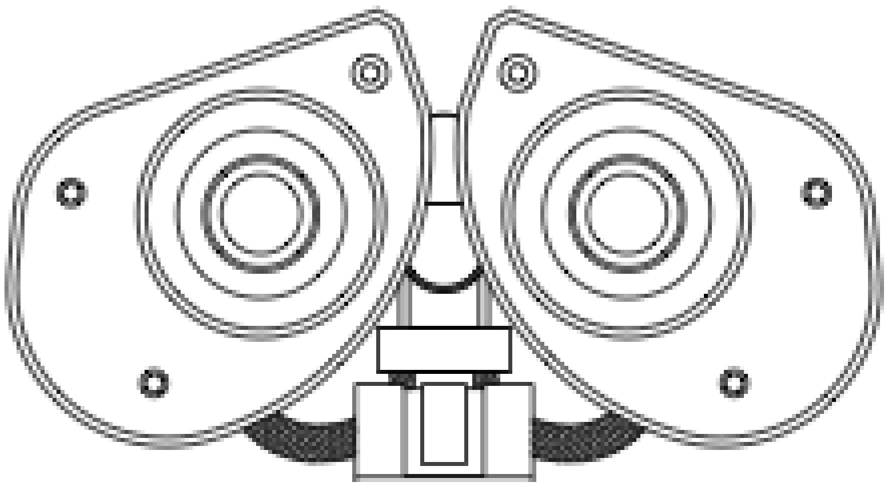

IGVC Design Report

Matthew Duncan, Justin Milam, Jonathan Hawes, Chris Thompson, Louis McAllister

I, Dr. Robert Riggins, Professor of the Department of Electrical Engineering Technology at Bluefield State College, do hereby certify that the engineering design of Wall•E has been significant and each team member has earned at least two semester hours credit for their work on this project.

Signed, Date

____

Phone: (304) 327-4134 E-mail: briggins@bluefieldstate.edu

Table of Contents

1. Introduction

1.1 Overview

The Bluefield State College (BSC) Robotics Team is pleased to present Wall•E for entry in the 18th Annual Intelligent Ground Vehicle Competition (IGVC). Inspired by the Disney movie Wall•E, the Bluefield State Robotics Team created this robot to emulate human senses and decision making as much as possible. Almost every child and robot enthusiast knows about the fictional Wall•E and how he seems human on the screen. Over the years of Bluefield State's involvement in the IGVC, we have strived to design algorithms by analyzing human thought processes and then relating these processes to a mobile intelligent ground vehicle. Therefore our newest project is appropriately based on Wall·E.

1.2 Wall•E Specifications

Table 1.1 lists Wall•E's physical specifications.

Weight	125 lbs (excluding 20lb payload)
Horizontal Center of Gravity	24 inches to rear, 20 inches to front
Vertical Center of Gravity	11 inches to ground
Height, Length, Width	5 feet, 32 inches, 31 inches
Wheel Diameter	13 inches Rear, 8 inches Front
Wheel Base, Ground Clearance	25 inches, 5.5 inches

Table 1.1

2. Design Process

2.1 Overall Design Methodology

The process used to design Wall•E began with an analysis of lessons learned from previous IGVCs. Immediately following each IGVC, the teams would always document what the goals and objectives of the next robot should be. The analyses from previous teams served as a starting point in Wall•E's design process.

One of the Wall•E team's desires was to be more environmentally conscious in the physical construction of the robot. This is in the spirit of mimicking the on-screen version. As a result, Wall•E is composed of 82% recycled materials and equipment.

Figure 2.1 shows the design process that the Wall•E team used. The Wall•E team divided into hardware and software groups, and each group followed this design process. First, participation of the previous team in the IGVC2009 resulted in a Post-IGVC analysis by that team. The team then met, brainstormed, and developed proposed design solutions. A decision about which solution to implement was made through the hierarchical team structure (See Section 2.2). At this point, we tested each solution in a simulation, real life, or both, and analyzed the results. If the results proved that our solution was successful, we proceeded to the next item on the analysis. Otherwise we re-analyzed the proposed solution. Figure 2.1 shows an abstract view of our design process.

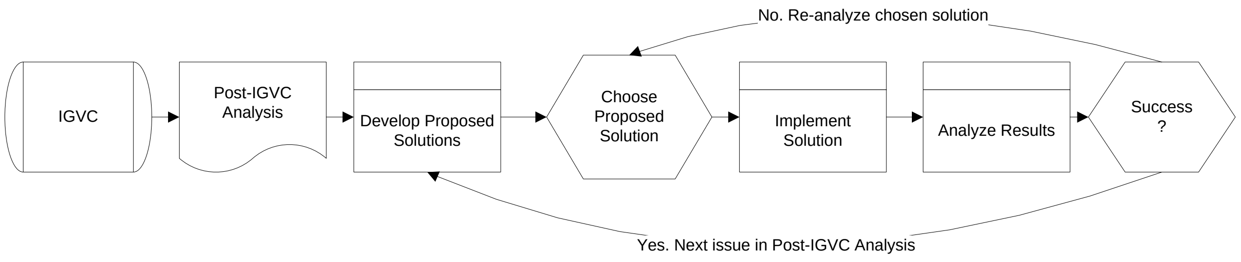
No. Re-analyze chosen solution
IGVC
Post-IGVC Analysis
Develop Proposed Solutions
Choose Proposed Solution
Implement Solution
Analyze Results
Success ?
Yes. Next issue in Post-IGVC Analysis

Figure 2.1

2.2 Decision Making Process and Team Structure

Our team is comprised of undergraduate students from the Electrical Engineering Technology, Computer Science, and Mechanical Engineering Technology departments at BSC. Our primary sponsor and helper was Shawn Anderson, previously a graduate student from Mountain State University. Each member contributed to all or part of the design and fabrication of Wall•E. We estimate that our team spent 500 hours designing and fabricating Wall•E. The team structure and members are listed in Figure 2.2.

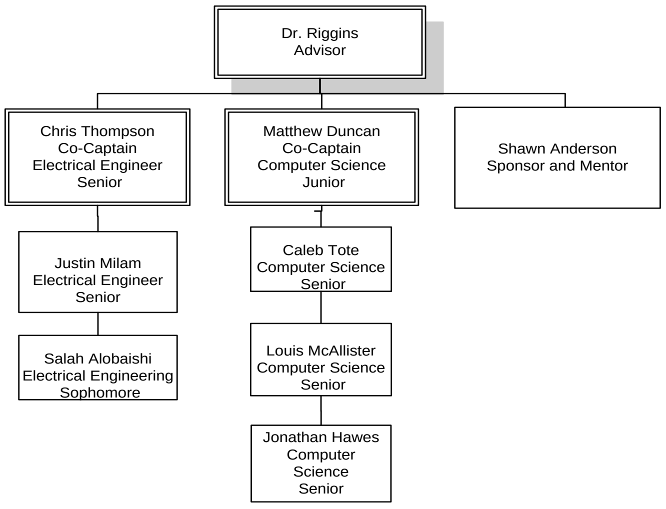

Figure 2.2

3. Hardware Design

The hardware design of Wall•E is divided into two sections: mechanical design and electrical design. Table 3.1 lists the replacement cost and the team cost of each component and the total cost of Wall•E.

Description	Replacement Cost	Team Cost
HP tx1000 Laptop	$1,300	$1,300
Sony camera	$700	$129
1 180 degree LMS (SICK)	$5,000	$3,000
DGPS-w/antenna/cables	$3,000	$300.00
Steel Frame	$50	$50
Wireless E-stop	$100	$100
Compass	$700	$700
24 to 12 volt dc to dc	$200	$50
LCD Display	$400	$35
4 35 amp 12V Batteries	$300.00	$170
Microcontrollers	$65	$65
Miscellaneous	$300	$300
Jazzy wheelchair	$5,977	$599
Total	$18,182	$6,498

Table 3.1

3.1 Mechanical Design

3.1.1 Body

The body of Wall•E was designed so that it would look like the movie version and be sturdy. The team wanted to require no more material than we deemed necessary to fit all of our equipment inside. The body is made of a steel frame and recycled materials such as fiberglass, plastics and metal. By using these materials, not only is the body very durable, but it is also very light. We wanted the body to be lighter than the drive train in order to keep Wall•E's center of gravity as low as possible.

3.1.2 Drive train

The drive train of Wall•E is the modified chassis of a Jazzy Jet 7 electric wheelchair. The frame is constructed of square steel tubing that is built to carry a payload of three hundred pounds. The chassis components include an active suspension system that allows all wheels of the drive system to travel independently as the vehicle moves across uneven terrain. The vehicle is elevated by thirteen inch solid tires (the drive wheels) and eight-inch front articulating caster wheels. The arrangement of the wheels provides five and a half-inch ground clearance which generates a low center of gravity. This, coupled with the active suspension system, gives the robot a curb-climbing height of six inches. The overall robot base of thirty-one inches wide by thirty-two inches long provides a tight turning radius of twenty-three inches.

3.1.3 Choice of Frame

The tam decided to modify a frame from a Jazzy Jet 7 wheelchair for three reasons. First, with a ready-made frame we saved both time and money designing and fabricating the bottom of Wall•E. In the past several IGVCs, the BSC teams have successfully used wheel-chair bases. Second, the Jazzy wheelchair was designed over the course of twenty years to be strong enough to carry a 300-pound person safely. This makes it strong enough to carry all of our equipment and the payload. Third, the rugged design of the base makes it reliable and durable.

3.2 Electrical Design

3.2.1 Power

Wall•E uses two 12-volt batteries to supply power to all of its electronic devices in a twenty four volt configuration. This battery set can be interchanged easily with another set. All devices are connected to this single power source, so we do not have to charge multiple power sources. Each battery is rated at 35 amp-hours. By dividing the total amp-hours of the batteries by the total amps consumed by all the devices, we calculate that the total battery life of Wall•E is 1.6 hours under normal operating conditions and 60% operating efficiency. Our observations concur with this maximum run time. See Table 3.2 for a complete list of devices and current draw.

Component	Average Current Draw (Amps)
LMS	1.5
GPS	1
Compass	.15
Camera	.5
Wireless E-Stop	1
Motor x 2	20
Controllers	2

Table 3.2

3.2.2 Sensors

The sensors on Wall•E are used for obstacle detection, positioning, and communication. The following list describes the sensors that Wall•E uses to sense its environment.

One SICK Laser Measurement Systems (LMS) – measures distance between robot and objects. Maximum Range: 8 meters. Resolution: 1^o. Sweep Angle: 180^o. Precision: one millimeter.

Maretron Solid State Compass – determines heading in degrees. Precision: 0.1^o.

Trimble Pathfinder XR DGPS Receiver and Antenna – determines latitude and longitude of the robot as well as velocity. Precision: 2 feet and 0.01 feet per second at one Sigma (67% of the time).

Sony Handycam – captures still images of Wall•E's surrounding environment. Resolution: 640x480 pixels.

3.2.3 Computer Systems

The computer in Wall•E is a high performance laptop with a dual core processor and two gigabytes of ram. In addition to the main computer, the robot also uses several microcontrollers to handle various tasks. An 8-bit microcontroller controls Wall•E's motors, another microcontroller controls the wireless ESTOP, and another microcontroller controls miscellaneous devices such as the indicator light. This distributed computing allows the main computer more time to perform the sensor integration and path planning described in later sections.

3.2.4 Actuators

The actuators consist of two twenty-four volt DC motors that move the vehicle. These motors are connected through a speed reducer to the drive axle. The speed reducer converts the high speed, low torque of the motors to the low speed, high torque required to move the 150-pound Wall•E at top speed. This produces a top speed of 4 mph for Wall•E. It also allows Wall•E to climb up to a 30^o incline. This calculation was predetermined by Pride Mobility Products Corporation for the Jazzy wheelchair.

3.3 Safety

Wall•E incorporates several safety features. First, the hardware utilizes multiple Emergency Stops (E-Stops) to provide several levels of control. Wall•E now has three ways to manually E-Stop the robot, and three wireless ways. Second, software programs continually monitor the system for errors in control, communications, and battery charge levels. Third, “Heartbeat” (fault monitoring) signals are monitored at critical points in the system. Finally, the software has the inherent “fail-safe” ability to abort the current mission and shutdown the vehicle in the event an error is detected. Wall•E has both a Soft E-Stop (a large red “Easy” button) and a Hard E-Stop located on the rear instrument panel. The on-board Hard E-Stop switches all power on/off. The on-board Soft E-Stop and the wireless E-Stop both switch the controller on/off, but they do not affect the main power. These two systems are independent of each other for additional safety. The wireless radio-controlled (RC) E-Stop has been installed to extend our control range to fifteen hundred meters.

For electrical safety, each device is connected to its power source through a fuse box, and the fuse for each device is the correct size to allow for the maximum safe current. Wall•E is also safe while charging. When the chargers are connected to Wall•E, the motor controller will not allow the robot to move using manual control methods or computer control.

4. Software Design

4.1 Signal Processing

The inputs to our software program come from the data collected from our external sensors, including the compass, camera, DGPS unit, and the LMS unit. Wall•E makes its decisions based on that data so the signal received from each device must be processed into a form that Wall•E can use. Data from the four basic sensors, LMS, GPS, compass, and camera, have radically different formats, but combinations of the data from different sensors must be used to make each decision.

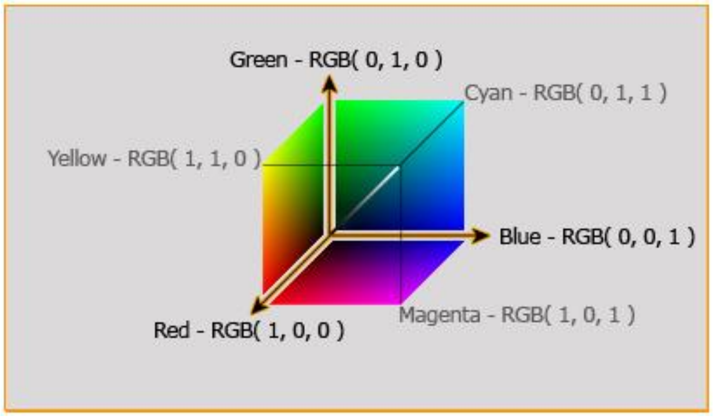

Figure 4.1

Of these four sensors the camera data processing is the most involved. A software frame grabber retrieves frames from the camera and saves them as images so that the software can analyze the content. The Wall•E team uses a process we call "color vectors" to analyze camera data. We analyze each pixel's red, green, blue (RGB) content and compare them to pre-stored values. Every color in the spectrum is made of a certain amount of red, green, and blue. Our software analyzes the RGB vectors to determine if the pixel's vector is close enough to the hypothesized vector of an object or marking (See Figure 4.1). Our software recognizes colors as either passable or not, based on the rules set by the IGVC.

4.2 Mapping the Inputs

Wall•E uses a "map" to represent obstacles inside the software. This map is a two-dimensional array of integers that measures 80 by 80 nodes. Many pixels make up each node and each node in the array represents about four inches of real-world space. We chose this size because it is approximately the size of the smallest feature on the courses at IGVC, slightly more than the width of a white line (3 inches.) After the computer receives the data from each of the sensors, it places that data onto the map. An obstacle is stored as a cost on the map. If a node has a high cost, the robot knows that it should not go through that space. The cost assigned to a node depends on what sensor reported that obstacle. For instance, the LMS data has a higher cost than the camera data because the LMS is more. Once all the data has been placed on the map, Wall•E can begin the process to choose a path that will avoid the obstacles.

4.3 Decision Process

Two similar sub-programs exist in Wall•E's software. One sub-program does the autonomous path-planning navigation while the other does autonomous waypoint navigation. These sub-programs are similar in that they use the same basic path-planning algorithm. The difference between them is in the cost equation that each one uses for determining the desired destination. First we will discuss the path-planning algorithm that is common to both sub-programs.

4.3.1 Path-Planning Algorithm

The first task in our path-planning algorithm is to decide on the desired destination. We call this destination the goal node and the program that finds the goal node is called the "set goal algorithm." To select a goal node, the software analyzes the nodes on the map based on a set of parameters and a special cost equation. The parameters and cost equation for selecting the goal node are the key differences between the two sub-programs and will be discussed in detail in Sections 4.3.2 and 4.3.3. Regardless of which method of selecting the goal node is used, one condition remains the same: the goal node can never be placed somewhere that is unreachable. The process of analyzing the nodes in the map is the same for both sub-programs. The software starts at the robot and begins to analyze the nodes in the map. Each “clear” node whose value is not representative of an obstacle is then entered into the cost equation. The node that minimizes this cost equation becomes the goal node.

The algorithm’s next task is to decide what path the robot must take in order to reach the goal node. Since the set goal algorithm described above cannot select a goal node that is unreachable, the program is guaranteed to find a path to the goal node. In order to do this we use a process called the “ripple algorithm” developed at BSC. The ripple algorithm assigns weights to the nodes starting at the goal node. The first group of nodes that surrounds the goal node, and are not obstacle nodes, are given a weight of one. The next group that surrounds the first group is given a weight of two. This continues until the algorithm reaches the position of the robot.

The next task in our algorithm is to find at least one path from Wall•E to the goal. This is done by using a “waterfall” algorithm, another process first developed by previous BSC teams and updated by this team. There will be a high weight on the node that represents Wall•E’s location and a weight of zero at the goal node. Starting at the robot, the waterfall algorithm looks for an adjacent node with the next lowest weight. There are situations where multiple nodes with the same weight are adjacent to the current node. In these situations, the waterfall algorithm will choose the node that is more in line with the direction to the goal node.

The path created by the waterfall algorithm is not a smooth path. The reason for this is that most of the next-node decisions made by the waterfall algorithm involve a 45^o or a 90^o change in direction. To smooth the path, we created an algorithm to smooth the jagged edges of the path. This process cuts the corners if possible, but it will not cut corners into an obstacle. By cutting corners, Wall•E is able to move close to objects without hitting them, thereby increasing its overall speed. Figure 4.2 gives an overview of Wall•E’s path planning algorithm.

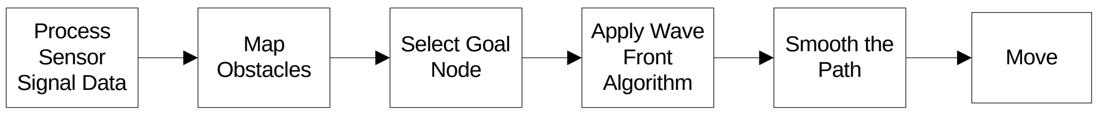

Figure 4.2

4.3.2 Decision Process Based on Lane Following

As stated in Section 4.3.1, there are two main sub-programs in Wall•E, and the key difference between the two sub-programs is the parameters of the cost equation. Our program for autonomous navigation based on lane following uses the following parameters to decide which node should be the goal node: *Distance, Slant, Gap, Confidence, and Straightness.*

The parameter *Distance* indicates how far a node is from the robot. It is not efficient to always choose a node that is close to the robot's current location as the goal node because when the goal node is close, Wall•E moves slower, and it becomes easier to get trapped. By placing a lot of importance on the parameter *Distance* the algorithm will tend to choose a goal node with the largest distance from Wall•E.

The parameter *Slant* indicates which way the path is likely to go. The best way to understand this parameter is to think about how a person driving a car knows which way a road turns. When a person drives down the road, he can get an idea which way the road will turn ahead of him by looking at the edges of the road. On the edges of the road, there are lines, tree-lines, and other such markings that tend to run parallel with the road. By noticing the trend that these lines follow, we can anticipate which way the path will go. In the same way, our algorithm measures the amount of slant on our software map. By placing a lot of importance on the parameter *Slant* the algorithm will tend to choose a goal node with a direction that lies parallel with the slant angle.

The parameter *Confidence* indicates how much confidence we have in the *Slant* parameter. Although we may calculate a slant of the map, the actual amount of slant can vary greatly. By placing a lot of importance on the *Confidence* parameter the algorithm assumes the map definitely slants according to the slant angle.

The parameter *Gap* indicates how much a given node lies between two markings or obstacles. By putting a lot of importance on the *Gap* parameter, Wall•E will prefer the path that has a reasonable gap with markings or obstacles on both sides with a separation of 1 to 3 meters.

The parameter *Straightness* is a measure of how straight ahead a node is from Wall•E. The robot prefers to go as straight as possible in order to maintain higher speed. By including straightness as a parameter, Wall•E chooses the route to a destination with the least amount of turning.

Our algorithm uses another operator-defined parameter called the importance vector. This vector assigns the relative importance to be placed on each of the five parameters described above.

These five parameters described above are entered into a cost equation as shown in Equation 4.1.

$$cost = f(distance, slant, confidence, gap, straightness, importance\ vector)$$

Equation 4.1

The node that minimizes this cost equation becomes the goal node.

4.3.3 Decision Process Based on GPS Waypoints

The parameters used in waypoint navigation are different than the parameters used above for lane-following. These parameters used for waypoint navigation include: straightness, waypoint pull, obstacles, and range. Each of these parameters is described below.

Straightness is the same as the straightness parameter used in the lane-following sub-program. We use this parameter in GPS waypoint navigation for the same reason described in the lane-following section above – that is, how straight ahead a node is from Wall•E.

Waypoint Pull is used to keep the robot going toward the current waypoint. This parameter is a measure of how much a node is in line with the waypoint. The more a node is in line with the waypoint, the lower the cost will be.

Range is what we use to cause the robot to go the farthest distance it can on its way to the waypoint. In the case of traps and heavy obstacles, we also use this parameter to cause the robot to temporarily "forget" about going to the waypoint and instead focus on getting around the obstacles in the robot's way. Without this, if there is a trap between the robot and the waypoint, it could get stuck because the robot wouldn't know to go around; it would want to go straight to the waypoint. For this reason, we added a variable to count the number of nodes that are marked as obstacles (called the *Obstacle* parameter.) If this variable goes above a certain threshold, we put more emphasis on the range parameter. Otherwise, we put more emphasis on the waypoint pull parameter.

The cost equation for waypoint navigation is very similar to the one based on lane-following described above. Again, the purpose is to minimize a cost associated with each node, resulting in a goal node selection.

4.4 Output

Wall•E's algorithm now sends speed and angle commands to the controller according to the path determined by the methods described above. In order to do that, both the desired speed and direction must be calculated based on the direction of the next node in the path. Speed is calculated based on how close the robot is to any objects. If Wall•E is in a situation where there are many obstacles close to it, then it will not go fast so that it can make tighter turns. If there are no obstacles around, then Wall•E will choose to go as fast as possible toward the goal.

4.5 Safety and Reliability

It is the intent of our team to make Wall•E's code safe and reliable. There are several ways the software accomplishes this task. First, the code provides safety in that when Wall•E detects obstacles within 0.6 meters, the hazard light will flash. Secondly, in the event the computer malfunctions, the microcontroller that governs the motors will not allow the robot to move until the computer resumes normal function.

5. Conclusion

The Wall•E team at BSC is confident that the robot will do very well at the 2010 IGVC. We know this because our algorithms have evolved from very successful algorithms created by former teams from BSC. We also chose to mimic a robot that is only fictitious and in the "movies". We made this choice in August 2009 for one main reason; that is, we want this robot to represent our push to make our robots seem more human-like in their capabilities. We feel very strongly that IGVC requirements can be satisfied by algorithms based on what the human mind would do. In future IGVCs we plan to encourage the students behind us to step up to this challenge. Wall•E may just be the inspiration they need.