Faculty Statement

I certify the "Wunderbot 4" has made substantial improvements in the areas of theory, hardware acquisitions, and software control from the previous "Wunderbot" entries into the IGVC competition.

Joseph T. Wunderlich, Ph.D
Project Advisor
Physics & Engineering Department
Elizabethtown College

Wunderbot Project
One Alpha Drive
Elizabethtown College
Elizabethtown, Pennsylvania
17022
ph: 717-361-1000
fx: 717-361-1400
www.etown.edu

Table of Contents

Team Members:
James Painter, CE&CS
David Coleman, CE
Jeremy Crouse, CE
Chris Yorgey, EE
Mike Patrick, CE
Dan Fenton, CE

Project Advisors
Joseph Wunderlich, Ph.D.
Thomas Leap, Ph.D.
Troy McBride, Ph.D.

1. Introduction

The Robotic and Machine Intelligence (RMI) Laboratory at Elizabethtown College is proud to announce its third entry – the Wunderbot 4 - into the 2008 Intelligent Ground Vehicle Competition. Coming off a mid-field showing in the 2006 competition, the Wunderbot 4 team has made significant improvements in the area of obstacle avoidance, GPS navigation, and white line detections. While the main chassis may bring back memories of the Wunderbot 3 (2006 competitor), the new acquisitions of vision, LIDAR, and digital compass – along with complete redesign of internal software including implementation of JAUS – are steps in the right direction for the versatile platform that will serve the Elizabethtown, PA community in web-based autonomous tours in the near future.

A listing of major developments since the 2006 IGVC onboard the Wunderbot platform is listed in Table 1. A thorough documentation of the overall design process, hardware implementation, software development, as well as IGVC challenge solutions will be discussed within this paper.

New for 2008	
Chassis	4' vertical tower for camera GPS/Compass integrate mount on tower Waterproof attempt
Hardware	Front-mount LIDAR DVT camera JAUS access point Remote control (for manual drive)
Software	PID control Sub-system partitioning JAUS implementation White-line detection Obstacle avoidance O^3 GPS navigation scheme

Table 1 – Wunderbot platform developments since 2006 IGVC.

2. Innovations

The Wunderbot 4 features a new navigation scheme known as O^3 for GPS navigation and obstacle avoidance within GPS navigation which alters the optimal sort explicit definition of GPS coordinates as provided by the IGVC judges. The O^3 method was published in March in the IEEE proceedings of the Advanced Motion Control Workshop (available in Appendix B). By using the three stages of O^3 (**o**ptimal explicit path planning, local points of **o**pportunity, and **o**bstacle avoidance) a path between two GPS coordinates is not chosen based of local availability, but rather in a global context and the relationship between the GPS coordinate and obstacle density in the vicinity. A sample course is shown in Figure 1.

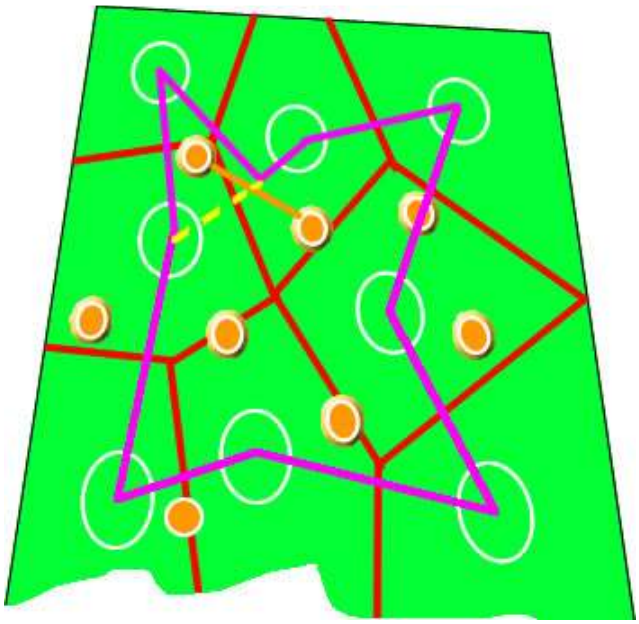


Figure 1 – Sample GPS waypoint challenge course with coordinates outlined in white. The chosen path is shown in pink with the O^3 ability of "path deviation" shown in a yellow dotted line.

2. Design Concepts

The primary focus of the Wunderbot 4 team was to improve on the systems currently onboard the Wunderbot 3 platform while developing new algorithms and techniques to solve the challenges faced at the IGVC 2008. A constant process of evaluating the current system, proposing new solutions, developing these solutions (combining theory and simulation results), implementing the solution with technology donated by corporate sponsors, and testing the solution to provide reassurance of quality, controllability, and proof of concept is shown in Figure 2.

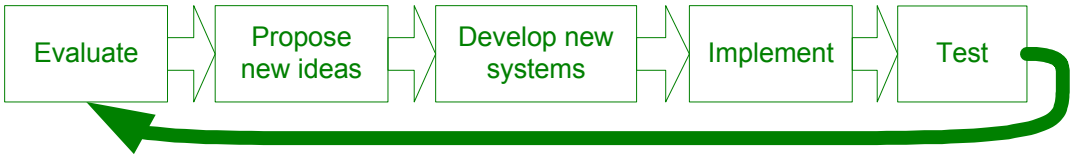


Figure 2 – Design process governing the transition from Wunderbot 3 to Wunderbot 4.

2.1. Evaluation – The IGVC 2006

The success of the Wunderbot 3 team in 2006 proved the systems developed to-date were at or above par to the rest of the competitors. However, there was significant room for improvement specifically in three areas: 1) obstacle detection - which was being handled by minimal processing on-board the camera – 2) GPS navigation – no optimal path between multiple nodes was present and 3) JAUS communication – no attempt was made in 2006. The complete evaluation period lasted about six months and the design process was about one year to develop and write the necessary items. The team's Gantt chart along with estimated 2000 man-hours is shown in Table 2.

Task	Class	Description	2007 Aug	Sept	Oct	Nov	Dec	Jan	2008 Feb	Mar	Apr	May	Length
1	Hardware	Design utility pole in SolidEdge											1 wks
2		Install utility pole											2 wks
3		Expand electrical system											8 wks
4		Install new systems (LIDAR, Vision, JAUS)											10 wks
5	Software	Review all code from Wunderbot III											18 wks
6		Design vision software											12 wks
7		Implement vision software											14 wks
8		Design JAUS interface											12 wks
9		Implement JAUS software											14 wks
10		Design LIDAR system											6 wks
11		Implement LIDAR software											18 wks
12		Design path planner											16 wks
13		Simulate working path planner											14 wks
14		Implement path planner software											12 wks
15		Troubleshoot entire system											4 wks
16	Club	Fundraising for Wunderbot 4											4 wks
17		Presentations											4 wks

Table 2 – Wunderbot 4 team Gantt chart showing concurrent works throughout the 30 weeks over two semesters of 2007-2008 academic periods.

2.2. Constraints on Wunderbot platform

By working with the Wunderbot 3 platform a few variables in robotic design were already guaranteed from previous successes including maneuverability, mobility, versatility, and safety as outlined in [1]. Additionally, the team balanced resources,

aesthetics, and functionality to develop the new systems as outlined in Table 3. Finally, the team reviewed the constraints of the IGVC as well meeting additional requirements of Elizabethtown College and travel abilities.

Description	Actual	Constraint	Placed by
Vertical Tower for Vision	5’	Under 6’	IGVC
Horizontal Rear Length for Vision	5’	3’-9’	IGVC
Front bumper	28”	36”	Door width in RMI Lab
Wireless E-stop	Yes	Yes	IGVC/RMI Lab
Waterproof	90%	None	Need

Table 3 – Constraints placed upon Wunderbot 4 design

2.3. Chassis Improvement

Two needs were addressed with the addition of a 4’ tall tower on the rear of the robot: 1) more stable GPS receiver mount and 2) more focused viewing region for the camera. The tower was designed in SolidEdge with the specifications shown in Figure 3. The tower is constructed from one-inch square galvanized steel piping at a height of 4’ (122 cm). It was spray painted black to match the look of the robot and to prevent rust. The tower has one shelving unit approximately halfway up the tower, which is protected by plexiglass and houses the GPS receiver and digital compass. At the top of the tower at 40.5cm behind the rear bumper the camera is mounted at an adjustable angle. The angle of camera is adjusted in a coarse manner by its angle mount and finely adjusted by its setscrews in the housing unit. More information on the camera mount can be seen in section 4 and Appendix C.

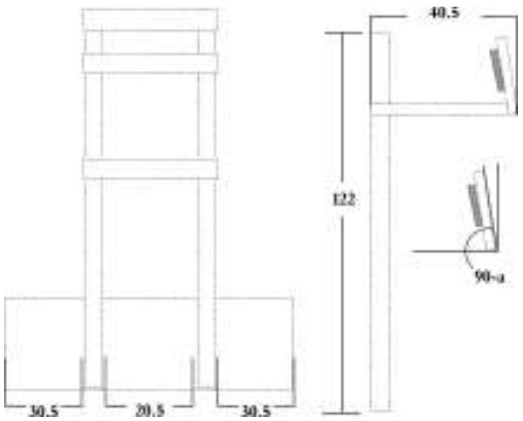


Figure 3 – Tower design measurements. The camera can be adjusted coarsely by its mounting unit and finely by its setscrews. This angle is marked by ‘a’.

3. Electrical System

The majority of the electrical system was carried over from the previous Wunderbot 3 platform with minor changes including 1) wider-conduit to hide additional communication cables, 2) increased distribution power blocks, and 3) increased number of fuses. A complete breakdown of the electrical system can be seen in Figure 4.

3.1. Power

Two 12V 60-amp hour batteries connected in series provides approximately two hours of operating time. A 480W 24V DC-DC ATX power supply provides voltage regulation for the onboard PC and all system components.

3.2. Control

Due to the castor resistance affecting the turning radius of the Wunderbot, this year's team made the decision to reverse the orientation of the drive system. It now features a rear-wheel drive controlled by two independently controlled RobotEQ AX2550 motor controllers. These controllers communicate via RS-232 and parameters are passed via LabVIEW. The RobotEQ motor controllers also feature safety parameters such as

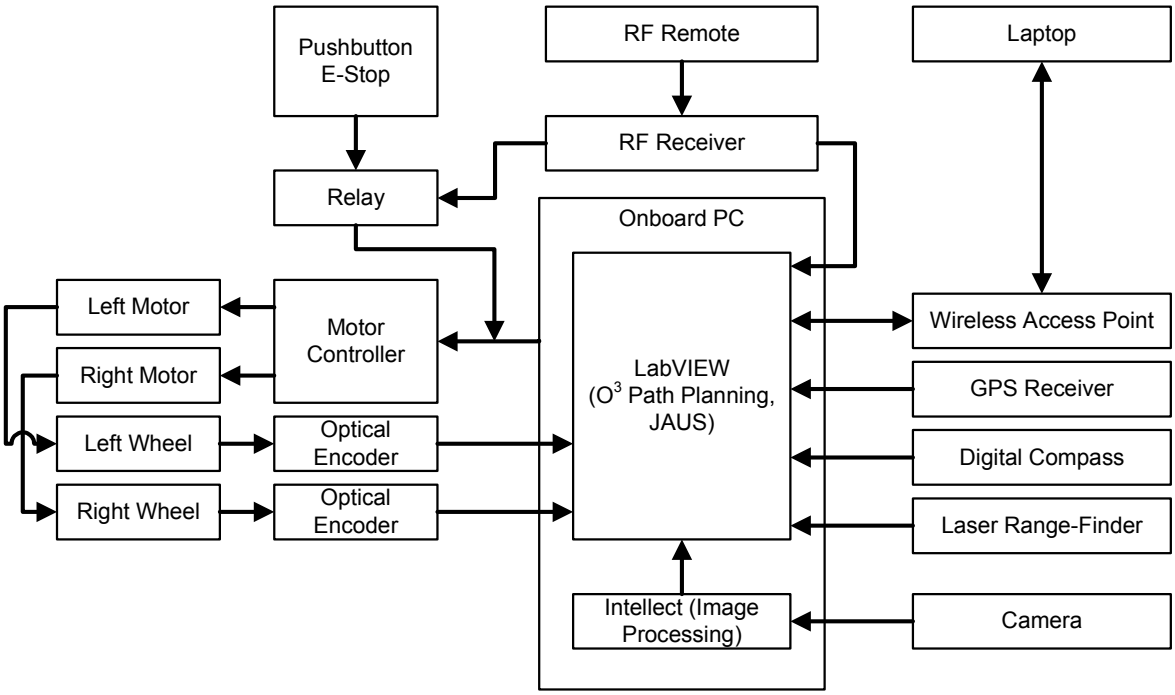


Figure 4 – Block diagram of Wunderbot 4.

temperature, battery voltage, and current draw to ensure proper performance and damage to equipment is prevented.

One of the most costly problems encountered during early testing was inaccurate vehicle motion. When operated on smooth indoor surfaces, Wunderbot was able to move roughly in the intended direction, but once the vehicle was tested outdoors on grass, motion response had a large degree of error. The largest cause for error is the front casters, which require a disproportional amount of force in order to change direction. This problem is a typical case for a PID controller to amend.

The PID closed-loop control was developed in LabVIEW and is very straightforward. The P, I, and D are all user-adjustable via the front panel, and feedback comes from the U.S. Digital optical encoders. Unfortunately, the robot itself is extremely difficult, if not impossible, to model via differential equations, hence classic methods of control theory could not be instituted to determine the value of the PID's constants. Instead, it was a trial-and-error procedure, which led to P=0.500, I=20.00, and D=0.001. Very subtle variations in the derivative constant led the robot to accelerate out of control. A PID controller's derivative constant in general is highly susceptible to noise, and therefore an adjustable low-pass filter was designed for the D. This kept the D from fluctuating too rapidly, while still allowing it to quicken the output's rise time.

3.3. Emergency Stop

The Wunderbot 4 now features four ways of stopping the robot: two hardware and two software. The onboard hardwired e-stop normally open button will instantly ground the motor controllers and motion will only be activated with a program reset. This same relay is available wirelessly through a remote switch.

With the addition of a remote control for manual drive an additional emergency stop was introduced and is controlled initiated through software. By activating this e-stop button the control program (LabVIEW) will immediately abort the program and the communication between the PC and motor controllers will be eliminated; thus stopping

the robot. Finally, anytime the remote LabVIEW program is stopped the robot will halt its motion and therefore, would be considered another e-stop method.

4. Software Strategy

The largest change made from competing in 2006 was the software strategy onboard the Wunderbot 4. The team used the previous code as a guide and first developed a base flow diagram of how the decision should be made within LabVIEW. This flow diagram is shown in (Figure 1 of) Appendix A and is elaborated upon within this section.

4.1. Vision System

The location at which the camera is mounted has enormous impact on the image-processing step of the vision system. Various configurations were tested, comparing the field of view and corresponding image processing times. With the camera 1.2m directly above the rear bumper and 40.5cm back, the viewing distance extends to roughly 2.25m, missing some data from directly in font of the front bumper, as seen in Figure 5.

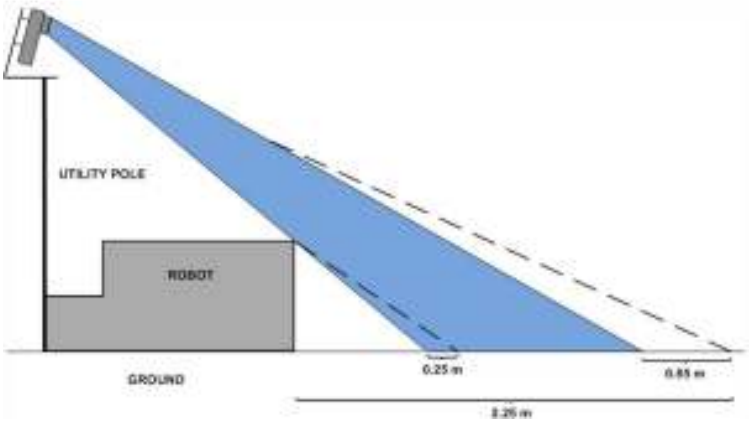


Figure 5 – Camera angle versus distance of sight. The blue region represents that of a fixed angle at a position equal to the rear bumper. The dotted lines represent the distance gained by moving the camera back 40.5 cm.

4.1.1. Signal Processing

The sacrifice of image data from in front of the bumper is acceptable, due to the trade off between seeing farther ahead and trimming the top edge of the image to reduce processing time. A feasibility study on the processing time reduction resulting from cropping the top edge of the image showed that the decrease in processing time was significant enough to allow for image cropping. The percentage speedup was a nearly linear relationship to the percentage of the image that was

trimmed, and the final implementation incorporated the cropping of the top 200 lines, for a reduction of about 110ms in processing time, to about 90ms.

4.1.2. Line Detection

The line detection is performed from within the camera's proprietary software, DVT Intellect v2.2. First, an erosion filter is applied to the image, using a 3x3 kernel. This closes many holes of noise, such as small dirt patches that appear through the grass, while still maintaining the shape of the desired white lines. Larger kernels could produce an even more accurate image, but processing times increase sharply as the kernel grows larger. Once noise has been filtered, an Intellect ``line thickness'' sensor is applied. This measurement sensor first uses a 60% intensity threshold to deduce a binary image. The sensor then scans every row in the image to find the two edges closest either side. Final line pass/fail conditions are used to filter shadows and other undesirable objects in the field of view. A maximum width condition of 300 pixels is combined with a ``straightness'' condition.

4.1.3. Path Planning

The data from the camera software is then sent to LabVIEW to be used for path planning. In general, when two lines are found, the following equation is used:

$$\left(\frac{x_{right} + x_{left}}{2}, y \right) \tag{1}$$

When only one line is found, the target becomes the point directly centered between that line and either the left or right edge of the viewable region. If the line is on the left, the target is placed on the right, and vice versa. If no lines are found, the target is placed in the center on the horizon, such that the robot will move directly forward at full-speed.

4.2 LIDAR

The LIDAR is a SICK LMS200 laser range finder mounted approximately six to eight inches from the ground on the front bumper of the vehicle. This equipment is used for obstacle detection in a planar view and can deliver 180-degree resolution up to 80 meters away. For the Wunderbot 4 the LIDAR is configured to deliver 360 data points (½ degree resolution) at approximately ten meters.

4.2.1. Signal Processing

The current obstacle avoidance on-board the Wunderbot 4 has two parts. The first is a simple local obstacle detection scheme with a dynamic viewable window. This viewable window goes through a two-fold level of calculations to determine the best path for the robot to follow. A radial filter is first placed on the incoming data of the LIDAR. The data transmission of the LIDAR is in polar coordinates so a radial filter is best. The radial distance is determined by the following equation:

$$Filter_{radius} = (Window_{height}^2 \times Window_{depth}^2)^{\frac{1}{2}} \quad (2)$$

After the filter is applied the obstacles found less than *r* are converted to (x,y) using the following:

$$x = r * \cos(\Theta) \quad (3)$$

$$y = r * \sin(\Theta) \quad (4)$$

4.2.2. Obstacle Avoidance

Finally, an obstacle is “within the window” iff:

$$x < \tfrac{1}{2} * (Window_{depth}) \quad (5)$$

$$\text{and} \quad y < \tfrac{1}{2} * (Window_{height}) \quad (6)$$

The result of the first part is if an obstacle is found within the window a decision is necessary. A polar histogram is developed from the "window" obstacles and is shown in Figure 6. This histogram [2] is useful in determined which direction (left, center, right) has the highest obstacle density and should provide the highest cost function, locally.

4.2.3. Path Planning

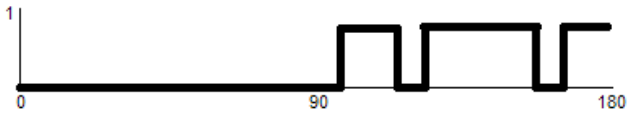


Figure 6 – A binary polar histogram is useful locally in determining which direction to turn based on the obstacle density within a region: left (0-45 degrees), center (45-135) and right (135-180).

Until now the methods discussed are local methods for obstacle avoidance. Each part does not contain starting and goal positions relative to the obstacle being detected. A* provides the ability to incorporate the obstacle detected with the end goal to develop a heuristic approach to obstacle avoidance and overall guarantee an optimal prune of the search arena.

The A* method was used in simulations and is implemented on-board the Wunderbot 4. The simulation versus implementation, however, is different as the simulation provided a few assumptions, which cannot be assumed on the real application. These assumptions are listed in Table 4.

As stated the benefit of using A* is the knowledge of the starting and destination coordinates in the cost function. The cost function of A* is three parts:

Simulation assumptions	Actual constraints
Uniform motion in any direction	Closed loop control necessary to maintain speed/direction after command issued.
Zero-degree turning radius	Turning radius approximately < ½ meter.
360-degree sensing	180-degree line of sight
No time delay in data acquisition	Equipment transmission delays
Coordinates of absolute location	

Table 4 – Assumptions made during simulations and actual constraints on project.

g = distance from start node

h = distance to goal node

f = g + h (7)

The distance calculations are done using the Manhattan method, which states that only square paths are to be taken in the X and Y direction independently. Diagonal paths are acceptable at higher costs.

The simulation results can be found in Appendix F with the actual implementation windows shown in Figure 7. Notice the only available windows for the cost function in the implementation are the three windows in front of the autonomous vehicle and that the global solution is maintained using a mix of local detection and global heuristics.

4.3 GPS/Digital Compass

The orientation in space is obtained through two sensors: 1) GPS receiver and 2) digital compass. The combination of these two sensors allows for specific path planning and destination within one meter. The GPS unit is a Trimble AgGPS 114 receiver with DGPS service provided by OmniStar. The digital compass provided by PNI features 3-axis roll, pitch, and yaw measurements.

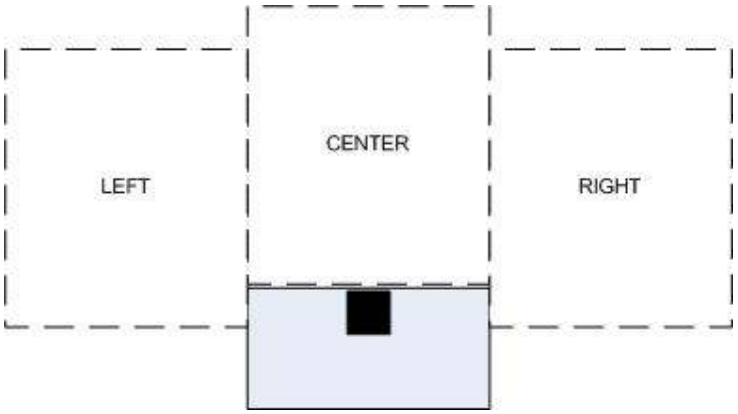


Figure 7 – The actual window and line-of-sight for the Wunderbot is only 3 of the 9 squares available compared to the simulation. This is due to the limited (180 degrees) vision of the robot.

4.3.1. Signal Processing

The GPS information is transmitted via RS-232 at a sample rate of one hertz from the GPS receiver to the PC in NMEA sentence of the following sample format:

$PTNL,GGK,172814.00,071296,32723.46587704,N,12202.26957864,W,3,06,1.7,EHT-6.777,M*48

The first nine of eleven fields are used by the Wunderbot 4, which include UTC time, UTC date, longitude, N/S orientation, latitude, E/.W orientation, GPS quality, number of satellites, DOP of fix.

The digital compass is also transmitted via RS-232 and is read in at a variable rate. In the control software the port is read when 60 bytes of information are available from the device. The transmit time therefore ranges from 10-20ms.

4.3.2. Waypoint Challenge

Using the O^3 method path planning is done in two steps: explicit (before motion) and implicit (during motion). By sorting the GPS points through traditional discrete algorithms an optimal order can be achieved in an ideal environment without obstacles. Furthermore, when the introduction of obstacle occurs – in real time discovery – the Wunderbot 4 is capable of implicitly changing its path to adapt to its environment.

Explicit path planning. Upon receiving the GPS coordinates from the IGVC judges a custom script (written in Matlab) is used to sort the points in an optimal order based on the distance matrix. Since the number of possible paths is on the order of

$$!(n-1) \qquad (8)$$

a method was developed using the Delaunay triangulation as outlined in Appendix B. This method has shown to bring the number of permutations from the complete set shown in equation (8) down to a reasonable set of size *n*. Additionally, it has also been proven that by using the Delaunay sub-graph the globally optimal solution has been preserved and the integrity of the solution has not been compromised.

Implicit path planning. As outlined in Appendix B, a non-traditional use of the Voronoi polygons has allows for more efficient traversal of the arena. An obstacle that expands neighbor polygons can allow for local points of opportunity that through testing have shown to be globally optimal.

4.3.3. Path Planning

The global path planning is done through the explicit and implicit defined above. The path planner in this challenge incorporates the data from three systems: 1) GPS, 2) digital compass, and 3) LIDAR. The vision system has been deactivated since its primary ability is detecting white lines on grass. In this challenge that would pose as threat more than an aid since the GPS coordinates are outlined in white lines. Extending from this issue is the case of boundary points. However, using the explicit graph developed with the original coordinates a convex hull can be developed and as long as the implicit path is within the convex hull the path is valid and the robot will execute the proper commands.

5. JAUS

After the 2006 IGVC competition JAUS research was started in order to participate in the 2008 JAUS challenge. Since the team did not participate in the 2006 JAUS challenge we had to start from the beginning.

The Wunderbot 4 can read and execute the JAUS message commands from the operator control unit through the 802.11g data link. During the JAUS challenge the Wunderbot 4 will be set to monitor for JAUS messages and check for incoming JAUS commands. At this level of implementation these messages will start the Wunderbot 4 moving forward in autonomous mode, stop the Wunderbot 4 from moving in autonomous mode, and activate a warning device (sound file output to speaker) and report position.

In the JAUS software the UDP Ethernet connection is opened to listen for JAUS messages coming in through that port and IP address. When a message is received it is

checked for the UDP header information containing the ASCII equivalent of “JAUS01.0” then parsed out of the UDP header. The incoming IP address and port are rechecked in code to ensure that they are correct for receiving data and carrying out commands. Next the message properties are parsed out and output to the front panel of the LabVIEW program. The next piece of information that is in the JAUS header is the command code followed by the destination ID, the source ID, the data control, and sequence number. For the competition there is no data control information being sent or sequence number. After the command code is parsed out of the byte array the command string is sent to the JAUS Command VI to carry out the command.

5. Performance

The Wunderbot 4 system maintained the same specifications as the previous platform; however, the results were retested and verified. These are shown in Table 5.

Category	Required Result	Expected Result	Confirmed Result	Unit
Speed	5	5	5	Mph
Ramp Climbing	15	45	30	Degree include
Stopping Distance	6 feet, 15% incline	Immediate	Immediate	--
E-stop range	50	1000	100	Feet
Payload	20	>20	100	Lbs
Battery life	30	240	120	Minutes

Table 5 – Performance specification on Wunderbot 4.

6. Cost

The budget for this year was minimal as many of the changes were software. Therefore, we have supplied the estimate in total robot costs from last year plus the additional hardware changes made by the current team to the Wunderbot platform. The updated budget is shown in Table 6 and a full budget breakdown is available in Appendix G.

Item	Price
Remote control	$200
Tower parts	$150
Plexiglass	$50
Electrical blocks	$60
Wiring conduit	$50
Laptops (3)	$1200
Subtotal	**$1710**
Estimated project 2006 net worth	$31,000
Total Cost	**$32,700**

Table 6 – Budget for the Wunderbot 4 team. This figure does not include additional costs incurred by the club including but not limited to IGVC travel costs.

7. Social Contributions

The Wunderbot 4 team has provided many social contributions both technical and non-technical. The technical avenue includes publishing an IEEE paper, meeting with industry sponsors, and structuring learning opportunities for other students within the department on topic such as control theory, signal processing, and quality assurance. In the non-technical venue the team has worked with high school students sparking their interests in robotic design. They have been featured in numerous media outlets including television, newspaper, and Internet publications. It is always on the team's agenda to further the discussion and developing of robots society to aid in all aspects in the home, classroom, and/or in space.

8. Conclusion

The Wunderbot continues to be a platform for undergraduate student research in the RMI Lab of Elizabethtown College and has provided many opportunities educationally and professionally to its students. The team would like to take this opportunity to thank the judges and organizers of the IGVC for their hospitality and the team looks forward to a successful competition this year!

9. References

[1] . Siegwart and I.R. Nourbaksh, Introduction to Autonomous Mobile Robots, The MIT Press, Cambridge, Massachusetts, 2004
[2] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots", IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16-21, 1998, pp. 1572-1577

Appendix Listing

Appendix A – Software flow diagrams

Appendix B – O^3: An Optimal and Opportunistic Path Planner (with Obstacle Avoidance) using Voronoi Polygons (ISBN:

Appendix C – Vision System for Wunderbot 4 Autonomous Robot

Appendix D – IGVC Way Point Navigation Solution; Case Study: Wunderbot 4

Appendix E – The Joint Architecture for Unmanned Systems: A Subsystem of the Wunderbot 4

Appendix F – Simulations results of Wunderbot 4 path planning

Appendix G – Complete budget of Wunderbot platform starting with Wunderbot 0.

Appendix A – Software Flow Diagrams

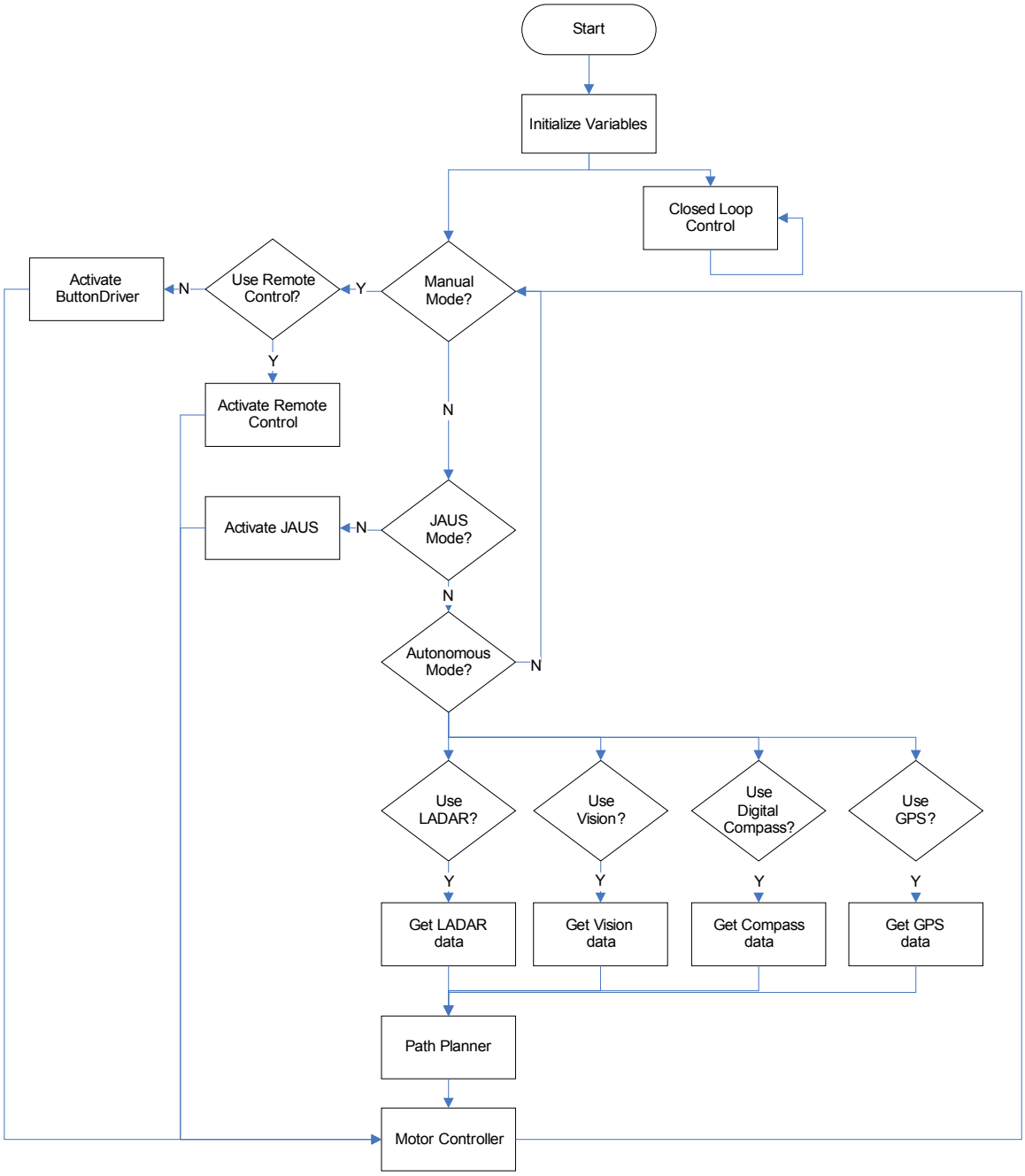


Figure 1 – Wunderbot Software Overview

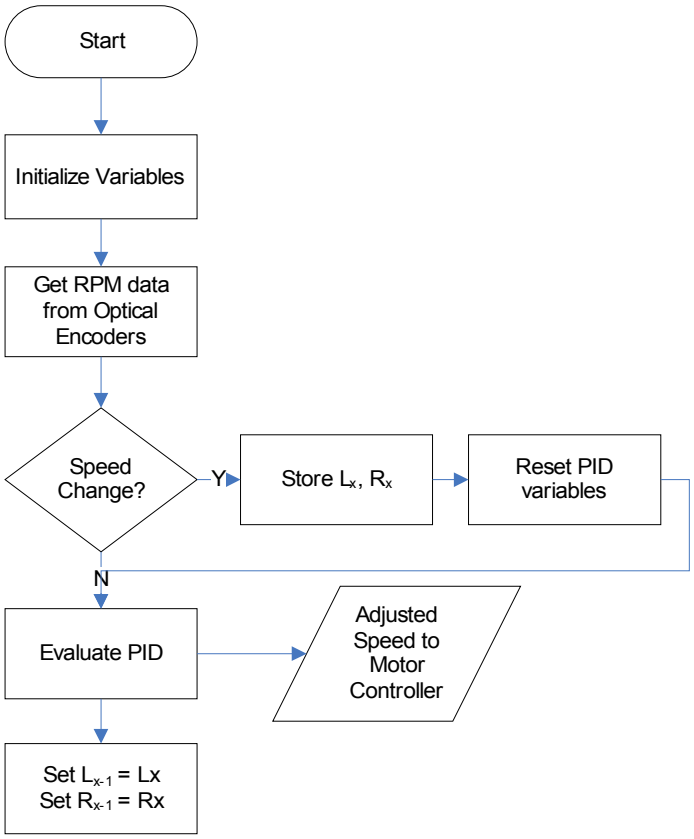


Figure 2 – Closed Loop Control Algorithm

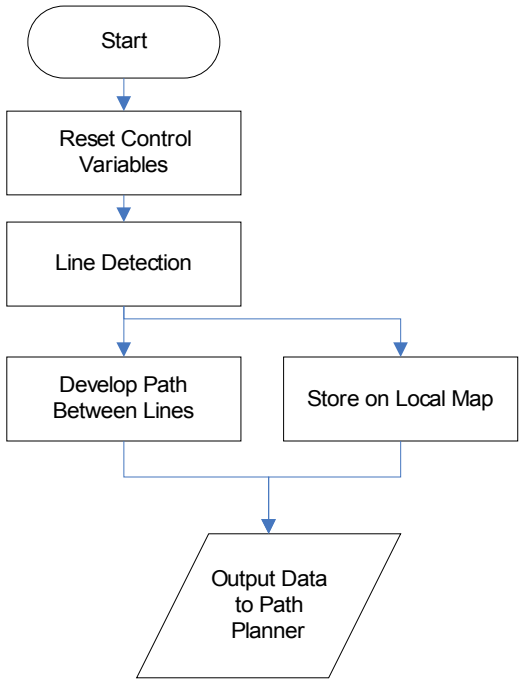


Figure 3 – Vision System Control and Decision Making Algorithm

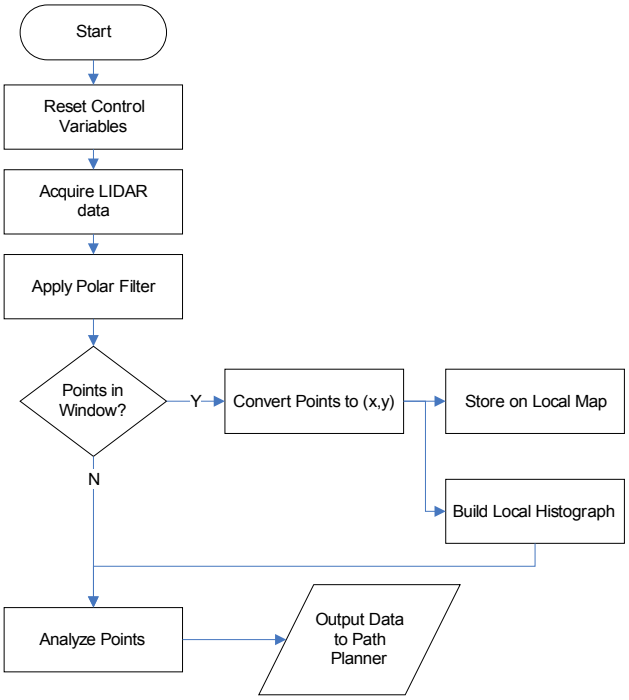


Figure 4 – LIDAR Data Filtering Algorithm

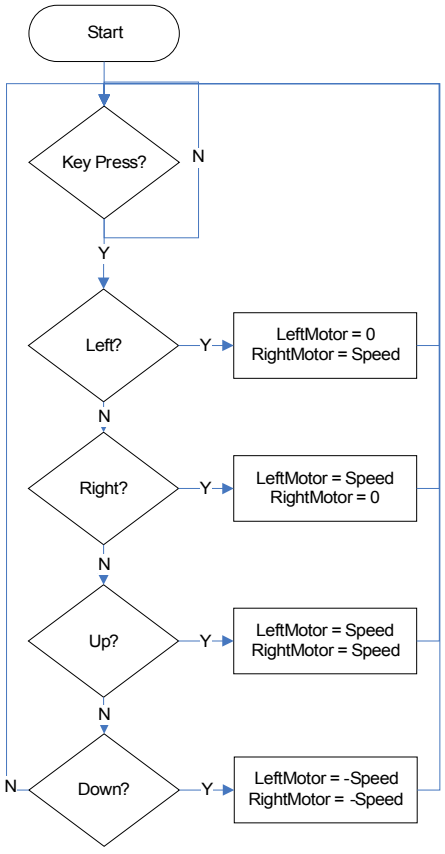


Figure 5 – Manual Mode Control Algorithm

O^3:
An Optimal and Opportunistic Path Planner (with Obstacle Avoidance) using Voronoi Polygons

David M. Coleman and Joseph T. Wunderlich, PhD.
Elizabethtown College, Elizabethtown, Pennsylvania, USA
{colemand, wunderjt}@etown.edu

***Abstract*- Traditional mobile robot research focuses on a robot navigating its environment to reach a single goal while avoiding obstacles. This paper proposes a new method called O^3 to solve the challenges presented at the Intelligent Ground Vehicle Competition (IGVC) where a navigation course includes multiple goals to be found in an optimal order. The O^3 technique includes improvements on traditional path planning and obstacle avoidance techniques while providing an explicit ability to change course as obstacles are discovered. This method uses modern trajectories such as minimum-weighted Hamiltonian circuits, A* algorithm for obstacle avoidance, and local points of opportunity to update the globally optimal path using Voronoi polygons. Environmental mapping is also used to speed up the search algorithms in static environments. Overall, the O^3 technique exploits local points of opportunity while avoiding obstacles and ultimately finding a globally optimal path through an unknown environment.**

This methodology will be implemented on an autonomous web-based tour guide robot to serve the Internet community reviewing Elizabethtown College. This methodology can be extended to other research areas where multiple locations need to be traversed independent of their order such as city map, trip planners, and distribution networks (power, internet, etc) due to its balance between weighted graphs and obstacle avoidance (objects, traffic, construction, etc).

I. Introduction

Mobile robotic motion control can be separated into two research areas: (1) simple obstacle-free path planning and (2) path planning which includes various obstacle avoidance strategies. An example of a simple obstacle-free path planning strategy is creating a Hamiltonian circuit through a set of given nodes [10]. *A priori* information may be added to plan a specific path around an obstacle [4]. However, obstacle locations are often not known ahead of exploration by the robot; this warrants developing more complex obstacle avoidance strategies.

In [1], a dynamic window approach provides a "local vs. global" relationship and is used to store obstacles in memory for later analysis. Heuristics along with additional feedback from sensors are used to provide motion and obstacle locations as seen in [9]. To improve the ease of mapping it is suggested in [3] that obstacles be scaled to match the dimensions of the robot. Other obstacle avoidance decisions are done using the cell-decomposition methods of VFH* [4] and Sentz's A* algorithm [7].

Most importantly the O^3 technique heavily relies on Voronoi diagrams. The Voronoi diagram is used in other path planners including the roadmap [14] and HGVG algorithms [5]. A formal definition of the Voronoi diagram can be found in [5], [14], and [16]. Related to the Voronoi diagram is the Delaunay triangulation. As defined in [16], the Delaunay triangulation T is the maximum planar subdivision of n points $P = \{P_1 \ldots P_n\}$ such that no points of P are bounded by the circumcircle of any triangle in T. The Delaunay triangulation will be used to restrict the domain of our algorithms and will be expanded upon throughout this paper. Examples of each of these are shown in Fig. 1(b), (c), respectively.

II. Path Planning

As stated in [10] and [14] a path planner must be correct and complete. Correct meaning the algorithm must be accurate and complete meaning an algorithm must return a failed value when a solution is not available in a reasonable amount of time. This is a requirement for both explicit (before motion) and implicit (during motion) methods.

A. Classical Approach – Explicit Methods

Example: Imagine walking into a room for the first time with the lights off and being asked to find the door on the opposite side of the room. You do not process any knowledge of the room's exact dimensions, obstacles, or the quickest path to the door. However, having the knowledge that a door exists (or assuming it exists), and knowledge that its location is approximately "across" the room, is enough information to plan a path (including an obstacle-avoidance strategy) – even though it may not be optimal.

Similarly, a robot enters an unknown environment and the complexity of tasks is increased with multiple nodes to be visited and exact distances and velocities to be rendered. The information given for target nodes may include GPS position. Thus specific distances can be calculated based on the entrance point of the robot; and since the end of the overall path plan often includes returning to the entrance, this often becomes a typical Traveling Salesman Problem (TSP) [11]. By solving the TSP for the given set, a solution will exist that is minimal in distance (and likely to be traversable by the robot).

A breadth-first search through all possible Hamiltonian circuits is logically the easiest, but most difficult computationally. Given a set of n points, as shown in Fig. 1(a),

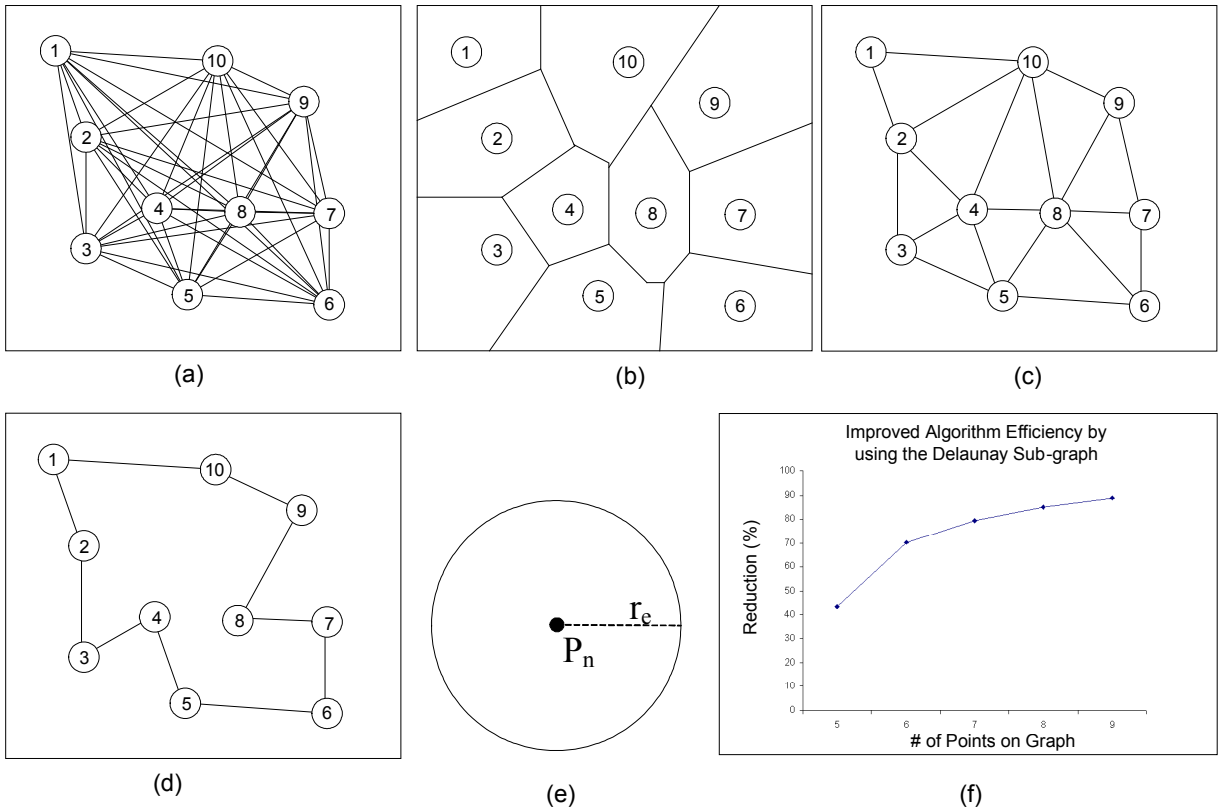


Fig. 1. (a) Environment of 10 goal nodes; (b) Voronoi Diagram of environment; (c) Delaunay Triangulation of environment; (d) Hamiltonian circuit derived by (c); (e) Expanded node showing r_e (f) Reduction in processing time using (c) instead of (a).

finding the TSP solution through a breadth-first search is on the order

$$O((n-1)!) \tag{1}$$

since every point has a degree of (n-1) and every point can be reached from any starting point. A general source code structure would look like the following:

```
% Input node locations
% Define adjacency matrix
Set (control_flags)
for i = 1…(n-1)!
     path = get(g, Hamiltonian);
     if ( is_unique(path) )
          P = path(information);
     end
     Reset control flags
end
return shortest(p)
```

Given the Voronoi diagram in Fig. 1(b) the Delaunay triangulation graph can be constructed by connecting points within touching Voronoi polygons and is shown in Fig. 1(c). By Euler's formula (V-E+F=2) [15] the sub-graph is said to have at most 3n-6 edges and an extreme less number of Hamiltonian paths. Testing has shown a breadth-first search of the Delaunay graph is on the order

$$O(n) \tag{2}$$

The actual number of Hamiltonian circuits that exist in the Delaunay sub-graph of any graph is dependant on the topology of the graph and therefore is not generally quantifiable. Introducing the Delaunay sub-graph into the previous code segment results has shown to improve the search for a TSP solution between 40% and 90%. The specific results and testing software will be explained later in this paper. Fig. 1(f) shows the improved algorithm efficiency and corresponds to Table 1.

Throughout this paper the term "TSP solution" will be used to identify the Hamiltonian circuit in the Delaunay domain. By previous arguments TSP solution can be said to be correct since the globally optimal solution was not lost in the Delaunay reduction. With the absence of obstacles and unreachable points within the graph the TSP solution and breadth-first algorithm are complete.

It can also be said that by using the TSP solution the robot will intersect the acceptable radius error (r_e) on each target

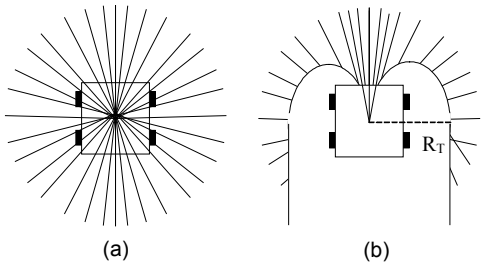


Fig. 2. Robot with (a) zero turning radii, (b) actual turning radii, RT

node. Once obstacles are introduced implicit methods are employed and the TSP solution is used as a governing overview path.

B. Implicit Methods

Recalling the thought experiment from the previous section, imagine starting to move in the direction you think is correct to reach the door. How do you go about detecting, avoiding, and overall re-evaluating the best path to follow to reach your goal? Do you just run into objects that may be in the room and bounce off, or do you feel with your arms and attempt to adjust your senses to the dark room? Let us also focus on achieving one goal and avoiding obstacles along the way.

A robot has many sensors that contribute to the "overview" of an environment. Feedback needs to be assessed in real-time to adjust the course of the robot. Two things need to be considered for path alteration: turning radius and size of the dynamic window as described in [13]. The specific algorithms for real-time obstacle avoidance are beyond the scope of this paper. Therefore, only key points will be discussed for completeness of the O^3 techniques.

As shown in Fig. 2(a), the easiest approach is to assume the robot is a point position and can change its course at any time. However, when implemented, this is not often the case as seen in [3].

As outlined in [1], a dynamic window is common among implicit algorithms for obstacle avoidance. The sensors available on the robot govern the exact dimension of this window. These can include laser range finders, sonar, machine vision, and magnetic/GPS positioning. An example of the window approach is seen in Fig. 3. Depending on the sensors used, only certain directions may need monitored (i.e. forward) and the previously explored areas can be stored in an environmental map. In static environments, a map in memory can be an effective tool to speed up processing time for revisited areas.

By using the dynamic window approach, heuristics can be employed for implicit path planning and obstacle avoidance. As seen in Fig. 4, there are situations where a decision is not obvious with the limited view of the sensors. As stated in [4] "a larger trigger distance would not eliminate the problem." In fact it could increase the run time for the algorithm computing all possible avenues, and thus be incomplete. By using a heuristic approach such as A*, a decision based on a cost function will be made and followed as shown in path C of Fig. 4. This cost function is explained in [7].

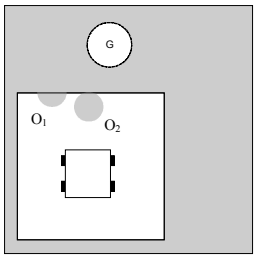


Fig. 3. Dynamic window approach. Obstacles are seen at O_1 and O_2 and the goal node is labeled G.

C. Local Opportunistic / Globally Optimal Points

To conclude our previous thought experiment, now imagine moving towards one goal and, through sensory information, an obstacle blocks the intended explicitly defined path. While avoiding the obstacle, another point becomes more desirable for traversal in distance and availability. Therefore a change in course should be analyzed to see if it is in fact globally optimal and not just locally opportunistic.

This locally opportunistic globally optimal visit of an out-of-order node is a combination of the TSP solution developed in the first section and the obstacle/motion techniques developed in the second section of this paper. An obstacle is shown in Fig. 5(a), which interferes with the TSP solution explicitly planned. By avoiding the obstacle using techniques previously discussed, a new point becomes locally opportunistic.

Consider the simplified graph shown in Fig. 5(c). This graph is connected with edges {D1, D2, D3, T1, T2, T3} as paths. Assume the ideal case where the turning radii (r_r and r_l) are neglected and straight paths are possible. Also assume there are no obstacles in the paths between {C, 8},{7,8},{7,*n*},{8,*n*}. Let C be the point where the path crosses into the unexpected Voronoi polygon as shown to the left of the obstacle in Fig. 5(a) surrounding point 8. Allow point *n* to be the next point to be traversed after the set of three points shown.

Two equations can be shown: the current path as specified

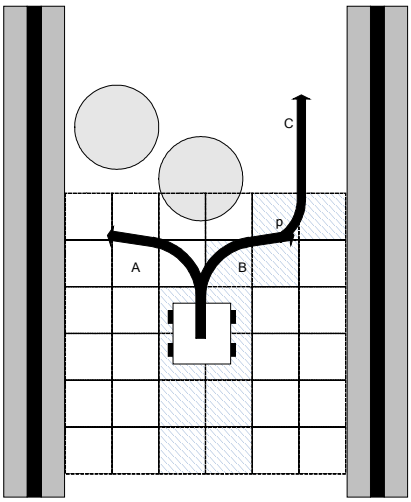


Fig. 4. Cell decomposition of region where one obstacle is in the middle of the robot's vision. A decision needs to be made to take path A (left curve) or path B (right curve). Using the heuristics invoked in A* path B is chosen and the best-fit curve (path C) is implemented at point P. The curvature at point P is equal to r_l. The grid surrounding the robot represents the environmental map being developed with A*. Its size is equal to the dynamic window shown in Fig. 3 and is govern by available sensors.

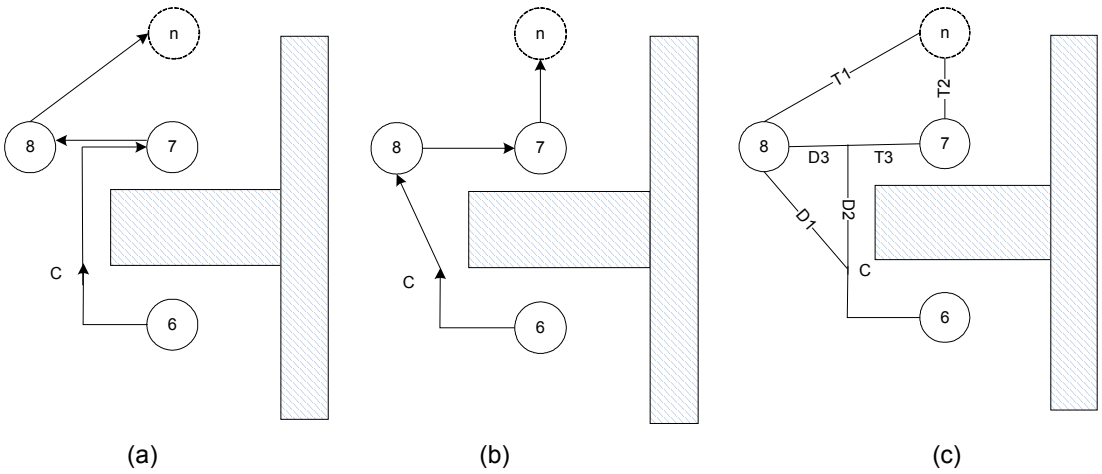


Fig. 5. Following the path prescribed in Figure 1(d) the robot should traverse the points as shown in (a). With the discovery of the obstacle it changes points to traverse the path shown in (b). (c) The combination of (a) and (b) for γ analysis.

by the TSP solution in Fig. 5(a):

$$\{6, 7, 8, n\} = (d_{6C}) + (D2+T3) + (T3+D3) + (T1) \quad (3)$$

and the opportunistic path shown in Fig. 5(b):

$$\{6, 8, 7, n\} = (d_{6C}) + (D1) + (D3+zT3) + (T2). \quad (4)$$

It must be shown that in order for the path to be optimal (not just ideally opportunistic) the new path eq. (4) is less weighted than the current TSP solution in (3).

$$(d_{6C}) + (D1) + (D3+T3) + (T2) < (d_{6C}) + (D2+T3) + (T3+D3) + (T1) \quad (5)$$

Subtracting (d_{6C}) from both sides:

$$(D1) + (D3+T3) + (T2) < (D2+T3) + (T3+D3) + (T1) \quad (6)$$

By trigonometry rules:

$$D1 = (D2^2 + D3^2)^{1/2} \text{ and } T1 = ((T3+D3)^2 + T2^2)^{1/2} \quad (7)$$

Combining (6) and (7):

$$((D2^2 + D3^2)^{1/2}) + (D3+T3) + (T2) < (D2+T3) + (T3+D3) + (((T3+D3)^2 + T2^2)^{1/2}) \quad (8)$$

Expanding and eliminating like terms:

$$\gamma = (D3^2+2*D3*T3+T2^2+T3^2)^{1/2} + T3 - T2 - (D2^2 + D3^2)^{1/2} + D2 \quad (9)$$

Therefore, when $\gamma > 0$ the alternative path is optimal and the course should be altered.

Once again it is essential that the local point-of-opportunity does not compromise the correctness and completeness of the global solution. Therefore it is necessary that the point of non-opportunity be the target point after the opportunity point. It is also sufficient in the case where multiple points-of-opportunity exists on the way to the original non-opportunistic point. This would solve the problem where a point of interests resides in an area of limited opening such as a box shown in Fig. 6.

In Fig. 6 the explicit TSP solution dictates an A-B-C-D path. However, after γ analysis, point C is determined to be locally opportunistic and globally optimal. Now an A-C-B-D solution exists. While on route to point B, point D is determined to be locally opportunistic and globally optimal. Once again the path is now changed to A-C-D-B. By this repetitive process the γ analysis will prove to be globally correct and complete. There is an assumption made that all points are reachable by the robot.

The γ analysis is a technique that is only necessary to perform when crossing over an unexpected Voronoi polygon. Similar to OPEN and CLOSED (and RAISED and LOWERED) sets in Sentz's A* algorithm a set needs to be created for Voronoi polygon crossings along the path between two points. By setting a flag for an unexpected Voronoi polygon crossing the γ analysis will be limited to only those of

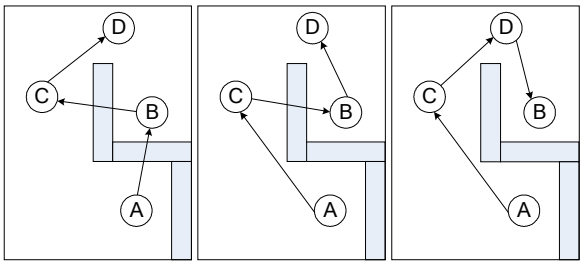


Fig. 6. Map with more than one point of opportunity. The original TSP dictates an A-B-C-D path. After the γ analysis, A-C-D-B is locally opportunistic and globally optimal.

Processing Time Comparison (in seconds)					
	Number of input points				
	Five	Six	Seven	Eight	Nine
Full graph	0.0330	0.3656	10.12	575.0	41570
Delaunay sub-graph	0.0183	0.1059	2.036	84.97	4547
Improvement (%)	43.68 ±20.42	70.31 ±7.422	79.29 ±3.915	85.01 ±3.016	89.06[†]

Table 1. Reduction of processing time shown by using the Delaunay sub-graph in Fig 1(c) over the complete connected graph of Fig. 1(a). [†]Standard deviation was not calculated, as only one test run was available due to the time interval required to complete one pass.

true opportunity points. Also, the polygonal crossings are used to speed up the "nearest" point question. By knowing specifically which Voronoi polygon the robot resides in, the nearest point is apparent.

By both these techniques (γ analysis and polygon sets) O^3 expands traditional robot navigation for a multiple-target environment while still employing traditional obstacle avoidance strategies.

III. Implementation

Our methods for path planning and obstacle avoidance are being implemented to solve the challenges at the IGVC competition in May 2008. One challenge is navigating a long, complex maze defined by white lines painted on approximately 3-inch tall grass, and riddled with various complex obstacles (e.g. ramps, pits, trees, fencing, and various cones). Using A* heuristics with dimensional constraints in the cost functions as outlined in [3] [4], both correct and complete decisions can be made. With the addition of environmental mapping in a static environment, obstacle-processing time can be minimal. In the second IGVC challenge of GPS navigation with minimal obstacle avoidance, our O^3 method is ideal. Our *Wunderbot 4* has laser ranges finders, vision system with color recognition, digital compass, GPS receiving, and optical encoders for sensor feedback.

IV. Preliminary Results

Table 1 shows the improved processing time seen in our code during multiple trials. Matlab R2007 was chosen to test our method because it supports graphing, provides functional toolboxes for efficient environmental geometry analyzing and straightforward integration into LabVIEW. LabVIEW provides the real-time operating controls necessary for the Wunderbot 4. The testing was done on an Intel Core Duo CPU @ 2.00GHz. Overall, the Delaunay sub-graph provided an average improvement over 73%.

V. Future Research

After competition, the *Wunderbot 4* will serve as a platform for autonomous web-driven tours on the Elizabethtown College campus. With our O^3 implemented, a pre-determined or "way-point" driven tour is not necessary. In fact, any path can be altered (or tailored) based on student traffic, construction paths, and/or availability of certain building/fields on campus. By changing the weights of certain paths in the TSP solution and cost function (with obstacles), a fully autonomous robot is possible with O^3.

VI. Conclusion

O^3 is a unique method that combines traditional implicit and explicit methods to offer an optimal and opportunistic (and obstacle avoidant) solution to the areas of path planning for autonomous robots. By continually implying graphical techniques such as Voronoi polygons short cuts in calculations can be achieved as well as opportunistic paths through an unknown environment. This methodology should also yield successful results for our Wunderbot 4 robot at international competition and as a robot tour-guide.

VII. ACKNOWLEDGMENTS

We would like to thank the math department at Elizabethtown College for their continuing support and collaboration of theories within this proceeding, especially Dr. Bobette Thorsen.

References

[1] O. Brock, O. Khatib, "High speed navigation using the global dynamic window approach", IEEE International Conference on Robotics and Automation, May 10-15, pp. 341-346 vol.1

[2] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots in cluttered environments", IEEE International Conference on Robotics and Automation, Cincinnati, Ohio, May 13-18, 1990, pp.572-577

[3] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots", IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16-21, 1998, pp. 1572-1577

[4] I. Ulrich and J. Borenstein, "VFH*: local obstacle avoidance with look-ahead verification", IEEE International Conference on Robotics and Automation, San Francisco, CA, April 24-28, 2000, pp. 2505-2511

[5] H. Choset and J. Burdick, "Sensor-based exploration: the hierarchical generalized Voronoi graph", The International Journal of Robotics Research, Vol. 19, No. 2, February 2000, pp. 96-125

[6] A. Stentz, "Optimal and efficient path planning for partially-known environments", IEEE International Conference on Robotics and Automation, May 1994

[7] A. Stentz, "The focused D* algorithm for real-time replanning", International Joint Conference on Artificial Intelligence, August 1995

[8] J.F. Canny and M. C. Lin, "An opportunistic global path planner", Algorithmica, vol. 10, pp. 102-120, 1993

[9] M. Lindhe, P. Ogren, and K.H. Johansson, "Flocking with obstacle avoidance: a new distributed coordination algorithm based on Voronoi partitions", IEEE Conference on Robotics and Automation, April 26-May 1, 2004

[10] R. Siegwart and I.R. Nourbaksh, *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge, Massachusetts, 2004

[11] S.S. Epp, *Discrete Mathematics with Applications*, 3rd ed., Brooks Cole, Boston, MA, 2003

[12] K. Hwang, *Advanced Computer Architecture: Parallelism, Scalability, Programmability*, McGraw-Hill, Boston, MA, 1992

[13] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance", IEEE Robotics & Automation Magazine, Vol. 4, pp. 23-33, March 1997

[14] J.J.A.M. Keij, “Obstacle avoidance for wheeled mobile robotic systems (literature exploration)”, Technische Universiteit Eindhoven, Eindhoven, The Netherlands, February 17, 2003, Report No. 2003.10
[15] D.I.A.Cohen, *Basic Techniques of Combinatorial Theory***,** John Wiley & Sons, Inc., pp. 292, 1978
[16] C. Kavka, M. Schoenauer, “Evolution of Voronoi-based Fuzzy Controllers”, International Conference on Parallel Problem Solving From Nature, Birmingham, UK, September 18-22

Vision System for Wunderbot IV Autonomous Robot
EGR494: Senior Project in Computer Engineering

James G. Painter
Department of Physics and Engineering
Elizabethtown College
Elizabethtown, Pennsylvania
Email: painterj@etown.edu

***Abstract*— The Robotics and Machine Intelligence Club of Elizabethtown College has maintained an ongoing autonomous robot project for nearly seven years, but the robot has yet to feature an effective vision system for unmanned navigation. Here we describe the steps taken toward the development of such a system, including physical mounting of a camera, image processing, and motor control. The resulting vision system establishes a platform for competing at the 2008 Intelligent Ground Vehicle Competition.**

I. Introduction

Autonomous robots serve mankind in areas ranging from search and rescue to space exploration. In attempt to elicit new, creative designs from college students [1], the Association for Unmanned Vehicle Systems International (AUVSI) holds the annual Intelligent Ground Vehicle Competition (IGVC), an engagement of roughly 30 unique robots designed by colleges and universities from various countries, including the United States, Japan, Canada, and India. Their objective is to excel at complex autonomous tasks used to measure the robots' navigability and strength of design.

At Elizabethtown College in Pennsylvania, the Wunderbot autonomous robot project has been progressing for over five years. From the financial assistance and donations of numerous corporate sponsors, new equipment and software is added to the robot between competitions. Its primary objective is to compete in the IGVC, with the potential of administering future automated tours on the college campus. Wunderbot IV is led by the team of James Painter, David Coleman, and Jeremy Crouse, with support from Chris Yorgey and Daniel Fenton, and advised by Dr. Joseph Wunderlich. The project has become a staple of the school's Physics and Engineering Department, attracting prospective students and drawing the attention of local industry and media.

II. IGVC

The IGVC provides an excellent opportunity for students to explore the possibilities of unmanned vehicles. 2008 will mark the 16th anniversary of the competition and the third entry for Elizabethtown's Wunderbot. IGVC consists of four challenges for the autonomous robot, each outlined below [2].

A. Autonomous challenge

The autonomous challenge pits the robot against an outdoor obstacle course, traversed by remaining on a path of grass approximately three inches tall, bounded by spraypainted solid or dashed lines. The robot must avoid obstacles, including fences, construction barrels, trees, and shrubs. Potholes, inclines, and sand pits may also be strewn about the course. Scores are calculated based on the distance traveled through the course and the elapsed time.

The autonomous challenge will rely on the robot's vision system moreso than any other challenge, due to the imperative condition that the robot remain within two-dimensional white lines, which go undetected by the laser range-finder.

B. Navigation challenge

In the navigation challenge, a field of approximately one acre is marked with a number of GPS waypoints (approximately ten). Each team is provided with the coordinates in latitude and longitude of each waypoint. Obstacles similar to those on the autonomous challenge course may also be placed randomly on the navigation challenge course. Scores for the navigation challenge are determined by the number of waypoints traversed by the robot and the time taken to do so. This challenge will, for the most part, neglect the capabilities of the vision system. All obstacles can be detected by the laser range-finder, and the GPS/compass sensing will be responsible for finding the targets.

C. JAUS challenge

The challenge for Joint Architecture for Unmanned Systems, although not mandatory, demonstrates the robot's ability to communicate using a standardized wireless messaging system that is growing in popularity in engineering fields [3]. A section must be included in the written report that describes the robot's JAUS capabilities, and the robot must demonstrate a pre-defined working ability to communicate using the JAUS message type. The JAUS challenge will ignore the vision system entirely.

D. Design challenge

The design competition exists as a separate entity of competition, in that the robot's performance has no influence on the design score. The design competition measures the team's procedures, workmanship, and innovation to determine product quality. Each team must submit a typed report prior to the main competition date, detailing the conceptual design of the vehicle and its components, and emphasizing design changes from the team's previous contest entry as well as technological

innovations that distinguish it from the rest of the field. Teams must also prepare a ten-minute oral presentation. The third component of the design challenge involves judges' hands-on examination of the robot, for such aspects as neatness, safety, originality, and style.

III. Related Work

The Wunderbot vision system's most closely-related work is in the IGVC competition itself. In any given year, roughly 30 other robots, all having identical objectives, are available for comparison. Following each competition, the organizers make all teams' design reports publicly available online, encouraging the spread of successful ideas. As such, we find trends in particular subsystems among the contest entrants. The laser range-finder, for example, has become standard for obstacle avoidance, having been built into nearly all of 2007's competing vehicles.

While many teams may share similar components, each year brings new innovation in overall design. Part of the initiative in this matter is the scoring of the design challenge. The scoring is partially dependent on the vehicle's display of a significant subsystem or software upgrade over that which represented the team previously.

The competition has seen many different vision configurations, as well as an assortment of corresponding software packages for image processing. Some teams opt for camcorders, some use industrial cameras, and others choose to mount webcams [4]. Stereo vision has given robots the advantage of a line of sight extending to the sides or rear of the vehicle [4]. Once images have been grabbed, teams have performed successful filtering through the use of Intel's OpenCV library [5], MATLAB [6], and LabVIEW [4]. Alas, the IGVC has no cookie-cutter winning formula. In fact, oft-champion Virginia Tech's traditional three entries per year are all structurally distinct from one another [7], a further testament to the competition's flexible path to success.

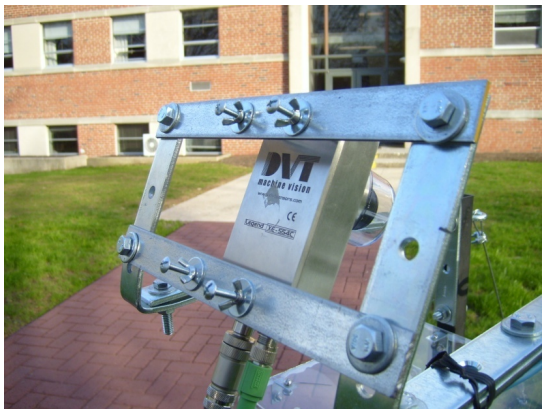
IV. Preliminary Work

Prior to the start of the project, a DVT Legend 554C XE high-resolution video camera was acquired and configured. In addition, a LabVIEW sub-VI was written to acquire the camera's TCP/IP communication string, in which is contained the manually-formatted results of any image processing performed. The camera is hard-coded to capture color images at 1280×1024 resolution. The image quality is far better than needed, but Section V describes how this can be used to our advantage by zooming out while still maintaining sharp objects in the distance.

V. Camera Mount

A steel camera mount, shown in Figures 1(a) and 1(b), was build atop the Wunderbot's utility pole, which also houses the GPS and digital compass. Two $30.5\text{cm}\times2.5\text{cm}$ flat steel bars were fastened to the utility pole supports using L-brackets. At the far end, two more L-brackets were attached, making the entire camera mount extend about 40.5cm back from the rear bumper of the vehicle. The L-brackets were bent to form roughly 45-degree angles. Two $25\text{cm}\times2.5\text{cm}$ steel bars were secured across the L-brackets in order to provide a stable mounting surface for the camera. Through the 25cm bars were inserted 10cm bolts that screw directly into the four threaded holes in the back corners of the camera. Wing nuts allow very precise fine-tuning of the angle at which the camera is directed. The data and power cables for the camera were concealed with the 2.5cm plastic conduit that runs along the utility pole.

(a)



(b)

Fig. 1. Wunderbot camera mounted on steel brackets.

The angle at which the camera was mounted played a crucial role in the eventual image processing step of the vision system. Mounting the camera farther from the front of the vehicle would widen and deepen the field of view. A more downward mounting angle would enable the camera to see directly in front of the bumper, while a more upward angle would extend the depth of view. This situation is illustrated in Figure 2. Another consideration was the image processing time on the software end of the system, which could be accelerated by trimming the edges of the rendered image. A larger field of view yields more unnecessary regions of the image, which can be eliminated to reduce the processing time.

Various configurations were tested, measuring the range of view and corresponding image processing times. For instance, with the camera positioned 1.2m directly above the rear

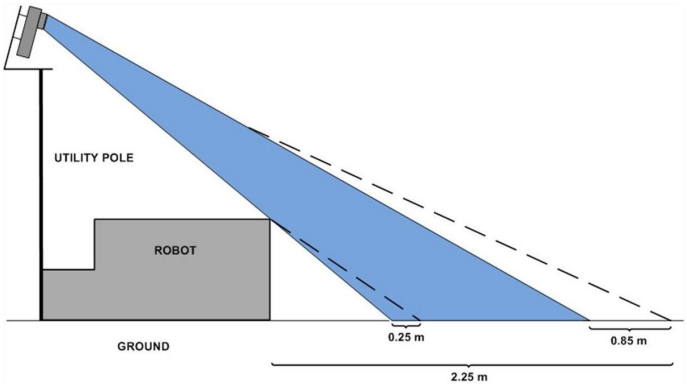
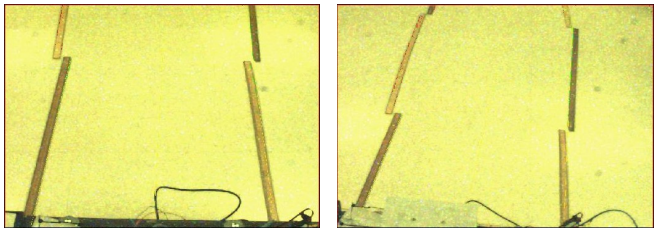


Fig. 2. Camera viewable region, with camera mounted directly above rear bumper (blue fill) and with camera shifted back 40.5cm from rear bumper (dashed lines).

bumper, the camera was able to see approximately 1.4m ahead of the front bumper, as depicted in Figure 3(a). With the camera at the same height, but 40.5cm back, the viewing distance was extended about 85cm to roughly 2.25m, at the expense of about 25cm lost directly in front of the bumper, as shown in Figure 3(b). This sacrifice was acceptable, since the tradeoff is either seeing farther ahead or trimming the top edge of the image to reduce processing time.



(a) (b)

Fig. 3. Viewable region of camera when mounted (a) directly above rear bumper, and (b) when mounted 40.5cm behind rear bumper.

Processing time reductions when trimming the image were significant enough to implement the feature. The percentage speedup was a nearly-linear relationship to the percentage of the image that was trimmed, and the final implementation incorporated the cropping of the top 200 lines, for a reduction of about 110ms in processing time.

Top Edge Cropped	Processing Time Speedup
15% (153 lines)	16% (90ms)
24% (246 lines)	25% (140ms)

VI. Image Processing

The vision system's image processing is performed from within the camera's proprietary software, DVT Intellect v2.2. First, an erosion filter is applied to the image, using a 3×3 kernel. This closes many holes of noise, such as small dirt patches that appear through the grass, while still maintaining the shape of the desired white lines, since the lines (including dashed lines) will always be wider than three pixels. Larger kernels could produce an even more accurate image, but processing times increase sharply as the kernel grows larger.

Once noise has been filtered, an Intellect "line thickness" sensor is applied. This measurement sensor first uses a variable 60% intensity threshold to deduce a binary image. The sensor then scans every row in the image to find the two edges closest either side. Optionally, all edges can be found and more accurately be used as input for the line fitting algorithm to follow; however, the extra computations lengthen the processing time roughly three-fold. To help eliminate noise, all edges less than 50 pixels wide are discarded. Next, a Hough Transform with resolution of four is performed on the detected edges in order to fit two lines, one closest to the left side of the image and one closest to the right. These final lines are measured for separation width, and the average of the two is measured for straightness, contrast, and angle.

Final line pass/fail conditions are used to filter shadows and other undesirable objects in the field of view. A maximum width condition of 300 pixels is combined with a "straightness" condition that will fail the test if the sum of the distances between the data point that is farthest away in one direction, and the one farthest away in the other direction, of the resulting average line.

A formatted string is sent via TCP/IP to the on-board PC. This string contains (in units of pixels), the dimensions of the viewing window, the x- and y-coordinates of the point on the left line with the lowest y-value (nearest to the robot), and the corresponding points on the right line. Once these are received by the PC, logic is used to determine the direction in which to turn.

VII. Motor Control

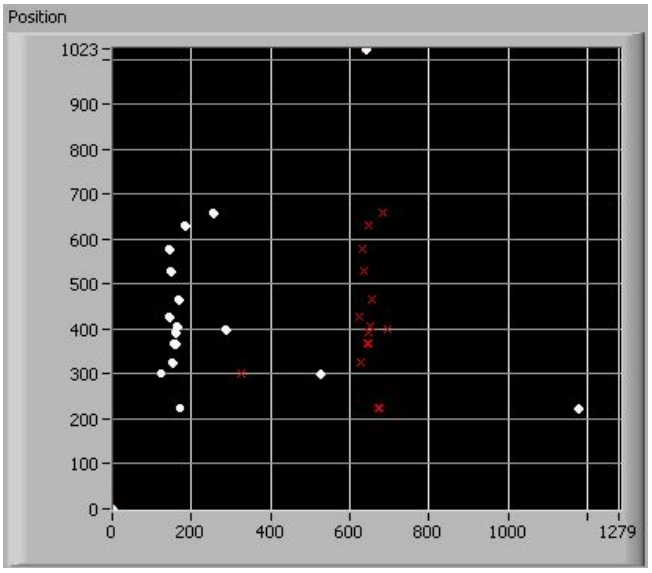
LabVIEW 7.1 was used to develop the all cognition of Wunderbot IV. This section explains the method for turning the vehicle and for achieving accurate motion response.

A. Turning

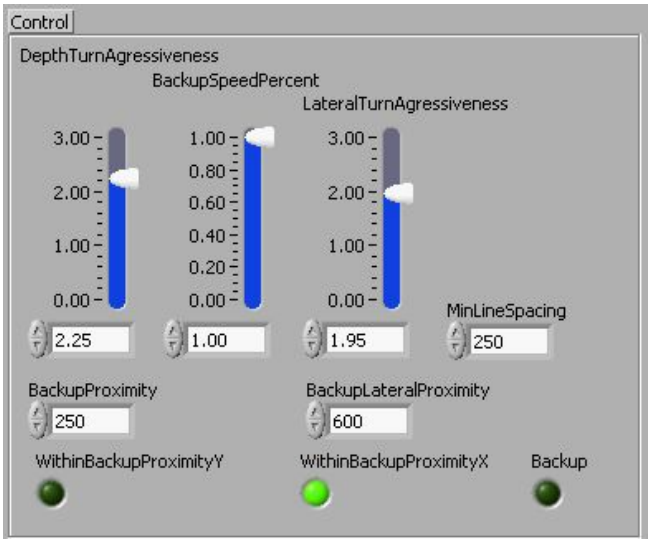
For responding to the white line positions parsed and sent by the camera, both the x- and y-coordinates are taken into consideration. Rough scaling factors for both the x and y direction were used to convert pixels to centimeters. Because the actual width of the view widens when extending outward, the scale is only an approximation. The scaling factors were then used to estimate the depth and width of the camera view. Once these measurements were obtained, they could be used to plot the detected lines on a local map with target locations, as seen in Figure 4(a). The physical locations of these points are critical in giving the Wunderbot a global sense of position, which is used to determine how sharply to turn away from white lines and how to coordinate with other sensing subsystems, such as the GPS, digital compass, and LIDAR.

In general, when two lines are found, the target location is the average of their x-coordinates and the actual value of their y. When only one line is found, the target becomes the point

directly centered between that line and either the left or right edge of the viewable region. If the line is on the left, the target is placed on the right, and vice versa. If no lines are found, the target is placed in the center on the horizon, such that the robot will move directly forward at full-speed.



(a)



(b)

Fig. 4. (a) Detected white lines and calculated target points, both plotted on local map using pixel scale. (b) LabVIEW control panel with adjustments for vehicle movement.

Controls, shown in Figure 4(b), were designed in the main LabVIEW sub-VI to adjust the weight of both depth and lateral position of the lines as they affect the vehicle's degree of turning. Additional controls enable the user to adjust the proximity (both depth and lateral - both must met) within which a detected line will force the robot to move in reverse, and another control sets the percentage of the forward speed to use when backing up.

B. PID Controller

One of the most costly problems encountered during early testing was inaccurate vehicle motion. When operated on smooth indoor surfaces, Wunderbot was able to move roughly in the intended direction, but once the vehicle was tested outdoors on grass, motion response had a large degree of error. The largest cause for error is the front casters, which require a disproportional amount of force in order to change direction. This problem is an typical case for a PID controller to amend.

The PID closed-loop control was developed in LabVIEW and is very straightforward. The P, I, and D are all user-adjustable via the front panel, and feedback comes from the U.S. Digital optical encoders. Unfortunately, the robot itself is extremely difficult, if not impossible, to model via differential equations, hence classic methods of control theory could not be instituted to determine the value of the PID's constants. Instead, it was a trial-and-error procedure, which led to P=0.500, I=20.00, and D=0.001. Very subtle variations in the derivative constant led the robot to accelerate out of control. A PID controller's derivative constant in general is highly susceptible to noise, and therefore an adjustable low-pass filter was designed for the D [8]. This kept the D from fluctuating too rapidly, while still allowing it to quicken the output's rise time. The resulting transient response can be seen in the figure below.

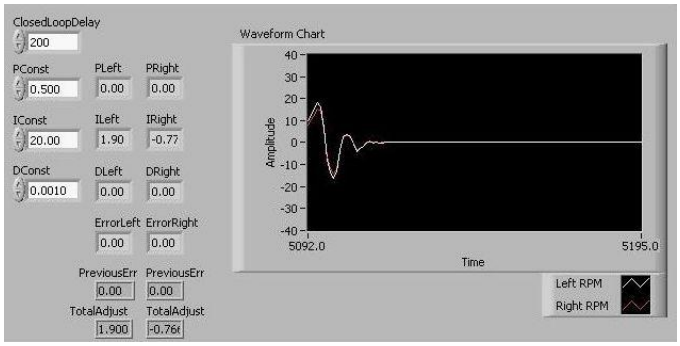


Fig. 5. LabVIEW control panel for PID controller with robot's resulting transient response.

VIII. Results

The effectiveness of the vision system, combined with a well-developed motion control system, was seen in several live demonstrations. Wunderbot IV was able to follow a white-lined, one-turn path in grass, albeit slower than desired competition speed. Future improvements include the creation of a global map, on which the position of the robot will be tracked by the optical encoders. In time, the global map will also incorporate the GPS navigation system and LIDAR in order to visually display all facets of the surrounding environment - target GPS points, white lines, and obstacles.

REFERENCES

[1] B. Theisen, "The 15th annual intelligent ground vehicle competition: intelligent ground robots created by intelligent students," in *Proc. of SPIE*, vol. 6764, Sep 2007.

[2] B. Theisen, et. al., *The 16th annual intelligent ground vehicle competition (IGVC): official competition details, rules and format*, "september 2008" ed., 2007.

[3] D. Carroll, K. Mikell, and T. Denewiler, "Unmanned ground vehicles for integrated force protection," in *Proc. of SPIE*, vol. 5422, 2004, pp. 357–377.

[4] M. Bovard, et. al., "Design evolution of the Trinity College IGVC robot ALVIN," *Journal of Robotic Systems*, vol. 21, no. 9, pp. 461–469, Sep 2004.

[5] M. Tedder, et. al., "An affordable modular mobile robotic platform with fuzzy logic control and evolutionary artificial neural networks," *Journal of Robotic Systems*, vol. 21, no. 8, pp. 419–428, 2004.

[6] M. Tedder and C. Chung, "Autonomous robot vision software design using MATLAB toolboxes," in *Proc. of SPIE*, vol. 5608, Oct 2004, pp. 99–106.

[7] AUVSI, "IGVC Design Reports," 2007. [Online]. Available: http://www.igvc.org/reports.htm

[8] J. H. Lumkes, *Control strategies for dynamic systems: design and implementation*. Marcel Dekker, Inc., 2002.

IGVC Way Point Navigation Solution; Case Study: Wunderbot 4

David Coleman[+]
Elizabethtown College

***Abstract*—This paper summarizes a solution towards the Intelligent Ground Vehicle Competition (IGVC) way point navigation challenge. The Wunderbot 4 of Elizabethtown College has been a test bed for new algorithms and techniques developed by numerous team members. Specifically, the navigation challenge has three main parts: 1) predefining an optimal path to traverse a set of given coordinates in the fastest time, 2) avoiding obstacles that may interfere with the straight-line path between way points, and 3) completing the course by positing the robot facing magnetic north. The first part can be solved "off-line" and delivered to the navigation system as a set of sorted coordinates. The second and third part (along with the actual moving between way points) must be done in a fully autonomous method.**

I. Introduction

As a third-time competitor in the IGVC the Wunderbot 4 has developed numerous methods for handling difficulties in previous venues. Starting in 2006, the Wunderbot 4 came together and devised new subsystems and methods for solving challenges such as obstacle avoidance, line following, directional driving, remote controlling for limited manual drive, GPS navigation, and government communication protocol. A few of these systems will be highlighted within this paper as they pertain to the IGVC challenge of way point navigation.

TABLE I
ACRONYMS AND DEFINITIONS

Symbol	Term
GPS	Global Positioning System
iff	If and only if[f]
IGVC	Intelligent Ground Vehicle Competition
LIDAR	Laser Range Finder
TSP	Traveling Salesman Problem
TSP-BG	TSP Bounded Graph

The IGVC

The Intelligent Ground Vehicle Competition (IGVC) invites robotics teams from around the world to solve challenges such such as obstacle avoidance, GPS navigation, and government standard communication protocols. It is an annual competition that features some of the highest ranked schools in the world from America, India, and Japan. The focus of the competition is to provide new strategies towards modern robotic challenges that will one day be made stream and offered to the public such as autonomous vehicles, service robots, and more precise machining tools.

Being a part of the IGVC has allowed the past (and current) Wunderbot teams the ability to showcase their education in modern robotics as well as provide a platform to share knowledge with other schools and industry leaders such as General Motors, AVIUS, and the U.S. Department of Defense.

II. The Navigation Challenge

The IGVC way point navigation challenge consists of three parts. On the first day of competition each team is provided a list of eight to ten GPS coordinates (latitude and longitude). The points are listed in random order and provide no additional information as to their placement, surrounding obstacle density, or traversability by an autonomous robot. The information that is available is that all the points reside in a 50x50 yard arena in which each GPS coordinate is outlined with white spray paint on grass and a two meter radius of acceptance is permitted. The entire arena is outlined by white lines to ensure the autonomous vehicle (which must recognize the white boundary lines during qualification) will stay within the arena and not pose a threat to other vehicles as well as spectators. A sample course is shown in Figure 1.

The second part to overcome in the navigation challenge consists of obstacle avoidance during trajectory traversal between the given set of GPS coordinates. Again, since the obstacle locations are not known before entrance on the course

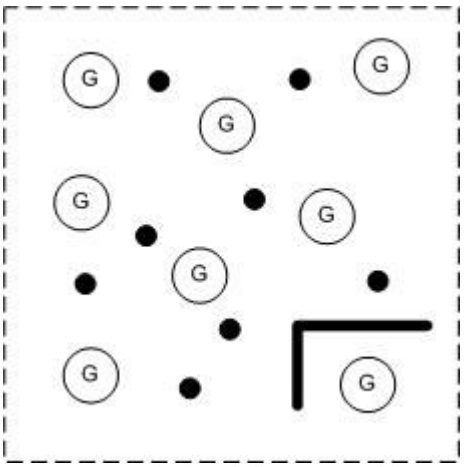


Figure 1 – A sample GPS navigation challenge arena with 8 GPS coordinates marked with the letter "G" and circumscribed by a circle with radius of two meters. The obstacles are shown in solid black with the thick black lines representing a mesh fence and the dotted lines representing the boundary lines.

– inspection of the course prior to traversal attempt will lead to disqualification – precautions and techniques must be employed to ensure the autonomous vehicle does not interfere with obstacles as well as not provide damage to the arena or itself. Obstacles that are found in the field range in dimension and color. Figure 2 shows some sample obstacles which include orange/white stripped construction barrels, orange cones, and a orange mesh fence.

Figure 2 – The GPS course will feature obstacles ranging from triangle cones to orange/white construction barrels. Additionally, there will be orange construction mesh fence and white spray painted lines on the grass.

The third and final part of the navigation challenge is a directional face towards magnetic north upon arrival at the final node. As specified by the rules, the autonomous vehicle must return to the starting GPS coordinate and face magnetic north to guarantee all aspects of trajectory traversal on-board an autonomous vehicle is established.

III. Previous Results

In 2006, at the 14th Annual IGVC the Wunderbot III team placed 18 among 32 teams. Specifically, in the GPS navigation challenge the team placed 9 out of 11 teams that attempted the course successfully completing one GPS coordinate traversal in one minute and nineteen seconds. The winning team in 2006 – Virgina Tech (Johnny-5) – traversed all nine GPS coordinates in two minutes and eight seconds.

IV. Explicit Path Planning

To begin designing a solution to the navigation challenge a mathematical approach was given to the first part of the challenge – defining the optimal sorted order of traversal to upload to the autonomous vehicle. Reviewing the requirements of the challenge includes the following: 1) visited the GPS coordinates in any order, 2) return to starting coordinate (and face magnetic north), and 3) fastest time wins.

To a mathematician this sounds like a very familiar but haunting problem – the "Traveling Salesmen Problem" [1]. The haunting task is in the complexing of the problem which has been proven to be NP-Hard as the number of solutions becomes exponential as the number of coordinates expands. Formally, the number of solutions that exists for a given set of n coordinates is shown to be on the order of:

$$O(n) = !(n-1) \quad (1)$$

Assuming that the number of coordinates defined in the IGVC challenge has never exceeded ten GPS coordinates there are still

$$!(10-1) = 9! = 362{,}880 \quad (2)$$

possible trajectories of traversal within the arena. Therefore, doing a simple hand sort could prove very demanding and require more than one pencil.

Beginning in 2006, a solution was research to make this method more feasible and less demanding. After all, any "optimal" solution defined in this stage will be ultimately not be guaranteed optimal once obstacles are introduced. The first looked at were Kruskal's algorithm and expansion trees. However, each of these do not provide the "return to start" that must occur. So, naturally this left only Hamiltonian circuit methods which lead to the TSP.

Another approach was offered. Instead of ignoring the Hamiltonian circuit approach and being burdened by the TSP limits, develop a method that would enhance the the Hamiltonian approach and will ultimately lead to a solution less than the TSP boundary. Exploring traditional explicit path planners where start/goal points are known as well as the obstacles in the environment lead to the knowledge of Voronoi polygons. The Voronoi polygons found "pathways" [2] that allow for safe traversal around obstacles while moving towards the goal. An example of this is shown in Figure 3.

Since the obstacle locations are unknown in the IGVC navigation challenge this traditional use of the Voronoi polygons cannot be guaranteed to work. However, related to the Voronoi polygons is the Delaunay triangulation. This is developed by connecting straight lines between touching polygons with the arena absent of obstacles. Comparing

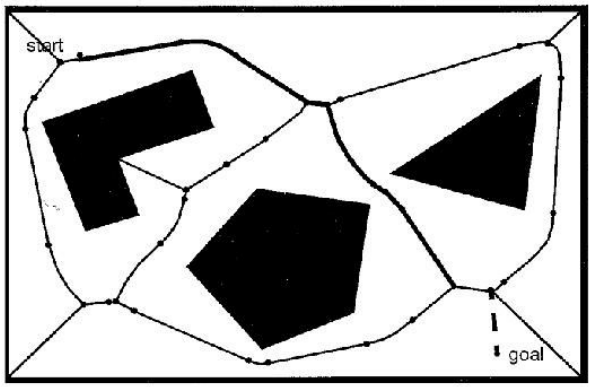


Figure 3 – Traditional Voronoi polygons are used with a fully-explored region with known locations for obstacles and coordinates as shown above. The edges of the polygon provide "pathways" to traverse to avoid the obstacles and reach the end coordinate.

Figures 4a and 4b where 4b is the Delaunay triangulation of 4a it is clear a brute force approach to finding the optimal solution is much easier. In fact, it has been proven to be on the order of:

$$O(n) = n \quad (3)$$

which n << !(n-1) and therefore is much easier to use even on the IGVC challenge where n <= 10.

Method

Matlab 2007a Student Edition was used to test the methods proposed (and is the method that gave the final value of "n" for the Delaunay method). The code for this portion of the navigation challenge simulation was straight forward.

1 Load in the unsorted list of GPS coordinates
2 Obtain the Delaunay triangulation points
3 Use brute force techniques to find a solution
4 Compare the weights on all solutions
5 Output best solution

Furthermore, two additional results were found: 1) the Delaunay triangulation provide a result in less time than the complete TSP-BG (as expected), and 2) the solution provided each time was exactly the same as the TSP-BG. The complete results of the simulation are shown in Figure 5 and show an improvement in TSP search algorithm at approximately 90% at the IGVC limits and a 70% in reduction in pure number of edges found within the graph. Finally, it should be addressed that the cost function from the complete graph to the Delaunay triangulation could not be calculated and was developed around existing Matlab functions.

Equipment

The Wunderbot 4 is equipped with a Trimble AgGPS 114 receiver with DGPS service provided by OmniStar. The information is transmitted via RS232 from the GPS receiver to the PC in NMEA sentence of the following sample format:

$PTNL,GGK,172814.00,071296,32723.46587704,N,12202.26957864,W,3,06,1.7,EHT-6.777,M*48

The first nine of eleven fields are necessary for GPS navigation in an autonomous vehicle. These parameters are listed in Table 2.

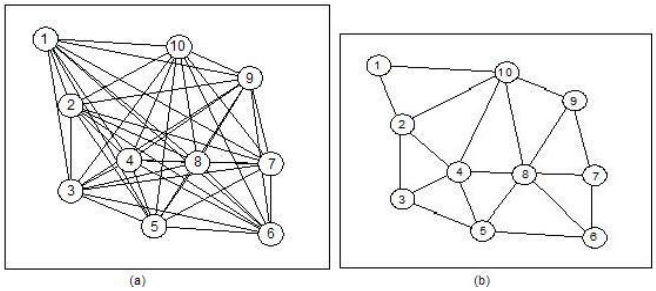


Figure 4 – Improved domain between a) original graph and b) Delaunay triangulation for the algorithm to find the optimal path between coordinates.

TABLE 2
GPS PARAMETERS

Field	Description
1	UTC of position, in hhmmss.ss format
2	UTC Date of position, in mmddyy format
3	Latitude, in degree and decimal minutes; ddmm.mmmmmmm
4	Direction of latitude (N,S)
5	Longitude ,in degree and decimal minutes; ddmm.mmmmmmm
6	Direction of Longitude
7	GPS Quality (0-4) with a value of 4 representing DGPS
8	Number of satellites used in GPS solution
9	DOP of fix

This information is sourced from the NMEA-0183 manual (page 25) in Appendix Binder 3.

V. OBSTACLE AVOIDANCE

Once an optimally sorted list of GPS coordinates is uploaded to the autonomous vehicle a trajectory can be executed restricted by two pieces of equipment: LIDAR and vision system. The information obtained by these two pieces of equipment are fed into the A* algorithm previous developed by A. Stentz from Carnegie Mellon in 1996 [3]. This method of obstacle avoidance uses heuristics to ensure the method of avoidance is towards the end goal node and not in a loss to the cost function of traversal.

The current obstacle avoidance on-board the Wunderbot 4 is two parts. The first is a simple local obstacle detection scheme with a dynamic viewable window. This viewable window goes through a two-fold level of calculations to determine the best path for the robot to follow. A radial filter is first placed on the incoming data of the LIDAR. The data transmission of the LIDAR is in polar coordinates so a radial filter is best. The radial distance is determined by the following equation:

$$Filter_{radius} = (Window_{height}^2 \times Window_{depth}^2)^{1/2} \quad (4)$$

After the filter is applied the obstacles found less than r are converted to (x,y) using the following:

$$x = r * \cos(\Theta) \quad (5)$$
$$y = r * \sin(\Theta) \quad (6)$$

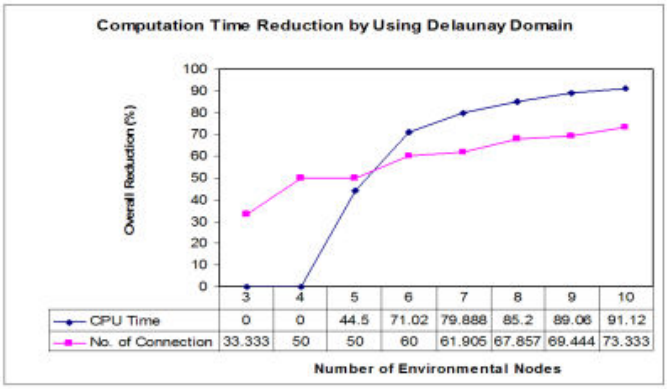


Figure 5 – Using the Delaunay triangulation over the original connected graph has shown a 90% reduction in processing time as well as a 70% reduction in the number of edges on the graph. These were shown through simulations.

Finally, an obstacle is "within the window" iff:

$$x < \tfrac{1}{2} * (Window_{depth}) \quad (7)$$
$$\text{and } y < \tfrac{1}{2} * (Window_{height}) \quad (8)$$

The result of the first part is if an obstacle is found within the window a decision is needed to be made. A polar histogram is developed from the "window" obstacles and is shown in Figure 6. This histogram [4] is useful in determined which direction (left, center, right) has the highest obstacle density and should provide the highest cost function, locally.

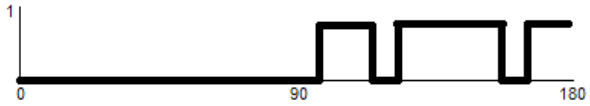


Figure 6 – A binary polar histogram is useful locally in determining which direction to turn based on the obstacle density within a region: left (0-45 degrees), center (45-135) and right (135-180).

To now the methods discussed are local methods for obstacle avoidance. Each part does not contain starting and goal positions relative to the obstacle being detected. A* provides the ability to incorporate the obstacle detected with the end goal to develop a heuristic approach to obstacle avoidance and overall guarantee an optimal prune of the search arena.

The A* method was used in simulations and is implemented on-board the Wunderbot 4. The simulation versus implementation, however, is different as the simulation provided a few assumptions which can not be assumed on the real application. These assumptions include:

1. Uniform motion in any direction
2. Zero-degree turning radius
3. 360-degree sensing
4. No time delay in data acquisition
5. Coordinates of absolute location

With the actual implementation constraints including:

1. Closed loop control necessary to maintain speed/direction after command issued.
2. Turning radius approximately < ½ meter.
3. 180-degree line of sight
4. Equipment transmission delays
5. Radius of acceptance on destination coordinate

14	10	14
10	X	10
14	10	14

Figure 7 – The cost function is dependent on the movement of the robot. Diagonals are cost heavy as those movements require more processing.

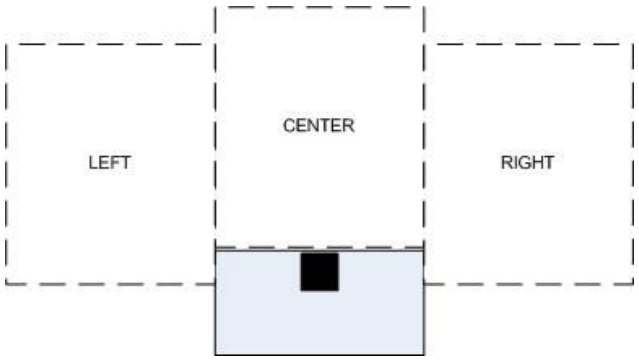


Figure 8 – The actual window and line-of-sight for the Wunderbot is only 3 of the 9 squares available compared to the simulation. This is due to the limited (180 degrees) vision of the robot.

As stated the benefit of using A* is the knowledge of starting and destination coordinates in the cost function. The cost function of A* is three parts:

$$g = \text{distance from start node}$$
$$h = \text{distance to goal node}$$
$$f = g + h \quad (9)$$

The distance calculations are done using the Manhattan method which states that only square paths are to be taken in the X and Y direction independently. Diagonal paths are acceptable at a higher costs as shown in Figure 7.

The simulation results can be found in Appendix Binder #2 with the actual implementation windows shown in Figure 8. Notice the only available windows for the cost function in the implementation are the three windows in front of the autonomous vehicle and that the global solution is maintained using a mix of local detection and global heuristics.

Equipment

The LIDAR is a two-dimensional view mounted approximately six to eight inches from the ground on the front bumper of the vehicle. This equipment is used for obstacle detection in a planar view and can deliver 180 degree resolution up to 80 meters away. For the Wunderbot 4 the LIDAR is configured to deliver 360 data points (½ degree resolution) at approximately ten meters.

The vision system on-board is configured to detect white lines painted on grass at a distance of one-and-a-half meters in front of the robot which is approximately three feet wide. For the GPS navigation the vision system plays a minimal role as the GPS coordinates are outlined in white spray paint so with the methods of current programming the robot will avoid or may even follow the lines instead of entering the two meter radius of acceptance. It is shown that as long as the autonomous vehicle is within the convex hull of the known GPS coordinates the vision system can be turned off and the LIDAR be the only method of obstacle detection.

VI. Implicit trajectories

Finally, an unique opportunity is available when the mix of obstacle avoidance and optimal path planning is mixed. This point of opportunity is available when "avoiding" an obstacle that is large enough to force a deviation from the optimal order in such a fashion that another coordinate becomes locally optimal to traverse out of order. This locally opportunistic yet globally optimal visit to the node out of order is a unique feature to the overall cost function of the Wunderbot 4 path planning scheme.

This method was recently published by David Coleman, "O3: An Optimal and Opportunistic Path Planner (with Obstacle Avoidance) using Voronoi Polygons." [5] Please see Appendix A for this theory and resulting calculations.

VII. Social Context

Extending from the IGVC challenges are numerous opportunities to embed this research in areas of autonomous vehicles for the public use, improving cost functions for current autonomous flight systems, GPS navigation in the car, and obstacle avoidance techniques in a machine robotic arm in a factory. The safety on-board the Wunderbot 4 is three fold: 1) a hard-wired relay on board the robot that will terminate power to all motor controllers, 2) a wire-less emergency stop controller that transmits a signal to the on-board relay, and 3) a software emergency stop controller which places the robot in a halt state. The first two are IGVC required with the third being precautionary but not mandatory.

VIII. Conclusion

As shown the Wunderbot 4 has had a complete overhaul of the software when it comes to the path planning scheme locally for basically trajectories as well as globally for the GPS navigation challenge at the IGVC 2008. The Wunderbot 4 has provided many opportunities for testing the methods developed since 2006 in mathematics, simulations, and now implementation.

IX. References

[1] S.S. Epp, *Discrete Mathematics with Applications*, 3rd ed., Brooks Cole, Boston, MA, 2003

[2] R. Siegwart and I.R. Nourbaksh, *Introduction to Autonomous Mobile Robots*, The MIT Press, Cambridge, Massachusetts, 2004

[3] A. Stentz, "The focused D* algorithm for real-time replanning", International Joint Conference on Artificial Intelligence, August 1995

[4] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots", IEEE International Conference on Robotics and Automation, Leuven, Belgium, May 16-21, 1998, pp. 1572-1577

[5] Coleman, D. and Wunderlich, J.T. (2008). O3 : An optimal and opportunistic path planner (with obstacle avoidance) using voronoi polygons. In Proceedings of IEEE the 10th international Workshop on Advanced Motion Control, Trento, Italy. vol. 1, (pp. 371-376). Pisticataway, NJ: IEEE Press.

The Joint Architecture For Unmanned Systems: A Subsystem of the Wunderbot 4

Jeremy Crouse Computer Engineering
Elizabethtown College
crousej@etown.edu

***Abstract*—The Wunderbot 4 is going to be competing in the Intelligent Ground Vehicle Competition (IGVC) and as part of the competition we, the team, must make changes to the current system. We are in the process of rewriting the entire program that runs the Wunderbot in order to make future implementation easier and current implementation of new subsystems understandable. As part of the team I am in charge of programming the Joint Architecture for Unmanned Systems (JAUS) protocol used by the Department of Defense (DoD) which is a challenge in the upcoming competition.**

***Index Terms*—**
OCU – Operator Control Unit
JAUS – Joint Architecture for Unmanned Systems
IGVC – Intelligent Ground Vehicle Competition
WGS – World Geodetic System
UDP – User Datagram Protocol
ASCII – American Standard Code for Information Interchange
GUI – Graphical User Interface
VI – Virtual Instrument

I. Introduction

The Intelligent Ground Vehicle Competition is a yearly competition where universities and colleges come together to compete and show their achievements in the field of robotics. There are four main competitions that take place at the IGVC: Design, Navigation, Autonomous and JAUS challenges. In the past the Wunderbot has not competed in the the JAUS part of the competition since it was optional and no one on the team had time to work programming that needed done to compete in this area. Needing someone to work on this project for the competition and being new to the team at the time I chose to undertake this project.

II. Research

The Joint Architecture for Unmanned Systems is an architecture specified for use in research, development and acquisition for unmanned systems. The JAUS protocol is an emerging universal command language that all robots under the Department of Defense will use so that different organizations don't have to upgrade to the most current hardware and software technology in order for older versions of autonomous vehicles to continue interoperability with newer autonomous vehicles. This is strictly a message set that was developed by the Department of Defense to allow for interoperability.

The JAUS architecture has to support all classes of unmanned system, rapid technology insertion, interoperable operator control unit, interchangeable/interoperable payloads, and interoperable unmanned systems. According to the IEEE interoperability is defined as the ability of two or more subsystems to exchange information and to use the information that has been exchanged [1]. Classes of unmanned systems can range from air, ground, water and size (e.g. throwable to ship-sized) should not to be an issue when it comes to interoperability.

Interoperable operator control units (OCUs) can range in size with each having different levels of functionality and JAUS should allow for the interoperability of the operator control units. As concepts are developed and technology improves the ability of payload capabilities, organizations must have the ability to update the payload without redeveloping the whole unmanned system and JAUS has to allow technical advancements while not imposing specific hardware or software implementations.

Since some missions are not possible with just one unmanned vehicle JAUS must allow for communication between multiple unmanned systems no matter what platform they are running. JAUS must also be independent of technology, should not allocate functions to any particular element of the system, should not define or measure system performance, should not infringe upon intellectual property and data rights, should facilitate diverse operational procedures should focus on open standards and architectures and should be flexible enough to support a wide range of possible missions and be independent of operator use.

JAUS was also made to be independent of vehicle platform, mission, computer resource, technology and operator use. Vehicle platform independence means that no assumptions are made about about underlying vehicle since unmanned systems include a wide spectrum of platforms.

Mission Independence mans that JAUS must isolate mission specific functions because of there bing so many different types of uses and missions that unmanned systems can be used for. This allows for JAUS to be independent of any specific mission or set of missions that may be built into the unmanned system.

Computer resource independence means that JAUS must maintain computer hardware independence in order to be applicable to all unmanned systems. Technology independence means that JAUS cannot be built around a specific technology solution or else it may eliminate a superior solution. Operator use independence is the last requirement of JAUS which means that it must be independent of any uses the operator will have for the unmanned system. JAUS is not supposed to restrain the user in determining the best approach to accomplishing a mission.

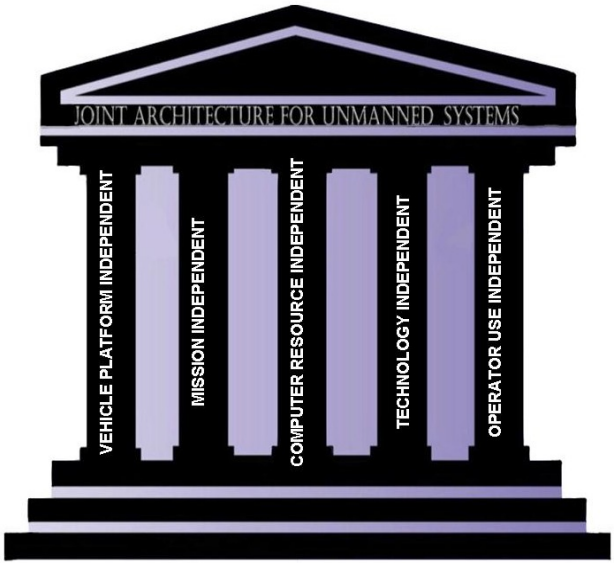


Figure 1: JAUS Independence requirements [3]

There are three different levels of compliance for joint architecture for unmanned systems. The levels are inter-subsystem, inter-nodal, and inter-component. According to the JAUS hierarchy a subsystem is a logical stand-alone operational entity such as an OCU or unmanned vehicle. Within a subsystem the computer processors are roughly defined as nodes and software processes executing within a node is defined as its components.

Level one compliance, inter-subsystem, covers the requirements of communication between subsystems and its purpose is to support the interoperability of the subsystems. Such subsystems that are included in level one compliance are robot to robot, robot to controller and controller to controller. Level two compliance deals with the requirements between nodes and its purpose is to support the interoperability of the nodes.

The nodes involved with this level of compliance are payload to payload and payload to on-board controller just to name a couple. Level three compliance addresses the requirements between the components. Since JAUS is still fairly new this level of compliance is still not defined within the JAUS standards and is not testable at this time.

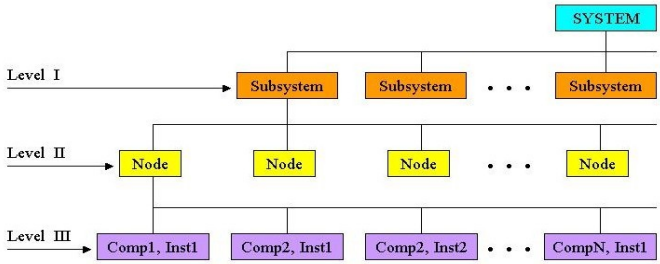


Figure 2: JAUS Levels of Compliance [4]

In the JAUS portion of the IGVC competition we will be using two of the many components that JAUS has implemented into its system. We will be sent information for the Primitive Drive (ID 33 given by JAUS) which is the component that performs basic driving and all platform related mobility functions.

This also includes other devices such as an engine and lights if present. Since JAUS has to promote interoperability the particular platform of wheels (e.g. tracked or not) and power plant (e.g. gasoline, diesel or battery) is not an issue.

The other component that we will be using in the JAUS challenge is the Global Pose Sensor (ID 38 given by JAUS). This component determines the global position and orientation of the platform and the reports the given latitude, longitude, and elevation in accordance with the WGS 84 standard. All this information is being sent via wireless Ethernet to an access point on the robot.

The packet structure that the Department of Defense has chosen to use is the User Datagram Protocol (UDP) protocol and as stated by JAUS compliance only one JAUS message is alloted per UDP packet sent.

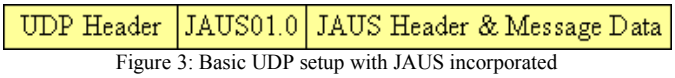


Figure 3: Basic UDP setup with JAUS incorporated

JAUS defines six different classes of messages at the component level. This segmentation is used as a method of organizing the messages and also allows message command codes to be masked prior to evaluation to help with the message transaction process.

Message Class	Offset Range (0000h to FFFFh)
Command	0000h – 1FFFh
Query	2000h – 3FFFh
Inform	4000h – 5FFFh
Event Setup	6000h – 7FFFh (Deprecate v4.0)
Event Notification	8000h – 9FFFh (Deprecate v4.0)
Node Management	A000h – BFFFh
Reserved	C000h – CFFFh
Experimental Message	D000h – FFFFh

Figure 4: Segmentation of Command Codes by class [6]

The command message class is used to effect system mode changes, actuation control, alter the state of a component or subsystem or to initiate some other type of action. The query

message class is used to solicit information form another component. The inform message class allows components to transmit information to each other (e.g. status reports, geographic position, state information). The event setup message class is used to setup the parameters for an event notification message and to have a component start the monitoring for the event trigger.

The event notification message class communicates the occurrence of the event (e.g. engine over temperature, oil pressure, etc). The node management message class is only used by the node management task and is used for node specific communications (e.g. configuration information, component registration). Lastly the experimental message class is used to provide a mechanism for experimentation with new messages that have not been defined in the JAUS reference architecture. This class is mainly there for the expansion of the current message set.

All JAUS messages are required to have a header and data fields. JAUS is setup in this format because it is common to all messages and it allows JAUS to employ an embedded protocol which means that certain fields within the header provide information on how to handle the message. Each field in the message header is interpreted as an unsigned integer value and the header contains information regarding the properties, data size, handling requirements, data encoding and decoding and message routing.

Field #	Field Description	Type	Size (Bytes)
1	Message Properties	Unsigned Short	2
2	Command Code	Unsigned Short	2
3	Destination Instance ID	Byte	1
4	Destination Component ID	Byte	1
5	Destination Node ID	Byte	1
6	Destination Subsystem ID	Byte	1
7	Source Instance ID	Byte	1
8	Source Component ID	Byte	1
9	Source Node ID	Byte	1
10	Source Subsystem ID	Byte	1
11	Data Control (bytes)	Unsigned Short	2
12	Sequence Number	Unsigned Short	2
	Total Bytes		**16**

Figure 5: JAUS message header data format in bytes [6]

The message properties outlined in field one of the message header data are split into six distinct bit-fields. The message priority supports values ranging from 0 to 15. The default priority is what is being sent to us in the IGVC JAUS challenge.

The acknowledge/negative acknowledge behavior is set to none for the IGVC JAUS competition. Property bit six is the service connection indicator which is always set to zero unless a service connection is needed. The experimental bit should always be set to JAUS and the version bits get set to two for the JAUS challenge. The rest of the message properties are not used at this time and are being used for future expansion of the JAUS protocol and are set to zero.

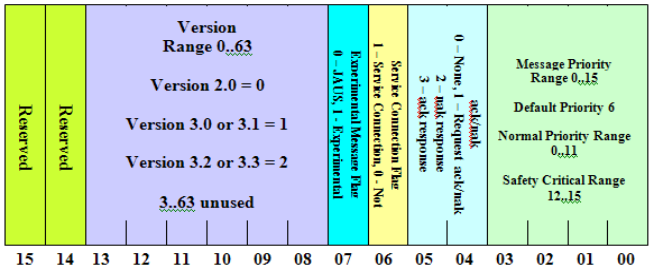


Figure 6: Message Property detailed structure [6]

The header defines the command code in field two which is a two byte numeric value that specifies the type of message and size of its required data and it encoding and decoding characteristics. The source and destination ids that are defined in fields three through ten identify where the message data is coming from (source) and where the message data should be sent to (destination).

These two fields deal with routing commands through the unmanned system. The data control field from bits zero to eleven are for the size of data per transaction and data flag bits from twelve to fifteen are used to control single/multi packet transactions. The last field is the sequence number which is used to serialize messages.

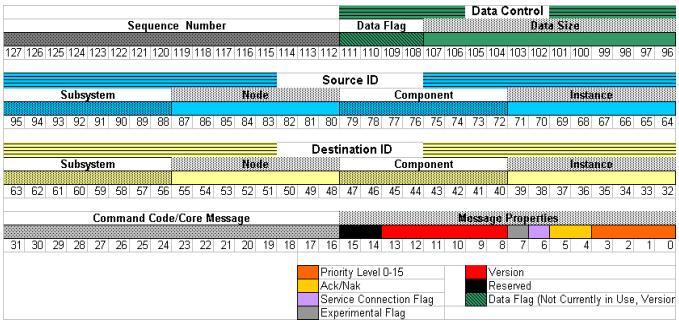


Figure 7: JAUS message header detailed structure [6]

III.JAUS & IGVC

A.Learning Process

After the 2006 IGVC competition JAUS research was started in order to participate in the 2008 JAUS challenge. Since the team did not participate in the 2006 JAUS challenge we had to start from the beginning. Reading through the reference architecture and doing some addition research on line was the beginning of the JAUS process.

After reading through the reference architecture and looking though the header structure coding was started to separate the difference parts of the structure. In order to have the Joint Architecture for Unmanned Systems ready for competition I need to program three different commands to pass the challenge, implement the LabVIEW code into our autonomous vehicle, and be able to receive the commands through wireless communication and have the commands carried out.

B.Implementation

With this implementation on the Wunderbot 4 can read and execute the JAUS message commands from the operator

control unit through the 802.11g data link. During the JAUS challenge our system will be set to monitor for JAUS messages and check for incoming JAUS commands. At this level of implementation these messages will start the unmanned system moving forward in autonomous mode, stop the unmanned system from moving in autonomous mode, and activate a warning device (sound file output to speaker).

C. Challenges Encountered

When this project was first started i did not have any test software except what i had written to test how the header was being separated and i did not know LabVIEW that well. Reading though all the documentation keep the project at a standstill until i could figure out how to decipher what i needed from it.

There were various coding issues that were encountered during this project with the way that the header file was being split up and deciphered. Until the JAUS Compliance Suite was received i had no way to test our software to make sure if it was ready for competition. i could not getting the compliance suite to work for a while until i received an email about a bug in the way I installed it.

IV. JAUS on Wunderbot 4

The router on the Wunderbot 4 is programmed to only allow messages from the JAUS OCU IP address (192.168.128.1) and the JAUS router on the Wunderbot is set to the IP address of 192.168.128.253. The router on the Wunderbot also checks for the JAUS port (3794) and does not except messages from any other port and the frequency channel is set to 802.11g.

In order to change data on the router you must plug the Ethernet cable into the router then to your computer and open the Network Connections window, once in there right click on the Local Area Connection and click on Properties. Then scroll down to Internet Protocol (TCP/IP) and click the Properties button.

When the window appears click the Use the following IP address and enter the IP address (192.168.128.2) and Subnet Mask (255.255.255.0) and then open an Internet Explorer window. Type the IP address (192.168.128.253) into the address bar and then a GUI will show up where you enter the user name (*Admin*), password (*admin*), and check the "I agree to the terms and conditions below" then click the sign in button. This should not need done unless a new router is obtained.

In the JAUS software I first open the UDP Ethernet connection to listen for JAUS messages coming in through that port and IP address. When a message is received I check for the UDP header information that is eight bytes containing ASCII equivalent of "JAUS01.0" then parse that out of the UDP header.

I then double check to make sure the incoming IP address and port are correct for me to be able to receive data and carry out the commands. Next in my code I parse out the message properties and output them to the front panel of my LabVIEW code. The next piece of information that is in the JAUS header is the command code followed by the destination id and the source id and the data control and sequence number. For the competition there is no data control information being sent or sequence number.

After the command code is parsed out of the byte array I send it to the JAUS Command VI to carry out the command. There will be a new portion of code to accommodate the new part of the challenge where the destination id will be parsed out and need to be sent to a new set of commands to get information from the GPS. Screen shots of router setup and code are in Appendix A.

V. Social Impact

With the development of unmanned/autonomous vehicles and technology there have been many social and ethical impacts that we have had to face. Should robots and unmanned systems do our thinking and decision making for us removing the human error or should we keep the human in the equation.

Also the interoperability of these systems has an impact to us. If an unmanned/autonomous search team is sent out to find you lost in the woods would not you want them to be able to communicate no matter what platform they are built on so they could work together to find you faster.

These unmanned systems reduce exposure of humans to harmful environments, preform tasks not possible for humans and provide cost effective solutions to repetitive tasks. A large number of unmanned system products are being introduced to the market and many of these systems are characterized as task dependent and non-interoperable.

That is where JAUS comes into play because it makes these non-interoperable systems interoperable no matter the platform and it also supports the rapid and cost-effective development of unmanned systems.

Appendix

A. Appendix A: Router Setup Screen shots

B. Appendix B: Software Screen shots

C. Appendix C: JAUS Documentation

References

[1] Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.

[2] Pedersen, Jorgen. A Practical View and Future Look at JAUS. May 2006.

[3] The Joint Architecture for Unmanned Systems, Compliance Specification (CS), Version 1.2, 25 October 2006.

[4] The Joint Architecture for Unmanned Systems, Domain Model (DM), Volume I, Version 3.2, 10 March 2005.

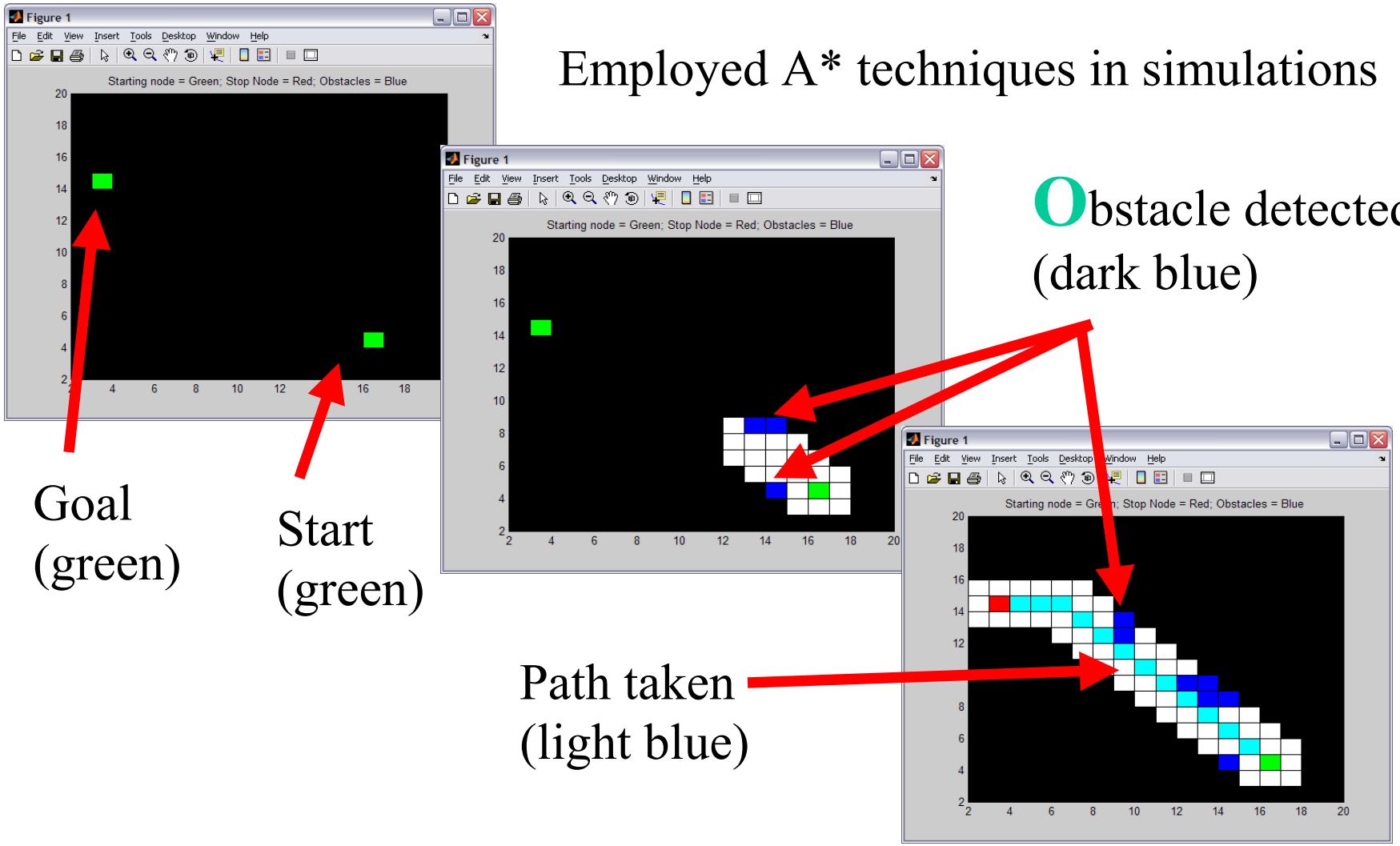
[5] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 1 Architecture Framework, Version 3.3, 27 June 2007.

[6] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 2 Message Definition, Version 3.3, 27 June 2007.

[7] The Joint Architecture for Unmanned Systems, Reference Architecture Specification (RA), Volume II, Part 3 Architecture Framework, Version 3.3, 27 June 2007.

Appendix F

Matlab was used to simulate the A* technique and to further its understanding. Below are screenshots of the simulation written by the Wunderbot 4 team.



Appendix G

The Wunderbot 4 team has enhanced the current Wunderbot platform. Below is the comprehensive budget since the beginning of the Wunderbot project.

W I	W II	W III	Product	Qty.	Act. Total	Our Total	Savings
•			NPC T-74 Motor	2	$598.00	$598.00	$0.00
•			NetGear Wireless PCI Adapter	1	$59.99	$59.99	$0.00
•			Linx RF Development Boards HP-II Series	2	$298.54	$298.54	$0.00
	•		MK M34 Batteries	2	$500.00	$147.20	$352.80
	•		Trimble AgGPS 114	1	$3,000.00	$0.00	$3,000.00
	•		Roboteq AX2550 Motion Controller	1	$495.00	$468.00	$27.00
	•		PNI TCM2-50	1	$769.00	$0.00	$769.00
	•		Over-Current Protection	1	$31.55	$31.55	$0.00
	•		80A Connectors & 8 AWG wire	1	$95.64	$95.64	$0.00
	•		6061 Aluminum Frame	1	$550.00	$350.00	$200.00
	•		NPC PT-5306 Flat Proof Wheel	2	$146.00	$146.00	$0.00
	•		8" Pneumatic Caster (Series 5000)	2	$133.66	$133.66	$0.00
	•		24-12V Power Supply Unit	2	$400.00	$400.00	$0.00
	•		Omnistar DGPS Subscription	1	$1,600.00	$0.00	$1,600.00
	•		LabVIEW Prof Dev System	1	$3,495.00	$0.00	$3,495.00
	•		NI Vision Development Module for LabVIEW	1	$2,595.00	$0.00	$2,595.00
	•		(Book) Img Processing with Labview and IMAQ	1	$80.00	$0.00	$80.00
	•		Laptop for remote monitoring	1	$1,500.00	$0.00	$1,500.00
	•		Silicon 8ga Wire (Black)	15	$22.50	$22.50	$0.00
	•		Silicon 8ga Wire (Red)	15	$22.50	$22.50	$0.00
	•		Hella Contour Master Power Switch	1	$21.49	$21.49	$0.00
	•		Replacement Soldering Tips	4	$20.24	$20.24	$0.00
	•		12 V 10A Battery Charger	1	$100.00	$100.00	$0.00
	•		Misc Frame and Body Materials	1	$527.49	$527.49	$0.00
		•	OS 3100 Laser Area Scanner	1	$5,030.00	$0	$5,030.00
		•	Phoenix Contact Wireless Router	1	$200	$0	$200
		•	Phoenix Contact IPC5500	1	$3,000	$0	$3,000
		•	DVT Legend 554C XE w/ lens and I/O board	1	$6,000	$0	$6,000
		•	Phoenix Contact Connectors and Fuses	1	$300	$0	$300
		•	Motor Controller Repair	1	$250	$250	$0
			Total		$31,841.60	$3,692.80	$28,148.80