Navigator Design Report
2011 Intelligent Ground Vehicle Competition

IEEE Student Branch
Rutgers, the State University of New Jersey
New Brunswick, NJ, USA

June 3, 2011

I certify that the design and engineering of the vehicle (original or changes) by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Spasojevic Predrag

Advisor: Predrag Spasojevic , Ph.D

Name	Major	Position	Hours
Adam Stambler	ECE 2011	Project Lead	480
Peter Vasilnak	ME 2012	Mechanical Lead	475
Micheal Koval	ECE 2012	Vision Lead	450
Cody Schaffer	ECE 2012	Electronics Team	60
Elie Rosen	ECE 2013	Electronics Team	60
Nitish Thatte	ME 2012	Mechanical Team	20
Rohith Dronadula	ECE 2012	Electronics Team	15
Siva Yedithi	ECE 2012	Assembly	10
Total			1570

Table 1: Rutgers IEEE Student Branch 2011 IGVC team members.

1 Introduction

The Rutgers Navigator is a brand new Intelligent Ground Vehicle Entry built by the Rutgers University IEEE club. Its 2011 debut marks the first entry of Rutgers into IGVC. The Navigator has been designed from the ground up for traversing an unknown outdoor environment. It is a three wheeled 190lb robot which uses a trinocular vision system, precision DGPS, and a laser range finder to perceive obstacles in its path and to navigate through rough terrain.

2 Design Process

The Navigator was constructed as a year long project of the Rutgers IEEE club. Our team met in the summer of 2010 to outline our goals for the project and design metrics. We wanted a rugged, low-cost, easy to machine robot whose code would take advantage of every open source project possible. As a result, we developed the Navigator chassis, with its blend of 8020 aluminum extrusion, custom CNC machined aluminum suspension, and water jet plastic casing. The software is built around Willow Garage's Robot Operating System (ROS) and takes advantage of a variety of linux libraries. This allowed us to innovate our own ideas rather than rebuild existing infrastructure.

The robot's design was constrained by budget, timeline, and competition requirements. We started as a new team with an unknown small budget, unknown faculty support, and operating entirely on our free time. We needed extreme flexibility. At the same time, we set minimum goals for competing, if not winning. Our robot needed to be able to continually localize itself with at least 1 meter global accuracy while avoiding and identifying obstacles, as it moved at least 1.5 m/s. It needed to be be able to accurately identify lane markings from 6 meters away, construct a map with both the lines and the obstacles, and to dynamically replan based on the changing map. Our processes was to identify the bare minimum hardware requirements to successfully reach these minimum goals, design for them, but also lay plans for improving the feature and adapting the work if given the extra time and money. Thus, our process was iterative: design, simulate, build, test, and use the results to improve the robot and gain support for the project. This allowed us to identify failure points in our design while allowing us to adapt to a growing budget.

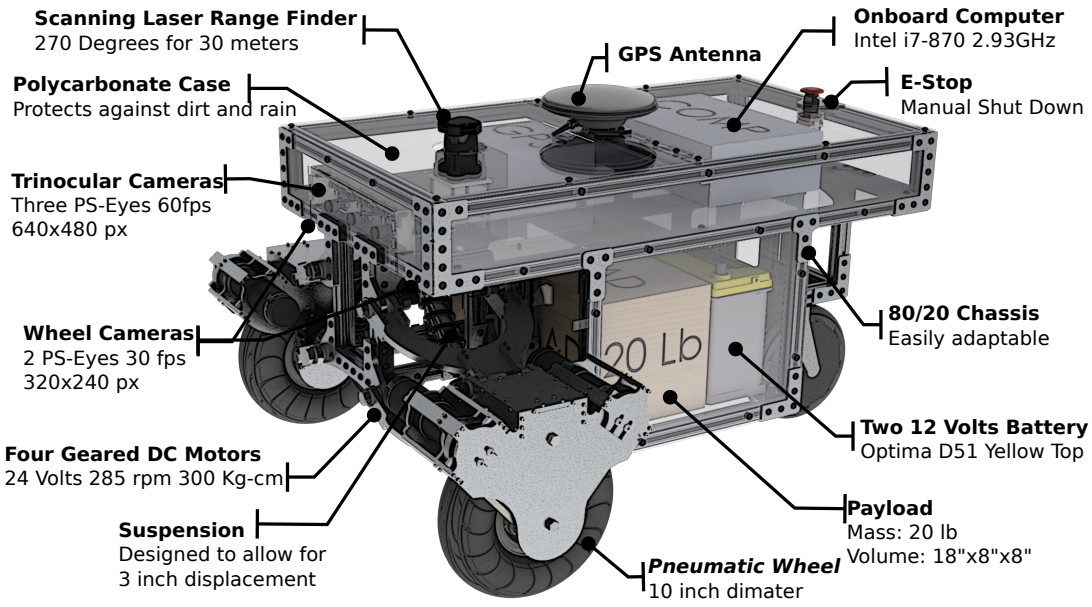


Figure 1: Navigator Hardware Diagram. The batteries and payload are mounted underneath the robot to provide a low center of gravity. All of the electronics and sensors rest in the top compartment for easy access and maximal sensor range.

3 Hardware

The Navigator is a three wheeled, front wheel drive robot weighing over 76 kg. It measures 0.83 m wide, 0.97 m long and 0.66 m tall. It features a custom-designed and machined chassis with a passive a suspension, a suite of twelve sensors, and an on board computer to perceive and plan.

The chassis was designed from the ground up to be both easy to machine and assemble while being light and adaptible. The chassis is framed by one inch 80/20 aluminum extrusion that provides light-weight structural support without requiring specialized manufacturing equipment. The extrusion has T-slots along the edges, which allow the mounting of hardware anywhere along its length. These slots allowed the chassis to be assembled using screws and joining plates rather than welding. Enclosing the 80/20 frame, water-jet cut polycarbinate acts as basic weatherproofing and provides support for the internal components.

The Navigator's signature chassis feature is its front suspension system, which protects the robot's delicate electronics from jarring shocks and vibrations. Each wheel is independently supported by its own shock absorber so that the platform can adapt to uneven terrain. Off-the-shelf suspension systems are not available for a vehicle of this size, so the Navigator uses go-kart shocks with an extremely high spring constant of 200 lb/in. This high spring constant means that the shock absorbers are too stiff to use directly, but are mounted such that they experience a 3:1 mechanical advantage. This creates gives the internal components a softer experience and allows the Navigator to easily traverse outdoor terrain.

Each of the Navigator's two drive wheels are chain-driven by two powerful gear motors. These high-end motors have a stall torque of 600 kg·cm, power that is sufficent to accelerate the Navigator at 2.5 m/s^2 and climb a 30° grade with a safety factor of two. Additionally, each motor contains a built-in Hall-effect encoder that accurately measures wheel rotations. Combined with its other sensors, these quadrator encoders aid the Navigator's localization ability.

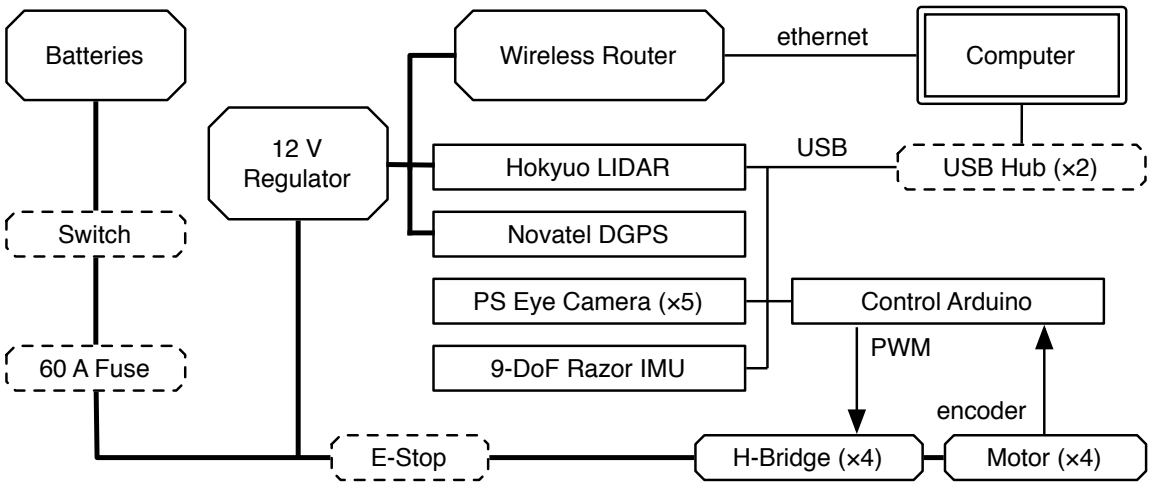


Figure 2: Electronics Diagram

For perception, the Navigator has five cameras and a scanning laser range finder. Its vision system uses specially modified commodity hardware to create a top-of-the-line stereo vision system. Mounted in front of the robot are three Playstation Eye USB cameras modified to capture hardware-synchronized images. Although only costing $40 each, these cameras offer frame rates of 120 FPS at the 320×240 resolution, 60 FPS at the 640×480 resolution, and a 75° horizontal field of view. and 60 fps at 640×480 pixels. Images from these cameras do not suffer from any perceptible motion blur and, once calibrated, produce a dense, accurate point cloud of textured obstacles and points on the ground. Taking advantage of its trinocular stereo camera, Navigator switches between using the two left cameras – when it needs narrow baseline for near obstacles – and to the two outer cameras – when it needs a wide stereo base-line for sensing far-away obstacles. The biggest downside of this vision system is the cameras' relatively small field of view: a flaw that is compensated for using two additional cameras and a scanning laser rangefinder.

These two cameras are mounted directly above each wheel and are tilted down at a 10° and out at 30°. The wheel cameras are the Navigator's primary line-detection sensors and assure that the robot always percieves a line before it moves outside the autonomous challenge course. Together with the front-facing stereo camera, these additional sensors give the robot's vision system a full 180° field of view. To further complement obstacle perception, the Navigator is outfitted with a Hokuyo UTM-30LX scanning laser range finder. This LIDAR unit scans in 270° arc with 0.25° angular resolution and has a maximum range of over 20 m.

For localization, the Navigator has three additional sensors. First, each motor contains a built-in Hall-effect encoder that gives gives 1000 ticks per meter the wheel rotates. Second, the Razor Nine Degree-of-Freedom Inertial Measurement Unit (IMU) serves as a capable and inexpensive replacement for a expensive Altitude and Heading Reference Systems (AHRS). Combining a three-axis MEMS accelerometer, a gyroscope, and a magnometer into a small package, the IMU communicates to the computer via an Arudino microcontroller and provides a heading that is accurate within 0.3°. Finally, the Novatel Propack-V3 Differential Global Position System (DGPS) and Omnistar's L-Band HP Differential Corrections allow the Navigator to localize itself anywhere on the Earth with an root mean square error of 0.1 m.

The remainder of the electronics support perception and navigation. Powering the robot are two 12V 38 A·H lead-acid batteries. The batteries were chosen for price and power: they were the cheapest batteries that

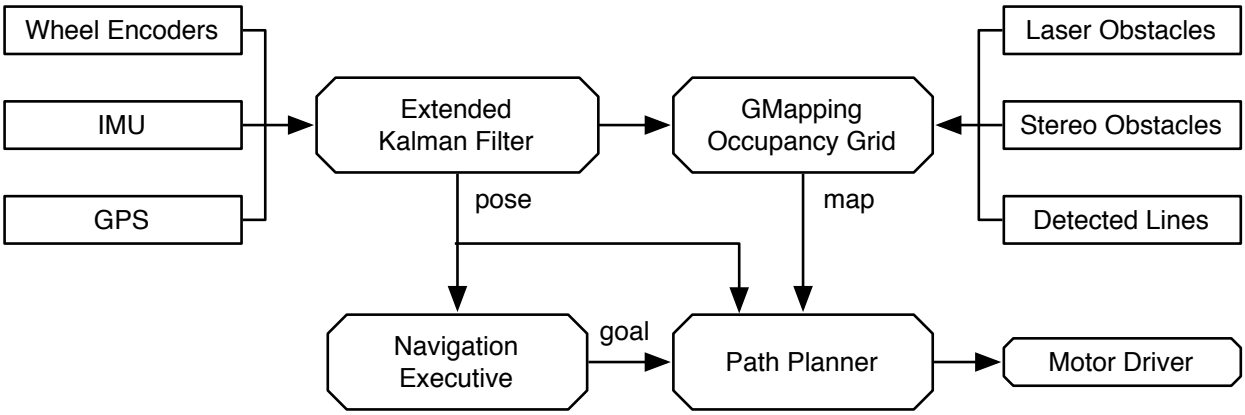


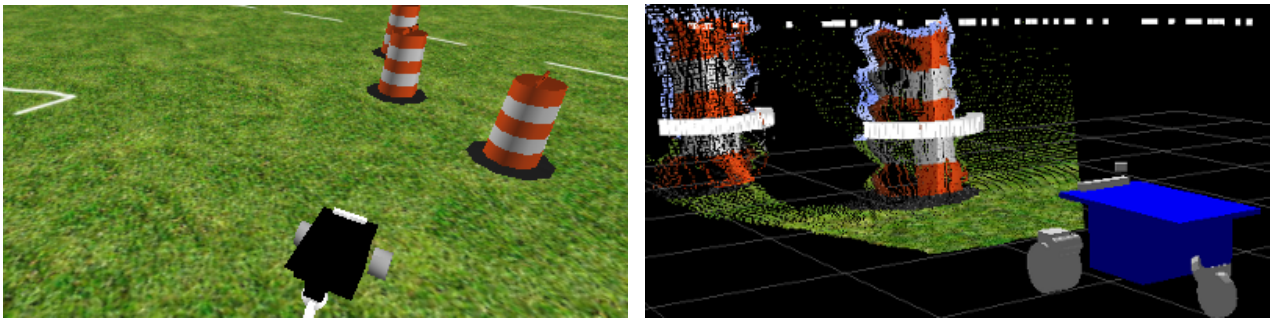
Figure 3: The Navigator has a unified software architecture for both the Navigation and the Autonomous Challenge. The software has three major components: perception, localization, and planning.

would provide at least 2 hours of continuous operation. Power calculations showed that the motors needed about 200 W, the computer needed 200 W, and the rest of the electronics needed another 20 W. This gave a theoretical battery life of 2.15 hours. Controlling the robot is a custom built computer with a Mini-ITX motherboard, 3 GHz Core i7 processor, and 4 GB of RAM. The computer is connected to all the sensors via two USB 2.0 hubs and supports USB 3.0 for future expansion. Additionally, the computer is connected to two additional 8-bit Arduino microcontrollers: one to perform closed-loop motor control via four Simple H-Bridge motor controllers and a second that controls the autonomous indicator light and emergency stop relay.

4 Software

There are three major components to the Navigator's software architecture: perception, localization, and planning. "Perception" processes the camera data and integrates it with the laser range finder. Within the camera processing, the Navigator uses stereo vision to find obstacles and a special line detector to identify the boundaries of the autonomous challenge course. "Localization" probabilistically combines the disparate odometry data into a single pose estimate for the robot. The Navigator's executive and path planner then uses both the pose estimate and its perceptual model to generate a global path through the world and low level motion commands for the motors.

The architectural components are actually composed of many distinct software processes unified by Willow Garage's Robot Operating System (ROS) [10] interprocess communication system. By using ROS, the Navigator avoids creating a huge monolithic program that would would be difficult to write and debug. Instead, each functional part of the robot is its own separate program that communicates using ROS's network-agnostic communication system. This allows our software to scale effortlessly with the number of CPU cores and the number of computers on the robot. Another advantage has been the wealth of robotics software libraries available in the ROS ecosystem. The Navigator has particularly benefited from RViz visualization system, the Gazebo robot simulator, OpenCV computer vision library, and the navigation stack.



(a) Gazebo Simulation (b) Simulated Data Visualization

Figure 4: The Navigator's path planning and perception code has been fully simulated using Gazebo. On the left is a screen shot of our Autonomous Challenge map in Gazebo while on the right is the simulated sensor data output.

4.1 Simulation

The vision and control software was tested using the Gazebo robot simulator. Gazebo naturally interfaces with ROS. Once the physical robot is described in its modeling format, the simulated robot is a drop-in replacement for the physical robot. It has the same kinematics, trinocular vision system, wheel cameras, and laser scanner. Furthermore, we even modeled the challenges. Our robot has completed a simulated autonomous and navigation challenge complete with white lines at the boundaries and construction barrels blocking its path. The simulation served as an early test and benchmark for the robot's perception and planning code.

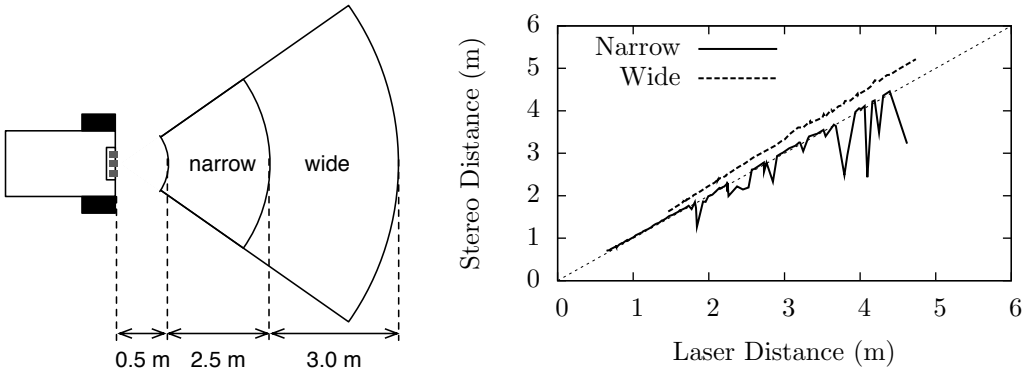
4.2 Stereo Vision

With the Hokuyo UTM-30LX laser range finder the Navigator is able to dodge obstacles in the rangefinder's field of view. While the laser rangefinder has unparalleled range, sample rate, and accuracy, it is not able to see every type of obstacle that the Navigator faces: flags, fences, and some road obstacles have a profile that is difficult to detect using a scanning laser rangefinder. To avoid colliding with such obstacles, the Navigator uses a custom-built trinocular stereo camera to localize obstacles relative to the robot. Stereo vision, the process of reconstructing a three-dimensional scene from two or more photos does not share these deficiencies, but suffers from a large minimum range and noise that directly proportional to distance. To solve these problems, the Navigator uses a custom trinocular stereo camera to simultaneously reconstruct the world using two pairs of cameras: the *narrow* (left-middle) pair for nearby objects and the *wide* (left-right) pair for distant objects.

4.2.1 Block Matching Algorithm

As the inverse of prospective projection, stereo vision aims to reconstruct the imaged scene as a three-dimensional point cloud by measuring *disparity*, the distance between points in two images of the same scene. When nothing is known about the relative position and orientation of the two cameras, each pixel in the left image could correspond to any pixel in the right image. To improve the naïve quadratic-time point-matching algorithm, the Navigator uses *image rectification* to reduce the size of the search space.

Using knowledge of the relative position and orientation of the two cameras, the Navigator transforms



(a) Stereo Range and Field of View

(b) Stereo Reconstruction Error

Figure 5: Performance of the trinocular stereo camera mounted on Navigator. Figure 5a shows how the narrow baseline is used for objects between 0.5 m and 3 m and the wide baseline for objects between 3 m and 6 m. This avoids the inaccuracies in the narrow baseline that begin to appear after 3 m, as is seen in Figure 5b.

both images into a rectified coordinate frame in which the cameras are exactly parallel. Searching for point correspondences is now simple: instead of searching the entire image, it is necessary only to search the corresponding row [8]. Once rectified, the pair of images is convolved with a texture-enhancing filter and each point in the left image is paired with its corresponding point in the right image with a *sum-of-squared difference block matching* (SSD-BM) algorithm [8].

The SSD-BM algorithm matches the each point in the left image to the point in the right image that will minimize the sum-of-squared difference between the patches of pixels that surround the two candidates. [8]. This process is performed again on the right image, and the two are checked for consistency. Any pixels that do not closely match their corresponding point or whose pair changes the second time SSD-BM is run are assumed to be noise and are discarded [4]. With a properly tuned similarity threshold, the Navigator's stereo camera produces a real-time (approximately 10 Hz) dense disparity map that is highly resistant to incorrect point correspondences.

4.2.2 Trinocular Baseline Selection

Even with the robust block-matching algorithm described above, the Navigator's ability to perceive an object is contingent upon two conditions: (1) regions of sufficient texture are simultaneously visible in both cameras, and (2) the object is close enough to produce a non-zero disparity. Mounting the cameras close together, equivalent to selecting a small *baseline*, guarantees that the cameras share most of their fields of view and gives the stereo system a small minimum range. Conversely, designing the stereo system with a large baseline increases the disparity at each distance from the camera and improves the system's maximum range.

Through consideration of the Navigator's size and the width of the navigation course specified in the 2011 IGVC rules, it was determined that the stereo system must be able to localize obstacles between 50 cm and 6 m with a maximum error of approximately 10 cm. Unfortunately, it is not possible to achieve both of these goals with a 640×480 pixel camera with a single baseline: 10 cm is the largest baseline that has a minimum range of 50 cm, but suffers from too much error to accurately reconstruct objects at 6 m.

To meet these demanding design requirements, the Navigator simultaneously reconstructs nearby objects

using the *narrow* camera pair with a 10 cm baseline and distant objects using the *wide* camera pair with a 20 cm baseline. The accuracy of both baselines was evaluated by comparing the reconstructed distance of a calibration object to highly-accurate distances measured by the Hokuyo laser rangefinder. As Figure 5 shows, intelligently merging the narrow and wide reconstructions satisfies all three design requirements.

4.2.3 Ground Plane Extraction

In addition to reconstructed obstacles, the output of stereo reconstruction also contains a large number of points on the highly-textured ground. If all reconstructed points were assumed to belong to obstacles much of the ground would incorrectly be labeled as untraversable and the Navigator's path planner would fail. By assuming that the ground is relatively flat and can be locally modeled as a plane, the Navigator uses the reconstructed ground points to constantly estimate the constantly-changing position and orientation of the stereo camera relative to the ground.

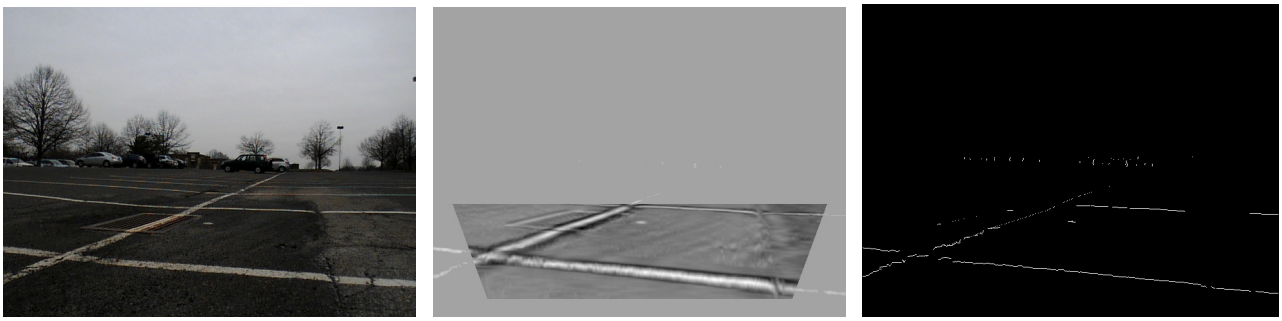
Under most circumstances, when the region directly in front of the Navigator is not occluded by obstacles, a high percentage of reconstructed points lie on the highly-textured ground plane. Because of this abundance of points, the ground plane is assumed to be the dominant plane in the image and capable of being identified by using RANdom SAmple and Consensus (RANSAC), a model-fitting algorithm that is robust in the presence of outliers, to fit a plane to a subset of the data [2]. Once fit, the resulting model parameters are verified using a series of manually-tuned thresholds: (1) size of the inlier set, (2) orientation and (3) vertical distance. If any of these tests fail, the fitted model parameters are discarded and computations continue using static defaults that were measured from the Navigator's CAD model. This redundancy allows the Navigator to continue functioning even when its ability to perceive the ground has been compromised.

4.2.4 Obstacle Detection

Once the ground plane has been extracted and removed from the reconstruction, the presence outliers caused by reconstruction error and incorrect point correspondences indicates that the data still contains a significant number of false-positives. Because the reconstruction is a point cloud consisting of one or more objects, true inliers are tightly clustered near other inlier points belonging to the same obstacle. Outliers, which are produced mostly due to incorrect point correspondences, do not share this spacial locality. To reject these outliers, the Navigator applies a *statistical outlier detection* (SOD) filter to candidate obstacle points. For each reconstructed point the SOD filter calculates the mean and standard deviation of the point's k nearest neighbors. Any of the k neighbors that are more than α standard deviations away from the mean are classified as outliers and are removed [11]. By tuning the parameters k and α, the SOD filter allows the Navigator's path-planning algorithm to directly add detected obstacles to its occupancy grid without additional post-processing.

4.3 Line Detection

Not only must the Navigator see road obstacles, it must also be capable of detecting the white painted lines that define the edges of the Autonomous Challenge obstacle course. Analysis of the 2010 IGVC results show that accurate line detection is crucial to performing well: a mere 2 of 28 teams lasted the full duration of the Autonomous Challenge without being disqualified. To avoid suffering the same fate, the Navigator simultaneously searches for lines in images from three cameras: the front-mounted stereo camera, the left



(a) Original Image (b) Matched Pulse-Width Filter (c) Non-Maximal Suppression

Figure 6: Intermediate stages of the Navigator's line detection algorithm. Each stage better isolates the true line points. After non-maximal suppression is done on all three cameras, each detected line point is projected onto the ground plane and transformed into a common coordinate frame

wheel camera, and the right wheel camera. By running the entire processing pipeline at 10-15 Hz, the Navigator is able to detect lines within in a 180° field of view with less than 100 ms of latency.

Lines in each of the three images are independently identified using a three-stage algorithm. Immediately upon receiving a new frame from the camera, the original color image undergoes a *color space transformation* (Section 4.3.1) that emphasizes white regions of the image. The resulting monochromatic image is then searched for regions of high intensity that match the expected width of the line using a *matched pulse-width filter* (Section 4.3.2). The output of the matched pulse-width filter is reduced to a set of candidate line points through *non-maximal suppression* (Section 4.3.3) and the local maxima from all three images are projected onto the ground plane in a common coordinate frame. Now that all the candidate line points lie in a common coordinate frame, With all of the candidate line points in a common coordinate frame, they are accumulated in an occupancy grid and *centerline extraction* (Section 4.3.4) is used to infer the structure of the lane. By modeling the entire lane instead of its individual edges, the Navigator's path planner functions equally well on planning algorithm from dashed boundary lines.

4.3.1 Color Space Transformation

Since the 2011 IGVC rules [12] specify that all course boundaries are marked by painted white lines, the first step is to remove regions of the image that have the wrong color. While images are captured in the in the *red-green-blue* (RGB) color space, this is not convenient for image processing: changes in illumination independently modify each channel in a non-intuitive manner. Before further processing, the images are converted into the *hue-saturation-value* (HSV) color space and processed by a custom whiteness filter.

Unlike the RGB color space, each axis in the HSV color space has an physical meaning: *hue* corresponds to color, a pixel, *saturation* controls the amount of color present, and *value* is brightness. Ignoring hue, the whiteness filter emphasizes white regions of the image by responding strongly to pixels that have low saturation and meet a minimum brightness threshold. Specifically, the whiteness, $W(x, y)$ of a pixel with the color $(H(x, y), S(x, y), V(x, y))$ is

$$W(x,y) = \begin{cases} 0, & \text{if } V(x,y) < T \\ 1 - S(x,y), & \text{else} \end{cases},$$

where each dimension of the color space is normalized to the range $[0, 1]$ and T is a threshold found using a

histogram of intensity values.

4.3.2 Matched Pulse-Width Filter

Even with regions of incorrect color eliminated by the whiteness filter, the Navigator could still detect false-positives on obstacles, shadows, and noise that are not part of the course boundary. Using the fixed line width, W, of three inches that is specified by the 2011 IGVC rules [12], the Navigator convolves the output of the color space transformation with a digital filter *pulse-width* filter that is *matched* to the expected width of a line in the image.

While the width of of the boundary lines is constant in real units, the observed width varies inversely with distance: lines that are near the robot appear much larger than lines that are far away. The rays between the camera center and each pixel on the image plane are intersected with the ground plane extracted from the stereo reconstruction (Section 4.2.3) to map each point in the image to its corresponding three-dimensional point on the ground [6, 7]. This point is then perturbed by 1.5 inches in each direction to simulate a line passing through this point and the four resultant points are projected back into the image. Using these four reprojected points, the height of a vertical or horizontal line is simply the distance between the corresponding pair of points.

These distances are used to construct two digital filters tuned to the appropriate widths: a row filter to detect vertical lines and a column filter to detect horizontal lines [6, 7]. Each filter consists of a centered, positive *pulse* of width w flanked by two negative *supports* of width αw. Normalized to sum to zero, each filter responds most strongly to white regions of the image with the same width as the line. The filter has a zero response to uniform noise [6]. As Figure 6b shows, the matched pulse-width filter successfully eliminates all regions of the image that do not match the width of a boundary line painted on the ground. By assuming line-width does not change along rows in the image [6], and through aggressive caching, the Navigator is capable of simultaneously processing images from all three cameras in real-time.

4.3.3 Non-Maximal Suppression

Once processed by the matched pulse-width filter, most of the false-positives have been eliminated and true positives enhanced. Before it can be added to an occupancy grid, this filter response must be quantized to a discrete set of points. By thresholding the filter response at approximately 14% of its maximum value and performing non-maximal suppression along the rows of the horizontal filter response and columns of the vertical filter response [6, 7], the Navigator reduces the filter response to a small number of points that are roughly in the middle of the line (Figure 6c), mixed with a small number of false positives.

Using its knowledge of the ground plane and the camera calibration parameters, the Navigator projects each of these local maxima into three-dimensional space [6] and transforms all of the reconstructed points into a common coordinate frame. Once in a common coordinate frame, the point clouds produced by all three cameras are merged and the statistical outlier detection filter (SOD) [11] described in Section 4.2.4 is used to eliminate false-positive maxima. By intentionally setting the filter response threshold to a low value and eliminating the false positives with the SOD filter, the Navigator is capable of reliably identifying and localizing distant lines without producing spurious detections on obstacles and noise.

4.3.4 Centerline Extraction

While the output of non-maximal suppression is sufficiently accurate to be used in global navigation, it suffers from a major flaw: the gaps in dashed lines are indistinguishable from safe regions. The Navigator's local path planner, discussed in detail in Section 4.6, avoids crossing these gaps by favoring trajectories that are near the lane's centerline. During each update, the lane's centerline is approximated using the accumulated line points in a 10×10 meter window around the Navigator's current position [6]. Using this data, the centerline is the line that is: (1) within the course boundaries and (2) maximizes distance to the observed points on the boundary lines. Unlike algorithms that individually model the left and right boundary lines, centerline extraction intelligently treats the lane as a single unit and reduces the likelihood of the Navigator inadvertently crossing the course boundary.

4.4 Calibration

Efficient perception and localization relies on fusing multiple sensors into a single model of the world. Without calibration, no matter how carefully designed, the sensors will disagree with each other. They will disagree because of errors in setup, machining tolerances, and variations in sensors. On the Navigator, there twelve sensors and the exact position, orientation, and covariance matrix of the sensors must be found in a common coordinate frame. To perform the calibration, the Navigator uses its most accurate sensor, the Hokuyo UTM-30LX laser range finder, as ground truth.

For calibrating the drive encoders and IMU Censi's [1] canonical scan matcher is used to track the Navigator's position. Combining this ground-truth position estimate with recorded encoder ticks and IMU data, a gradient descent-based non-linear optimizer is used to solve for the robot's rotation center, effective drive diameter, and sensor covariances. Using the covariance matrices found through this calibration process, the Navigator is able to use an enhanced Kalman filter to fuse the odometery, IMU, and DGPS information into a coherent accurate position estimate.

Additionally, since the cameras' positions are not known relative to laser's coordinate frame, a separate technique is used to calibrate the three stereo cameras relative to the laser rangefinder. By using a custom calibration stand faced with large black-and-white chessboard pattern the Navigator is able to simultaneously localize the object using its laser rangefinder and stereo cameras. Each sensor uses a different approach of localizing the chessboard: the laser rangefinder detects the two sharp changes in depth on either side of the stand and the camera tracks the internal corners of the chessboard. Using the calibration object as a common point, least-squares estimation is used to solve for the (x, y) displacement and yaw of each camera with respect to the laser's coordinate frame.

To calibrate the intrinsic parameters of each camera, several views of a large calibration chessboard were captured by each camera. The chessboard is detected in each image with subpixel accuracy and its corner locations are used to solve for each camera's intrinsic parameters and distortion coefficients. Once the intrinsic parameters of each camera are known, the chessboard is simultaneously localized in three-dimensional space using two or more of of the cameras. Using the estimated position of the chessboard in space, a non-linear optimizer is used to solve for the coordinate frame transformation between the two camera coordinate frames of interest. By calibrating the extrinsic parameters of all cameras relative to the left stereo camera, all of the camera relative to the laser rangefinder, all of the sensor data can be transformed into a common coordinate frame for further processing.

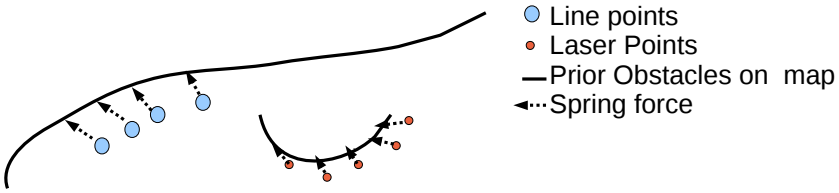


Figure 7: The Navigator uses an Iterative Closest Point (ICP) algorithm within the GMapping to align the detected line points and obstacles with the current map. At each ICP iteration, the sensor points are matched with the closest existing map obstacle with a spring (dotted arrow above). ICP aligns the data with the map to minimize total force in the springs. This minimizes the error between our prior map and current sensor data and creates a consistent world map.

4.5 Localization and Mapping

The Navigator charts its course by fusing all of its calibrated sensors into a single position estimate while making a map of its surroundings. This allows it to overcome the errors in each individual sensors and create a model of of the world that is better than any single sensor could provide. The system uses two classic probabilistic robotics algorithms for this task: an extended Kalman filter and a FAST-SLAM particle filter mapping framework, GMapping.

The extended Kalman filter (EKF) [13] is the Navigator's default localization algorithm. The filter fuses measurements from the wheel encoders, the IMU, and the GPS to create an optimal position estimate. It is optimal in the sense that it minimizes the weighted mean square errors between each sensor and a model of the robot dynamics. It works by modeling the robot's pose using a multivariate normal distribution. At each iteration of the EKF, the filter predicts the robot's most likely position and the position's covariances using the robot's differential drive motion model. It then updates the prediction with each sensor by weighting each update with sensor's calibrated covariance. This allows the Kalman filter to overcome the deficiencies in each individual sensor. Wheel encoders provide a perfect position estimate over perfectly flat smooth ground, but on real terrain they slip and go over bumps, which causes wheel rotations that do not match robot movement. The IMU and compass provide excellent instantaneous measurements of heading, angular velocity, and acceleration, but they suffer the random noise and drift in the zero point of the sensors. If measurements of either sensor are integrated over time, the errors grow without bound as they compound on each other. On the other hand, the GPS gives position estimates to within a fraction of a meter but does not provide an accurate orientation or instantaneous velocity. Luckily, by combining the sensors and modeling the vehicle dynamics, the EKF helps to eliminate these errors and makes a more accurate pose estimate.

To combine its localization estimate and laser, stereo, and line detection data, the robot uses the GMapping [5] simultaneous localization and mapping (SLAM) framework. GMapping is a FAST-SLAM based particle filter which tracks not only the most likely pose of the robot, but also the most likely world map. It represents the map as a 2d occupancy grid. Each cell of the grid is initially unknown. As the robot detects obstacles, each obstacle grid cell is marked as an obstacle and every cell in a line between the obstacle and the sensor is marked as open space. GMapping was designed to work indoors with laser scanners and ultrasound. Indoors, these sensors produce feature rich maps that allow the algorithm to match the current sensor data with the prior map using an iterative closest point (ICP) technique. This alignment is used to improve the pose estimate of the robot and to register the new data with the old. Outside however, there are much fewer obstacles, no walls, there is frequently nothing to sense. When there is no perceptual data,

the pose estimate and the maps diverge and create an inconsistent global map. Our key insight is that while there are no obstacles to sense outside, there are white lines acting like walls in the autonomous challenge. This allows the Navigator fuses its stereo data, detected lines, and laser data into a single obstacle data set which can be utilized by an open-source GMapping implementation. In the absence of perception data, the robot simply keeps track of its state using its extended Kalman filter.

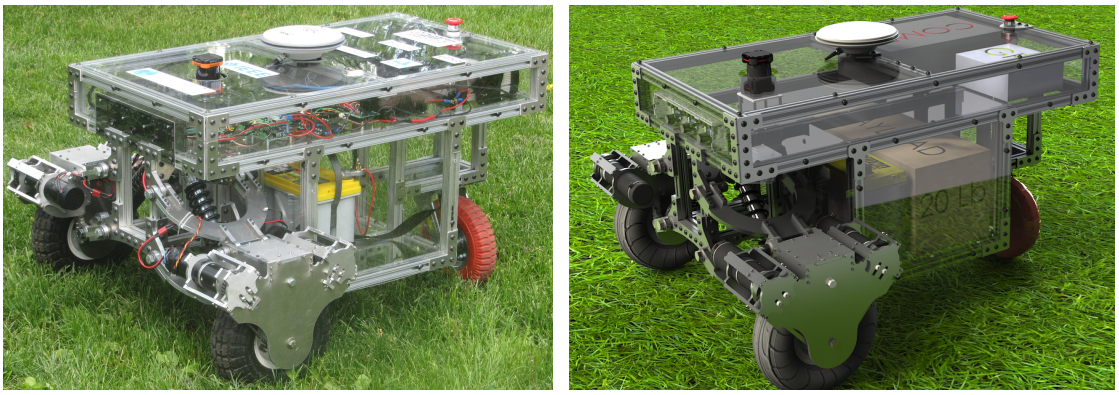
4.6 Path Planning

The Navigator's *navigation executive* takes the pose estimate from the localization and the occupancy grid from our mapping system and determines the order in which the robot should visit the goal GPS way-points. On first receiving the list of way-points, the executive converts the latitude and longitude coordinates into (x, y) displacements from the robots start position. the executive then chooses the order of the goals that minimizes the cost of the round trip between the waypoints by assuming that the robot can travel in straight line between waypoints. At each step, the executive greedily chooses the next closest goal and informs the path planner so that it can determine a detailed motion plan.

The Navigator's *path planner* is an adaptation of the standard path planner provided in ROS navigation stack [9]. This path planner receives the goal from the navigation executive and creates a coarse plan using a graph-based *global path planner*. Ignoring the robot's dynamics, the global path planner models the robot as a point that can move in any direction and inflates obstacle in the occupancy map by half the width of the robot plus a small safety factor. By treating each unoccupied cell in the occupancy grid as a node in a graph and the adjacency of cells as edges, the global path planner uses the A* search algorithm to efficiently plan a rough path through the occupancy grid. This coarse plan is then fed into a local planner, which is active throughout the robot's journey along the path.

Accepting the output of global path planner as input, the *local path planner* is responsible for directly issuing velocity commands to the Navigator's drive system. The local path planner takes into account the robot's physical size and dynamics to plan locally optimal paths and quickly reacts to new sensor data. The local planner works by tracking and planning on a 10×10 m local occupancy grid around the robot rather than the global map. It then uses the *dynamic window approach* (DWA) [3] to search for safe velocity commands within that window. The DWA works by sampling all possible velocity commands, constrained by the robot's acceleration and velocity limits, and simulates each using knowledge of the robot's dynamics. Each candidate command is scored based on how closely it follows the global path, proximity to the centerline of the course, proximity to obstacles, and the robot's speed. Paths that cause the robot to hit an obstacle or exit the course are immediately discarded and the optimal command is sent to the motor driver. The Navigator runs the local path planner at 20 Hz and samples 500 trajectories for each iteration.

Combining the global and local path planners allows the Navigator to efficiently deal with the most difficult terrain. The global path planner produces a near-optimal path given accurate knowledge of the environment, but is too computationally expensive to quickly react to new sensor data. The local is capable of quickly reacting to new sensor data, but can easily get caught on local minima without the global path planner specifying a long-term goal. This is especially apparent in courses with U-shaped center islands: the robot thrashes about because no set of commands both moves it closer to the goal and further from obstacles. With the global path planner active, the robot plans a new path that leads out of the trap. Furthermore, this planning combination allows the Navigator to deal with the dashed lines without explicitly modeling the dashes. Since the local planner heavily favors trajectories that are near the course centerline, the Navigator will equally avoid solid and broken lines.



(a) Physical Construction (b) CAD Render

Figure 8: Comparison between the final version of the Navigator and a CAD render.

Description	**Retail Cost**	**Team Cost**
Hokuyo UTM-30LX Laser Rangefinder	$5590	$0
Novatel ProPack-V3 Differential GPS	$8000	$0
Computer (Intel Core i7, 4 GB RAM, 60 GB SSD)	$900	$900
OmniStar Subscription (90 days)	$850	$0
80/20 Chassis Hardware	$577	$444
IG52-04 24VDC 285 RPM Gear Motor with Encoder (×4)	$560	$560
Aluminum Sheet Stock	$450	$450
Optima Yellow Top Batteries (×4)	$420	$130
Water-Jet Cut Polycarbonate Casing	$350	$350
Simple H-Bridge (×4)	$320	$320
Misc Hardware	$300	$300
Playstation Eye Camera (×5)	$185	$125
Sparkfun 9-DoF IMU	$125	$125
Total	$18,627	$4,708

Table 2: Approximate cost of each component on the Rutgers Navigator.

5 Results

In one year, the Rutgers Navigator has developed from a few back-of-the-envelope sketches to a real life autonomous outdoor vehicle whose design model is almost indistinguishable from reality. All of the team's original design goals have been met and exceeded. The robot travels at a continuous 2.5m/s, easily climbs the required 15 percent grade incline, and provides a smooth, stable ride to its electronics with its custom suspension. The robot has an top of the line stereo system built from scratch by the team using commodity PS3 cameras. This custom vision system cost only $200 and provides obstacle detection along with robust line detection. The robot navigates outside, adapts to new obstacles it it sees them, and reaches its goals to within a 0.5 meter accuracy. Furthermore, the team was able to radically expand our expected budget from a little over $3500 to over $18,000 and successfully integrate new hardware as the funds became available.

6 Acknowledgements

The Rutgers IGVC team would like to extend our gratitude to everyone that has helped us along the way this year. We could not have finished the robot without the generous support of our sponsors: Dr. Stuart Shalat of the Advanced Robotics Environmental and Assessment Lab, Optima Batteries, Novatel, Omnistar, Github, IEEE, 80/20 Inc., the Knotts Company, the Rutgers Engineering Governing Council, Dr. Michael Littman of the Rutgers Laboratory for Real-life Reinforcement learning (RL3), and Dr. Dimitris Metaxas of the Rutgers Computational Biomedicine Imaging and Modeling Center (CBIM). In addition to our sponsors, we would like to thank Dr. Kristin Dana for answering all of our questions and Dr. Predrag Spasojevic for serving as the Rutgers IEEE Student Branch's faculty adviser. Finally, we would like to thank Joe Lippencott, Steve Orbine, John Scafidi, and the departments of Electrical, Industrial, and Mechanical engineering for their continued support.

References

[1] Andrea Censi. An ICP variant using a point-to-line metric. *2008 IEEE International Conference on Robotics and Automation*, pages 19–25, May 2008.

[2] M.A. Fischler and R.C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[3] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance, 1997.

[4] P. Fua. A parallel stereo algorithm that produces dense depth maps and preserves image features. volume 6, pages 35–49. Springer, 1993.

[5] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1):34–46, February 2007.

[6] Albert Huang. *Lane Estimation for Autonomous Vehicles Using Vision and LIDAR*. PhD thesis, Massachusetts Institute of Technology, 2010.

[7] A.S. Huang, D. Moore, M. Antone, E. Olson, and S. Teller. Multi-sensor lane finding in urban road networks. *Proceedings of Robotics: Science and Systems, Zurich, Switzerland*, 2008.

[8] K. Konolige. Small Vision Systems: Hardware and Implementation. In *Machine Vision and Applications*, volume 8, pages 203–212. MIT Press, 1998.

[9] Eitan Marder-eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The Office Marathon : Robust Navigation in an Indoor Office Environment. *System*, 1998.

[10] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. ROS: an open-source Robot Operating System. In *ICRA Workshop on Open Source Software*, 2009.

[11] R.B. Rusu, Z.C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[12] B. Theisen. The 19th Annual Intelligent Ground Vehicle Competition, 2011.

[13] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probablistic Robotics*. MIT Press, Cambridge, MA, 2006.