring that the robot behaves correctly.

6. Robot Cost

Item	Unit Cost	Total Cost
Gearmotors*	$450	$900
Motor Controllers	$100	$200
Batteries	$30	$60
Chassis Materials	$75	$75
Pulleys and Belts	$100	$200
Bicycle Wheels	$100	$200
Caster	$12	$12
USB Frame-grabber	$75	$75
Security Camera	$120	$120
Sonar	$50	$150
Shaft Encoders	$5	$10
Electronics Interface Board	$68	$68
Emergency Stops	$30	$30
Power Inverter	$50	$50
Laptop Computer	$1000	$1000
Miscellaneous Parts		$250
	Total	**$3400.00**

* This item was donated to our team by Bodine Electric Company.

7. Conclusion

Our robot, Little Blue, is a small, agile entry into the IGVC autonomous challenge. It has a solid, powerful chassis and drive train that will enable it to drive where it needs to go. By simplifying the image processing and navigation algorithms Little Blue is able to respond quickly to the images gathered during a run and can correct any errors in navigation before it gets trapped or leaves the course. The simplified processing and control will also enable Little Blue to successfully navigate the course at speeds near 4.5 mph. Through this design experience we have faced and overcome many of the challenges in developing fully autonomous robots and we have shown that a simple control approach is adequate, even in moderately controlled environments.

Certification of Design Work Performed on Little Blue by the Cedarville University Design Team

I, Dr. Clint Kohl, Professor of Electrical Engineering at Cedarville University, certify that the members of the Cedarville University engineering design team that created Little Blue have done significant engineering design work on the robot that is equivalent to the work which is awarded credit in a senior design course.

Signed:_____

Date:_____