University of Illinois, Intelligent Ground Vehicle Robotics Team

Rajarshi Roy, Bhaskar Vaidya, Jason Lefley, Kevin Armstrong, Urvashi Khandelwal, Adana Pappas, William Tadekawa, Lam Vu

{roy18, bvaidya, lefley1, armstr13, khndlwl2, tadekaw2, anpappa2, lamvu2}@illinois.edu

IGVRT
attention: Kevin Armstrong
1715 Westhaven Drive
Champaign, IL 61820-7051

2012 I.G.V.C. Competition

May 10, 2012

Faculty Statement: This report with the description of its vehicle, IGVRT, underwent construction and design by the Intelligent Ground Vehicle Robotics Team that occurred during the 2011-2012 academic school year with effort on the magnitude of a senior design capstone course.

_____

Dr. Timothy Bretl

Contents

Overview

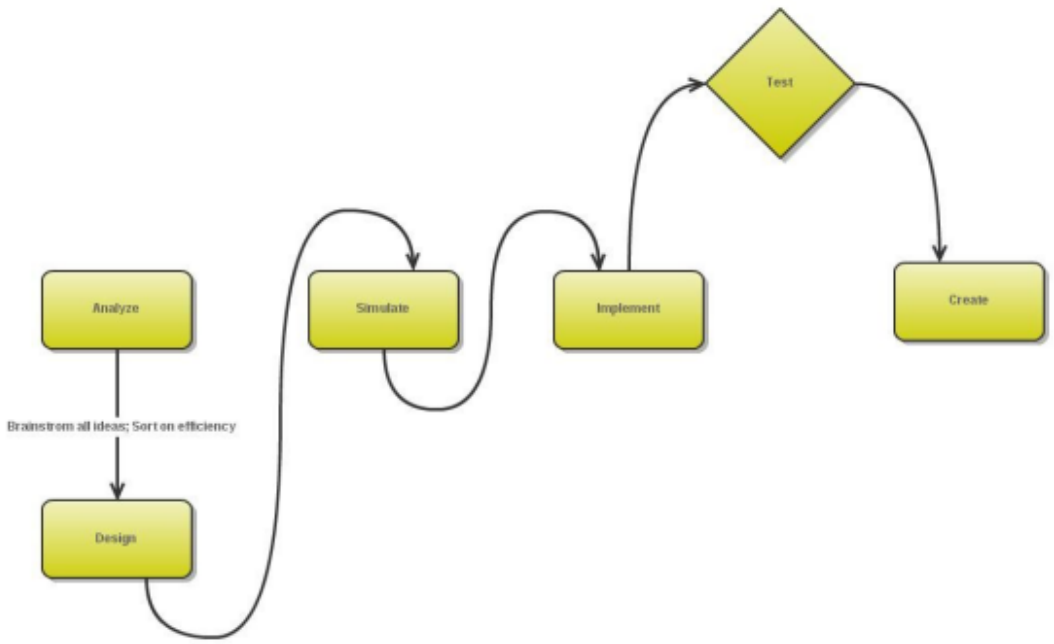
The Intelligent Ground Vehicle Robotics Team is the first of such robotics teams formed

at the University of Illinois, Urbana/Champaign, and consists of dedicated members from Electrical Engineering, Computer Science and Mechanical Engineering, all undergraduate students. We approached the current task of building a robot for this competition by grouping our team into four primary sub-teams: Artificial Intelligence, Computer Vision, Embedded Systems, and Mechanics. While there are courses for undergraduates at the University touching briefly on the topic of robotics, none go into the depth that this competition requires, and thus provides team members with the unique opportunity to teach themselves many new skills that they would not necessarily get from their course of study.

Our team, though only consisting of eight undergraduates, had a unique opportunity to share knowledge amongst engineering students at various different points in their study. With four of the eight undergraduates being freshmen, we upperclassmen had a two-fold challenge: working on our vehicle, and stimulating the interest in this competition and the pursuit of self-knowledge amongst a younger generation. As one will discover upon reading our report, our team underwent many revisions, with engineers fading in and out of the picture. However, the steady foundation of our team has been the eight individuals displayed on the cover of this report. Due to irregularity in our initial team structure, we all eventually began getting our hands wet in all four of the sub-teams. The Artificial Intelligence sub-team served as our algorithm machine, churning out the Hough transform for camera image accuracy, to the Gaussian filter structure for obstacle movement. Although steep in algorithm analysis, the Computer Vision team served as the overall coders and merged the A.I. team's algorithms to see what effects might result from their flow in one machine. Embedded systems focused on the placement of sensors and how we were to detect incoming obstacles, either with a camera, a set of sensors, sonar, or all of the above. When it came to the physical structure of our vehicle, our Mechanics team proposed a variety of different structures, eventually going with the most cost-efficient design as elaborated in Section #3.

Design Process

The process which our group produced our vehicle adhered to the following flowchart:



We studied the 2012 IGVC competition requirements and constructed a plan tailored to fit our newly established group. Our efforts primarily focused on achieving the following tasks:

1. Operate within a $3500 dollar budget
2. Have our vehicle carry a 20 lb payload
3. Max speed no more than 5mph over grass
4. Configure a GPS system to less than 1m error
5. Use a wireless e-stop for our vehicle
6. Distinguish between objects based on color and size
7. Differentiate between colored lane markings
8. Efficient running time algorithms
9. Scale for a 20% incline
10. Design/construct within two semesters

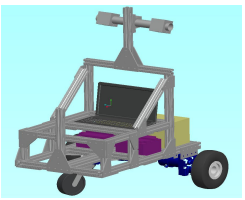
Our primary focus was cost efficiency. While our vehicle's design may not be the most aerodynamic, we worked under a limited budget, and thus required our team to think beyond the physical scope and simplify our hardware design to make use of used physical frames that would qualify as a legal vehicle for the competition. Objectives 5 through 7 focus on sensing and embedded systems described in Sections 4.1 and 4.2. Our GPS and compass system, objective #4, is described in Section 4.5. The software structure on how our algorithms were chosen is described in detail in Section 4.3. Mechanics objectives #2, 3 and 9 are described in the following section.

Hardware

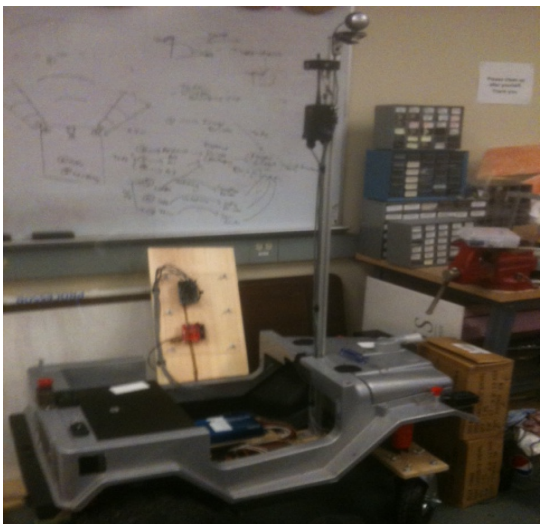
The IGVRT, as our vehicle is titled, comes from a gutted PowerWheels toy that had remained in storage with one of our team members' homes. The vehicle takes on the following dimensions:

Measurement assessed	Measurement in respective units
Width	36 in
Distance from ground	11 in
Height (including camera stand)	72 in
Weight	~165 pounds
Front wheels' diameter	10 inches
Back wheels' diameter	22 inches

The camera stand was based on an 80/20 framing system, a system that was initially used for our IGVRT's first sketch in the AutoCad software, Pro-Engineer, shown below:



Pro-Engineer, or more affectionately referred to as ProE, is a 3D CAD system created by Parametric Technology Corporation. The software specializes in three main functions: engineering design, analysis and manufacturing. Not just a digital representation of our robot, ProE provides us with analysis tools that allow us to test our robot design under thermal, dynamic, static and fatigue conditions. Below, is a present (March/April 2012) representation of our vehicle:



As one may notice, we scrapped our 80/20 framing system for the body of the vehicle, with exception to the camera stand. We can generate tests on center of mass and weight under various conditions before completion of vehicle with 20lb payload. Below is a list of the parts we used to create IGVRT, the robot:

Item	**Cost**	**Cost for Team**
Amp snap connectors 300V insulated	$1	$1
Radio Shack 250 Volt, 1 Amp fuses	$2	$2
PHOPHET PLUS Battery charger	$85	$26
Silicon bridge rectifier	$2	$1
Sparkfun.com GPS Evaluation	$40	$32
Arduino XBee series 2	$26	$21
Sparkfun.com sonar sensor	$26	$26
Pololu OJ2907 cmp01a, compass	$30	$30
HORNET Remote Control E-Stop	$40	$40
Arduino MEGA board	$60	$50
PEG Perego Gaucho Grande Car frame	$500	$0
2 front wheels, 10“ diameter	$20	$20
Power Patrol 12 V battery	$60	$50
hp ProBook laptop	$500	$250
LV MaxSonar-EZO	$30	$30
2 motor controller batteries, 8.4V	$80	$60
RoboteQ DC Motor Controller	$600	$600
Serial coupler	$38	$10
Serial USB converter	$29	$11
Misc. mechanical supplies	$300	$300
Camera	$30	$12
Total	**$2499**	**$1522**

Hardware Techniques

As a new organization at the University of Illinois, our team had to work under a limited budget. Fortunately, funding support from John Deere allowed our team to purchase the RoboteQ DC motor controller and HP ProBook, enabling us to assemble and program the basic embedded systems portion right away. Since one of our former teammates had an old Perego-Pro toy car sitting in his garage, we saved on the cost of constructing an entire 80/20 framing system, and purchased only the camera stand. Much of our funding for sensors and velcro assembly came from the later formed Illini Robotics Organization, which worked with Microsoft and Engineering Design Council at the University of Illinois to equip us with funds for insurance, entry fee, transportation to and from the competition, along with various necessities like the wireless E-Stop and camera.

Even with the provided Perego-Pro toy car, we had to disassemble much of the inner workings of the vehicle in order to satisfy the minute turns and twists of the IGVC obstacle course. We removed the front two wheels, not connected to the motor, and purchased two smaller wheels about 10“ diameter, roughly 11“ smaller in diameter than the original wheels. This allowed the IGVRT greater maneuverability and turn at a smaller degree while still having sturdy wheels that will endure muddy conditions.

We created a wooden platform with hinges for our HP ProBook to rest on and also serves as a purpose of protecting the motor and batteries that power it from variable weather conditions. Our sturdy velcro system to hold motor, batteries and laptop allows for a more modular design and greater access to space within the vehicle if needed. We have a 12V lead acid battery that sits inside the “hood” of the car. We removed the accessory features, including the original batteries that powered the system and replaced with our own, more industrial 12V battery. This battery powers the back wheels, which run on a motor, gearbox, and gear and chain later described in this section. The embedded systems such as the front sensors, sonar, camera, compass, GPS and Arduinos are powered by a separate RoboteQ DC controller that has two speedpower battery packs, each at 8.4V. We acquired the HP ProBook for half the price because we found a used one online and refurbished it while adding more RAM for our processing demands.

Mechanical Design

Although briefly highlighted earlier, our mechanical design is unique in that we re-used material and transformed it into the autonomous vehicle, IGVRT. Given that the payload of two children well exceeded the 20lb payload, even on outdoor terrain, the Peg Perego contained the perfect motor to power the back wheels. However, just to be on the safe side, we simulated the motor, knowing the gear to chain ratio to be 134:1. We assumed that the gearbox efficiency was 90% in our calculations for each of the following cases:

Case	Power Required	Motor Power	Power per side
Full speed forward on grass	60W	61.2W	30.6W
Static rotation on grass	20W	40.88W	20.44W
forward up 20 degree incline	300W	360.36W	180.15W

The PEG Perego motor uses a 12V battery supply, and with the additional RoboteQ DC motor controller powered by a collective total of 16.8V allows the vehicle to achieve a maximum power of 28.8^2 or 829.44 Watts. By adding this motor, we can add to the vehicle’s payload a heavier lead-acid battery amongst other devices like sensors while still being able to power the

20 lb payload up a 20 degree incline.

However, this analysis would not have been complete without a technical examination into our theoretical gearbox ratios and wheel diameters. By doing so, we needed to construct a power curve for the motor, using the measured values: nominal voltage, stall current, stall torque, no-load speed, and no-load current to determine the motor power curve. Based on our 134:1 gear ratio, we chose 10“ diameter wheels for the front and maintained the 22“ diameter back wheels powered by a 130 : 1 planetary gearbox and a 20: 1 chain drive. As seen in the provided test cases, this will permit more than enough power to suffice the three tested conditions. Yet, even with safe gearbox, chain drive and gear ratios, we still lose some energy from friction. Fortunately our 250 V, 1 Amp, and 250 V, 10 Amp fuses protect our circuit from exceeding the power rating of our motors and our devices. Our speed did not exceed 10 mph, nor did it stagger below 1 mph, as one can see from the aforementioned three cases:

Case	**Speed** (m/min)	**Total Power** (Watts)	**Motor Power** (Watts)	**Current** (Amps)
1	34.64	61.2	30.6	2.5
2	30.12	40.88	20.44	1.71
3	34.64	360.36	180.15	15

In order to calculate the acceleration we used the velocity, calculated from analysis and experimentation, with the equation acceleration=0.5v^2. As one can see with both our predicted analysis performed on Matlab and our results measured during our test run, our robot appears to be under the 10 mph limit.

Electrical Design

Our electrical design requires an entire orchestration amongst the sensors, camera, and Arduino feed. The delineation between Software and Electrical Design is remarkably thin, as we discovered in our design process. The sensors feed into the programmed drivers in the Arduino, which depend on the filtering of the camera feed, which are then used by the A.I. to determine a direction. This stacked structure begins with the sensor feeds, dependent on a 1 V signal sent to the Arduino. For this portion of the electrical design, we will articulate the specifics in section 3.7. The blend of computing and the electronic components, their delineation in addition to their shared components, are expressed in greater detail in section 3.6.

Each of the components used on the vehicle IGVRT have a certain power rating that cannot be exceeded. Also, each component has a specific power requirement in order to perform its given function. Below is a brief chart that outlines the power requirements for each system of components:

Avg. Component Power (Watts)	**Peak (Watts)**	**Idle (Watts)**
Driving Straight (50 % duty cycle)	30.6	0
Turning (45% duty cycle)	26.4	0
Climbing (5% duty cycle)	180.15	0
Drivetrain Weighted Average	79.05	0
GPS	3	1.5
Compass	3	1.7
Camera	3	1.3
Sonar	3	3
Sensors Total	12	7.5
Computer	90	90
Wireless E-Stop	5	5

Misc. Electronics	10	8.84
Total	**354.15**	**111.34**

However, the hardware is powered by our 12 V battery and its circuitry guarded by the larger 20 Amp fuses. By maintaining this power range, we are able to keep IGVRT running at a minimum of one hour with continuous running upwards to two hours in idle state.

Fabrication and Assembly

For most of our fabrication stage, we already had the shell of our vehicle and back wheels crafted by the manufacturers at PEG. Our subsequent ProEngineer CAD drawing allowed us to simulate the proper parts that would be necessary for IGVRT. For the camera stand, we used an 80/20 framing system, allowing us to secure hinges with relative ease and little damage to the aluminum support. For the interior of the vehicle, we drilled a hinge attached to a wood panel that covered what was once the foot area for the vehicle. This wood panel can lift up and out, and return to rest across the foot area, covering the inner RoboteQ DC motor controller. This wooden panel protects the inner devices from excessive rain damage and allows us to set testing equipment on the panel without damaging the motor controller or its batteries. Our minimal use of external framing parts allowed us to focus on spending our money on a quality GPS, camera, compass and sensors for the vehicle.

Testing and Redesign

Our first setback came early in the design phase came when a major sponsor ended up withdrawing funding for our team due to re-structuring of their R&D program. This required our team to scrap our initial ProEngineer CAD design displayed earlier and re-formulate the structure based around a PEG Perego car framing system. This creative thinking from our team ended up reducing our original mechanics budget of $4500 to under $1000. This savings really came from the expense of an 80/20 framing system for the entire vehicle versus a free (yet much modification required) PEG Perego framing system.

After we sorted out our car frame issues, the next difficulty came with the camera angling. Initially, we had the camera angled downward at 45 degrees. However, this required programming the Hough transform and reduced reliability by as much as 30%. Further analysis of our camera can be found in section #4.

Electronics and Computing

We used an HP ProBook customized with extended RAM to perform the necessary memory large processing commands from the RoboteQ motor controller and the Arduinos. These same commands govern the wireless E-Stop along with instructions that regulate the movement of the back axle gearbox. The HP ProBook has AMD Dual Core processors that allow such instructions to be conducted between all devices on the IGVRT. Our increase in RAM went from the standard 2GB 800 MHz SDRAM to 100GB 800 MHz SDRAM, allowing more cache memory for actions being processed. Attached to the side of IGVRT is a small breadboard with an 802.11g WIZnest hardwired TCP/IP ethernet controller. This little device allows reliable transmission up to 25 Mbps, and has a simple interface to work with reducing developing time. This device requires only 3.3V for operation and a 5V I/O signal tolerance. Using the WIZnet with the customized laptop makes it possible for us to also enter the JAUS competition.

Sensors

We used a combination of sonar sensors from sparkfun.com that we knew worked well

with the Arduino boards, along with an optical camera to measure the surrounding environment. The sonar sensors were ideally placed, one on each side where car headlights would rest, and one in the center between the other two sensors. The placement was used based on range and width of signal. The center sonar sensor was used as redundancy to confirm any immediate objects the vehicle might approach on its course. The camera has a range of 25 meters, with a minimum distance of 2 meters, allowing for the sonar sensors to kick in and guide IGVRT away from the obstacle. Without the sonar sensors, there would be a "dead zone", where any obstacle would remain undetected by IGVRT, and most likely crash into them. We will explore the exact layout and specifications in section 4.3.

In order to keep track of its specific location, IGVRT comes equipped with sparkfun.com's GPS evaluation board. This GPS evaluation board allows interfacing with USB over the FT232RL and RS232 interface. The laptop also serves to power this board via USB with 5V. The GPS evaluation board also comes with special, programmable software, allowing us a more modular working of the GPS with various supporting devices. One of these supporting devices used on IGVRT was the Pololu comp01a compass and accelerometer. With a voltage range of 2.2-5.5V, the Pololu compass fits nicely with the GPS evaluation board, not over-drawing power. The specific range for the magnetometer is: ±1.3, ±1.9, ±2.5, ±4.0, ±4.7, ±5.6, or ±8.1 gauss. For the accelerometer, it is:±2, ±4, or ±8 g (accelerometer). Given this wide set of ranges, it will allow IGVRT a more precise course of navigation. Both devices were chosen because they interface well with the Arduino board network. The software component for both the GPS and compass is further explored in section #4.3 with the sensors.

Safety

Part of our safety system with our hardware was the 10Amp and 20 Amp fuses to prevent an accidental short circuit from overloading the rest of the mechanical devices with power. We also incorporated a wireless E-Stop system that can be activated both on the vehicle and from a distance of 120 feet. This E-Stop has a manual location next to the laptop towards the center-rear of the vehicle, and is three inches in diameter. Both disabling mechanisms work in the same manner, breaking the circuit connected to the RoboteQ motor controller. Given that the competition max speed is 10 mph, IGVRT has a max speed well below that, and therefore does not need to use specific hardware stops for speed regulation. Also, the IGVRT has a safety light that turns a solid red color when the power is on, and begins to flash in autonomous mode. After coming out of autonomous mode, the light returns to a solid color. The LED light is regulated by a signal from the laptop to the breadboard, indicating power is on, and also issues a signal when the IGVRT has engaged in autonomous mode.

Testing and Simulating Obstacles

While we did not have time to assemble, test, and simulate switchbacks, center island dead end traps, and potholes, we did run some depth perception code that we used in the assistance with determining complex obstacles. With respect to depth perception, we used an optical camera and a filtered the image with a set of weights to identify depth levels on the terrain surface. The image below articulates this process in action:

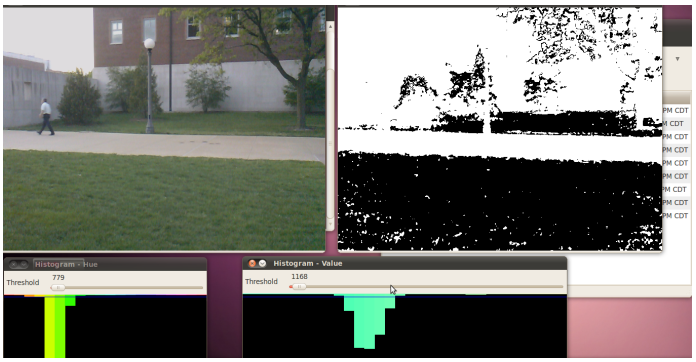


Figure #1: Here, one can see that objects of different colors are distinguished between each other.

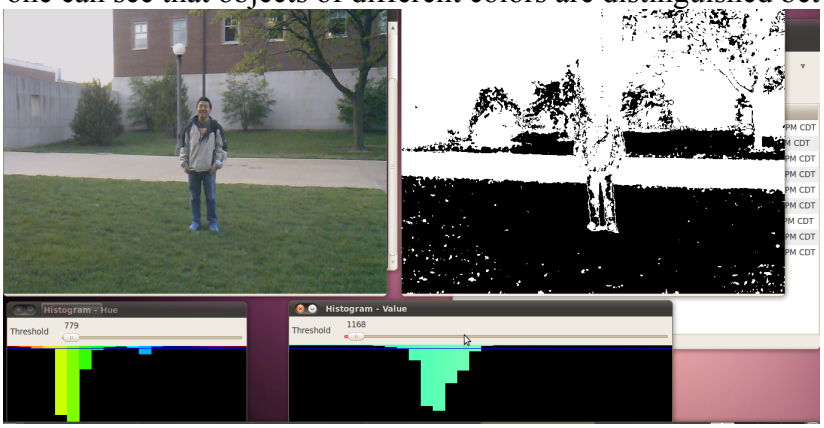


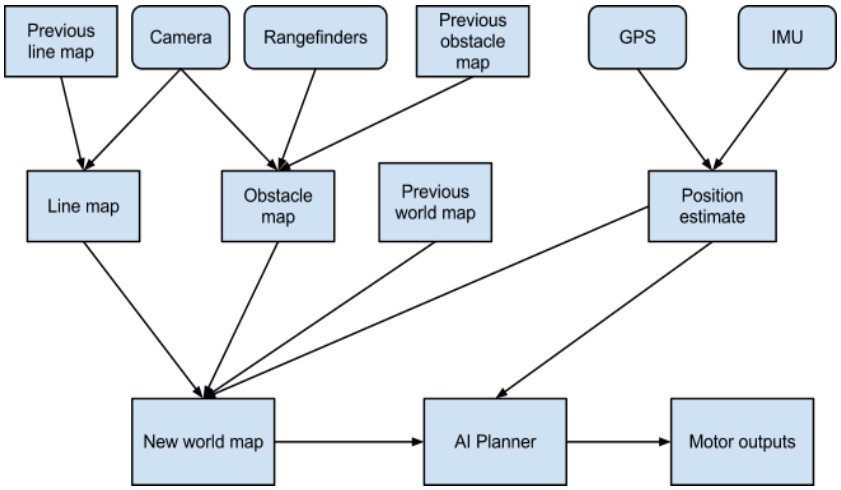
Figure #2: We also tested objects of a more complex magnitude, like a person.

In both pictures we can see a person identified, one far away and one up close, with figures 1 and 2 respectively. In each case, the relative depth (or rather the height) of the object is identified on the histogram with turquoise bars, and is used in a global mapping mechanism.

Figure #3: As one can see, even at an angle, the object's position and depth is pinpointed by a weight per location on the image.

Software

Software Schematic



- Line map(camera, previous map)
    - Filter out colors and focus on anything white
    - Run a line/curve matching algorithm
    - Connect line segments with best guess that may be obscured by obstacles. Also takes into account lines found in previous frames/previous line map.
- Obstacle map(camera, rangefinders, previous obstacle map)
    - Find blobs of color that is not grass (color histogram).
    - Group blobs with the aid of openCV library
        - http://opencv.willowgarage.com/wiki/cvBlobsLib
    - Correlate camera and rangefinder data
        - If camera sees something but rangefinder does not, it is probably a pothole, so the 2d plane transform will give the exact bounds of it on the ground.
        - If camera and rangefinder see something, it is probably a barrel, so group the barrel into a blob, find where the barrel meets the ground, and use the 2d plane transform to calculate where the barrel should be marked on the ground. We do not want it to mark the entire barrel as being on the ground because that would extend well behind the actual obstacle.
    - Correlate previous obstacle/blob data with current frame to possibly match the same blobs.
- Position estimate
    - Kalman filter state machine to deal with error. This is included with the driver.
- World map(line map, obstacle map, previous world map, position estimate)
    - Fit these maps together by using the change in position, and possibly by rotating/ shifting and template matching the map images to find the best fit.
    - Combine the maps by fading out values from older map versions
        - new map = 0.4*generated map + 0.6*old map
        - We do not want new map to dominate the world map, since new map may have temporary errors
- AI planner(world map, position estimate)

    - Threshold the world map values into a binary map (>0.5 = 1, else 0).
    - Floodfill out from every obstacle point the width of the robot + some safety constant. The center of the robot can now be treated as a single point.
    - Pick a short-term goal position a few meters down the course between the white lines.
    - Run A* between that position and the current position.
- Motor outputs
    - Transform the path given into a series of turn and drive commands.

Camera Transform

For the robot IGVRT, we needed to angle the camera approximately 30 degrees below horizon in order to gather an accurate mapping selection of obstacles. Since this angle causes distortion, we used a transform that creates a 2D image of the field in front of the robot. The image transform appears accurate up to ten feet in front of the robot and generates a map that is compared against the sensor map detection described in section II. Below is the pseudo-c algorithm that is implemented by the Hough Transform:

```
int min q;
int max q;
For min q = q to max q do[
        For min p = p to max p do[
        S(p,q) = 0;
        For y=L to U do[
                x=round(q+((p-q)/(U-L))(y-L) ]
        For n=x-w to x+w do[
If I(n,y) = 1 then S(p,q) = S(p,q) + 1;
                ];
        ];
            ];
 ];
```

As on can see, this iterative process is dependent on q's boundaries. These boundaries are directly correlated to the edge of the picture frame that the camera sees in its image process.

Mapping

In order to generate an accurate map of the field in front of the robot, we used two sets of mapping: camera (optical) and sonar sensor (infrared). The camera uses a transform to create a 2D image that undergoes a filtering selection. This filtering selection applies a contrast based on the depth of the obstacle. For potholes and uneven terrain, a depth perception is critical for IGVRT to interpret for its decision-making navigation. After the filtering weight is applied, a certain hue grouping mechanism smooths out pixels that share a similar weight using nearest-neighbor selection. Thanks to OpenCV library, we were able to create a variable threshold with python that would allow us to adjust this grouping scheme threshold through a GUI interface.

GUI Grouping Threshold

In addition to optical mapping, there is a second map created via infrared sensors. This

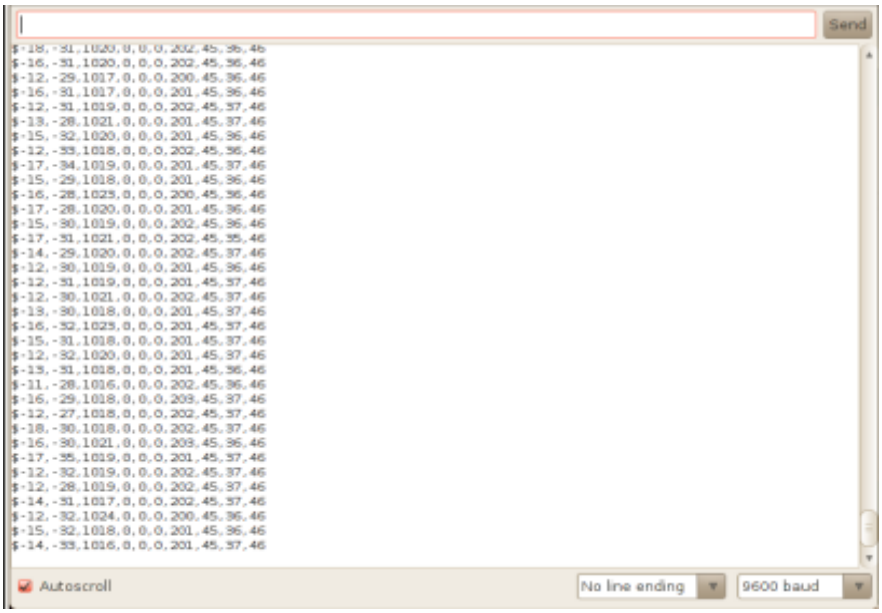
map is a 2D map that displays erected obstacles within less than a foot to twenty-five feet away from IGVRT. The sonar points are then grouped together into single objects and then mapped on to a global map. This global map combines both the optical and infrared maps for accuracy. This global map then uses the downloaded obstacle map along with its GPS compass to pinpoint exactly where IGVRT is on the course.

Waypoint Selection

IGVRT's waypoint selection scheme simply extends the boundaries of the lane in a continuous manner outwards. This is achieved by placing two web cameras on each side of the 80/20 framing system (to the left and right of the center camera). These two cameras measure where the lane markings are to each side of IGVRT.

Navigation

IGVRT uses its sonar sensors to detect obstacles directly in front of IGVRT up to 256 inches away. During our testing of the sonar, we discovered that at certain nodes, the sensor alignment seemed to super-impose the distance an object was away from the sensors. These nodes existed between the left sensor box and the center sensor box, as well as between the right sensor box and the center sensor box. Below is a screen shot of our data results from testing the sensors:



The last three numbers are referring to the three sonar readings: center, right, left. As one can see, the sensors are picking up an obstacle roughly 50 inches away from the sensors. Each number refers to the distance in inches from its respective sensor.

Conclusion

We believe that IGVRT will perform with success, given the present software design and hardware ingenuity. We pride ourselves on designing an effective robot that will traverse the course with under such monetary constraints. Much of the knowledge acquired for building this vehicle did not come directly from coursework, but rather from external educational resources outlined in the appendix.

Appendix

- Na, Yu. "Orientation Codes-based Template Matching Method on Workpiece Detection." *Power Electronics and Intelligent Transportation System (PEITS), 2009 2nd International Conference* 3 (2009): 70-73. *IEEE Xplore*. IEEE, 20 Dec. 2009. Web. 19 Mar. 2012. <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5406896&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5406896>.
- Rahman, Shahedur, Abu Syeed Md. Zakaria Shah, and Gill Whitney. "Computer Vision Based Navigation System for the Visually Impaired." *SIGGRAPH '04 ACM SIGGRAPH 2004 Posters* (2004). *ACM Digital Library*. ACM, 2004. Web. 28 Feb. 2012. <http://dl.acm.org/citation.cfm?id=1186489&bnc=1>.
- Duffany, J.L. "Artificial Intelligence in GPS Navigation Systems." *Software Technology and Engineering (ICSTE), 2010 2nd International Conference* 1 (2010): V1-382-1-387. *IEEE Xplore*. IEEE, 5 Oct. 2010. Web. 10 Mar. 2012. <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5608862&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5608862>.
- Bradski, Gary. "OpenCV Library." *OpenCV Wiki*. Wikipedia, 13 Feb. 2006. Web. 19 Apr. 2012. <http://opencv.willowgarage.com/wiki/>.
- Chien-Cheng, Tseng, Cheng Hsu-Yung, and Jeng Bor-Shenn. "A Lane Detection Algorithm Using Geometry Information and Modified Hough Transform A Lane Detection Algorithm Using Geometry Information and Modified Hough Transform." *18th IPPR Conference on Computer Vision, Graphics and Image Processing* (2005): 21-23. Print.