# USMA INTELLIGENT GROUND VEHICLE COMPETITION "IGGY" SUBMISSION

**Alissah McGill, John Oberholzter, Keith Schneider, John Tazioli, Edward Woodruff, MAJ Dominic M. Larkin, LTC Christopher Korpela Ph. D.**

## INTRODUCTION

The United States Military Academy at West Point has formed an interdisciplinary team of 6 cadets from the Electrical Engineering and Computer Science majors to participate in the 2016 annual Intelligent Ground Vehicle Competition (IGVC). This years IGVC team will be representing the United States Military Academy at West Point with the Iggy. This model uses several pieces of sensory hardware over the modular Robot Operating System (ROS) interface in order to successfully navigate the IGVC obstacle course.

## DESIGN PROCESS

All seniors in the CS/EE Capstone Program are required to use the Agile Development Process based on the Scrum Model. The requirements and constraints for this project are identical to those provided by www.igvc.org for the 2016 competition. Our project is funded by TARDEC, providing funding for our hardware budget, and supervised by the Electrical Engineering and Computer Science (EECS) faculty as USMA.

In August of 2015, our team began with the chassis that the 2013 team designed and built, and was modified significantly by the 2014 and 2015 teams. This was the source of many advantages as well as disadvantages. One of the major challenges we faced with was trying to learn a system that two different teams designed and modified but did not always properly document the changes or create follow-on how-to manuals. Furthermore, besides the lack of how-to documentation, there was also a lack of clear understanding which components were completely functioning, partly functioning, or were not functioning at all. The majority of the team took a significant amount of time, in excess of one whole sprint, just to learn how to properly interface with the robot.

In some areas, our team felt comfortable continuing previous work but had to start from scratch in other areas. Much time was lost determining what sub-components belonged in each category. This was further complicated by the fact that only 3 of the cadets on the project had robotics experience, with only 2 with ROS experience. As we learned new hardware components and software APIs, we would verify our understanding by testing the current design. When something did not work or make sense, we would assume that the current design was flawed and would change the design to a simpler, more effective design.

### Design Overview

Given that AY2015 teams vehicle meets most of the mechanical design requirements for the vehicle (height, width, etc), our team decided to keep the vehicle design and focus on replacing some of the internal hardware components and progressing the software components. The design relies on a simple acrylic glass design that supports internal components in a box with sensors mounted on a tower to provide maximum field of view. The entire vehicle is driven by two wheels and is powered by twelve military grade batteries (model: BB-2590) located on the undercarriage of the vehicle. The auto-navigation concept relies on continuous map building through the ROS navigation stack.

To create the majority of the map, the Velodyne H64 LiDAR provides a pointcloud that is passed into software to place obstacles onto a costmap. Additionally, a stereoscopic camera detects white lines that the vehicle cannot cross and flags that the vehicle must pass to the left or right of depending on the color. This creates a 2D map of locations that the robot can and cannot navigate. This costmap is later used for path planning when the robot is given a navigation goal.

To supplement the LiDARs data and provide dead-reckoning capabilities, the IMU (inertial measurement unit) provides linear and rotational acceleration information. Finally, the navigation grade GPS receiver provides an absolute position of the vehicle and a compass to determine heading, and help determine where the waypoints are on the map.

The output of the onboard sensors and their respective software suites are combined through the usage of SLAM (Simultaneous Localization and Mapping) algorithm. Through SLAM, Iggy creates and updates a continuous map of the environment as the vehicle moves in the environment. Maps are created temporarily and not stored after the ROS parent process dies. The navigation algorithm will use the map along with the current location and known GPS waypoints to determine the most appropriate path while avoiding obstacles on the map.

### Team Organization

The Iggy team is comprised of six undergraduate students from the EECS Department at USMA. The team is led by Electrical Engieering student Edward Woodruff. Responsible for all the hardware design and implementation are two additional Electrical Engineers, John Tazioli and Keith Schneider. Responsible for the software design and development are two computer science majors, John Oberholtzer and Alissah McGill. Throughout the year, the teams have primarily worked separately in different teams, but worked together at the end of the year to combine their separate work. This workflow allowed everyone to work on their own parts of the robot without being delayed by the others work.

## MECHANICAL SYSTEM

### Base

The wheel base of the vehicle has seen a couple of iterations, both before the time of the current cadet team. In the first iteration, the vehicle operated on an electrical wheelchair base, with six wheel and the middle pair of wheels driving the vehicle. Last year the system was upgraded to a

four-wheel system, with a 2x4 rear wheel drive. Currently, we are using the same four wheels as last year.

### Chassis

The chassis was designed three years ago by cadets in the Mechanical Engineering department. The chassis itself has not changed since then, however we have improved on the level of weather-proofing by using rubber tubing to cover any electrical wires protruding from the chassis (such as the cables to the GPS, Bumblebee camera, and LiDAR). Also, after reviewing competition specifications, we lowered the tower of the robot to below the 6 foot height maximum. Also, the LiDAR has been leveled to produce a more accurate pointcloud.

## POWER SYSTEM

The robot is an entirely DC system requiring both 12V and 24V sources to power its various subsystems. Previous teams used 3.2V Lithium Phosphate batteries in series to generate the required 24V source with a DC to DC converter that dropped the voltage to 12v as needed. The batteries were rated for 100Ah. The power is routed using DIN rails mount terminal blocks, 8 AWG wire, and terminals and circuit breakers. Each circuit breaker feeds into one component of hardware so that each component can be powered on or off independently from the rest. The circuit breakers also prevent any hardware from burning out. The Lithium Phosphate batteries were replaced with five BB2590s (military grade batteries) after their charge dropped too low, draining the lithium phosphate cells permanently. The five BB2590 system was a quick fix that needed to be improved upon.// This year, the Electrical Engineer students on the team conducted a power analysis of the robot and decided to redesign the power system using 12 BB2590s. This system made the most efficient use of the limited available space on the robot, regained the same amp hour rating as the old lithium phosphate batteries, and meets all voltage and current requirements. Additionally, a circuit breaker was added between the power bus output and the robot subsystems allowing the power bus to be easily disconnected from the subsystems. This conserves battery charge by preventing any unnecessary drain when the robot is not in use.

## SENSOR SUITE

### LiDAR

Our Light Detection and Ranging (LiDAR) device is a Velodyne HDL-64E. It is capable of identifying obstacles in a 360 degree field of vision and converting these instances into point clouds that the computer processes. Our LiDARs function for Iggy is to provide the point clouds for the obstacles our robot finds while navigating the course so that our computer can place them on our robots working map and perform obstacle avoidance when necessary.

### Camera

Iggy uses a Point Grey Bumblebee2 stereoscopic camera for image capture. The camera is responsible for capturing video of the area directly in front of the robot and sending those images to the Vision3D software for visual processing. The purpose of the camera system is provide images for identifying the location of white lines, red flags, blue flags, and pot holes.

### GPS

The robot is equipped with a Novatel 702L GPS dual-frequency antenna. The antenna receives GPS satellite signals, both simulated and actual, and converts them into fixes for computational use. Iggy uses this GPS antenna in order to provide access to GPS signals for the CNS-5000 to process.

### Inertial Measurement Unit (IMU)

The Novatel CNS-5000 IMU is integrated with the GPS receiver. The IMU opens a serial connection with the computer in order to receive commands from the computer and process the necessary data. The CNS-5000 provides IMU readings (linear and angular acceleration) that are integrated with the GPS data at 10hz as well as raw IMU readings at a rate of 100hz.

## COMPUTING

### Computing Hardware

The computer is based on a custom-built computer. It uses an ASUS mother board, AMD 8-core CPU, NVIDIA 690 GTX GPU, and a firewire PCI card. This allows all of our devices to connect to the motherboard and enough computing power to perform all function in real time. During operation, our computer averages about 50% CPU time usage, peaking up to 60% during route planning and other computationally intense events.

### Robot Operating System

The Robot Operating System (ROS) is incredibly useful in our project. Iggy relies on ROS to integrate all sensory and GPS waypoint input and run all software subsystems, including goal setting, white line detection, flag detection, obstacle avoidance, and waypoint navigation.

### Visual Processing

Our team this year spent a large amount of time modularizing the vision processing code that was written in previous years. Previous teams relied on a piece of software that was written without modularity in mind. While that worked for white line detection alone, it did not allow for the seamless integration of the red and blue flag filters.

Thus, our team broke the vision process down into three stages.

1. The first stage is the production of two images from a stereo camera and producing a disparity image. This disparity image is used to convert the $(row, col)$ coordinates on an image into the $(x, y, z)$ coordinate system needed to produce a pointcloud.
2. The second stage is the filtering stage. The filtering stage is the stage where images are taken from the camera in stage one and processed to produce binary mask images that represent detected objects that need to be filtered. The filtering stage finishes and publishes mask images (black and white mask images with the white pixels representing the object of interest. ).
3. The final stage is the "Vision" stage, which converts masks to pointclouds. During this stage, the software iterates over every given mask, and if the mask is white (255,255,255), then the point is converted through the disparity image and put into the pointcloud and published on a topic.

*White Line Detection* Iggys white line detection subsystem is responsible for identifying the white lines that represent the lane boundaries for the course. The ROS component receives the images produced by the Bumblebee2 stereoscopic camera and runs them through both OpenCV's Canny line edge detection and hough transform to detect the white lines. The resulting image is a fully filtered image with all white lines identified.

*Flag Detection* Iggy uses a python OpenCV2 filtering script that takes the images produced by the Bumblebee2 stereoscopic camera and produces masks where it sees red or blue objects within a certain distance from the camera. This image is handed to the Vision stage of the three stage process where two separate red and blue point clouds are produced.

### Navigation

The robot incorporates the variety of sensor data to create a cost map. This map assigns costs to obstacles and we set the obstacles and white lines to a high cost so that the path planning algorithm cannot go through them. With this map in place, we use ROS's base local planner to plan the path. Using odometry and dynamic map building we can navigate through the costmap.

*Goal Setting* A user interface provides the ability for the user to input GPS waypoint coordinates. These coordinates are placed in a navigation stack that the robot uses to do path planning.

*Simultaneous Localization and Mapping* SLAM allows the LiDAR and the White Line Detection algorithm to populate a 3D representation of the environment around the vehicle. Each time the vehicle moves, sensor data from the IMU and GPS update the map so that obstacles will move on the 3D map relative to the robots position just as they do in the real environment.

*Global Path Planning* After the 3D map is created, the robot determines the shortest path between the current location to the first GPS waypoint on the navigation stack. Then commands are sent to the motor controller to move the vehicle. The path planning algorithm has two aspects. Global Path Planning shows the start point to end point, akin to a map with the major highways and roads. The Local Path Planning looks at the surroundings and adapts, akin to a driver driving on the road and passing other cars and staying in his lane.

## PERFORMANCE PARAMETERS

Over the course of the year, our team conducted sub-component testing to ensure that everyone's products worked before combining them all during the last sprint. We did not experience any tests in any of the subsystems that would indicate that our system integration would not be able to meet the performance parameters required at IGVC. Integration testing was conducted indoors with plans to conduct outdoor testing in late May.

## PERFORMANCE ANALYSIS

The current robot can perform LiDAR ground segmentation to separate the pointcloud into two, only keeping the pointcloud that represents obstacles on the map. The GPS and IMU can provide sensor data when in plain view of the sky. The Bumblebee camera, through the use of the newly created Vision3D system, can perform White Line Detection and Flag Detection. At this point, we have successfully integrated all of the components into the robot. The robot receives a GPS coordinate, localizes, and sees obstacles given from the LiDAR and the Vision3D subsystem. At this point, our teams final tasks before the day of the competition are doing test runs so that we can determine what edge cases we have no accounted for yet.

## SUMMARY

In retrospect, our team this year learned a lot about the trials that a development team faces when working separately towards a common goal. After overcoming the challenge of working with the robot given little to no documentation, everyone eventually learned how Iggy worked and was able to successfully contribute to her performance in some way. By redesigning the Vision3D system, the robots vision processing software is modular enough to add more filters with ease in the future. The newly designed power system allows the robot to run a significantly larger amount of time, making performance both in testing and on the course much more convenient.

## CONCLUSION

Furthermore, our contributions to this project mean that in the future when we join a more advanced and complicated project such as a self-navigating vehicle that research teams might work on, we can understand the fundamentals of what is required for auto-navigation as well as combining different discipline strengths together.