LinBot

California State University
Northridge

Team Members
Acosta, Flavio
Douglas, Rolando
Park, Sam
Pismenny, John
Santiago, Clement
Sarmiento, Paco
Song, Suk Bae
Wood, Steve

Required Faculty Advisor Statement:

I certify that the engineering design of the vehicle described in this report, LinBot, has been significant and that each team member has earned four semester hours, of senior design credit for their work on this project.

C. T. Lin
Department of Mechanical Engineering
California State University, Northridge

1. Introduction

California State University Northridge is proud to present LinBot for entry into the 16th Annual Intelligent Ground Vehicle Competition. The LinBot team is comprised of multi-disciplinary engineering students. This year's main goal was to completely redesign and upgrade the Navigation and Autonomous software suites while reusing and slightly upgrading the vehicle platform.

1.1. Team Organization

This years' team was based on a chain of command structure. The team leader helps facilitate communication with the different groups as well as keep the project as a whole in perspective and on track.

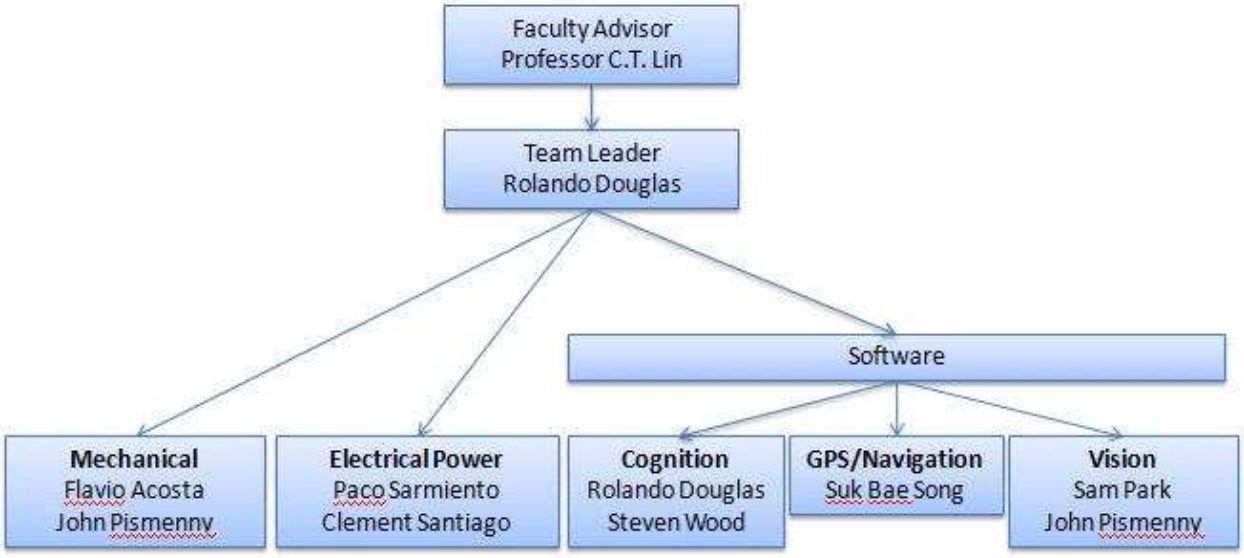

Figure 1. Team organization chart.

1.2. Design Innovations

As the third year of the team, we have brought new innovations to the design of our vehicle. The first is the shared variable engine. Using shared variables in Labview, we share data from several parallel processes simultaneously. Second is the vision system uses particle analysis, line probability mapping, and a line continuity algorithm in its filtering process to extract accurate line data and generate path goals. The third is local and global probabilistic obstacle mapping which are constructed from Laser Range Finder (LRF) and line data. Our cubic spline algorithm uses this mapping to generate a path for both the autonomous and navigation challenge. Fourth is the cognition system which uses Kalman filtering of several sensors to get an incredibly accurate robot position and heading. Last is the Power distribution panel. Every sensor has a switch creating a modular electrical system that can be easily fixed.

2. Mechanical Systems

2.1. Mechanical Development

The development of the vehicle went through 4 stages. These stages are illustrated in figure2.

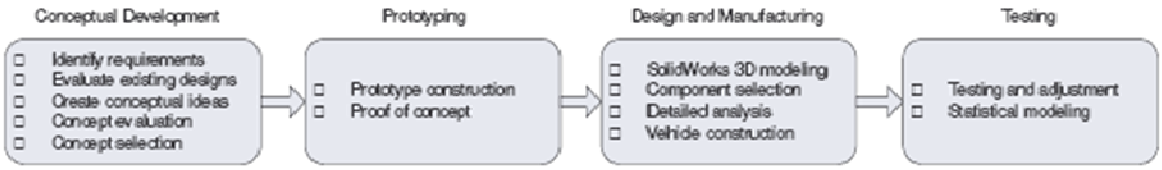

Figure 2. Mechanical development process.

2.2. Chassis

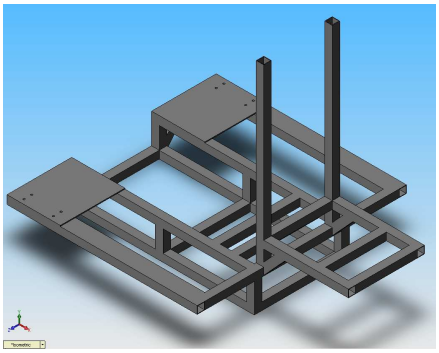

Figure 3. SolidWorks rendering of chasis.

The chassis for LinBot was designed to have a lower center of gravity and a strong frame (Figure 3). Size, position and weight of all components were taken into consideration and were design with this in mind. The frame is constructed of square steel tubing that has been welded together and polished. Due to miscalculations in the gear sizing, two sets of gear boxes were sheared right before last year's competition. The new gearboxes were too large for the steel chassis, there fore a quick light weight solution had to be devised. The team decided to mount the drive train onto a wood board which was fastened under the steel chassis. This unconventional chassis of wood and steel has proven to be reliable and sturdy, it was used in last year's competition and it has been used throughout the whole year without any problems.

2.3. Wheel Configuration

The choice of differential drive set the base point of the wheel configuration. For higher stability, a three wheel configuration was chosen having the drive wheels located in the back with one pneumatic castor wheel for support in the front of the vehicle.

2.4. Mobility

A differential drive robot such as LinBot has only on kinematical restraint on robot mobility. That restraint limits lateral motion of the drive wheels and makes the problem of vehicle control much simpler.

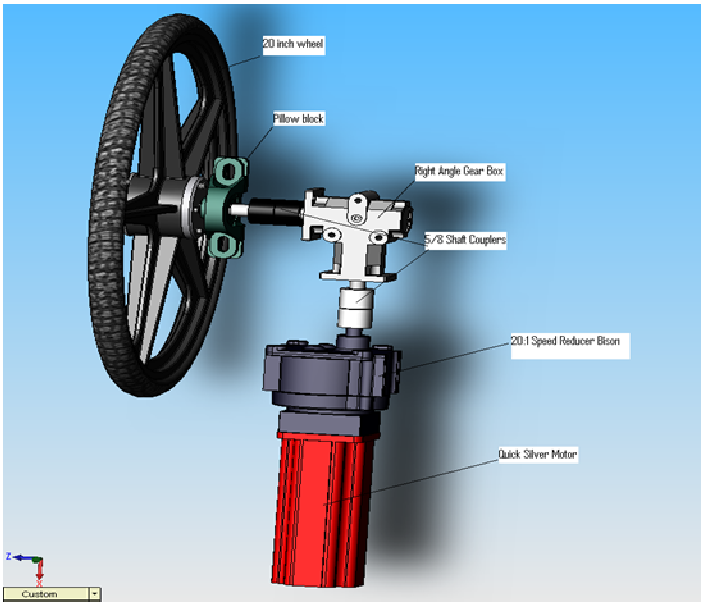

Figure 4. Right angle drive train configuration.

2.5. Drive train

Two Quicksilver 34HC-2 motors were selected to power the drive train. They are connected to Bison gearboxes having a gear ratio of approximately 21:1. In order to reduce the width of the vehicle, we used right angle gearboxes to achieve a right angle configuration as shown in Figure 4. Couplers are used to connect the gearboxes and the wheels.

2.6. Shell

The design requirements took into account the frame and the placement of the internal components. The shell is fabricated of fiberglass with structural reinforcements. The shell is designed to be weatherproof, dustproof, and able to take high amounts of stresses. One sheet of 4 millimeter thick Nomex honeycomb core was used as reinforcement to support the payload. Figure 5 shows the mold for the shell split into 7 pieces. These pieces were produced for our NC Code which was used for and placed into the CNC machine. Figure 6 shows an isometric view of our final shell design. The resin and fiberglass set to cure. The shell was then removed from the mold and then cut to specification. The shell has doors installed into them for easy access to the electrical components.

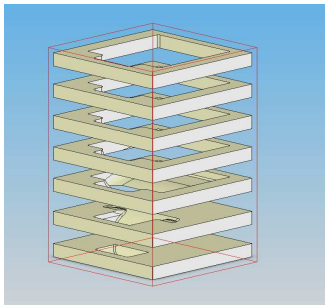

Figure 5. Layered view of shell mold.

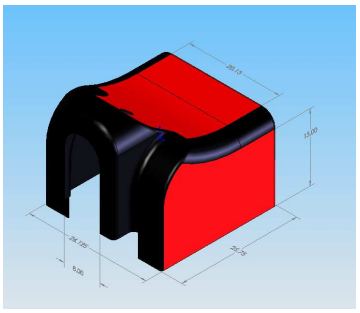

Figure 6. SolidWorks rendering of final shell design.

3. Electrical System

3.1. System Layout

Figure 7 is the design of our electrical system layout. Green blocks represent high power outputs, red and blue circles are switches, dark orange rectangles are output power, light green rectangles are the backing for the fuse holders and the light blue is AC power supply input.

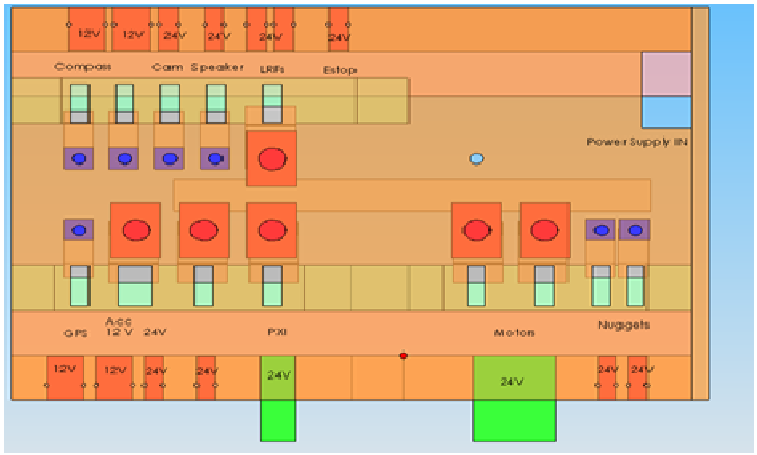

Figure 7. Electrical layout.

3.2. Power Distribution

Two 12 volt batteries are connected in series to provide 24 volts of power. These batteries are connected to each other and to the main switch

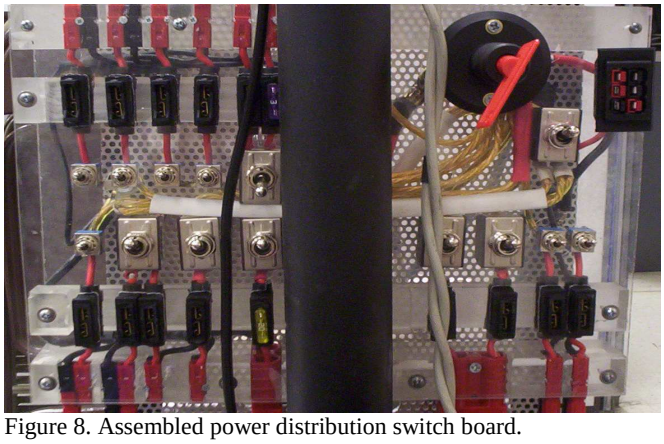

Figure 8. Assembled power distribution switch board.

board (Figure 8) using oversized power cables for low voltage drop at peak current. These cables were also chosen to handle our nominal current need of 30 and the maximum current need of 57 amps under extreme motor conditions. All systems have been setup to run on DC power of 24 volts, or 12 volts. The power distribution box is custom made with Anderson connectors, switches, and fuses for every sensor. A remote control system is implemented using standard remote control car/airplane parts. All sensors are set to run on DC sources. The laser range finder, camera, PXI, and motors are 24 volts; and the digital compass, remote control, router, and GPS use 12 volts.

3.3. Batteries

LinBot is powered by 24 volts system using two SVR 12 volt batteries (Figure 9) connected in series. These batteries are lighter than the previous Optima batteries. Each battery provides 28 amp hours giving us approximately one hour of power.

Figure 9. SVR 12V battery.

4. Sensors

4.1. System Integration

The Robotic Systems Mission Computer (RSMC) is the new mission computer to replace the old PXI aboard the LinBot. It is fully designed from the ground up meeting all specifications to handle the computational tasks and communication requirements with the LinBot sensors. An advantage of the RSMC over the PXI is that it's cheaper and faster to replace parts because their widely available off the shelf unlike the PXI which is composed of proprietary parts. In addition, RSMC runs an Intel Quad Core 2.4GHz processor with 4GB of 800MHz DDR2 memory which computes many more tasks at a faster clock speed compared to the PXI which is running a single Pentium processor with 1GB of DDR memory. To handle all the sensors onboard the LinBot the RSMC is equipped with a special serial card which contains 8 customizable ports at customizable speeds. Each port can be set to RS-232, RS-422, and/or RS-485 with any baud rate ranging from 50bps up to 1.8432Mbps depending on the serial communication used in addition to its 6 USB ports, a FireWire port, and an Ethernet port for all the other sensors. RSMC incorporates a DC power supply which eliminates the need for an AC to DC converter onboard the LinBot. The end result is a robotics systems mission computer enclosed in a mini tower case that conserves space, weight, and handles the required computations with minimal effort. Figure 10 represents the current system

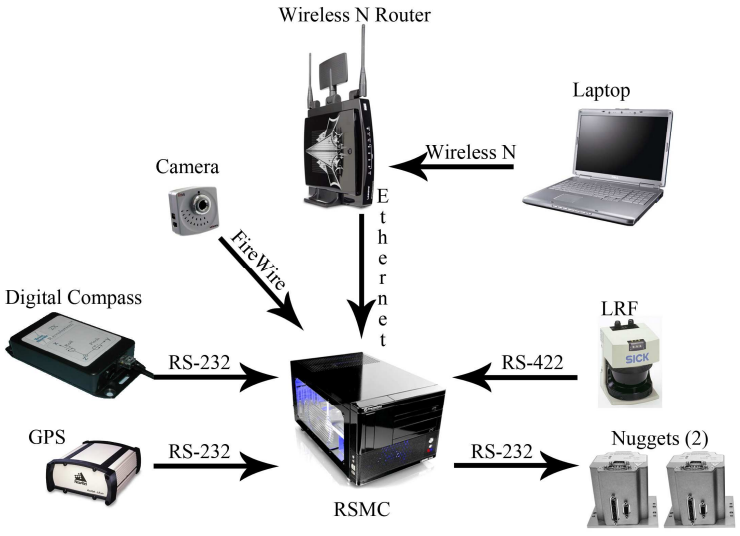

Figure 10. Assembled power distribution switch board.

integration of the LinBot sensors communicating with the RSMC.

4.2. Digital Camera

LinBot's vision system consists of a digital camera created by the Unibrain company as seen in Figure 11. The UniBrain has resolution capabilities of up to 640X480, selectable frame rates of up to 30fps, IEEE 1394 and USB data interface along with a 2.5mm Focal lens. The vision system will detect and process the necessary data of the lines which the robot is specified to stay within. The UniBrain is then used in conjunction with a linear polarized lens along with a neutral color filter.

Figure 11. UniBrain camera.

4.3. GPS

The GPS system used in LinBot is the Novatel Pro-Pak LBplus system seen in Figure 12. It is shock, water, and dust resistant. It can provide position accuracy of 1.5 meters CEP (Circular Error Probable) without a correction signal. The system uses Omnistar HP differential corrections to achieve an accuracy of .10 meters CEP. CDGPS and WAAS corrections can also be acquired to improve positioning. The system communicates with the RSMC through an RS-232 port. It can be powered by +7 to +15 VDC and it consumes 3.7W of power (typical). The system will be used in the navigation challenge to localize the vehicle and guide it towards the waypoints.

Figure 12. GPS receiver.

4.4. Laser Rangefinder

LinBot's main source of obstacle detection is a SICK LMS291-SO4 laser rangefinder seen in Figure 13. This device is capable of scanning a range of 180º in 0.25º increments, measuring distances up to 80m away. The settings used for LinBot make the device scan a range of 180º in 1º increments, measuring distances up to 8m away and returning values in mm. Anything further than 8m will not be considered by the obstacle avoidance algorithm, and 1º increments are sufficient at this distance. An RS-422 serial interface was used in order to obtain a data transfer rate of 500kbaud.

Figure 13. Laser Range Finder.

4.5. Digital Compass

The compass that we're going to be using for the LinBot is 2X Revolution by True North is depicted in Figure 14. This device has an accuracy of 0.3 degrees and can be updated frequency of 12Hz. It communicates using an RS-232 serial communication interface for data transmission.

Figure 14. True North 2X Revolution Digital Compass.

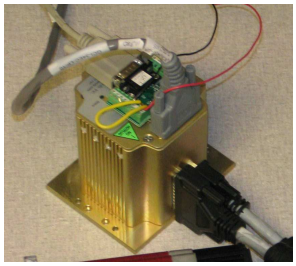

Figure 15. Silver Nugget Motor Controller.

4.6. Motor Controller

The Silver Nugget N3 M-Grade controller/driver has a 10-bit ADC for single input from a signal range of 0 to +5 VDC. The controllers have a RS-232 serial interface and baud rate of 57.6k with a variety of preset commands for controlling and interfacing. The controller communicates with the encoder by pulse width modulation at a TTL logic level of +5VDC. The encoder provides 16000 counts/rev with maximum rotational speed of 4000RPM.

5. System Software

5.1. System Communication

The software used to program LinBot was entirely written in LabView 8.2. This GUI application was very user friendly when interfacing all our sensors together. By using the embedded functions that LabView provided, sending and receiving commands from our outside sources became quite simple.

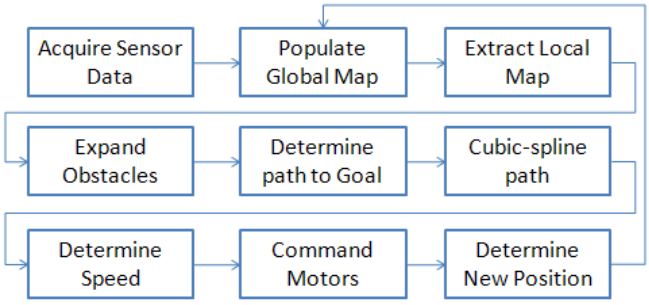

Figure 16. Software system.

5.2. Vision

The goal of vision is to provide cognition with locations of white lines and a general heading for the robot to move towards. This is accomplished by a continuous sequence: Acquiring vision data from a camera, filtering to accent white lines, analyzing the image to remove noise and extract white lines, and examining line data to determine goal heading.

The vision code has been written for portability so it can be moved to any platform and used as a viable vision solution. This program, called the "Vision Shell", allows the same piece of code to be tested through several mediums (videos, still images, or live camera input) with minimal additional programming.

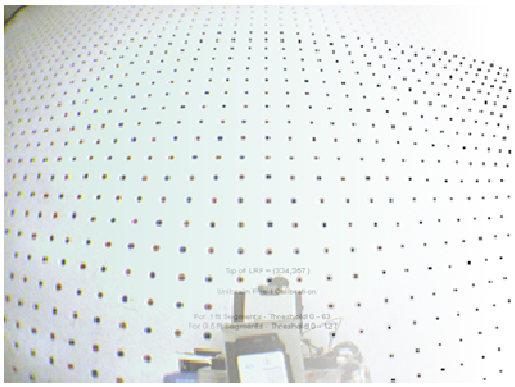

Figure 17. Camera calibration. Image fades from camera photo to digitally edited image for calibration.

Because cognition needs a birds-eye view of the obstacle course, the camera image must be corrected for both perspective and lens distortion. A grid of evenly spaced markers is laid in front of the camera, as seen in Figure 17, to shows the extent of the perspective and lens distortion. Corrective parameters from this image are extracted and saved as a specific camera profile to be applied back in the vision process.

Vision data is filtered by way of channel mixing, where the green and blue channels are extracted from the camera input and the inverse of the green channel subtracted from the blue channel. The resulting grayscale image is then passed on to particle analysis, which groups adjacent white pixels as particles and generates several properties of each particle such as center of mass, bounding rectangle, and area. Given that lines are generally long and not compact in form, the set of particles retrieved from the analysis is further filtered by extracting only particles

whose properties match the characteristics of a line. Remaining particles are further filtered by classifying all particles into sets describing lines with the criteria that particles in a set should be within maximum distances and orientations thresholds. The longest sets are considered to be the real lines. The output particles are corrected for

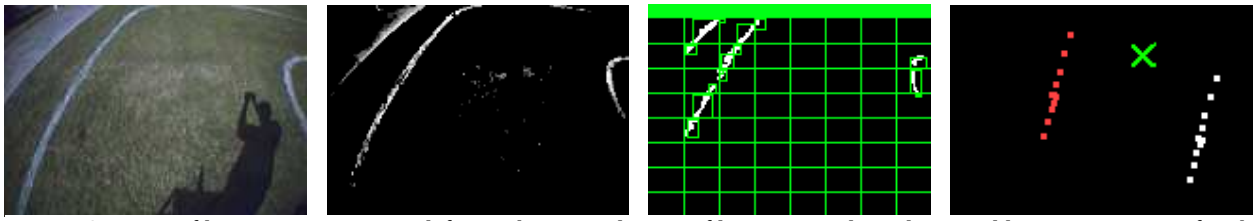

Figure 18. Vision filtering process. From left to right, original image, filtering, particle analysis, and line continuity. In fourth image, line continuity rejects the short lines replicates the strong left line with an offset. Green X is calculated goal.

perspective and lens distortion using information obtained from a calibration image of the camera and the results are passed on to vision mapping. The process is shown in Figure 18.

Once filtering is complete, the result is passed to the vision map, which takes additional inputs from the compass, GPS, and motor dead-reckoning to orient and position local vision data onto a global probabilistic map. Instances of both the obstacles and the local area seen by vision are kept and used to calculate the probability an observed obstacle both exists and is not noise. Each pair of cells in the obstacle and area map can be unique and thus a dynamic probability threshold is achieved even with a more or less static probability filtering threshold. This final output is passed to cognition as a vision map along with the computed goal from line continuity.

The advantage of the particle analysis over the previous year's Hough-Line transform analysis is accuracy and speed. The Hough-Line is a best-fit formula and can easily fail and wildly oscillate from small amounts of noise. Due to the nature of the algorithm, a single white pixel implied a large set of possible lines that had to be calculated. This caused a significant slow-down that the particle analysis easily trumps. The time saved from particle analysis allowed this year's vision algorithm to incorporate more complex filtering to achieve greater and more accurate vision results.

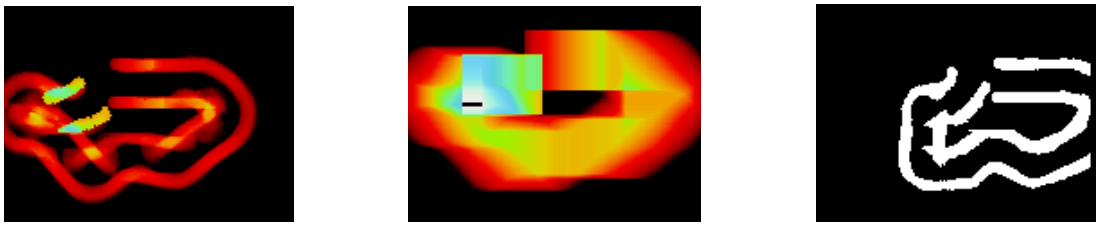

Figure 19. Probability mapping test run output. Left window showing obstacles seen; middle, the area seen; and right, the output.

5.3. Obstacle Avoidance, Mapping, and Path Planning

LinBot uses a laser range finder to build a Cartesian style occupancy grid map around itself. The laser range finder outputs a 181 array of distance points at 20Hz. The points are converted into XY points and used to populate the grid map. There are two different style maps that LinBot uses. A local map and a global occupancy map, Figure 20.

The Global map is fixed at a coarse scale to fit the entire map in an N x M array. This array is large enough to fit the entire map but coarse enough to not waste system memory. Once the immediate area around the robot is mapped and saved in the global map, a 16 x 16 m² area is extracted and used in a local reference. This map is what is used to perform our immediate moves. The obstacle and white line points are expanded by an offset equal to the radius of the robot. This is done because the robot sees itself on the map as a dimensionless point. This new array is then fed back to the path-planning program. Filled cells are obstacles and unfilled cells are clear. This map is used to determine incremental steps towards the desired occupancy grid goals. With the use of parallel processing, we can designate one process to solely building the global map and another process to extracting the local map and using a path planning algorithm called Grass Fire. Occupancy grid goal cells are usually located on the edge of the map, pointing to the next waypoint relative to LinBot or the goal point found by the vision system during the autonomous challenge, with LinBot being at the center. The grass fire algorithm fills in empty cells with numbers representing the distance to the goal until all of the cells around LinBot are filled. Then a path finding algorithm outputs a cell to cell path based on the information provided in the grass fire algorithm as shown in Figure 21.

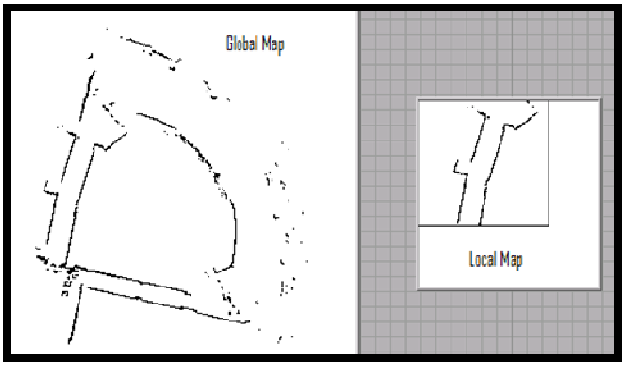

Figure 20. Two maps used for localization.

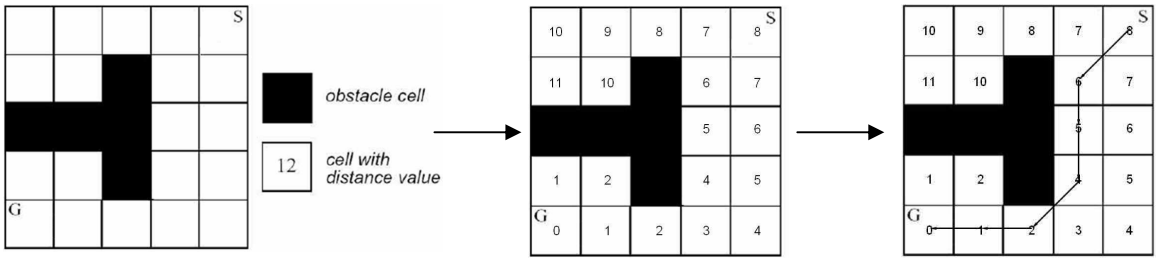

Figure 21. Grass Fire Process.

The output cell array is then converted to discrete x and y coordinates. These discrete coordinates represent the incremental steps towards the desired occupancy grid goals. Since there are parallel processers computing the path and extracting data, LinBot will continuously create new paths. With the new found path to the goal, a filter needs to be implemented to smooth the trajectory. To do so, a cubic-spline algorithm, Figure 22, is implemented.

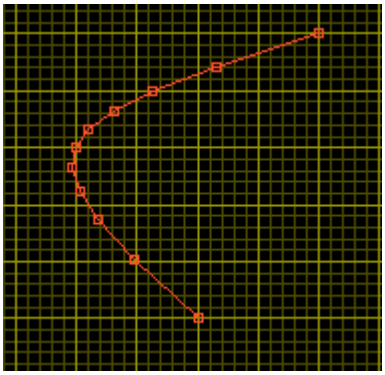
Figure 22. Cubic spline of a Grass Fire path.

5.4. Autonomous

The Autonomous Challenge and Navigation Challenge both use the same core cognition algorithm with a few modifications. To accomplish this, LinBot must first gather all information from each sensor. This is done with parallel processing which ensures that

the data is received from each sensor at their maximum rate. The key sensors used in the Autonomous Challenge are the LRF and the digital camera. Both of these sets of data are fused into one local map and placed in a larger, coarser global map. This map contains the entire competition course. Our local referenced goal is generated by the vision algorithm's analysis of the line data.

LinBot computes its immediate position using odometry but will deviate substantially over long distances due to losses of contact of the wheel and the ground. To correct this, we Kalman filter the odometry position with the GPS / IMU position. This corrects the inaccuracy of the long distance while maintaining the accuracy at short ones. LinBot also computes its' heading using the same odometry but once again, over long distances, it distorts. The same filter is applied with the digital compass and IMU for this correction.

5.5. Navigation

Motion Control for the Navigation challenge was coded similar to the Autonomous Challenge in efforts to make the algorithm compatible with both challenges. Instead of using the goal from vision, GPS will run in a parallel process to compute the global goal in reference with the lat / long position represented in Cartesian coordinates. The Cartesian coordinates are converted from latitude and longitude which are fed into motion control to plan its' path to the next navigation point while avoiding obstacles. Due to an average standard deviation of 7 meters using just the GPS solution, minimization of error is required to perform well during the navigation challenge. Therefore, OmniSTAR HP is used to decrease the solution error down to an average standard deviation of 0.7 meters using just the GPS coordinates. We can improve this even better while the IMU. This in conjunction with the GPS gives us an average standard deviation of 0.3 meters.

Azimuth is supplied from the GPS receiver and compared to the digital compass data to give a more precise heading. This helps reduce the error in the digital compass caused by magnetic interference. Since the global goal is outside of the local map, a new goal is generated with the same heading as the global goal until the global goal is inside of the local map. At that point, the same procedure occurs as mentioned before.

5.6. JAUS

Joint Architecture for Unmanned Systems was a new system that we incorporated on the LinBot. A great learning curve was required to understand the material. Numerous documentations provided from www.jauswg.org were read through to aid in the development of the new JAUS software along with learning about UDP packet transmissions through a network. In addition, the IGVC JAUS rules were also of great help by making some parameters constant. Although the most difficult process was learning about the correct values in the JAUS packet.

JAUS was developed using LabVIEW, being built upon basic UDP transmission. An OCU in addition to the LinBot subsystem node was developed to send and receive JAUS messages across the same network. This would verify, the messages being sent and received. The parameters are set by the user in the OCU which is a GUI application to reduce time creating different JAUS messages. The OCU takes the user defined parameters and creates a JAUS UDP packet in ASCII hexadecimal representation, the packets are then sent by the OCU across the same network to the destination IP address specified for the subsystem LinBot on port 3794. The subsystem LinBot receives the JAUS message and parses it. Various parameters in the header section are compared to the values specified on the IGVC JAUS rules for validity. If the message complies with the parameters set forth by the IGVC

JAUS rules and the JAUS architecture documentations on www.jauswg.org. The message data portion will then be compared to the ASCII equivalent of "JAUS01.0" to verify the message data. If all requirements are met as of yet, it will check the command code to determine what type of command the OCU desires. The subsystem LinBot node then composes a JAUS packet and sends it back to the OCU specifying if the JAUS packet it received from the OCU was a valid JAUS message or not. Level 1 JAUS compliance is completed because the current software does not take action from the command code. A representation of our JAUS architecture is represented in Figure 23. Level 2 and Level 3 will be implemented in the following year to come.

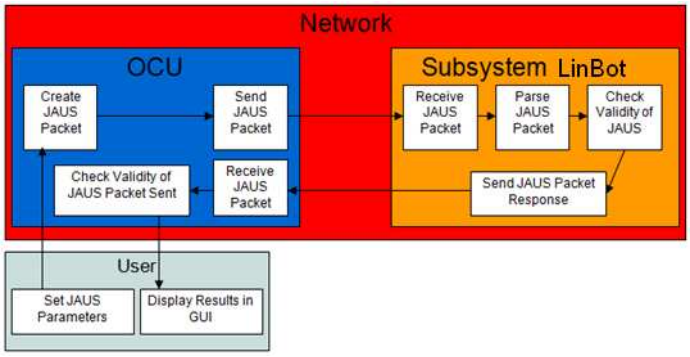

Figure 23. Our implementation of the JAUS protocol.

Many challenges were faced during the development of the JAUS software. One such challenge involved the nature of UDP packets not requiring a handshake unlike TCP packets. As a result UDP packets are transmitted much faster but at a cost of being dropped. This became an issue in our implementation of our JAUS software. Developing the OCU to send a UDP packet across the network, it would wait for a reply from the LinBot subsystem node. When a UDP packet was dropped, the OCU would freeze in to an infinite loop waiting for the message to be received. To account for this problem, a new check system was implemented in the development of our JAUS architecture by waiting 25ms for a response back from the LinBot subsystem node. If the message was not received by this time, it would stop waiting and send another request. This process would be continued for a total of three continuous requests. If the OCU did not receive a message from the subsystem LinBot after three continuous requests, then the OCU would stop. If a message is received in between the additional requests, then the process is reset to zero. Through many trials, this method has proven successful with a maximum drop of two continuous packets. Another problem arose from choosing a type of JAUS architecture that would work efficiently. Many trial architectures were proposed and implemented but our current architecture proved to be the best since it has not failed yet. The process was a challenge because of the creation of our own JAUS architecture but as a result it gave us a much better understanding of JAUS.

6. Predicted Vehicle Performance

6.1. Speed

The motors are capable of outputting a combined peak power of 0.76HP at 24 volts and run at an optimal speed of about 1200 RPM. This translates to an optimal vehicle speed of 3.4 mph, which gives the greatest motor efficiency. The robot has the ability to travel faster, but it is programmed to travel at a maximum speed of 5 mph.

Reaction Time	
Process	Time (ms)
Sensor Input	110
Grass Fire	440
Map Building	140

6.2. Ramp Clearing

LinBot has the ability to climb an incline of up to 30 degrees. The motors are capable of driving the vehicle up a 30 degree incline at 1.3 mph.

6.3. Reaction time

With parallel processing and shared variables, NorMAN now computes each process separately without the need to wait for any other process to finish. This allows it to process each sensor input and execute manipulation VIs at its maximum potential.

6.4. Battery Life

The robot requires and average of 28 amps of continuous power. Two SVR batteries are used providing 28 amp hours allowing the use of approximately one hour of operating time.

6.5. Obstacle Avoidance

The laser range finder, which outputs polar coordinates, is used as our main obstacle sensor. The points are observed and have various math functions applied to the data. The results of the functions are then applied to a case statement. This case statement will determine the behavior of the robot.

6.6. Accuracy of arrival at navigation waypoints

The accuracy of the robot's arrival to navigation waypoints is limited by the standard deviation of the GPS, which ranges from 7m with the free service and 0.6m with OmniSTAR.

7. Safety Configuration

When creating any autonomous vehicle it is very important to take safety into account. A careful safety analysis is essential to ensure that no one will be harmed while operating, observing, or working on the vehicle. Safety considerations have been of the utmost importance when designing LinBot. LinBot includes two forms of emergency stopping, wireless and manual. For the wireless e-stop, a Radio Shack 61-2667A, used to wirelessly interrupt power to the motors. Whenever the "ON" button is pressed on the remote control of the Radio Shack unit, AC power to the motors is interrupted. The mechanical switch, Mouser 653-A22E-MP-01, is connected to the DC power to the motors as well. Another consideration of safety for LinBot was that no one would be injured while working on the vehicle. All connections that pose an electrical shock hazard have been covered in non-conductive material. In addition to this all sharp edges were de-burred during assembly.

8. Cost

Throughout the design and development of LinBot there was an effort made to reduce cost without sacrificing sensor accuracy or vehicle performance. This was accomplished with the help of industry sponsors. Many of the component costs were eliminated or reduced due to the generous donations received by our sponsors. A full analysis of component and material costs can be seen in the following Table 2:

Item	Quantity	List Cost	Cost to Team
NI PXI-1000B Chassis	1	3,173.50	2,856.15
NI PXI-8187 Controller	1	5,640.26	5,076.23
NI PXI-8252 IEEE1394 Interface Board	1	544.5	490.05
NI PXI-8430/4 RS232 Serial Interface	2	709.5	638.55
NI PXI-8430/4 RS485 Serial Interface	2	764.5	688.05
SCB-68 I/O Connector Block and Cable	1	385	346.5
Quicksilver Motors and Controllers	2	4,040.00	1,743.94
Gear Boxes	2	1009	759
Novatel DGPS Unit	1	5,942.93	2,922.75
SICK Laser Rangefinder	1	7,675.00	4,891.00
Honeywell Digital Compass	1	540.17	540.17
UNIBrain Digital Camera	1	120.00	120.00
Mechanical Drivewheels and Chassis	N/A	782.27	782.27
AVR 12V Batteries	2	279.18	279.18
Electrical Components	N/A	253	253
LMS Driver	1	720	0
Total			22,386.84

Table 2. Cost of LinBot.

9. Conclusion

LinBot is an autonomous ground vehicle designed, built, and tested by the students of the CSUN IGV Team. We believe that with our simple yet refined mechanical design, dependable electronic layout, and sophisticated sensors and algorithms, LinBot will be very successful in this year's Intelligent Ground Vehicle Competition. Our use of positional Kalman filtering, probabilistic obstacle mapping, vision line continuity algorithm, and shared variable engine will set LinBot apart and establish new standards in the field of autonomous robotics.