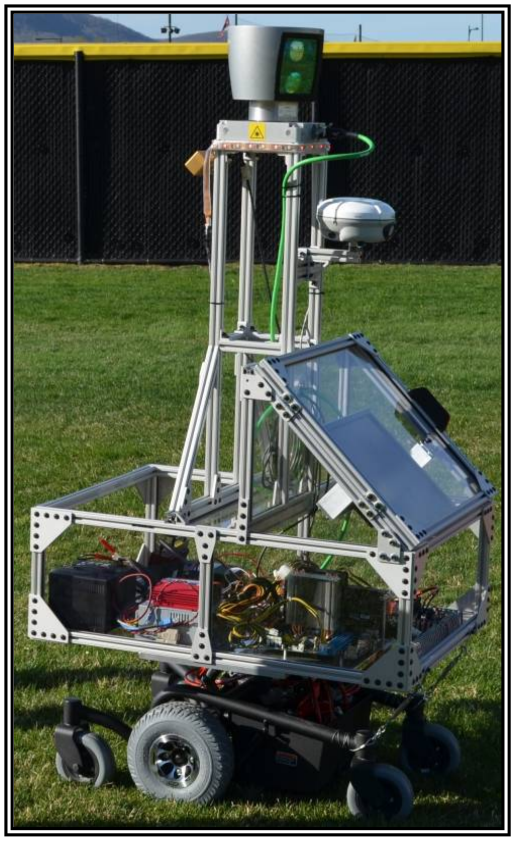
BLACK KNIGHT RISING DESIGN REPORT

UNITED STATES MILITARY ACADEMY
WEST POINT, NY 10996



The 21st Intelligent Ground Vehicle Competition

Oakland University, Rochester MI

Cadets Stuart Baker, Kyle Yoder, Nicholas Fettinger, Wei-hung Chen, David Choe, Brett Reichert

Faculty Advisor: COL Robert W. Sadowski, Ph.D.

As a first year project, I certify that the engineering design present in this vehicle is significant and satisfied the requirements of our two semester senior design project course in an accredited, undergraduate EE and CS curriculum.

Robert W Sadowski

Robert W. Sadowski
Colonel, US Army

1. INTRODUCTION

For the first time in nearly a decade, the United States Military Academy is proud to enter the annual Intelligent Ground Vehicle Competition (IGVC) with *“Black Knight Rising.”* This past year, an interdisciplinary team of six Electrical Engineering and Computer Science cadets designed and built an IGVC competitor from the ground up. The physical chassis is constructed of an 80/20 aluminum frame mounted atop a two-wheel differential drive wheelchair platform. An array of sensors including a stereoscopic color camera and a 3D LIght Distance And Ranging (LIDAR) are integrated through a hand-built computer platform. A student developed ground segmentation algorithm was offloaded onto the graphics processing unit for parallel point cloud processing.[1] The Robot Operating System (ROS) integrates the hardware while providing a modular, yet efficient software backbone. Without any previous IGVC platform or experience, cadets designed, built, and tested the system in two semesters and look forward to the competition.

1.1. Design Process

To begin our project, we chose to use the six phase design process based on the Salt and Rothery model[2] depicted in Figure 1. For this project the requirements and constraints are all clearly outlined in the rules given for the competition. The customers for our project are the judges at the competition as well as our project advisor.

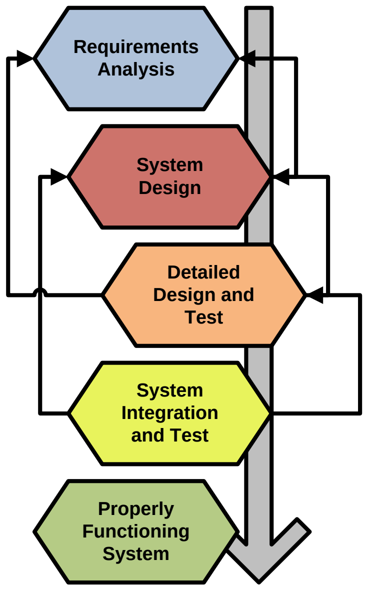


Fig. 1. Design Process

Starting without a previous chassis presented both advantages and disadvantages. One of our chief concerns was the amount of iteration required in the platform design process partially due to our inexperience as EE&CS majors, but also from not having an existing platform to compare against. Having a functional platform also provides a significant advantage for initial algorithm development. We mitigated this risk by using a four wheeled Mobile Robotics P3-AT as a proxy platform. However, not being constrained by a preexisting platform enabled us to focus on meeting minimum competition requirements and not spend considerable resources both understanding previous designs and alleviating their challenges. We decided from the outset to use an approach that maximized flexibility for future expansion while leveraging commercial off the shelf components that greatly eased assembly and testing.

[1] S. Baker and R. W. Sadowski, “GPU Assisted Processing of Point cloud Data Sets for Ground Segmentation in Autonomous Vehicles,” *Proceedings of the 2013 IEEE International Conference on Technologies for Practical Robot Applications (TePRA),* April 2013.
[2] E.J. Salt and R. Rothery, “*Design for Electrical and Computer Engineers*,” Wiley and Sons, Inc., NY, p. 16, 2002

1.2. Design Overview

Given limited mechanical engineering support, we modified an existing powered wheelchair base with an attached platform constructed from sections of 80/20 aluminum mounted on an aluminum plate. We centered our navigation design concept on continuous map building. To perceive the environment, we outfitted the chassis with both a LIDAR for depth mapping and a stereoscopic camera for white line and flag detection. To assist LIDAR point cloud registration and provide dead-reckoning capabilities, we employ an inertial measurement unit that provides linear and rotational acceleration information. We use a survey grade GPS receiver for absolute position in conjunction with an onboard compass module for heading. We fuse the output from the onboard sensors using Kalman filtering where appropriate and combine information via Simultaneous Localization And Mapping (SLAM) algorithms. Our SLAM based system creates and updates a continuous map of the environment as we navigate. The navigation subroutines use this map in combination with both current location and a navigation goal to maneuver to the desired waypoint using the fixed obstacles as points of reference. Despite being an international design competition, our advisor restricted us to US sourced components wherever possible in accordance with federal purchasing guidelines.

1.3. Team Organization

Team Black Knight Rising is composed entirely of undergraduate students with three computer science majors and three electrical engineering majors as shown in Figure 2. The team lead is a junior electrical engineering major that is pursuing a mechanical engineering double major. The small size of our team enabled a compact leadership structure and excellent communication across the team. We spent roughly 1,500 hours evenly distributed among the six members over the course of our two semester design project capstone course designing, building, and testing the platform.

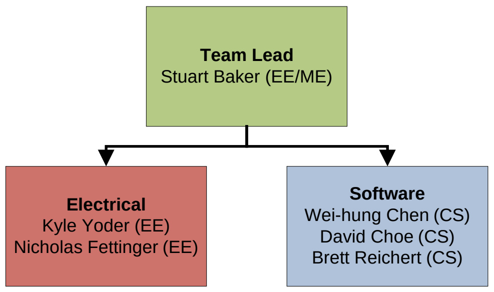


Fig. 2. Team Structure

2. MECHANICAL SYSTEM

The core design principles we emphasized in the mechanical system design were ease of manufacture and flexibility. Ease of manufacture was important because we were limited on manpower available and mechanical engineering expertise. Also the less time we spent assembling the platform left more time to work other issues. Flexibility was stressed as we wanted to make future modifications as easy as possible. We started with a motorized wheelchair base. We utilized 80/20 extruded aluminum framing because assembly is analogous to using an erector set and thus within our skill set as EE&CS majors: using 80/20 does not require welding or machining skills beyond cutting to size. Figure 3 below shows an initial sketch of the attached platform along with a picture of the final design.

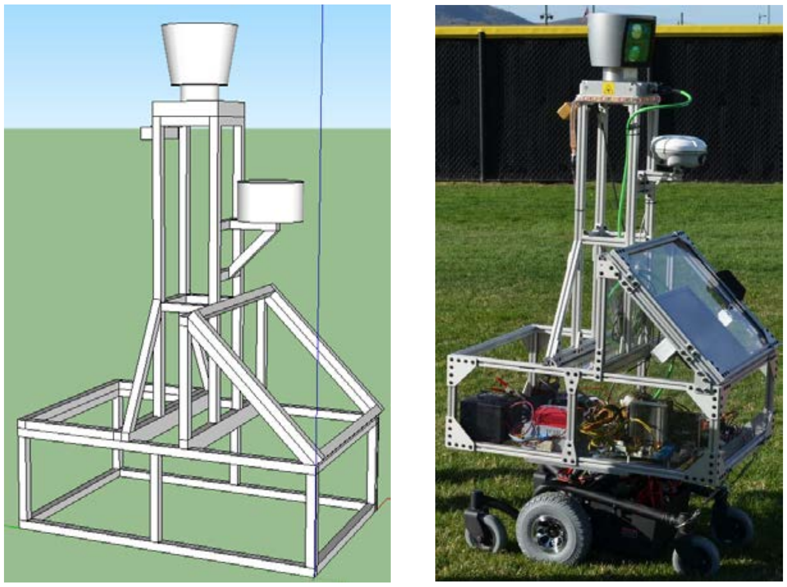

(a) (b)

Fig. 3. (a) Initial chassis frame sketch (b) and final frame design on commerical wheelchair base

2.1. Base

We used a preexisting wheelchair base to streamline the development process reducing design and fabrication time for the motor housings and drive train. During the selection phase, we compared several different wheelchair platforms along with existing robotic platforms including a PackBot and Talon. We opted for the Invacare Pronto M41 wheelchair platform because it is designed to carry several hundred pounds, operate for long periods of time continuously, traverse grassy terrain, and has a very small turning radius due to its center-mounted differential drive wheels.

2.2. Chassis

Purchasing a preexisting base platform helped reduce some engineering overhead, but we still required a custom chassis housing the sensors and processing equipment. For chassis construction we compared several different design styles including welded aluminum cage; plate and bolt construction; and finally extruded beam systems. Building on our primary design criteria of ease of manufacturing and flexibility, we used 80/20 construction. 80/20 is a square aluminum extrusion with T-slots on each of the four faces. Assemblies are created by cutting the raw extrusion to length, sliding T-nuts into the slots, and then bolting the pieces together using plates. This system allows for fast construction and is easily modified for future needs. We drew initial design sketches in Trimble SketchUp and transferred chassis blueprints over to SolidWorks for fine tuning.

As shown in Fig. 3, the chassis is composed of three distinct sections. The bottom portion of the chassis is a large rectangular compartment bolted to an aluminum plate that contains the majority of the power and processing components. The plate is attached via the chair bracket and wire cable stays. Mounted atop a large tower in the center of the chassis are the LIDAR, stereoscopic camera, and GPS receiver. On the rear of the chassis is a large angled compartment that houses the onboard monitor and has all of the other interface controls including the emergency stop.

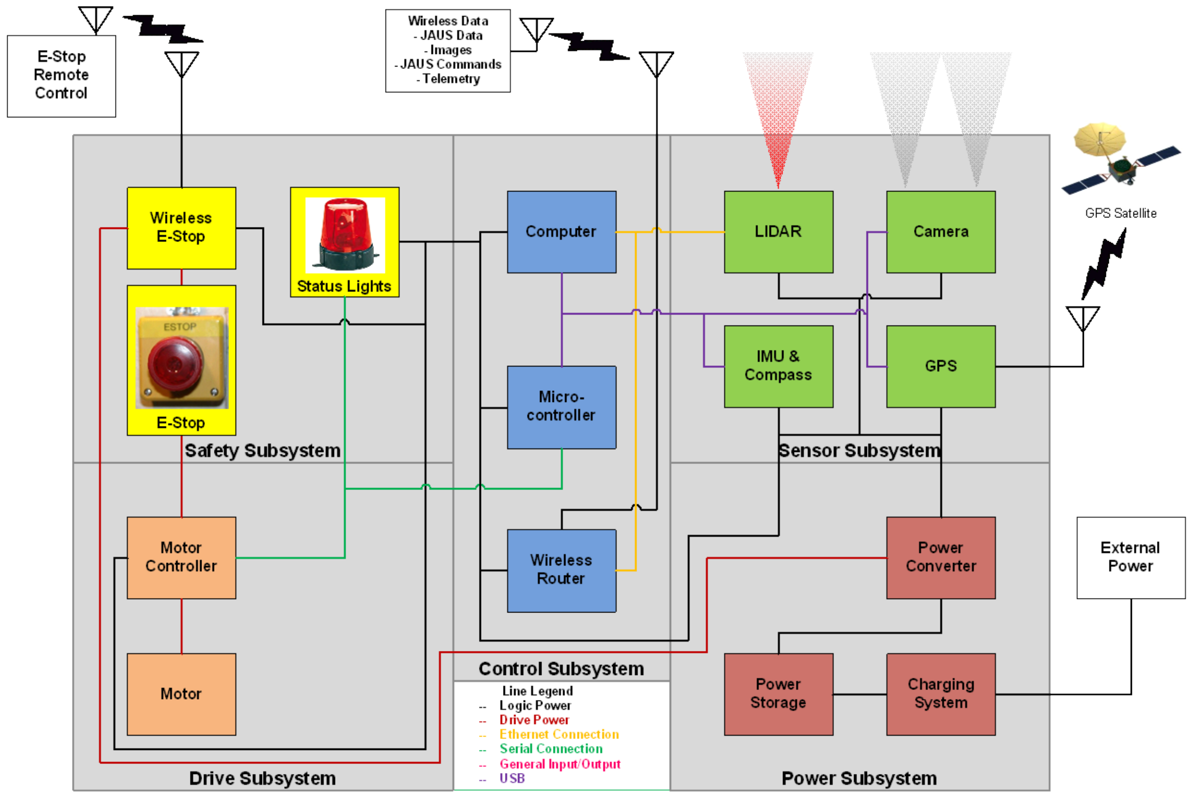


Fig. 4. Hardware Block Diagram

3. ELECTRICAL/POWER SYSTEM

As with the mechanical system, we stressed flexibility in design of the electrical system to enable potential future expansion. In addition, we emphasized circuit protection and modularity on the main power bus. Given the large number of delicate systems running from the main power bus, we deemed protecting those systems from both over voltage and over current conditions as critical to our success. This led us to use multiple DC-DC converters rather than a single large capacity DC-DC power regulator. We deliberately used singular DC-DC converters to isolate key components at the expense of cost and power efficiency. Protecting our $75k LIDAR investment with a dedicated and fused converter is one example of where we sacrificed efficiency to protect key components.

3.1. Hardware Block Diagram

Figure 4 on the previous page depicts the six major subsystems that comprise our design. The control subsystem (blue) provides the processing and operation. The sensor subsystem (green) comprises elements used to perceive the environment. The power subsystem (red) consists of the batteries, battery charger, and power converters. The drive subsystem (orange) propels the chassis. The safety subsystem (yellow) allows the user to halt operation and provides visual feedback. Lastly, the mechanical subsystem (gray box) comprises the chassis and powered wheelchair base that houses all the components.

3.2. Power Bus

In order to simplify wiring and allow for future expansion, the chassis is outfitted with a DIN rail depicted in Figure 5. Based upon standards originally developed by Deutsches Institut für Normung, DIN rail is a standardized section of metal channeling that is typically used to mount relays, circuit breakers, and other industrial control devices. For the purposes of our chassis, this rail acts as a central routing location for all power and control lines thereby allowing easy integration of subsystem control and protection using relays and fuses. We employed standard DIN rail compatible fused blocks that accept either automotive blade or cylindrical fuses. We could tailor maximum current draw protection by simply using the correctly rated fuse. The DIN rail is easily configured and enabled us to rapidly modify our power system several times since its initial installation. The flexibility was critical as our platform continuously grew with component integration. Subsequent reorganization occurred without substantial redesign as elements shifted location within the chassis.

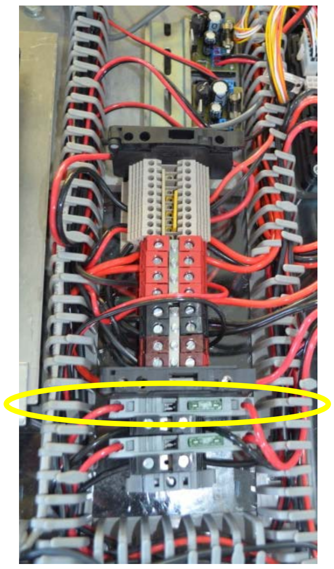


Fig. 5. DIN Rail with standard automotive blade fuse in circled element

3.3. Power Control Board

The power control board is a custom designed printed circuit board (PCB) that houses the lower level power control systems. This PCB houses several DC-DC voltage converters for some of the onboard systems as well as a series of relays and fuses that are not mounted on the DIN rail for circuit protection. Two other critical subcomponents are also located on this PCB: the wireless e-stop switch and the microcontroller that interfaces with the motor controllers.

TABLE I: POWER SYSTEM ANALYSIS

Regulated Power	Voltage (V)	Current (A)	Power (W)
LIDAR	12.00	3.00	36.00
Microcontroller	12.00	0.75	9.00
Wireless Router	19.00	2.00	38.00
Safety Light	12.00	0.75	9.00
Wireless E-Stop	12.00	0.50	6.00
Camera	Powered from computer		
IMU	Powered from computer		
Compass	Powered from computer		
		Regulated Power Consumption (W)	98.00
		w/90% Rated Converter Efficiency (W)	108.9
Unregulated Power	Voltage (V)	Current (A)	Power (W)
GPS	24.00	1.00	24.00
Computer	24.00	7.00	168.00
Motor Controllers	24.00	20.00	480.00
		Unregulated Power Consumption (W)	672.00
		Total Power Consumption (W)	780.9
		Nominal current draw from 24V batteries (A)	32.54

3.4. Drive System

We replaced the original Shark Controller with two Curtis 1212-24 single-channel motor controllers that drive two permanent magnet, brushed DC motors on the wheelchair base. The 1212 motor controllers are designed for powered wheelchair applications and provide sophisticated motor control in terms of acceleration/deceleration, top speed, and current limiting while remaining easy to configure via a serial programmer. Headquartered nearby, Curtis allowed us to purchase programming tools for their entire controller line normally reserved for OEM manufacturers. This gave us the capability to use our own controller settings that met IGVC safety constraints such as top speed. After initial calibration and testing to validate our settings, we control motor speed by a 0 to 5 V analog voltage input. For wheelchair applications, this is normally accomplished via a potentiometer in the joystick. We generate the analog control signal from the microcontroller located on the power bus PCB. The microcontroller takes digital command values from the host computer and produces an analog control signal for both motor controllers.

3.5. Battery System

The wheelchair chassis comes equipped with a pair of 12V 55A·hr series connected, sealed lead-acid batteries. Power system analysis, outlined in Table I, indicates that the batteries should last a minimum of one hour. Through testing we determined that the battery pack is capable of driving the chassis for at least two and a half hours of continuous use. The batteries are recharged using a regulated battery charger that has been integrated into the power bus.

4. SAFETY SYSTEMS

4.1. E-Stop

The chassis is outfitted with a manual e-stop switch located on the top of the rear control panel. The manual e-stop is wired in series with the wireless e-stop and controls power to the motor controller logic enable. If the e-stop is depressed the motor controllers are disabled, preventing the chassis from moving.

4.2. Wireless E-Stop

The wireless e-stop is an active on system that consists of a standard R/C remote control paired with a servo-signal controlled relay. Wired in series with the manual e-stop, the relay controls power to the motor controller logic enable. When the trigger on the R/C controller is pulled, the relay activates providing power to the motor controller logic enable. If the R/C controller trigger is released, the relay opens cutting power to the motor controller logic enable and stopping the chassis.

4.3. Warning Lights

To provide an indication of the robot's state, the center mast is equipped with a warning light system. As shown in Figure 6, the light system consists of a strip of LEDs attached below the LIDAR. They are solidly lit whenever the robot is powered, but flash when the robot is in autonomous mode. In addition to delineating between powered and autonomous modes, the warning light system can also display more complex information. Each LED on the light strip is actually an individually addressable Red, Green, and Blue (RGB) LED. This enables us to actually use color changes and provide additional system information depending on state information, waypoints reached, or errors that occur.

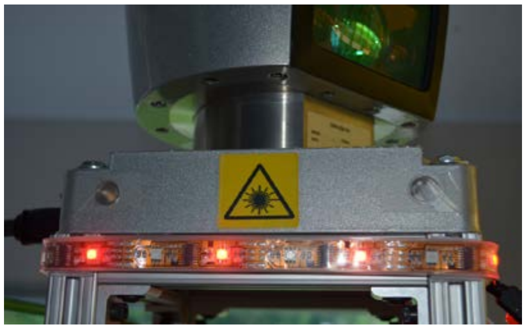


Fig. 6. Warning Light

5. SENSOR SUITE

5.1. LIDAR

To perceive the environment, the chassis is equipped with a Velodyne HDL-64 S2 LIDAR. The LIDAR is built of a large rotating laser head on top of a mount. The laser head contains 64 lasers that fire in series as the head rotates. As the system spins, the hardware uses time of flight data to calculate object distances in the environment to within one centimeter. The Velodyne can generate a 360° horizontal, 26° vertical depth map of the environment at a 5 to 15 Hz rate. The drawback of the LIDAR point cloud data is the 1.3 Mega points per second that must be processed to create an obstacle map and also suffers from a roughly 3-5 foot radial dead zone. From the depth information, we segment out the obstacles and use them to generate a collision map that helps the robot to navigate. The LIDAR sends depth map scans via multicast Ethernet through the chassis's onboard network. The LIDAR is powered by a regulated 12V bus.

5.2. Camera

The LIDAR system generates depth information only. We must augment it with a color vision system to detect lane markers and flags. We use a PointGrey BumbleBee XB2 stereoscopic camera to detect the white lines and colored flags on the course. The stereoscopic camera consists of two high definition, RGB cameras spaced 10 centimeters apart. The two cameras provide color sensing and create a depth value for each pixel using image disparity in the same way as our eyes function. By segmenting out everything but the white lines in the image, we are able to generate a 3D representation of the lane markers by fusing them with our obstacle map to avoid crossing them. The image information is sent from the camera to the host computer using FireWire (IEEE 1394). The camera is powered via the FireWire connection.

5.3. GPS

In order to provide absolute position information the chassis is equipped with a Trimble R8 GPS receiver. This survey grade GPS receiver has sub-meter accuracy and is used to determine if the vehicle has reached the desired waypoint goals. The GPS communicates with the host computer by way of a serial connection through an RS232/USB converter. Power is provided to the GPS via the tethering cable, which can auto-switch between the raw battery voltage and the built-in battery pack. The built-in battery pack is useful to maintain or generate an initial fix while the other subsystems are powered down while moving the vehicle into position. This can save considerable time by avoiding the usual 30 second or longer time to calculate a positional fix from a cold start.

5.4. Inertial Measurement Unit (IMU)

One issue observed during testing was that the LIDAR refresh rate was insufficient and hampered SLAM map building. Using a Microstrain 3DM-GX3-15 IMU for linear and rotational acceleration values gave us a best-guess estimate for initial LIDAR point cloud orientation and positioning during the SLAM registration phase. The inertial information is Kalman filtered for better localization of the robot chassis within the generated map. The IMU is connected to the host computer via USB, which provides communication and power.

5.5. Compass

The final chassis sensor is a KVH C100 fluxgate compass that has been successfully employed on military vehicular platforms. It is partially gimbaled to compensate for slope, is vibration resistant, and provides absolute heading information. The compass also communicates with the computer via USB that provides communication and power.

6. COMPUTING

Much like the mechanical and electrical systems, the primary emphasis with the software system was compartmentalization and modularity. To achieve our goal of a flexible, durable, and easy to use software architecture we chose to use the Robot Operating System (ROS) in conjunction with a hand-built computer. ROS will enable easier transition of software modules to follow-on teams than a custom developed control system.

6.1. Computing Hardware

The nerve center of the platform is the onboard computer. During the initial design phase we compared a series of different laptop and desktop computers. Unfortunately, we were not able to find a platform that provided both the features and performance required within our budget. We built our own desktop computer that provided us with significantly more performance than any prepackaged system.

The computer consists of a Bulldozer core AMD FX-8350 running on an ASUS M5A99FX motherboard that is equipped with 16GB of DDR3 RAM. The AMD FX8350 was selected because within its processor generation it shows the greatest multi-thread performance. The motherboard is also equipped with an EVGA GTX-660 TI graphics card that is used for parallel processing of the LIDAR point cloud data sets and the stereoscopic camera images. The 660 TI graphics card provided a reasonable balance between CUDA core numbers than enhance performance and price.

For remote diagnostic and control, the computer is also tied to a wireless router. From a remote laptop we can send a series of GPS goal positions to fill the waypoint queue. The remote interface also allows the operator to start, pause, resume, and stop navigation as well as remove waypoints from the queue. In addition, the host computer sends back telemetry and debugging information while the vehicle is navigating aiding in performance analysis.

6.2. Robot Operating System

In order to implement the software on the computer, we chose to use the Robot Operating System (ROS). ROS is an open-source, pseudo-operating system that provides a framework for the development of robotic systems. As with any operating system, ROS provides hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. ROS is built of a network of peer-to-peer processes coupled through a central master. The ROS master acts as a name-service that connects processes with each other, but once connected all data is transferred directly between processes. Each process, or node, can be a subscriber and/or publisher to any number of topics over which data is passed in the form of messages.

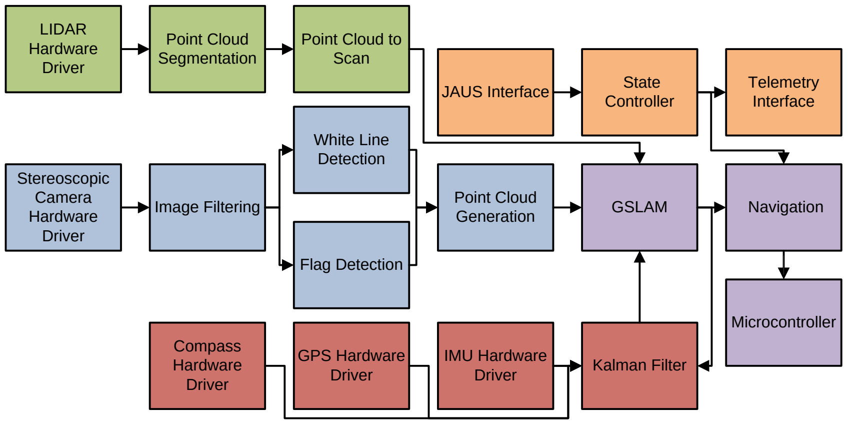


Fig. 7. ROS Node Diagram

All of the software on our platform is implemented in the form of ROS nodes. The nodes can be modeled as black boxes because the information flowing into and out of the nodes is of a fixed data type. By using ROS nodes, all of the software is compartmentalized making is very easy to interchange nodes for experimenting with different algorithms and integrating new code. Additionally, all of the messages between the nodes are readily visible making debugging extremely easy. Figure 7 shows the overall structure of the software system. The ROS node diagram illustrates the general design of the software system. The ROS nodes can be broken up into five groups: navigation (purple), LIDAR processing (green), camera processing (blue), sensors (red); and command and control (orange).

6.3. Graphical Processor Unit (GPU) Based Point Cloud Processing

The point cloud pipeline takes in raw LIDAR data and produces an obstacle map of the environment. Point clouds are groups of points that represent object laser reflections in a 360° sweep. After assembling the raw point cloud, the system uses the onboard GPUs on the graphics cards to segment the ground from obstacles. The obstacle cloud is flattened to 2D and supplied to the SLAM system that builds a map of the environment. Although we can implement the ground segmentation within ROS, the 1.3 Mega pixels per sec created by the LIDAR would drastically slow down the robots reaction time. We developed a novel GPU-based, ground segmentation scheme to offload the parallel processing of the point cloud to the onboard GPU processors. Figure 8 illustrates the segmentation process while Figure 9 depicts an example indoor map that was built of our lab and surrounding hallways.

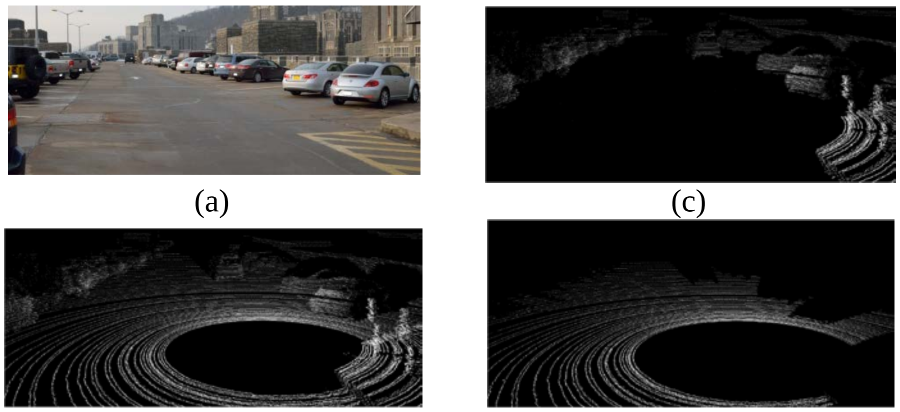


Fig. 8. Segmentation in a single test case: (a) Initial visual view, (b) LIDAR point cloud, (c) segmented obstacles, and (d) segmented ground plane

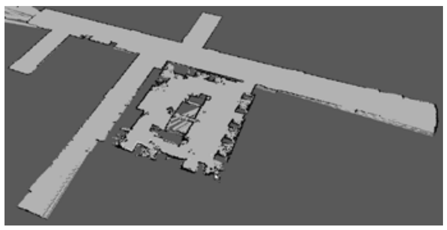


Fig. 9. Example flattened indoor map

The ground segmentation algorithm running on the GPU is the heart of a LIDAR pipeline that can be broken down into four stages. First, the hardware driver collects packets that the Velodyne broadcasts, applies calibration, converts from polar to Cartesian coordinates, and then assembles for one complete rotation of the sensor. Second, the algorithm creates a gradient field based on the height difference from each point to its neighbors. Third, the points are classified as ground or not ground. A point is considered ground if its gradient is below a threshold, it has a neighbor that is ground, and the height difference to that ground neighbor is within a specific range. Finally, the transition points are removed from the point clouds to reduce noise. Transition points are those that have a gradient that is too large to be a ground point, but is within a small height difference of neighboring ground points.

6.4. Visual Processing

6.4.1. White Line Detection

The first step in the image pipeline is to transfer the image from the red, green, and blue (RGB) color space to the Hue, Saturation, and Value (HSV) color space. The HSV color space is useful because this representation simplifies colors detection in varying lighting conditions due to the way the pixel information is encoded. The algorithm modifies image hue to help the obstacles stand out from the grass and the Value is adjusted to compensate for varying lighting conditions. Once the image has been normalized, the algorithm performs shadow correction by altering the hue and value of pixels that have a value more than two standard deviations below the average. The algorithm thresholds the image pulling out the white pixels and turning all non-white pixels black. Next, an outlier filter is used to eliminate the noise and extraneous points in the image leaving only large bodies of white. The filtered image is converted to a point cloud using the camera depth information. Pixels with z-values above a certain margin from the ground plane are part of obstacles and discarded. Finally, the lane marker point cloud is fused with the LIDAR obstacle cloud adding the lines to the global and local cost maps.

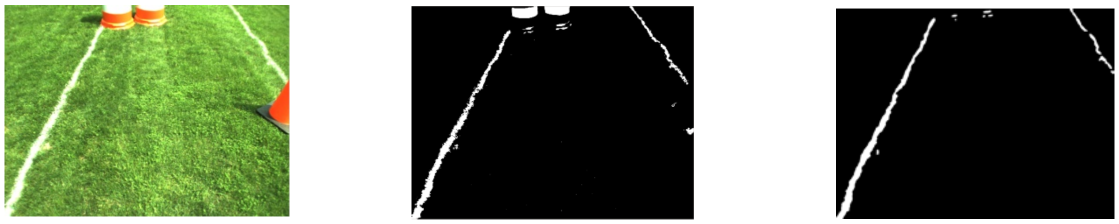


Fig. 10. (a) Initial RGB capture, (b) after thresholding, (c) and post filtered

6.4.2. Flag Detection

The flag detection algorithm is very similar to the line detection algorithm as it begins by converting the camera image into HSV color space. The hue and value of each pixel is adjusted to help the flags stand out and correct for lighting conditions. The shadows in the image are corrected as before and the image is then passed through a threshold filter to block out everything except the flags. The image is then passed through an outlier filter to remove noise and is then converted into a point cloud using the camera depth information. A *z*-filter is used to eliminate everything that is not within the area we expect to see a flag. The resulting point cloud contains the red and blue flags as seen by the camera. To navigate around the flags, the algorithm places a six inch diameter circle at the centroid of the flag projected onto the ground plane. Once the collision boxes are generated, the algorithm finds the left most red flag and creates an artificial barrier between that flag and the nearest blue flag to the right of it. This forces the path planning algorithm to take the path that has the blue flags on the left side and the red flags on the right side of the path.

6.5. Navigation

6.5.1. Goal Setting. The robot navigates the course using a queue of goals. At the start of the competition the required goals are loaded into the robot and it then either uses them in the order provided or attempts to determine the best order based on course knowledge. To begin navigation, the robot pulls the first goal off of the queue and translates it from a GPS coordinate to a location in the local map. The global and local path planning systems then take over and navigate the robot to the desired location. As the robot moves towards the goal, it continuously checks the distance to the goal and periodically re-plots the point to reduce accumulated error. Once the robot is within the specified distance of the goal, is pulls the next goal out of the queue and navigates towards that point.

6.5.2. Simultaneous Localization and Mapping (SLAM). In order to navigate through the obstacle course our platform first builds a map of its immediate surroundings. SLAM is the process of building a map of an environment without prior knowledge of the surroundings. The particular method we implemented, *gmapping*, uses a Rao-Blackwellized particle filter to reduce the number of particles in each scan match set. The reduced particle space is then registered against the existing map to generate a translation and rotation for the complete system. The exiting map and the new scan are then merged together using the offset to expand the map. The map generated by *gmapping* is used to produce a large, low-resolution global cost map while the small, high-resolution local cost map is generated directly from sensor information.

6.5.3. Kalman Filter. During testing, robot speed became a challenge while employing SLAM. While producing an incredibly detailed 3D map of the environment, the LIDAR’s 10 Hz update rate is relatively slow compared to the movement of the robot. Fast movement or rapid platform rotation can cause the SLAM algorithm to fail since it could not register the successive scans. To solve this, we employed a Kalman filter to merge GPS position information, IMU inertial measurements, LIDAR, and visual odometry information to generate a fused representation of the robot’s position within the environment. This not only provides more accurate positioning for navigation, but also compensates for the slow LIDAR update speed. The SLAM algorithm takes in the pose estimate generated by the Kalman filter output to help register successive LIDAR scans enabling greater platform velocity.

6.5.4. Global Path Planning. The purpose of the global path planning node is to generate a long-range path which is then fed into the local path planner. The global planner assumes a round robot and attempts to find the minimum cost path from the current position to the goal on the global cost map. In order to traverse the global cost map, the global planner uses Dijkastra's graph search algorithm to find the shortest path between the current position and the desired goal.

7. TABLE II: PERFORMANCE PARAMETERS

Performance Parameter	Units	Predicted	Tested
Top Speed	mph	5	7.3
Max Incline	°	7	30
Reaction Time	s	0.05	0.05-1.00
Battery Life	hr	1.75	2.5
Obstacle Detection Dist	m	120	100
Waypoint Accuracy	cm	20	38

7.1.1. Local Path Planning. The local path planning node takes the global path along with the local cost map to generate the robot's actual movement on the course. The local path planner uses the dynamic window approach to generate a series of circular paths based on the velocity and acceleration capabilities of the platform. The paths are weighted based on adherence to the global path, proximity to the goal, goal cost, and velocity. This process is run at a 20 Hz minimum to ensure a smooth path to the goal.

8. PERFORMANCE ANALYSIS

Table II depicts predicted and measured performance parameters. The predicted speed and incline performance parameters were initially derived from the manufacturer's literature on the wheelchair chassis. Since we replaced the preprogrammed controllers, we could significantly increase top speed and incline performance over the factory specifications. However, we still set initial motor controller current limits to constrain chassis speed to 5 mph. The lower motor load, and intermittent use at peak performance, also helps to increase battery life over the maximum predicted value.

Although not entirely unexpected, obstacle detection distance, waypoint accuracy, and reaction times have all fallen below predicted values. While we have detected objects out to 150m with the LIDAR, it required flat, reflective surfaces. The sensor can consistently detect objects at roughly 100m. Although stationary accuracy was achieved with the GPS, operational waypoint accuracy was less than the manufacturer's listed value partially due vehicle movement. While the system is capable of meeting the desire 20 Hz update rate, running the graphics user interface (GUI) and the virtual network connection (VNC) server for remote control significantly increased workload for the platform and reduced the update rate below 20 Hz. However, the reaction time of the chassis is partially controlled by the user. These advanced features used in remote debugging and telemetry can be deactivated once desired performance characteristics have been adjusted.

9. SUMMARY

"Black Knight Rising" represents the United States Military Academy's first entry into the Intelligent Ground Vehicle Competition since 2005. Without any previous IGVC experience and starting from scratch, six cadets designed and built a competitor from the ground up using a modified wheelchair platform in two semesters. We integrated an array of sensors including a stereoscopic color camera and a 3D LIDAR using a hand-built computer platform running the Robot Operating System (ROS) and using a student developed ground segmentation algorithm offloaded onto the graphics processing unit. All that remains is a successful showing in the competition next month.

10. APPENDICES

10.1. Vehicle Cost

TABLE III: BUDGET

Part	Unit Cost	Required	Gross Cost	Provided	Net Cost
Materials, Fasteners, & Connectors	$1,500.00	1	$1,500.00	0	$1,500.00
Wheelchair Base	$1,300.00	1	$1,300.00	2	$0.00
Additional Batteries	$100.00	2	$200.00	0	$200.00
LIDAR	$75,000.00	1	$75,000.00	1	$0.00
Stereo Camera	$2,300.00	1	$2,300.00	0	$2,300.00
GPS	$19,000.00	1	$19,000.00	1	$0.00
IMU	$1,850.00	1	$1,850.00	0	$1,850.00
Microcontroller	$50.00	1	$50.00	0	$50.00
Computer	$1,000.00	1	$1,000.00	0	$1,000.00
Wireless Router	$160.00	1	$160.00	0	$160.00
Motor Controller	$125.00	3	$375.00	0	$375.00
E-Stop	$15.00	1	$15.00	1	$0.00
Wireless E-Stop Parts	$100.00	1	$100.00	1	$0.00
		Prototype Development Cost	$110,450.00	**Residual Expenses**	$7,535.00