The University of Detroit Mercy
Presents

Revenant

IGVC 2013 Design Report

Team Members
Chris Smalley
Gizelle Guerra
Joseph Gilsenan
Chao Tong
Leonardo Martinez Roman
Di Zhou
Hong Nguyen
May Tamer
Hongyi Zhang
Yu-Ting Wu

Faculty Advisors
Dr. Chaomin Luo
Dr. Utayba Mohammad
Dr. Mohan Krishnan
Dr. Mark Paulik

Consultants
Cheng-Lung Lee
Bo Cui
Xingzhong Zhang

We certify that the engineering design in this vehicle undertaken by the student team, consisting of undergraduate and graduate students, is significant and qualifies for course credits in senior design and in the Master's program respectively.

____

Dr. Mohammad, Dr. Luo, Dr. Krishnan, Dr. Paulik

1. INTRODUCTION

For the 2013 Intelligent Ground Vehicle Competition (IGVC), the University of Detroit Mercy (UDM) is introducing *Revenant*, shown on the cover page, a skid steered vehicle purchased from Clearpath Robotics. *Revenant* is the product of many significant innovations, ranging from extensive software enhancements to a completely new ROS-based software platform.

The team that expanded upon the previous year's efforts is composed of senior undergraduate students as well as several graduate students who implemented solutions to the higher-level needs of the competition, including its new requirements.

2. DESIGN INNOVATIONS

There were a considerable amount of advancements in the development of *Revenant.* Many of these advancements resulted from the evaluation of previous robots that the University of Detroit Mercy designed for other IGVCs. This year several innovative concepts were pursued and engineered in order to give *Revenant,* the best opportunities to be successful. These developments resulted in not only a more competitive vehicle, but a vehicle that is more practical and adaptable for future competitions. These innovations are listed below and will be further explained in Section 7.

a) Robot Perspective Head Mount Display
b) Heuristics for Image Understanding and Interpretation
c) Data Fusion between LIDAR and Camera

3. PROJECT MANAGEMENT

3.1 Design Process

The undergraduate members of the team were in a Capstone Senior Design course. The graduate team members took other classes covering material relevant to the project. A systematic design and development effort was followed that accommodated continuous improvement of the product through several cycles of effort ranging from initial concept development of sub-systems through design and evaluation to final integration. A framework expediting continuous dialogue and collaboration within the team and also with faculty was established using wikis and discussion threads and weekly face-to-face meetings.

Early in the process, a thorough assessment of the literature was performed to identify algorithms potentially suitable to address some of the modifications in the competition rules; this was done in conjunction with a reflection on the performance of our last competition entry in 2012. This was followed by a detailed feasibility study that narrowed down the options and laid the foundation for an initial design concept formulation. As part of the design process, deadlines were set for the various project sub tasks. Pace of progress was enforced and assessed through weekly review meetings and progress reports. Slippage in terms of achievement of goals, perhaps due to unforeseen technical difficulties, was addressed through change in strategy or reallocation of resources. The iterative strategy adopted provided an excellent framework for overall project management.

3.2 Team Composition

The cross-disciplinary team was made up of two EE and two ME undergraduate students along with six graduate students working on specialized tasks. Task assignment was based on area of expertise, interest, and the project needs. Figure 1 indicates the team structure and assumed responsibilities. The total time devoted to the project was approximately 3000 hours.

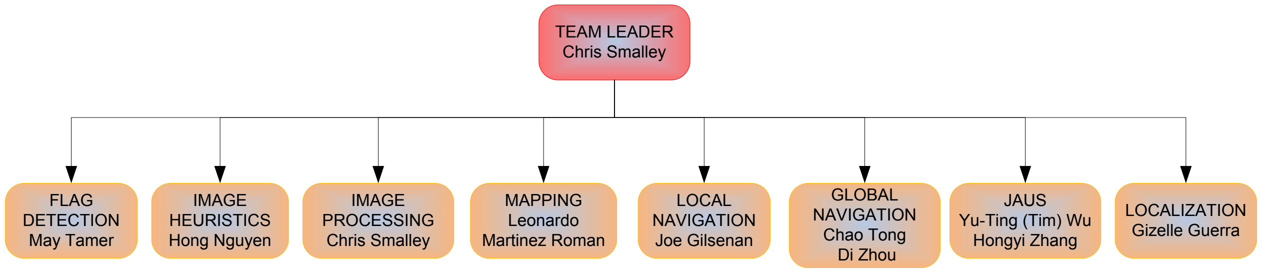
TEAM LEADER Chris Smalley
FLAG DETECTION May Tamer
IMAGE HEURISTICS Hong Nguyen
IMAGE PROCESSING Chris Smalley
MAPPING Leonardo Martinez Roman
LOCAL NAVIGATION Joe Gilsenan
GLOBAL NAVIGATION Chao Tong Di Zhou
JAUS Yu-Ting (Tim) Wu Hongyi Zhang
LOCALIZATION Gizelle Guerra

Figure 1: Team Organization

3.3 Design Objectives

In accordance with the principle of continuous improvement, the enhancements incorporated into *Revenant* were based on IGVC performance in the previous competition year taken in conjunction with the rules for this year's competition. Both faculty advisors and team members met to review our previous vehicle's performance at the 2012 competition and that of other teams. The document generated at that meeting served as the basis for the development of the design objectives for *Revenant.* These were:

i. Meet all IGVC requirements.
ii. Learn and deploy the ROS robot meta-operating system to implement algorithms.
iii. Develop and implement a new method of local navigation to replace VFH.
iv. Enhance the vision strategy to increase redundancy and resistance to failure, as well as to handle flag detection requirements.
v. Develop an EKF-based algorithm to increase localization accuracy and permit increased mapping accuracy.

3.4 Cost Summary

While *Revenant* was purchased, it needed to be developed and modified considerably to fit the needs of the competition. The systems incorporated were either designed and built in-house or specifically purchased to best suit the project objectives in terms of functionality, performance, cost, and development time. Table 1 shows the approximate retail cost of the various sub-systems in *Revenant* as well as the team cost (some items were donated, reduced in price, or re-used from an earlier project).

Description	Retail Cost	Team Cost	Comments
Husky Robot Chassis	$12,500	$6,250	Discounted by Clearpath
Custom Power System & Drive train updates	$2,280	$890	Discounted by Clearpath
Wheel & Mast	$600	$300	Discounted by Clearpath
KVH	$16,000	$0	Donated
Digital Compass	$ 1,890	$ 0	Donated By Spartan
Batteries (4)	$ 225	$ 225	Purchased
Sensor Mast System	$880	$880	Purchased
Custom Top plate	$480	$480	Purchased
Weather tight laptop enclosure	$345	$345	Purchased
Camera, Lens, Adapter	$ 937	$ 0	Salvaged
LIDAR	$ 6,700	$ 5,176	Purchased
DGPS & Antenna	$ 6,000	$ 0	Salvaged
Digital Compass	$ 1,350	$ 0	Donated By Spartan
Wireless Gigabit Router	$146	$146	Purchased
Dell	$ 3,584	$ 3,584	Purchased
Minibox	$1,100	$1,100	Purchased
OmniStar HP	$ 1,250	$ 0	Donated
Incidentals, cables, etc.	$1000	$1000	Purchased
TOTAL	**$ 57,267**	**$20,376**	**Savings = $36,891**

Table 1: Various Sub-Systems on *Revenant* and Associated Cost

4. MECHANICAL SYSTEM

This year we aimed to put more focus on the software challenges of the competition. We bought the Husky A200 base platform from Clearpath Robotics. It provided a strong, solid base which included wheels, motors, chassis, battery etc. The mechanical configuration of *Revenant* (pictured on the cover page) provided our team with a vehicle that was more durable, more accessible, and more versatile than previous competition designs. The chassis is 39 inches in length and 26 inches wide. *Revenant*'s total height, with a custom sensor mast system, is 70 inches. The maximum speed of the vehicle is 5 mph, which meets the speed limit requirement but is still fast enough for Revenant to navigate the course optimally.

5. ELECTRICAL & ELECTRONIC SYSTEMS

The electrical systems design was made up of enhancements to the already robust Husky chassis. Because several additional sub-systems were added, attendant upgrades were required that addressed safety, efficiency, and functionality. The design changes led to the creation of a schematic for a new and improved printed circuit board. The improvements incorporated into this new board are discussed below.

5.1 Power Distribution

To provide adequate power to all sub-systems of *Revenant*, the requirements of each were first assessed under normal and worst-case conditions. The power for *Revenant* comes from a 24V sealed lead acid battery and is then distributed via the custom-designed PCB to the vehicle sensors, wireless router, motors, and two computers. The overall power distribution scheme is shown in Figure 2.

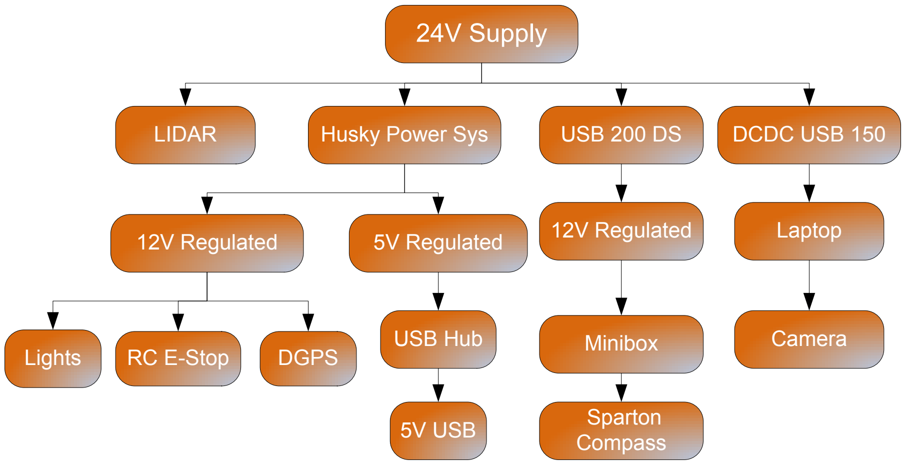

Figure 2: Power Distribution

5.2 Electronic System Protection and Safety

For user safety, *Revenant* is equipped with hard, soft and remote E-stops controlled by a mechanical button, the microcontroller, and the remote control respectively. The remote control, which can operate in one of two modes - Computer Controlled (PC) or Remote Controlled (RC) - is made up of a custom-designed PCB housed within a durable Futaba remote control shell. When the remote control is set to operate in PC mode, it transfers control of the motors to the computer (retaining E-stop control). If placed in RC mode, the operator can manually drive the vehicle. The transceivers that are used in *Revenant's* design are Aerocomm AC4490-200A transceivers. Although the vehicle only needs to be controlled from a maximum distance of 50ft, with the implementation of the aforementioned transceivers and full antenna extension, the vehicle is capable of being controlled from nearly a mile away. A twist-to-release remote E-Stop button is integrated into the remote control unit. As an added measure of security and immunity to interference, we transmit encrypted data over a spread spectrum wireless link for two-way communication between the vehicle and remote.

5.4 Sensor System

Revenant incorporates five sensors into its compact design: a camera, a LIDAR, a DGPS, a digital compass, and an IMU. Each sensor is enclosed in a waterproof case and firmly mounted to the vehicle while, at the same time, permitting easy removal for servicing. The following is a brief description of the sensors that are used by *Revenant* as shown in Figure 3.

Camera: The AVT Stingray F-080C 1/3" CCD camera was selected as the vision sensor for this vehicle. This camera uses the IIDC IEEE 1394B

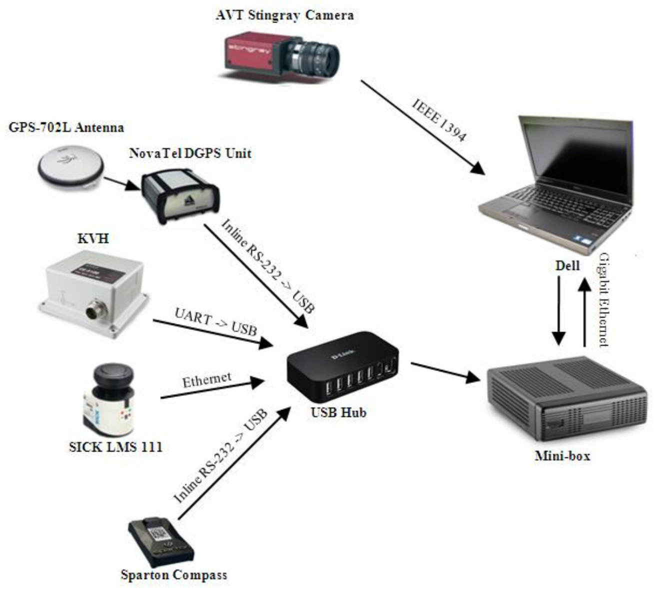

Figure 3: System Communication

protocol to relay images, which is ideal for machine vision applications, because the frames are uncompressed and various options such as region of interest and lookup tables can be set and executed in hardware. Also, the camera's progressive scanning and high frame rates minimize motion blurring. The CS-Mount lens design enables the camera to accept a very wide-angle low-distortion lens, which provides a 125^0 field-of-view. This makes navigation heuristics easier to implement.

LIDAR: A 270^0 SICK LMS111 LIDAR unit was employed for the purposes of obstacle detection. The unit is capable of collecting data over a 270° field-of-view with 0.25° resolution, a maximum range of 20 m, and a 25 Hz scanning rate.

DGPS: To obtain positioning data in the Navigation Challenge, Novatel's ProPak-LB Plus DGPS system was selected. The DGPS antenna is mounted to the top of the vehicle's mast while the receiver is securely positioned inside the chassis. Using Omnistar HP's DGPS system, the signal is corrected to provide up to +0.1m accuracy. This system provides data at a rate of 20 Hz, which is adequate for *Revenant's* expected speed and desired performance.

Digital compass: The Sparton AHRS-8 (Altitude Heading Reference System) was integrated into the vehicle to help determine heading. The AHRS-8 provides accurate heading readings by eliminating external magnetic disturbances that affect heading accuracy, it also provides 3D absolute magnetic field measurement and full 360° tilt-compensated heading, pitch, and roll data. This enables the compass to provide a heading accuracy of 0.2° and updates at 20 Hz, which is sufficient for the vehicle's speed and desired performance.

IMU: The KVH CG-5100 IMU was generously donated to our team by KVH Corporation; this inertial measurement unit (IMU) incorporates highly accurate fiber optic gyro (FOG) based sensors coupled with industry proven MEMS accelerometers. This sensor was added to the interior of the robot chassis and will also be used to determine heading, and is able to provide an accurate reading, with only 1° of drift per hour and a refresh rate up to 100Hz.

5.5 Data Communication Interfaces

The various electrical and electronic systems were interfaced in the manner illustrated in Figure 3. The AVT Stingray camera is connected via Firewire 1394B to the Dell computer. The DGPS system is connected to the internal mini-box computer through a USB interface using an inline RS-232-to-USB adapter. The SICK LMS111 LIDAR is connected to the computer via an Ethernet link. The SPARTON AHRS-8 digital compass is also connected to the mini-box and uses RS-232; finally, both of the computers are networked via gigabit Ethernet.

6. SOFTWARE DESIGN

The primary goal in the software area this year was to develop innovations in conjunction with the use of the ROS environment. Accordingly, a new software strategy was developed and implemented, including a modified image interpretation and heuristics strategy and an improved mapping and localization module. This was accomplished through the use of identifying problematic vehicle behavior from the previous year and using the ROS development environment to discover and implement new solutions in those areas. After initial algorithm development and testing, new tools contained in ROS were used to validate the algorithms through simulation, finally deploying the successfully validated algorithms and testing them on the new vehicle platform.

6.1 Development Environment

In previous years, the UDM team has used the Player/Stage environment as the framework for the software implemented on the robot. This year, ROS will be used as the primary development environment. As in previous years, MATLAB, set up to communicate to ROS through an IPC bridge, will continue to be used for execution of image processing algorithms for the associated ease of development. ROS is an open source robotics platform that operates on a peer-to-peer model; this structure provides some advantages over the client-server model of Player/Stage. In peer-to-peer networks all nodes can communicate directly with each other, without invoking a centralized server. As the scale of the system increases to support more nodes, the load on the server remains manageable.

Revenant's software development architecture utilizing ROS is shown in the Figure 4 below. ROS has three layers of functionality. At the lowest (graph) level ROS operates on the basis of a network of nodes. These are modular processes that execute a specific task (such as read in and convert GPS data, transform coordinate frames, send velocity commands, etc.). Nodes communicate with each other by sending messages, using topics with specific names. Nodes can either publish messages to a specific topic or subscribe to one or more topics. The processing of these messages is used to direct the algorithmic flow, as needed.

The file system level of ROS consists of Packages and Stacks; Packages are units that may contain ROS nodes, libraries, or any other type of related file, and Stacks are collections of Packages making up a higher level application. The third and outermost level of the ROS universe is the Community level, which consists of the online distributions and repositories of code addressing various applications.

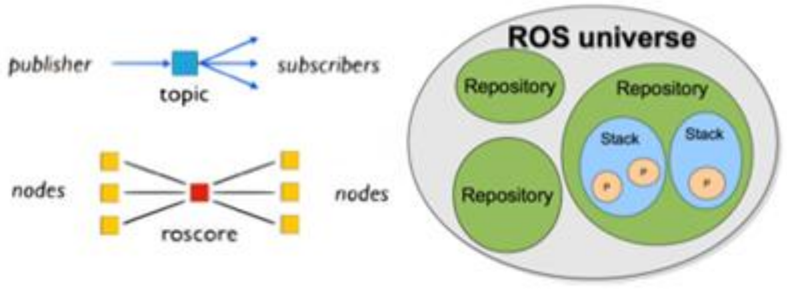

Figure 4: ROS Architecture and Resources

The ROS environment is effective not only for final implementation but also for simulation as it allows for the use of the Stage simulator. Throughout the progression of algorithm development, Stage accommodates testing via simulation. The ROS/Stage simulator is a representative and comprehensive simulation environment, complete with models for the robot, obstacles, GPS, camera, LIDAR, etc.. Stage also provides a graphical depiction of robot motion within its environment, which offers a powerful tool to gauge the effectiveness of algorithms.

6.2 Data Acquisition and Processing

With a steady stream of data from numerous sensors being made available, multiple computers are used to distribute tasks and ensure that all data is acquired and processed in a timely manner. The task distribution on *Revenant* is carried out as discussed below.

Sophisticated vision algorithms are central to a successful strategy for the Autonomous Challenge due to the complex obstacle, lane, and terrain features present. If, when implementing these algorithms, the associated computational complexity causes the overall image frame processing rate to drop too low, the vehicle may not be capable of operating effectively at higher speeds. In order to favorably address this tradeoff, a multi-pronged strategy to increase the frame rate was adopted. *Revenant* distributes its computational tasks over two computers, one of them a laptop and the other a 'headless' computer inside the robot chassis. Using ROS facilitates setting up this distributed computing architecture and provides a wealth of existing open-source code to draw from. The top computer is entirely devoted to running components of the overall vision algorithm. This computer utilizes the MATLAB® parallel processing toolbox to spawn multiple workers, which exploit parallelism in several of the IP routines (e.g., color segmentation and adaptive thresholding) by deploying the algorithm over multiple processor cores. The second computer is responsible for running the navigation and path planning algorithms as well as handling the data from the vehicle's sensors.

6.3 Image Processing

In order to address the increasing complexity of the vision task in the Auto-Nav Challenge course, *Revenant* has adopted a new image processing methodology. This strategy combines basic image processing techniques to extract white lane lines from images with development of higher-level image interpretation and heuristics. The image heuristics module retains information from previous image frames in order to track the lane lines including maintaining a sense of the left lane line versus the right lane line. This in conjunction with a mapping strategy allows for accurate goal selection throughout the lane following portion of the Auto-Nav Challenge.

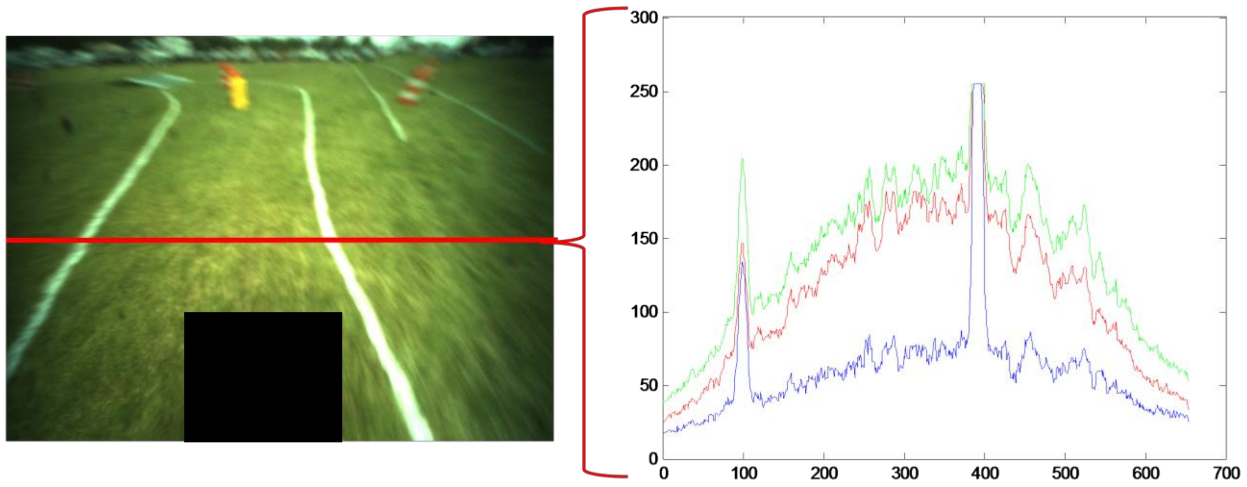

Figure 5: Profile of Image Line Segment

The images to be processed were captured in RGB format, which represents the image color components as red (R), green (G), and blue (B). In order to develop a process of lane line extraction, images were analyzed using the Matlab function *improfile*. This function shows the intensity values of the red, green, and blue planes along selected line segments. Figure 5 shows an example of a profile taken of an image. The graph on the right side of Figure 5 corresponds to the intensity values of red, green, and blue in the pixels along the red line segment that is superimposed on the image. This profile shows that the red plane and the blue plane are only similar in regions where the image is white. In most other areas of the image the red plane is about two times as large as the blue plane. For this reason the basic image processing used 2*Blue – Red in order to highlight white and suppress all other colors in the image. This technique provides a grayscale image which is then processed further using regional thresholding, morphology, and connected component analysis in order to produce a clean binary image. These techniques could be interchanged depending on the conditions and the algorithm can be tailored to fit specific course attributes.

Once the binary images were obtained, a series of Hough transforms were applied. This process results in extraction and tagging of the left and right lane lines, when available, which is then subjected to heuristic tracking. Maintaining the identity of the left and right lane lines is valuable for awareness of the forward direction for robot motion. The Hough transforms also fills in the intermittent and broken binary images that result from basic image processing. The resultant outputs of straight line segments are also more convenient to place on a map.

6.4 Goal Selection

The goal selection algorithm is concerned with determining the "forward" direction for the course. For the Navigation portion of the Auto-Nav Challenge this is relatively easy, as forward is towards the next waypoint. However, in the Autonomous portion of the Auto-Nav Challenge, the forward direction is less easy to determine, since it requires the vehicle to "go around the course" without turning around. The forward direction has to be established from the results of the vision algorithms, which are not 100% reliable. Further complicating the situation is the presence of course features such as switchbacks and ramps, which can fool the robot into turning around.

The combination of image heuristics and mapping provide a basis for creating a goal selection algorithm in the Autonomous (lane line) portion of the Auto-Nav course. Tracking the identities of the left and right lane lines as the robot moves can be used to continually be aware of the forward direction. The map development also helps through knowledge of where the robot has been, which additionally helps maintain forward motion. Thus we can realize improved performance through the redundancy of simultaneously using both techniques.

6.5 Global Navigation (Path Planning) – D*Lite

Performance in the Navigation portion of the Auto-Nav course is considerably enhanced if global path planning is utilized, which results in optimal routes and avoidance of traps. Given a set of waypoints in the Navigation portion of the course, they are first sequenced for optimal traversal. D*Lite is a very effective algorithm to use for planning a path through unknown terrain. It can work off a partially complete map of the field, while accommodating map discovery through re-planning as the robot moves. Using the next waypoint as input, the D*Lite algorithm provides an optimal path. Suitably positioned breadcrumbs are identified along this path. The local navigation algorithm (described below) then drives the robot through those breadcrumbs.

6.6 Local Navigation (Driving the Path) – SND Method

This year, along with the change to ROS, came changes in the local navigation strategy. In previous years, our team used a variant of VFH, which has proven difficult to tune. Specifically, its parameters have to be tuned through trial-and-error in actual runs; also, the parameters that work best for the Autonomous (lane line) portion of the course do not do as well in the Navigation (open-field) portion. The new Auto-Nav Challenge makes it difficult to utilize two sets of parameters in a single run. Through extensive research, we came across a method that is well suited for the new course characteristics. The Smooth Nearness Diagram algorithm, a relatively new variation of the Nearness Diagram navigation method, is a gap-tree based navigation approach developed for use in narrow spaces with a high obstacle density.

The Smooth Nearness Diagram method allows the robot to navigate a course and avoid obstacles by using the obstacle data (read in from the laser scan and map) to create a nearness diagram from which to determine navigable regions. A nearness diagram is a plot showing obstacle nearness per sector (where sectors are defined by the resolution of the LIDAR) with respect to the robot's center. Figure 6 below, shows an example of a possible situation a robot could encounter with various obstacles surrounding it. Figure 7 shows the nearness diagram that results from such a situation; where the higher the amplitude of a given sector, the closer the obstacle in that sector is to the robot.

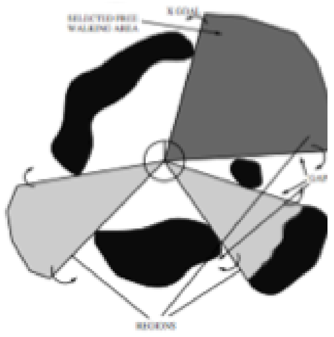

Figure 6: Potential Environment

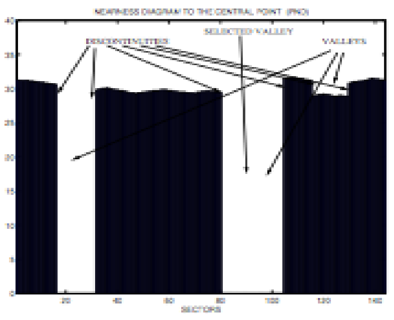

Figure 7: Nearness Diagram

Analysis of this nearness diagram allows the robot to locate navigable regions. The first step is to determine gaps; gaps are defined in the algorithm as discontinuities in the nearness diagram that are greater than twice the robot radius, ensuring enough room for the robot to pass. Once the gaps are determined, the algorithm then looks for valleys, which are navigable regions defined by two consecutive gaps, which are more than twice the robot radius in width. For each valley, a *rising gap* angle and *another gap* angle are determined; the rising gap is the angle corresponding to the sector that contains the gap closest in distance to the goal heading, and the other gap is the angle corresponding to the sector containing the second of the two gaps in the valley.

With this information, the algorithm can now select the valley that makes the best progress toward the goal, by comparing the distance of each valley's rising gap angle to the goal; the valley with the closest rising gap to the goal is called the *best valley*. Once a best valley is determined, a safe rising gap angle is determined by deflecting the rising gap angle from the obstacle that defines the gap.

This adjustment to the rising gap angle will point the robot in a direction such that the obstacle creating the rising gap will not enter the robot's safety distance (a specified radius around the robot into which obstacles may not enter) as it moves toward the gap. However, in some cases, where the best valley is narrow, it is possible that the deflection of the rising gap angle will leave the heading pointing too close to the other gap angle; often in these cases it is better to direct the robot towards the angle which bisects the best valley, ensuring the safety of the robot as well as providing a direction that still takes the robot closer to the goal. The desired heading for travel will then be set equal to whichever of the safe rising gap angle and bisecting angle is closer to the original rising gap angle.

Finally, a last check is done to ensure that no obstacles within view enter the robot's safety distance; this is done by measuring the 'threat level' of each detected obstacle, which is a measurement of each detected obstacle's distance to the robot. If any obstacles are found to be within the safety distance, a second deflection coefficient is computed from a weighted sum of the threat level measurements of obstacles found to be within the safety distance. The final heading for the robot is then calculated by adding the deflection adjustment to the desired heading.

The final heading value represents the direction the robot will travel in to the point decided by the algorithm; this value will then be used by the trajectory planner to determine optimal angular and linear velocities to get the robot to the desired point.

6.7 Localization Strategy

In order to optimize the performance of *Revenant*'s navigation and mapping algorithms, it is necessary to establish a framework for localization to provide an accurate pose estimate; this is the first year that such a formal framework will be incorporated into the UDM IGVC software strategy. The framework for this system will be comprised of a ROS package *husky_ekf*, containing a node that subscribes to the measurement messages published by the robot's wheel encoders, as well as the sensors seen in Figure 8.

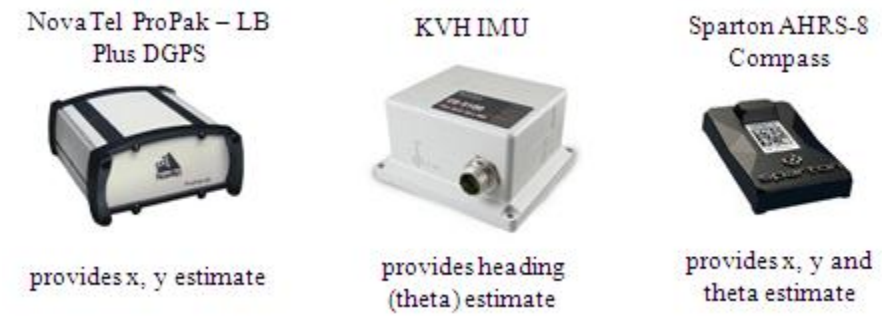
NovaTel ProPak – LB Plus DGPS
provides x, y estimate
KVH IMU
provides heading (theta) estimate
Sparton AHRS-8 Compass
provides x, y and theta estimate

Figure 8: Localization Hardware

Localization is implemented through the use of a Sequential Update Extended Kalman Filter. Given consistently available GPS data, an Extended Kalman Filter which fuses the DGPS and digital compass data with the robot's other relative sensors (such as the odometry and IMU) can be used to provide an accurate pose estimate; this method is a relatively mature area of study and is well documented in the literature.

The Sequential Update Extended Kalman Filter was chosen as a method of localization because it has been shown to produce unbiased, minimum error estimates when the state equations for the system and the measurements are largely linear, and the error is an independent Gaussian noise process, which is in fact the case for our system, as the only nonlinear characteristics will arise from the system model that reflects the encoder readings.

This approach also accommodates asynchronous inputs to the filter resulting from sensors having different refresh rates. Multiple sensor measurements are incorporated into the observation matrix by using a sequential update of the state for each of the different measurements, whenever they become available, with each sensor observation considered an independent event. Also, the overall error for the state output of the Kalman filter is smaller than that of the best of the individual sensors, which means that the Kalman estimate is always an improvement.

The non-linear relationships resulting from the Husky's system model are handled in the EKF by replacing several matrices in the KF equations with Jacobians.

6.8 Mapping

The mapping algorithms used by *Revenant* are designed to provide local navigation, global navigation, and the image heuristics module current and historic information as needed for the basic operation of each of these algorithms. For this reason one general map is kept with a complete set of information and several other maps are kept containing specific information as required by these algorithms (a map for obstacles, a map for lane lines, a map for position, etc.). The generated maps use different values in each cell in order to represent different properties. The

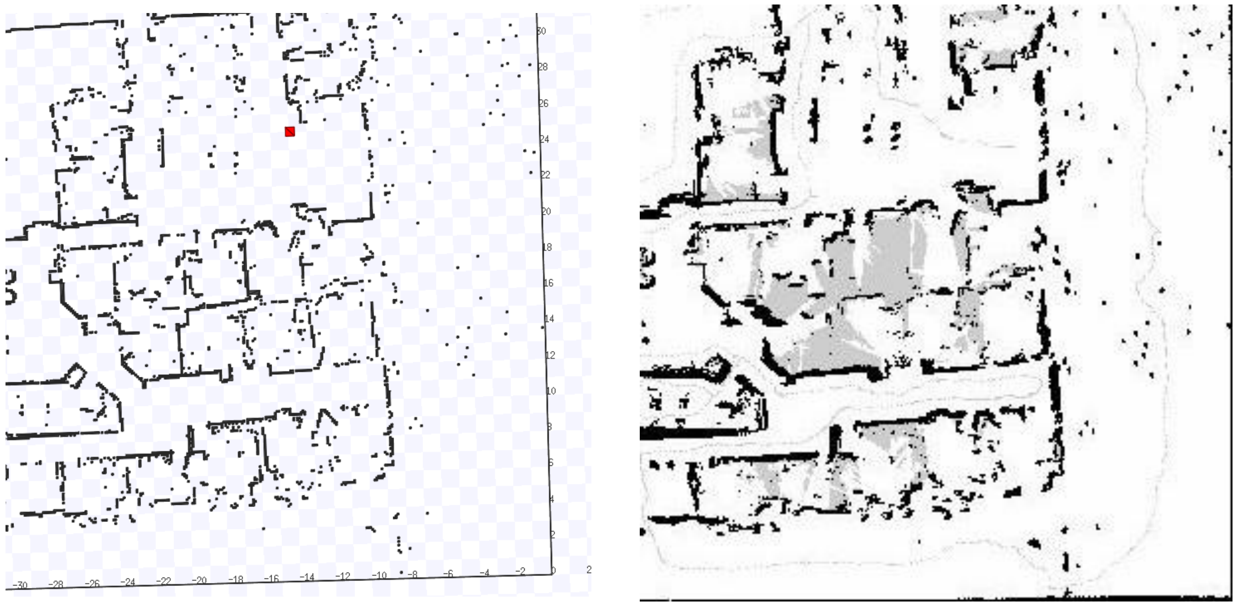
Figure 9: Stage Mapping Simulation

general map includes all the objects of interest needed for navigation; these objects include lane lines, obstacles, red and blue flags, and previous positions. Figure 9 shows a side-by-side comparison of Stage simulation environment and the associated map generated by driving a vehicle manually through the environment.

7. DESIGN INNOVATIONS

7.1 Robot Perspective Head Mount Display

A key innovation and debugging tool that was used this year is a Robot Perspective Head Mount Display. This unit allows the user to remotely subscribe to the topics published on ROS by *Revenant;* for instance, topics such as the raw camera data and outputs of the image processing algorithms. These topics can be overlaid and viewed in real time to see exactly what the robot is seeing, how it is interpreting the images, and what causes it to fail. This is a powerful diagnostic tool for initially getting *Revenant's* algorithms to work and subsequently fine tuning their performance. All aspects of operation such as the local navigator, image analysis and interpretation, path planning, etc. can be debugged through the robot perspective display.

7.2 Image Heuristics

Correct lane detection is a difficult problem even with high-level image processing strategies. In some cases such as switchbacks or in portions of the course where there are a large number of barrels the lane lines can be lost. In these situations it is beneficial to apply a deeper level of understanding to image frames, so that even when lane lines or portions of lane lines are lost, the robot still has an idea of where it is on the course and where it needs to go.

Essentially the goal is to tide over momentary problems in the form of incorrect results by using heritage information from the immediate past, which leverages requirements of lane continuity. Figure 10 shows an overview of the algorithm used to track the left and the right lane lines. The image was broke down into a series of sub images on which Hough transforms were performed. These operations resulted in masks which when summed together created straight line representations of lane images. The results of this algorithm on several images can be seen in Figure 11 where the left and right lane lines are differentiated using different colors.

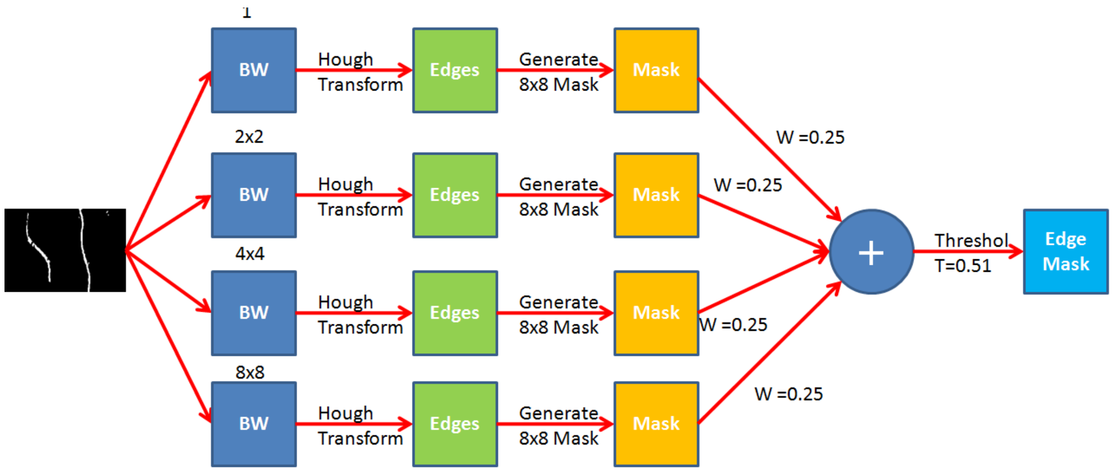

Figure 10: Image Heuristic Algorithm

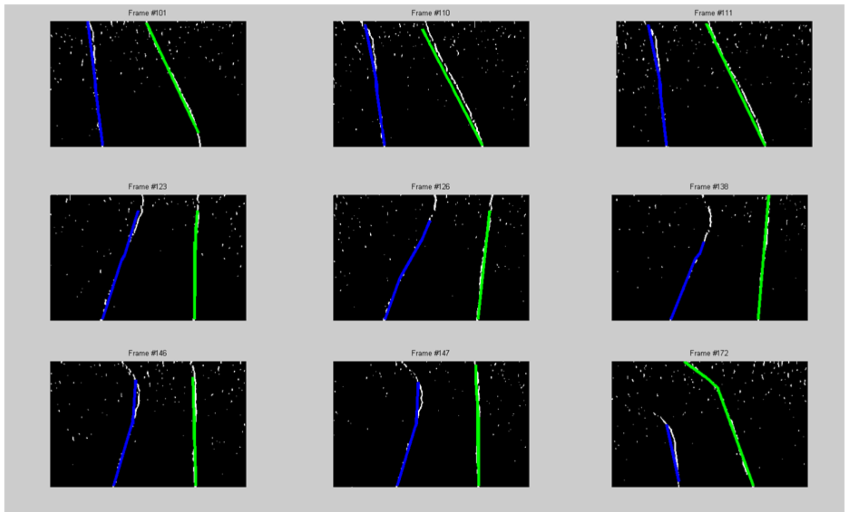

Figure 11: Lane Tracking Results

7.3 LIDAR Camera Data Fusion

Combining information from different sensors is a great way to improve the processing of information; in particular, the inherent difficulties associated with image processing can be mitigated by LIDAR data in conjunction with camera information. Because the LIDAR is an extremely reliable sensor for detecting 3D obstacles it facilitates the processing of camera images. Obstacles such as barrels and saw horses can create false positives for the presence of white lane line objects in a scene. The first step is to calibrate the camera and LIDAR data as well as transfer them to the same coordinate frame. Once this is done, the LIDAR information can be placed on images. Obstacles that were seen as roundish by the LIDAR can be interpreted as barrels and an associated mask is created for the image. This sets every pixel value where there is a barrel imprint to zero (essentially the barrel is removed from the image). This eliminates the possibility of white stripes on barrels creating false positives for lane lines. Furthermore, it reduces the computation time for image processing, which can avoid processing those areas. Figure 12 shows a simulated scene with lane lines and barrels and the associated mask produced. This type of mask can be directly applied to images captured by the camera and will improve the reliability and stability of the basic image processing steps.

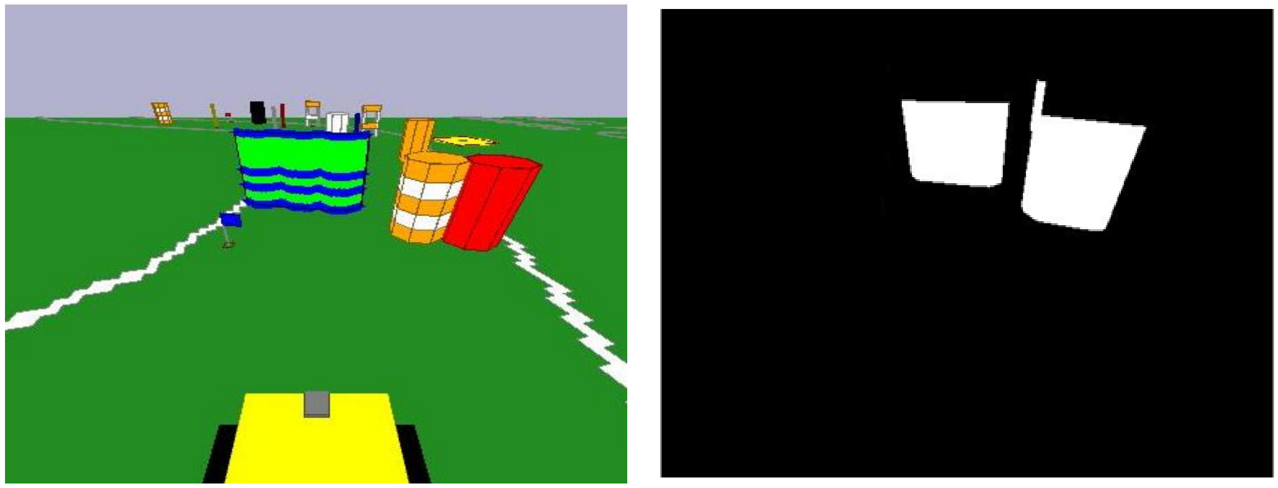
Figure 12: LIDAR/Camera Data Fusion

8. SYSTEM INTEGRATION

This project was divided into subtasks to facilitate development and assignment of tasks to individuals. The ROS environment is suitable for a breakdown of tasks because individual students can develop separate nodes that publish data and subscribe to other nodes' data. Of course, team members need to agree on the data format, but ROS essentially supports the independent development of algorithms for the various tasks. All hardware interaction was done through the capabilities available in ROS.

Software system integration for *Revenant* was carried out using a test course built on campus to represent possible scenarios that might be encountered at the IGVC. The physical integration of the subsystems, to fit properly and make efficient use of space, was given careful consideration during the chassis design process, by taking their dimensions and locations into account, including the space needed to accommodate connectors and wires.

9. JAUS

Our goal for the JAUS challenge was to implement a system that would meet the SAE JAUS requirements, while at the same time be compatible with each of the research robots used in our Advanced Mobility Lab. We chose to implement JAUS using Jr Middleware. It is SAE AS-4 AS5669A compliant software that handles routing JAUS messages on a network. It provides an API that allows a program to create and send out a message through the use of a function call.

Our JAUS implementation interfaces with ROS to obtain information, and perform JAUS-related tasks. The advantage is that our code could be compiled and run on any system that uses ROS. To verify the functionalities of the system a COP program was written that would send out the messages expected at the competition and display the incoming messages from our system. This allowed us to verify the functionality of messages that are not supported this year by the JAUS validation tool.

10. PERFORMANCE

10.1 Speed

The maximum speed of *Revenant* is approximately 5mph.

10.2 Ramp Climbing Ability

Based upon the rated torque output of the motors, the size of the vehicle's wheels and the selected gearing, calculations and testing have revealed that *Revenant* has ample torque to ascend an incline with a gradient of up to 30% (16.7°) without stalling. According to the IGVC rules, the vehicle needs only to be capable of climbing a 15% (8.5°) incline.

10.3 Reaction Time

For the Autonomous Challenge, it takes approximately 100 ms (10 frames per second) to run the system algorithms (based on software timing estimates). At 5 mph, which is the maximum speed for *Revenant*, this cycle time translates to a decision being made approximately every 22 cm of travel. In the Navigation Challenge, the algorithms take approximately 40 ms to complete. At the 5 mph speed limit, this cycle time corresponds to a decision being made approximately every 9 cm.

10.4 Battery Life

Table 2 lists the power consumed by the vehicle components under normal as well as worst-case operating conditions. Using these values, it is expected that the vehicle will be able to run for approximately 3 hours under normal operating conditions and slightly less than 2 hours under the worst-case conditions. These estimates have been exceeded in actual runs. A battery charger is also positioned inside the vehicle to enable the battery to be recharged, whenever needed.

Power Distribution

	Normal Conditions			Worst-Case Conditions		
Device	Volts	Amps	Watts	Volts	Amps	Watts
LIDAR	24	0.4	9.6	24	0.5	12
Dell Laptop	19	4.8	33	19	5	40
Minibox	12	2	24	12	2.5	30
DGPS	12	0.2	2.4	12	0.2	2.4
Compass	5	0.02	0.1	5	0.02	0.1
Camera	12	0.17	2.04	12	0.17	2.04
Motor/Controllers	24	3	72	24	10	240
Wireless Router	12	0.5	6	12	0.5	6
TOTAL POWER			230			621

Table 2: Power Consumption Estimates

10.5 Distance at which obstacles are detected

The LIDAR unit on the vehicle is capable of detecting objects at a distance of 20 meters; for *Revenant*, the LIDAR is configured for a range of 10 meters. The camera is set up to view a shorter range to reduce glare and horizon effects (approximately 5 meters).

10.6 Accuracy of arrival at waypoints

The waypoints at the competition will be designed as concentric 2m and 1m radius circles centered on the GPS coordinates of the waypoints. *Revenant's* DGPS system provides an accuracy of + 0.1 meters in DGPS mode, and + 0.01 meters in real-time kinematic (RTK) mode. It can be seen that this accuracy is more than sufficient. This has also been verified through actual experimentation.

11. SAFETY, RELIABILITY, DURABILITY

Revenant, with its Husky A200 chassis, is a rugged and high performance vehicle, which includes several features that not only contribute to its performance, but also increase its safety, reliability, and durability. Three E-Stop systems are implemented to ensure that the vehicle can be stopped safely, quickly, and reliably. These are the soft, hard, and remote E-Stops, which are controlled by the microcontroller, the manual mechanical button on the front of the vehicle, and the remote control, respectively. The vehicle is also weatherproofed such that light rain will not cause electrical short circuits. This involves the incorporation of NEMA enclosures for the power distribution system, which are fit inside the shell that surrounds the vehicle chassis and the various components. All additional circuits are carefully fused to prevent electrical damage. *Revenant* is also equipped with a lockout mode, which provides yet another E-stop method, for periods in which the vehicle is left unattended; when in lockout mode, the robot will still power on but motors will not receive commands to drive.

Revenant implements three levels of "watchdogs" on the motor controllers to prevent unintended vehicle operation. The first watchdog is a hardware watchdog, which prevents vehicle operation in the event of a hardware failure. Every 500 ms the computer must send a specific message to the motor controller. If the message is not sent, an E-Stop is triggered. In the event of a hardware failure or computer crash, the controllers will not receive the message and the vehicle will stop. The second watchdog is a software watchdog, to prevent vehicle operation in the event of a software failure. The motor driver will expect a new velocity command from the software algorithm at least every 2 seconds. If such a command is not received, the driver will halt the motors until a new command is received. The third watchdog monitors smooth wireless data transmission between the remote control and the vehicle. Any failure of the remote control or jamming of the wireless signal will trigger the E-Stop, acting as a hardware watchdog.

12. CONCLUSION

The UDM team is excited at the prospect of competing at the 2013 IGVC with *Revenant*. UDM's team has incorporated many changes to *Revenant* this year including improved localization and mapping strategies, better image processing techniques supplemented by heuristics, and a new local navigator. We hope that the design features added to *Revenant* will make us very competitive and successful at the competition.