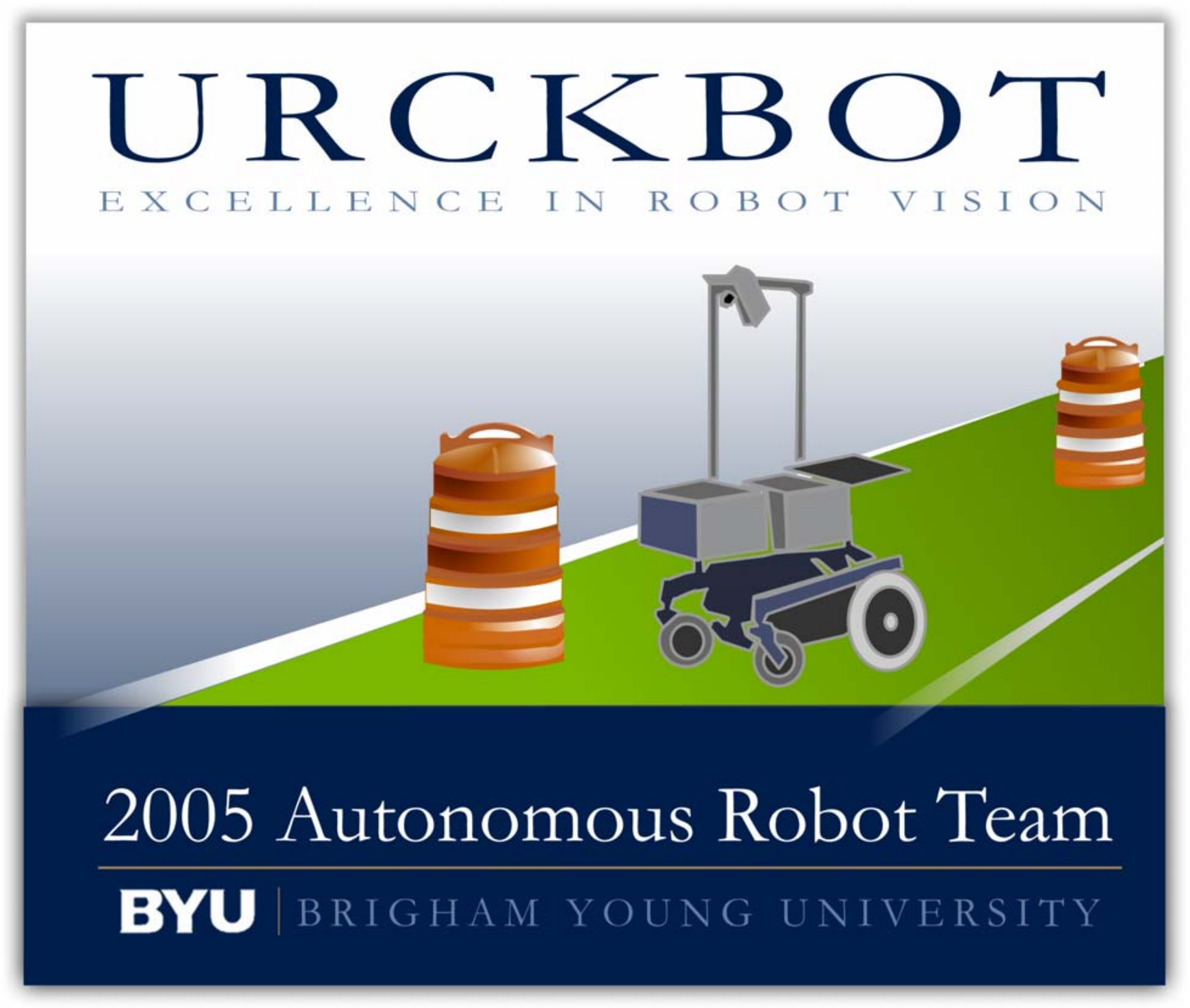
URCKBOT
EXCELLENCE IN ROBOT VISION
2005 Autonomous Robot Team
BYU | BRIGHAM YOUNG UNIVERSITY

TEAM MEMBERS

Jon Anderson, Chris Archibald, Jon Barlow, Tyler Campbell, Chris Kosanovich,
JJ Mejia, Evan Millar, Ben Peterson, Garrett Smith, Jason Smith, Marcus Urie

FACULTY ADVISOR STATEMENT

We, Dr. D. J. Lee and Dr. James Archibald of the Department of Electrical and Computer Engineering at Brigham Young University, certify that the design and development on Urckbot has been significant and that each student performed this work as part of a senior design project and as an extracurricular project.

D. J. Lee James Archibald

1.0 INTRODUCTION

Team Urckbot was the winner of a local autonomous vehicle competition held at Brigham Young University in April 2005. The competition was modeled after the International Ground Vehicle Competition. The competition was the result of a one semester, robot vision senior project class that tested the students' abilities to design, code, and build a robot that fulfills the requirements of the IGVC competition.

The vehicle name came from the television show "Family Matters." In one episode, a character named Steve Urkel builds a robot which he names Urckbot. Urckbot was a completely autonomous robot of humanoid form. Having this name gave us a humorous reminder of the ultimate goal of human-like response to the situations our Urckbot would be placed in.

Urckbot was designed to be cost effective while still maintaining a competitive edge. With limited resources, Urckbot was required to have simple software solutions to the many challenges associated with autonomous robotics.

2.0 DESIGN PROCESS

Designing a complex system like an autonomous robot is no easy task. It requires planning and a clear design process that will allow all team members to contribute, as well as clearly identify the requirements of the project, and help lay a path and schedule to accomplish the goals of the task.

2.1 Identifying Audience and Goals

The audience for Urckbot includes: ourselves, other students in our senior design class, our professors and judges of the BYU-IGVC competition, and the judges and peers at the national IGVC competition in Michigan. The basic goals of the Urckbot project are to design and build a functional, competitive, fully autonomous vehicle for the IGVC competition. We also want our robot to be fun to watch and to educate others about autonomous vehicles.

2.2 Identifying Product Specifications

Many of our product specifications were set for us when we decided to compete in the IGVC competition. These requirements include: size restrictions, e-stop capabilities, strength to carry cargo, and the nature of the tasks that Urckbot must perform. Specifications that we were able to determine were: chassis design, battery longevity, durability, software design, variety of sensors, optimal speed of robot, etc. Each of these specifications was enumerated by us in a design document along with metrics so that we could determine if we had met our goals when the design process was complete. These specifications were a guide to us as we designed, built and tested Urckbot.

2.3 Design Stages

The design process can be broken down into several stages. We organized the team, considered our audience, determined product functionality specifications, generated and selected concepts, designed and implemented the concepts, tested and debugged the implementation of the concept, and then demonstrated the functionality to the rest of them team. This was not a linear process; many concepts were found to be lacking or better ideas were generated as concepts were implemented and tested (see Fig. 1.) Good team communication was vital to ensure that all of the required tasks were completed and function to the specifications. This process on a smaller scale was applied to each task and challenge with which we were faced during the entire design process.

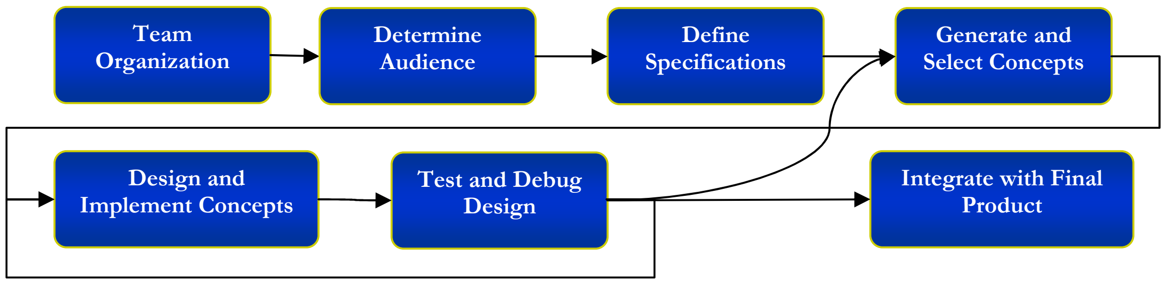
Team Organization
Determine Audience
Define Specifications
Generate and Select Concepts
Design and Implement Concepts
Test and Debug Design
Integrate with Final Product

Fig 1: Urckbot Design Process

2.4 Team Organization

Once we had determined a basic design for the chassis and we were confident that it would provide us a stable platform to run our code, we divided the team up so that each team member was working in an area of expertise. We had meetings at least twice a week to stay synchronized with each other and to keep our goals in focus. Through the combined efforts of all teammates we were able to win the BYU competition and progress towards the IGVC competition in Michigan.

Team Member	Area of Contribution
Jon Anderson	Project Manager, test course design
Chris Archibald	Software design, algorithm design, testing
Tyler Campbell	Mechanical design
Chris Kosanovich	GPS, compass, system integration
Evan Millar	Software design, camera integration, testing
Ben Petersen	Electrical, mechanical design
Garrett Smith	Mechanical design
Jason Smith	GPS and compass integration, motor control
Marcus Urie	Software optimization, testing

3.0 MECHANICAL SYSTEM

3.1 Robot Chassis

The basic design of Urckbot's chassis was the first major design decision faced by our team. We thought long and hard about different base chassis designs that we could use. Our final decision was to use an electric wheelchair base, and to modify it to meet our specifications.

3.1.1 Selection of Chassis

We selected the wheelchair after careful consideration of the options we had. After brainstorming and price-checking, we were left with 3 basic options. We could locate and purchase an electric wheelchair, build a robot entirely from scratch, or utilize an electric golf caddy. As with all of our major design decisions, we constructed a decision matrix, assigned different weights to each requirement of our final product, as determined earlier, and then gave each concept a score from 0 to 6 based on how well it met that requirement (6 being the best). This was a valuable design tool that gave us factual basis for our decision, and helped our team be united as we moved forward.

Figure 2: Chassis selection matrix.	Cost (x4)	Reliability(x2)	Durability	Construction Difficulty	Add-on difficulty	Development time (x2)	Difficulty to implement safety	Time to obtain	**TOTALS**
Purchase Wheelchair	*6*	*6*	*6*	*6*	*4*	*6*	*5*	*6*	***75***
Construct our own chassis	3	3	3	0	5	1	5	1	**34**
Purchase Electric golf caddy	0	4	4	6	1	3	4	1	**30**

3.1.2 Modification of Chassis

After purchasing the wheelchair we brought it back to the University for testing and modification. We determined that the vehicle needed new wheels and two new batteries. The wheelchair ran on 2 12-volt deep-cycle batteries. The team purchased two deep-cycle batteries from Costco and two new tires from a local healthcare accessories store. To modify and streamline the vehicle for the competition, all unnecessary parts were removed including the seat, leg supports, and seatbelt. (see Figure 3.)

After the wheelchair was stripped, the team added two large weatherproof electrical boxes that were donated by an electrician. The boxes were bolted to the vehicle for stability and to house the electrical components of the robot. The rear box holds the motherboard, which was mounted to a piece of Plexiglas. After the motherboard was mounted, the power supply was mounted in the rear box so it would be close to the motherboard. The power supply converts the 24-volt battery supply given by the deep-cycle batteries into an array of different voltages required by the different robot components. In order to prevent wires from shorting out, the team fashioned a piece of sheet metal that would secure all loose electrical ports. The sheet metal was bolted over a large hole that was cut into the side of the rear electrical box. All of the ports were secured onto the sheet metal and can now be accessed from the outside of the box.

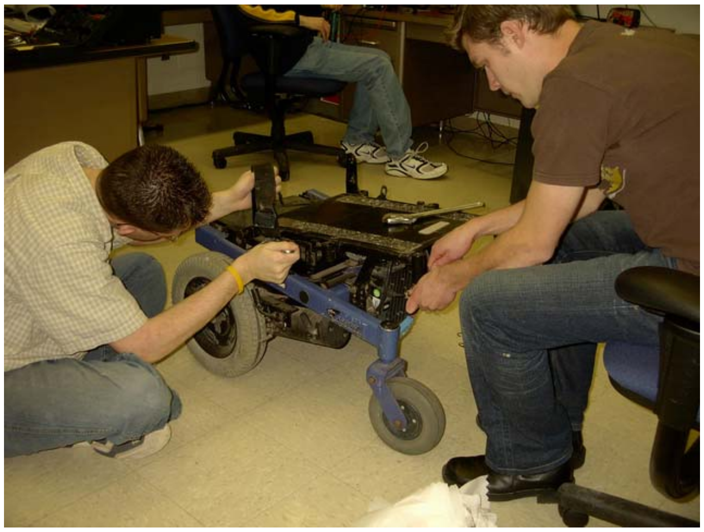
Figure 3: Stripping the chassis of unnecessary parts

The final mechanical additions that were made to the vehicle involved the camera. A waterproof camera enclosure was made from a plastic container. The camera was mounted on precision-machined plastic that was then glued onto the plastic container. The camera lens sticks out of the enclosure, while the uncovered electrical circuitry is protected inside the box. The camera box was fitted to the camera mount, which was welded from aluminum piping. The camera mount design allows the camera angle to be adjusted which allowed us to experiment and determine the optimal camera angle for Urckbot.

3.2 Safety

In order to ensure that Urckbot would be a safe vehicle, we constructed a platform out of Plexiglas at the rear of the robot, upon which we mounted our emergency stop. We mounted it at a height that would provide easy access and put it in an obvious position, so that we could stop Urckbot promptly and surely in the event that it needed to be stopped. The emergency stop button itself is red and very sturdy, providing a stable and safe way of bringing Urckbot to a complete halt almost instantaneously.

4.0 ELECTRICAL SYSTEM

4.1 Sensors

As a fully autonomous robot, Urckbot requires the ability to gather information from its environment in order to make decisions. We selected sensors for Urckbot based upon price, necessity, practicality and reliability. Urckbot makes use of a color video camera, a global positioning unit, and a digital magnetic compass as its sensors.

4.1.1 Camera

The onboard video camera gathers information used by the software. In the autonomous competition Urckbot relies entirely upon information from the camera. In the navigation competition the camera is used to detect obstacles. Because of this, a reliable color camera was required. Urckbot uses the SenTech STC-630c series CCD camera. This camera delivers an NTSC color video signal, supports disabling of auto white balance, could output either S-video or composite video signals, and is powered by a 12 Volt DC source.

4.1.2 Frame Grabber

The Sensoray 311 Frame Grabber was selected to convert the signal from the camera into digital information that could be used by the software. The Sensoray 311 has many characteristics that make it a good fit for the robot's purposes. It accepts both S-video and composite inputs, which make it compatible with the camera. It also uses a PC/104+ bus that can connect directly to the robot's motherboard. The Sensoray 311 also allows for image scaling (the ability to scale the x and y dimensions of the video frames), which allows for greater flexibility in working with different size images. Another major benefit of the Sensory 311 is that the board came with both linux and windows drivers. The interface between software and the frame grabber is also very simple. The Sensoray 311 came bundled with a simple API that allows a frame to be requested through a C function.

4.1.3 GPS

There are a wide range of GPS units available on the market today. We required a GPS unit that was within our budget, accurate to the specifications of the navigation competition, simple to interface with, and reliable. Given these requirements we selected the CSI Wireless MiniMax. Despite its cost in relationship to other GPS units, under ideal circumstances, it is capable of less than one meter accuracy 95% of the time. These circumstances include having the appropriate signals from the satellites orbiting the planet as well as from the WAAS navigation buoys provided by the U.S. Coast Guard.

4.1.4 Digital Compass

A digital compass is necessary for the navigation competition in order to determine Urckbot's heading, so that we can accurately locate the GPS waypoints. We selected the PNI V2Xe

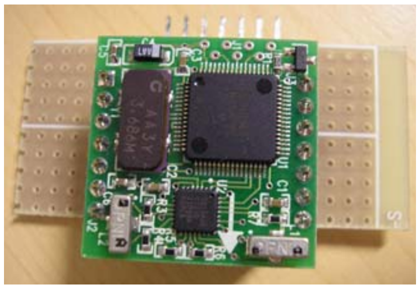
Figure 4: PNI V2Xe compass in place on the carrier board.

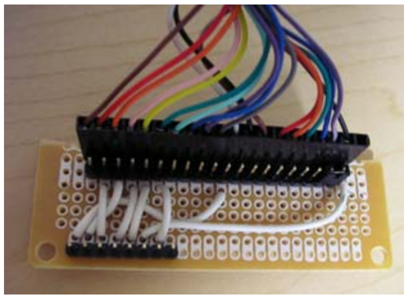
Figure 5: Pass through board

digital compass because of its cost. Interfacing with the digital compass proved to be more difficult than expected. The PNI V2Xe compass shown in Figure 4 uses a Motorola SPI bus for communication. Since Urkbot's computer does not have an integrated SPI bus, the compass was physically interfaced to the computer through the parallel port. The compass is powered through the parallel port pin because it has a small current draw.

To allow the parallel port to be used for other components at a later date, a small circuit board was built which used only those pins required by the compass and leaving the rest open. This circuit board is shown in Figure 5.

4.2 Control

An important consideration for the team was controlling the wheelchair motors from our software. We took apart the wheelchair's joystick and determined what voltages were being sent to the motor controller and what they did. After consideration, we decided to implement two Pulse Width Modulation (PWM) channels utilizing a Microchip PIC16F688 processor. This processor communicates with the computer via a serial port, allowing an easy interface to the calling software. Each channel simulates an axis of the joystick. A design decision that was very innovative and paid big dividends in terms of ease during testing, was the decision to leave the joystick intact, and merely have a switch that determines whether the motor controller gets its input from our processor or from the joystick. This enables us to drive Urckbot manually as we position it for testing, and then flip a switch and be driving from our software.

4.3 Safety

The emergency stop button was wired so that it cuts power to the motor controller of the wheelchair. A receiver for the wireless emergency stop performs the same function when the transmitter button is pressed. When there is no power to the motor controller, the motor controller locks the wheels of the wheelchair. This was one advantage to using the motor controller already in place on the wheelchair. This method has proven to be reliable and very effective.

5.0 SOFTWARE DESIGN

5.1 Software Platform

All of the software for Urckbot was created in the C programming language. We chose this language because most of the team members were very familiar with it, and it also gave us the control over serial ports and parallel ports, so that we could communicate with our sensors as we desired. The motherboard that our does all of our computing utilizes an Intel Pentium 3 processer.

Because it is optimized for Intel processors, the Intel OpenCV computer vision library was employed to give us fast and proven image processing tools at our fingertips.

5.2 Obstacle Detection and Avoidance

In order for the robot to react to different types of obstacles in the desired manner it was necessary to be able to recognize these different obstacles in each video frame. Instead of trying to detect actual shapes the team decided to focus on using the colors of the obstacles for obstacle detection. For this we needed to be able to process each image in such a way that we could extract the color information from each image. The end product of this process is a binary image that has a 1 in each pixel where there is something of interest (an obstacle, white line, or pothole), and a 0 where there is nothing of interest.

5.2.1 Orange Barrel Detection

One major type of obstacle to be recognized is orange barrels. After extracting the orange, we want the image to tell us where the orange was in the picture. To extract the orange from the image, we first convert the image from RGB color space to HSI color space. We then copy each channel of this 3 channel image into an image of its own using an OpenCV function designed for this purpose. The H image contains the hue information for each pixel in the image. We then use another OpenCV function that converts all the pixels whose hue falls within a region we specify (around the value of pure orange) to a 1. We also found that orange barrels had distinctively high saturation values. We applied a threshold to the S image to get all the pixels that are above the requisite saturation level. Now that we have the two processed 1 channel images, we multiply them together, so that we will see a barrel only where we had the right hue and saturation values. This

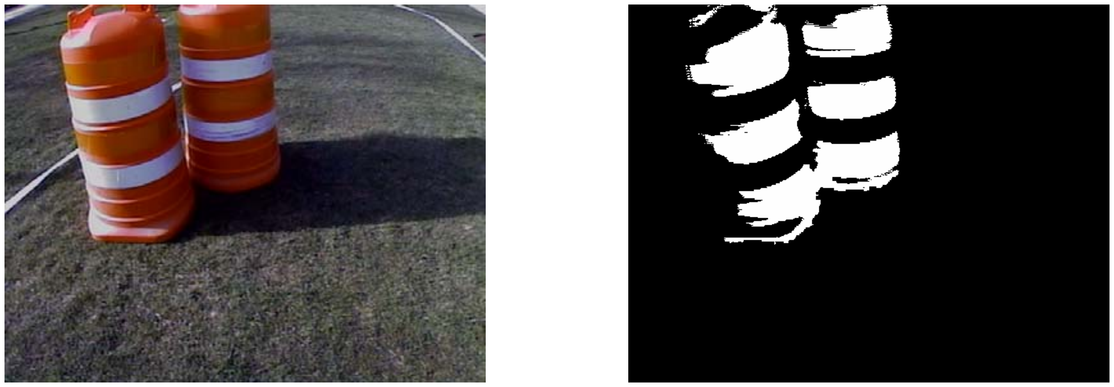

Figure 6: Original and orange barrel segmented image.

multiplication prevents us from recognizing orange in the grass as a barrel because it does not have high saturation, and also does not allow us to recognize things with high saturation as barrels because they do not have the right color of orange. The threshold values that we used for this color segmentation process were determined through experimentation. To detect white obstacles, including potholes, we processed the white image in the manner detailed in the following section, and then added that image to the orange-processed image.

5.2.2 White Line Detection

The first type of obstacle that we addressed was the white lines. The goal of color segmenting the image for white lines is the same as with the orange barrels: to produce a binary image that contained white pixels where the white lines were, and black pixels everywhere else. To accomplish this we again converted the image to HSI color space and then looked for the pixels that were a certain threshold above the average intensity of the image. This allows us to handle more variations in lighting conditions and still pick up the white line. By adjusting this threshold value, through trial and error, we were able to reliably extract the white lines from images. An example of a white line segmented image compared with the original image is shown in Figure 7.

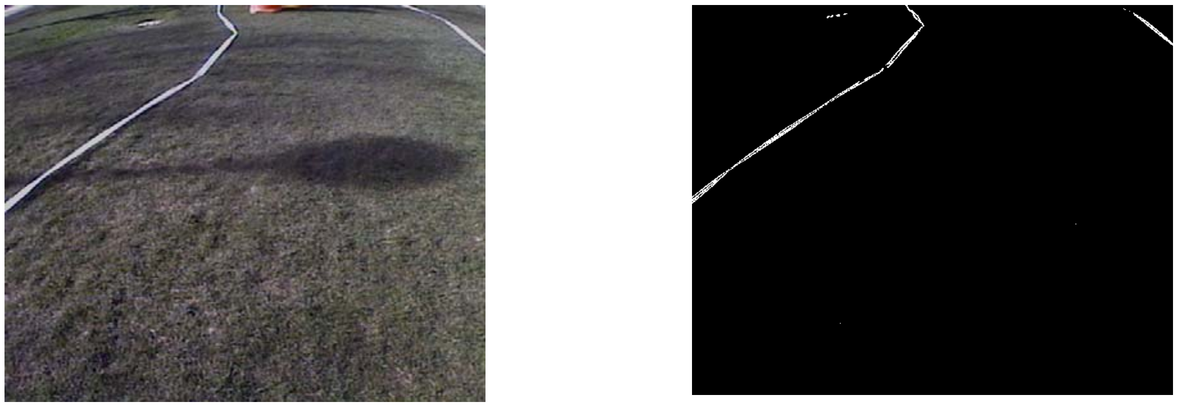

Figure 7: Original and white line segmented image.

5.2.3 Dashed Line Detection

To get complete and accurate white line information, in the event that there is a dashed line in the image, we constructed our own function that looks at the filled squares in our 10 x 10 grid, and determines if any 3 of the squares are situated on a roughly collinear path. If this is the case, we extend the line until it touches the edge of the image or is superseded by a white line closer to the robot. We have tested this method with various dashed line configurations it has performed well.

5.3 Autonomous Competition

Once both the orange barrel image and the white line image are obtained, Urckbot needs to decide where to go. During the design process, we thought extensively about a motor control strategy that would take advantage of the strengths of our "joystick" style motor controller interface. We also wanted a simple design that would be easy to debug and that would be easy to change as our project progressed. This allowed us to easily add functionality to avoid more obstacles and to deal with more difficult barrel configurations.

The basic algorithm that we came up with to use our strengths was to move forward constantly while examining the images as quickly as possible. This allows the robot to detect obstacles and lines and steer so as to avoid them. A turn command is generated from both the obstacles and the lines, and these values are combined to give us the overall turn command for the robot. The combining of the obstacle and white line commands proved to be one of the most formidable tasks in our path-planning process. We had to consider all the different combinations of commands that we could have. This included commands that differed in direction, severity, or both.

The process of obtaining a turn command from a binary image is similar for both binary images. The following explanation will only refer to the white line image and resulting command. Each image is divided into a ten by ten grid (see Figure 8.) Using OpenCV's *cvCountNonZero* which counts the number of non-black pixels in a region

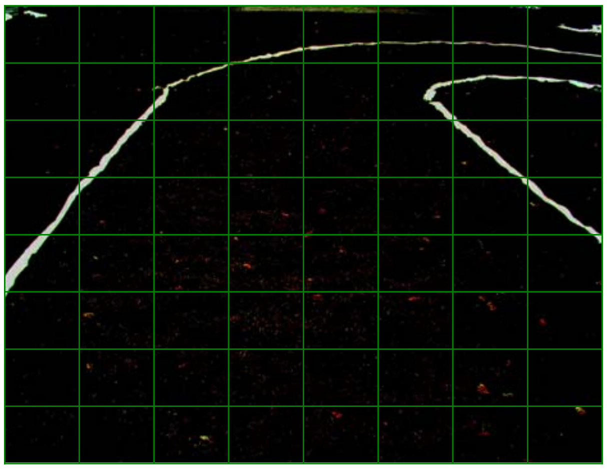
Figure 8: White line binary image with grid (8x8).

of the image, each cell in the grid is marked as occupied by a white line or not. Each cell in the grid has a turning magnitude associated with it. By observing which cells are marked and the turning magnitudes associated with each marked cell a desired turn magnitude is generated. The magnitudes that were assigned to each cell were determined experimentally with the largest magnitudes assigned to the cells near the bottom parts of the image since the robot will have to turn hardest to avoid obstacles immediately in front of it.

To determine our obstacle command, we determine where the white line is in relation to the edges of the obstacle, and then we determine which side of the obstacle has the most space between it and the white line. We then choose to go around the obstacle in that direction, which helps us to avoid potential traps and trouble spots.

5.4 Navigation Competition

The GPS unit and digital compass are used along with the camera during the navigation competition. The software selects an order for the waypoints to be visited, and we visit them in that order. For each waypoint, we get our current GPS position, and determine the heading that we need in order to go in the desired direction. We then get our current compass heading from the digital compass, and from that and the desired heading, figure out the direction that we need to turn in order to have our heading be as desired. Once a direction is determined, we get an image from the camera and process it to determine if there are any obstacles in Urckbot's path. This is done using the same image processing techniques as used in the autonomous competition. The obstacle avoidance code returns two values, corresponding to how hard we have to turn due to current obstacles, depending on whether we want to go to the right or to the left. The navigation software determines from its turn command and the obstacle avoidance commands the optimal command that will point Urckbot in the right direction and not collide with obstacles. We then move forward with a constant velocity and the turn command thus determined.

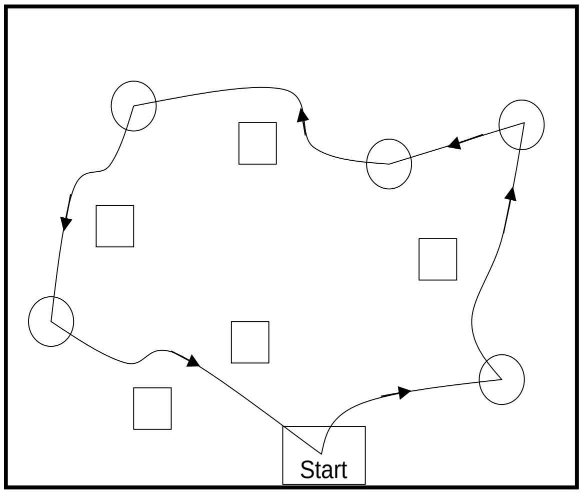

Figure 9: Sample navigation route.

An innovative design feature of Urckbot's navigational system is that after avoiding an obstacle, it reevaluates which waypoint is most accessible and should be pursued. This enables us to react dynamically to obstacles that steer us towards a closer waypoint (see Fig. 9.)

5.4.1 Waypoint Detection

During the navigation process, if we get a GPS reading and it is within 2 meters of the waypoint, we mark that waypoint as visited and move on to the next waypoint in the list. Once we have visited all of them, we return to the start box and the program is at its completion.

5.5 Safety

For safety reasons, we designed the software so that the closer the robot is to an obstacle or white line, the slower it goes. This resulted in very smooth, controlled, and safe performance.

6.0 Predicted Performance and Testing

6.1 Battery Life

Battery life is a critical issue in the IGVC competition to insure proper functionality of components throughout the entire course. It was also important to us as we tested Urckbot, because we wanted to have as much time as possible to test, without having to wait for batteries to be recharged. The wheelchair chassis that we used is designed to give optimal battery life to the people who use it, often for a whole day at a time. Urckbot was expected to have reduced battery life, due to the computer, camera, and GPS units all consuming power. We predicted that Urckbot would be able to run for up to 8 hours on a single charge, and through testing we found that to be the case. This longevity was a great boon to us, enabling long testing sessions during which much was accomplished. We also purchased an automatic charger designed for the wheelchair, and this enabled us to plug Urckbot in at the end of one day, and find the batteries fully charged by the next.

6.2 Speed

The capability to drive as quickly as our processing capabilities would allow (up to 5 m.p.h) was a major design concern. With the choice of the electric wheelchair chassis as our base we were able to get differing speeds by changing the voltage values we send to the motor controller, as well as set the maximum speed by turning a speed dial on the exterior of the wheelchair joystick. During testing we were able to reach speeds of up to 5 m.p.h., and we can travel at any speed up to that. This flexibility will be a big advantage for us in getting the best time possible out of our robot.

6.3 Ramp Climbing

In order to compete in the IGVC, we had to be able to climb a ramp with at least a 15% grade. The wheelchair frame used in our design comes with that capability, being originally designed to transport persons up wheelchair access ramps. This was one reason that the wheelchair chassis was so desirable. In testing, Urckbot was able to successfully traverse ramps of 15% grade (see Fig. 10.) It did so with minimal strain, and it is expected that Urckbot could successfully ascend and descend ramps with significantly more slope.

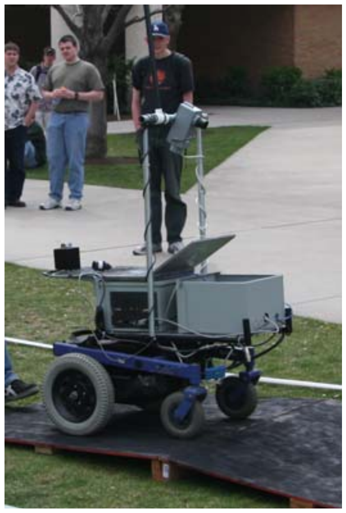

Figure 10: Urckbot climbing ramp.

6.4 Processing Speed

We wanted to be able to process enough information so that we could drive fast enough to be competitive in the autonomous and navigation competitions, and also so that our reaction time to obstacles and white lines would be adequate. Since our autonomous algorithm relies solely on the image data from the camera, frames per second was our metric. We got the autonomous code to process about 5 frames per second, which allows us sufficient time to react to changes in our field of view, even at a speed close to 5 m.p.h. This result was achieved through use of the OpenCV code library, as well as optimization of the software that we wrote.

6.5 GPS Accuracy

In order to successfully compete in the IGVC navigation challenge, we must be able to locate GPS coordinates within 2 meters of accuracy. This level was accuracy was one of our motivations for purchasing the selected GPS unit. During testing, we routinely came within 1 meter of our target waypoints, which will ensure that we get within the 2 meters specified.

6.6 Safety

Safety was a big concern throughout our entire design process. The first thing we added to Urckbot was its emergency stop capability. We have also separated electric components so that their wires could be less tangled, and we made a metal plating that housed all of the connections from the motherboard, so that we would be able to access it safely. The emergency stop has been tested thoroughly, being used every time we test the robot, usually every test run, and Urckbot has always come to a stop immediately. We also designed our software so that we drive slower and in control when near obstacles or white lines so that we do not pose a safety risk to any objects or people. We are confident that Urckbot is a safe robot, meeting the specifications for safety as outlined by the IGVC guidelines.

7.0 COST ANALYSIS

Vendor	Part	Part Number	Qty	Total Cost	Team Cost
Modular Industrial Solutions	Mother Board	LBC 9453 Embedded Board Computer	1	$732.00	$732.00
Sentech	Camera & Lens	STC-630CS	1	$341.00	$341.00
Sensoray	Framegrabber	Model 311	1	$190.00	$190.00
Navtech GPS Supply	GPS	MiniMax	1	$1899.00	$1899.00
Invacare	Wheelchair	Arrow Storm Series	1	$7365.00	$28.00
PNICorp	Compass	V2Xe	2	$75.00	$150.00
CostCo	Batteries	Deep Cycle Batteries 75 A/hr	2	$45.00	$90.00
	Chassis		1	$224.00	$145.00
Petersen Medical	Wheels		2	$60.00	$120.00
Totals				$10,931.00	$3505.00

8.0 CONCLUSION

Urckbot has been designed to compete on an autonomous course as well as a navigation course using vision as its primary sensor. It was also designed to be a cost effective solution to the problems posed by the IGVC competition. By utilizing both hardware and software, Urckbot will be able to avoid obstacles and function to the specifications of the IGVC, and will be able to compete in all 3 categories of the 13th annual IGVC.