Team CIMAR's NaviGATOR: An Unmanned Ground Vehicle for the 2005 DARPA Grand Challenge

Carl D. Crane III[1], David G. Armstrong II[1], Robert Touchton[1], Tom Galluzzo[1], Sanjay Solanki[1], Jaesang Lee[1], Daniel Kent[1], Maryum Ahmed[1], Roberto Montane[1], Shannon Ridgeway[1], Steve Velat[1], Greg Garcia[1], Michael Griffis[2], Sarah Gray[3], John Washburn[4], and Gregory Routson[4]

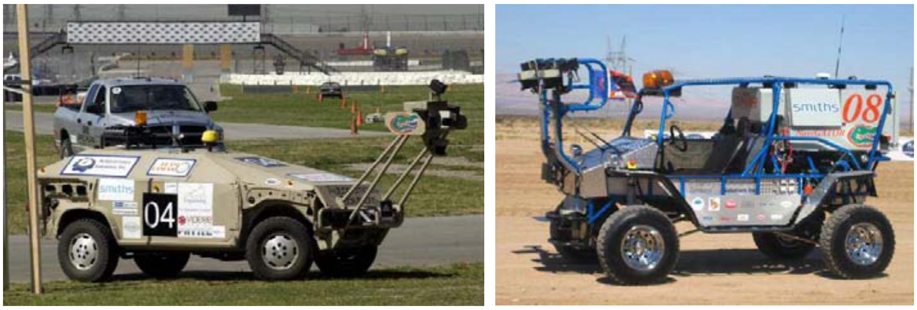
[1] University of Florida, Center for Intelligent Machines and Robotics, Gainesville, Florida
[2] The Eigenpoint Company, High Springs, Florida
[3] Autonomous Solutions, Inc. Young Ward, Utah
[4] Smiths Aerospace, LLC Grand Rapids, Michigan

Summary. This paper describes the development of an autonomous vehicle system that participated in the 2005 DARPA Grand Challenge event. After a brief description of the event, the architecture, based on version 3.2 of the Department of Defense Joint Architecture for Unmanned Systems (JAUS), and the design of the system are presented in detail. In particular, the "smart sensor" concept is introduced which provided a standardized means for each sensor to present data for rapid integration and arbitration. Information about the vehicle design, system localization, perception sensors, and the dynamic planning algorithms that were used is then presented in detail. Subsequently, testing results and performance results are presented.

Keywords: DARPA Grand Challenge, autonomous navigation, path planning, sensor fusion, world modeling, localization, JAUS.

10.1 Introduction

The DARPA Grand Challenge is widely recognized as the largest and most cutting-edge robotics event in the world, offering groups of highly motivated scientists and engineers across the U.S. an opportunity to innovate in developing state-of-the-art autonomous vehicle technologies with significant military and commercial applications. The U.S. Congress has tasked the military with making nearly one-third of all operational ground vehicles unmanned by 2015 and The DARPA Grand Challenge is one in a number of efforts to accelerate this effort. The intent of the event is to spur participation in robotics by groups of engineers and scientists outside the normal military procurement channels including leaders in collegiate research, military development, and industry research.



(a) Team CIMAR's 2004 DARPA Grand Challenge Entry

(b) Team CIMAR's 2005 DARPA Grand Challenge Entry

Fig. 10.1. The NaviGATOR

Team CIMAR is a collaborative effort of the University of Florida Center for Intelligent Machines and Robotics (CIMAR), The Eigenpoint Company of High Springs, Florida, and Autonomous Solutions of Young Ward, Utah. The goal of Team CIMAR is to develop cutting-edge autonomous vehicle systems and solutions with wide ranging market applications, such as intelligent transportation systems and autonomous systems for force protection. Team CIMAR focused on proving their solutions on an international level by participating in both the 2004 and the 2005 DARPA Grand Challenges.

In 2003, Team CIMAR was one of 25 teams selected from over 100 applicants nationwide to participate in the inaugural event. Team CIMAR was also one of the 15 teams that successfully qualified for and participated in the inaugural event in March 2004; and finished in eighth place. Team CIMAR was accepted into the inaugural DARPA Grand Challenge in late December 2003 and fielded a top-10 vehicle less than three months later. The team learned a tremendous amount from the initial event and used that experience to develop a highly advanced new system to qualify for the second Grand Challenge in 2005 (see Figure 10.1).

10.2 System Architecture and Design

The system architecture that was implemented was based on the Joint Architecture for Unmanned Systems (JAUS) Reference Architecture, version 3.2 (JAUS, 2005). JAUS defines a set of reusable components and their interfaces. The system architecture was formulated using existing JAUS-specified components wherever possible along with a JAUS-compliant intercomponent messaging infrastructure. Tasks for which there are no components specified in JAUS required the creation of so-called "Experimental" components using "User-defined" messages. This approach is endorsed by the JAUS Working Group as the best way to extend and evolve the JAUS specifications.

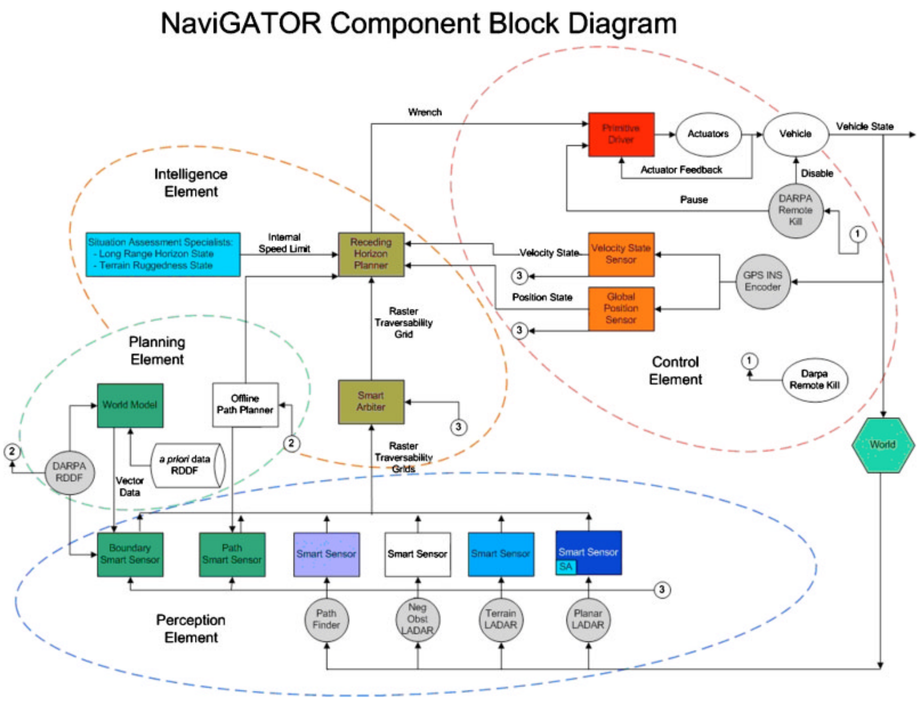


Fig. 10.2. The NaviGATOR's JAUS-compliant architecture

10.2.1 High-Level Architecture

At the highest level, the architecture consists of four fundamental elements, which are depicted in Figure 10.2:

- Planning Element: The components that act as a repository for *a priori* data, including known roads, trails, or obstacles, as well as acceptable vehicle workspace boundaries. Additionally, these components perform offline planning based on that data.
- Control Element: The components that perform closed-loop control in order to keep the vehicle on a specified path.
- Perception Element: The components that perform the sensing tasks required to locate obstacles and to evaluate the smoothness of terrain.
- Intelligence Element: The components that act to determine the "best" path segment to be driven based on the sensed information.

10.2.2 Smart Sensor Concept

The Smart Sensor concept unifies the formatting and distribution of perception data among the components that produce and/or consume it. First, a common data structure, dubbed the Traversability Grid, was devised for use by all Smart

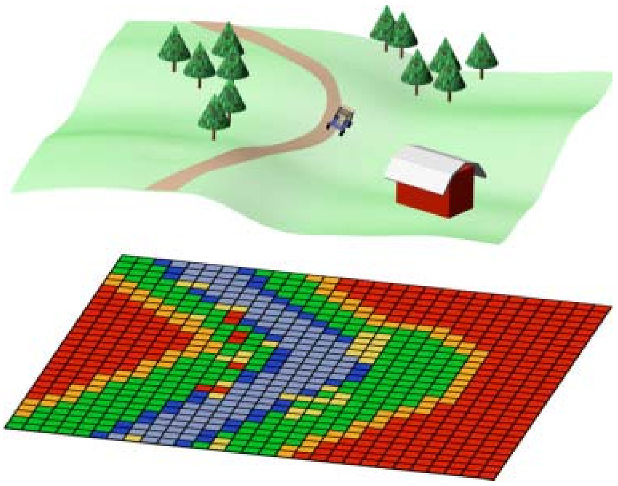


Fig. 10.3. Traversability grid portrayal

Sensors, the Smart Arbiter, and the Reactive Driver. Figure 10.3 shows the world as a human sees it in the upper level, while the lower level shows the Grid representation based on the fusion of sensor information. This grid was sufficiently specified to enable developers to work independently and for the Smart Arbiter to use the same approach for processing input grids, no matter how many there were at any instant in time.

The basis of the Smart Sensor architecture is the idea that each sensor processes its data independently of the system and provides a logically redundant interface to the other components within the system. This allows developers to create their technologies independent of one another and process their data as best fits their system. The sensor can then be integrated into the system with minimal effort to create a robust perception system. The primary benefit of this approach is its flexibility, in effect, decoupling the development and integration efforts of the various component researchers. Its primary drawback is that it prevents the ability of one sensor component to take advantage of the results of another sensor when translating its raw input data into traversability findings.

The Traversability Grid concept is based on the well-understood notion of an Occupancy Grid, which is often attributed to Alberto Elfes of Carnegie Mellon University (Elfes, 1989). His work defines an Occupancy Grid as "a probabilistic tesselated representation of spatial information." Sebastian Thrun provides an excellent treatise on how this paradigm has matured over the past 20 years (Thrun, 2003). The expansion of the Occupancy Grid into a Traversability Grid has emerged in recent years in an attempt to expand the applicability and utility of this fundamental concept (Seraji, 2003), (Ye & Borenstein, 2004). The primary contribution of the Traversability Grid implementation devised for the NaviGATOR is its focus on representing degrees of traversability including terrain conditions and obstacles (from absolutely blocked to unobstructed level pavement) while preserving real-time performance of 20 Hz.

The Traversability Grid design is 121 rows (0 – 120) by 121 columns (0 – 120), with each grid cell representing a half-meter by half-meter area. The center cell, at location (60, 60), represents the vehicle's reported position. The sensor results are oriented in the global frame of reference so that true north is always aligned vertically in the grid. In this fashion, a 60m by 60m grid is produced that is able to accept data at least 30m ahead of the vehicle and store data at least 30m behind it. To support proper treatment of the vehicle's position and orientation, every Smart Sensor component is responsible for establishing a near-real-time latitude/longitude and heading (yaw) feed from the GPOS component.

The scoring of each cell is based on mapping the sensor's assessment of the traversability of that cell into a range from 2 to 12, where 2 means that the cell is absolutely impassable, 12 means the cell represents an absolutely desirable, easily traversed surface, and 7 means that the sensor has no evidence that the traversability of that cell is particularly good or bad. Certain other values are reserved for use as follows: 0 → "out-of-bounds," 1→ "value unchanged," 13→ "failed/error," 14→"unknown," and 15→ "vehicle location." These discrete values have been color-coded to help humans visualize the contents of a given Traversability Grid, from red (2) to gray (7) to green (12).

All of these characteristics are the same for every Smart Sensor, making seamless integration possible, with no predetermined number of sensors. The grids are sent to the Smart Arbiter, which is responsible for fusing the data. The arbiter then sends a grid with all the same characteristics to the Reactive Driver, which uses it to dynamically compute the desired vehicle speed and steering.

The messaging concept for marshalling grid cell data from sensors to the arbiter and from the arbiter to the reactive driver is to send an entire traversability grid as often as the downstream component has requested it (typically at 20 Hz). In order to properly align a given sensor's output with that of the other sensors, the message must also provide the latitude and longitude of the center cell (i.e., vehicle position at the instant the message and its cell values were determined). An alternative approach for data marshalling was considered in which only those cells that had changed since the last message were packaged into the message. Thus, for each scan or iteration, the sending component would determine which cells in the grid have new values and pack the row, column, and value of that cell into the current message. This technique greatly reduces the network traffic and message-handling load for nominal cases (i.e., cases in which most cells remain the same from one iteration to the next). However, after much experimentation in both the lab and the field, this technique was not used due to concerns that a failure to receive and apply a changed cells message would corrupt the grid and potentially lead to inappropriate decisions, while the performance achieved when sending the entire grid in each message never became an issue (our concern about the ability of the Smart Sensor computers, or the onboard network, to process hundreds of full-grid messages per second did not manifest itself in the field).

In order to aid in the understanding, tuning, and validation of the Traversability Grids being produced, a Smart Sensor Visualizer component was developed. Used primarily for testing, the SSV can be pointed at any of the Smart Sensors, the Smart Arbiter, or the Reactive Driver and it will display the color-coded Traversability Grid, along with the associated vehicle position, heading, and speed. The refresh rate of the images is adjustable from real time (e.g., 20 Hz) down to periodic snapshots (e.g., 1 s interval).

10.2.3 Concept of Operation

The most daunting task of all was integrating these components such that an overall mission could be accomplished. Figure 10.4 portrays schematically how the key components work together to control the vehicle. Figure 10.4 also shows how the Traversability Grid concept enables the various Smart Sensors to deliver grids to the Smart Arbiter, which fuses them and delivers a single grid to the Reactive Driver. Prior to beginning a given mission, the *a priori* Planner builds the initial path, which it stores in a Path File as a series of global positioning system (GPS) waypoints. Once the mission is begun, the Reactive Driver sequentially guides the vehicle to each waypoint in the Path File via the Primitive Driver. Meanwhile, the various Smart Sensors begin their search for obstacles and/or

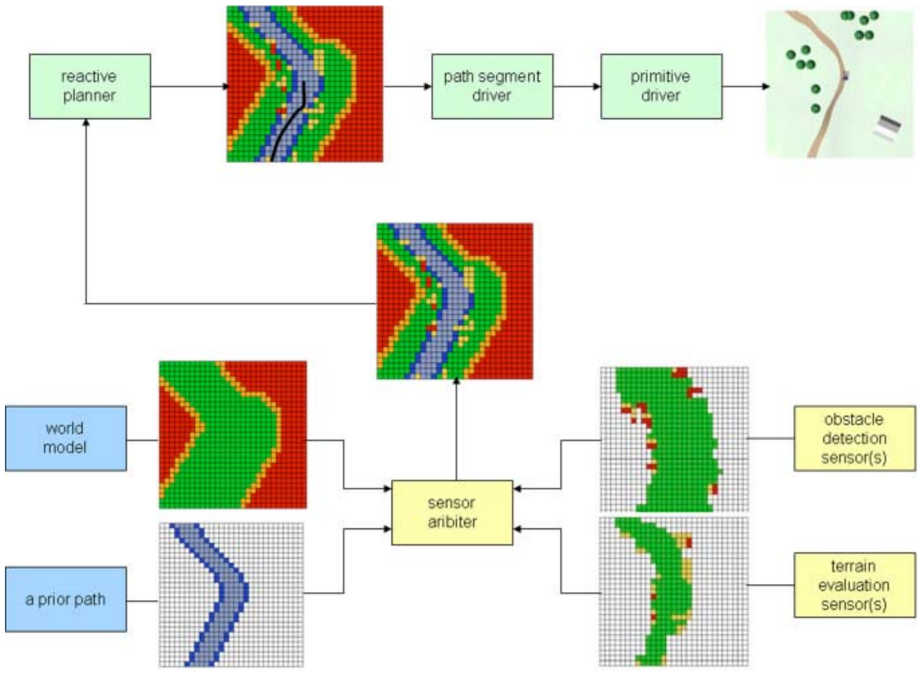


Fig. 10.4. Operational schematic (including traversability grid propagation)

smooth surfaces and feed their findings to the Smart Arbiter. The Smart Arbiter performs its data fusion task and sends the results to the Reactive Driver. The Reactive Driver looks for interferences or opportunities based on the feed from the Smart Arbiter and alters its command to the Primitive Driver accordingly. Finally, the goal is to perform this sequence iteratively on a subsecond cycle time (10 to 60 Hz), depending on the component, with 20 Hz as the default operational rate.

10.3 Vehicle Design

The NaviGATOR's base platform is an all terrain vehicle custom built to Team CIMAR's specifications. The frame is made of mild steel roll bar with an open design. It has 9" Currie axles, Bilstein Shocks, hydraulic steering, and front and rear disk brakes with an emergency brake to the rear. It has a 150 HP Transverse Honda engine/transaxle mounted longitudinally, with locked transaxle that drives front and rear Detroit Locker differentials (4 wheel drive, guaranteed to get power to the road). The vehicle was chosen for its versatility, mobility, openness, and ease of development (see Figure 10.5).

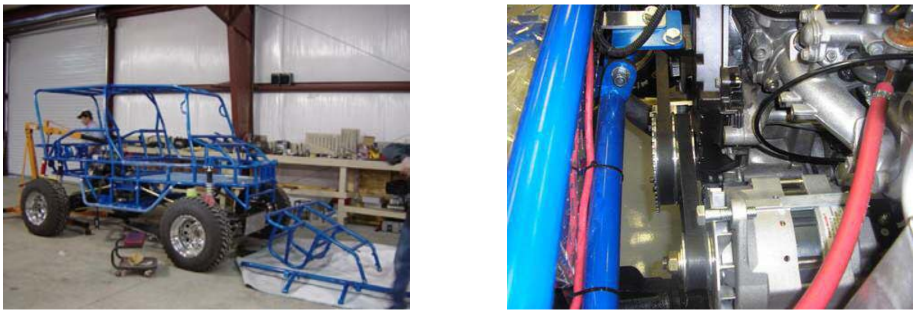

(a) (b)

Fig. 10.5. Base mobility platform

The power system consists of two, independent 140A, 28V alternator systems (Figure 10.5a). Each alternator drives a 2400W continuous, 4800W peak inverter and is backed up by 4 deep cell batteries. Each alternator feeds one of two automatic transfer switches (ATS). The output of one ATS drives the computers and electronics while the other drives the actuators and a 3/4 Ton (approx. 1kW cooling) air conditioner. Should either alternator/battery system fail the entire load automatically switches to the other alternator/battery system. Total system power requirement is approximately 2200W, so the power system is totally redundant.

The system sensors are mounted on a rack that is specifically designed for their configuration and placement on the front of the vehicle (see Figure 10.6). These

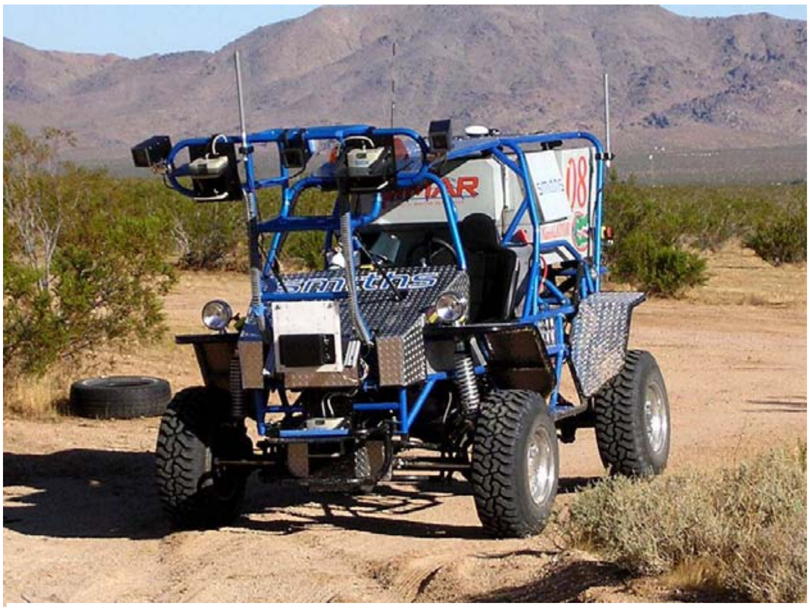


Fig. 10.6. View of sensor cage

sensors include a camera that finds the smoothest path in a scene. Equipped with an automatic iris and housed in a waterproof and dust proof protective enclosure, the camera looks through a face that is made of lexan and covered with a polarizing scratch-resistant film. Also mounted on the sensor cage are two SICK LADARs that scan the ground ahead of the vehicle for terrain slope estimation; one tuned for negative obstacle detection and the other for smooth terrain detection. Also, an additional SICK LADAR aimed parallel to the ground plane is mounted on the front of the vehicle at bumper level for planar obstacle detection. Additional sensors were mounted on the vehicle for experimental purposes, but were not activated for the Darpa Grand Challenge (DGC) event. Each sensor system is described in detail later in this paper.

The computing system requirements consists of high-level computation needs, system command implementation, and system sensing with health and fault monitoring. The high level computational needs are met in the deployed system via the utilization of eight single-processor computing nodes targeted at individual computational needs. The decision to compartmentalize individual processes is driven by the developmental nature of the system. A communications protocol is implemented to allow interprocess communication.

The individual computing node hardware architecture was selected based on the subjective evaluation of commercial off-the-shelf hardware. Evaluation criteria were centered on performance and power consumption. The deployed system maintains a homogenous hardware solution with respect to the motherboard, random access memory (RAM), enclosure, and system storage. The AMD K8 64

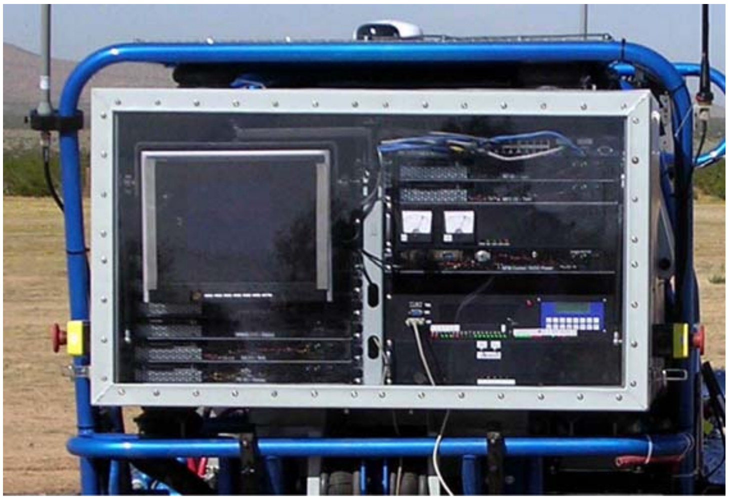


Fig. 10.7. Computer and electronics housing

bit microprocessor family was selected based on power consumption measurement and performance to allow tailoring based on performance requirements with the objective of power requirement reduction. Currently, three processor speeds are deployed: 2.0 GHz, 2.2 GHz, and 2.4 GHz. The processors are hosted in off-the-shelf motherboards and utilize solid-state flash cards for booting and long-term storage. Each processing node is equipped with 512 to 1028 MB of RAM. JAUS communication is effected through the built-in Ethernet controller located on the motherboard. Several nodes host PCI cards for data i/o. Each node is housed in a standard 1-U enclosure. The operating system deployed is based on the 2.6 Linux kernel. System maintenance and reliability are expected to be adequate due to the homogenous and modular nature of the compute nodes. Back-up computational nodes are on hand for additional requirements and replacement. All computing systems and electronics are housed in a NEMA 4 enclosure mounted in the rear of the vehicle (see Figure 10.7).

10.4 Route Pre-planning

The DARPA Grand Challenge posed an interesting planning problem given that the route could be up to 175 miles in length and run anywhere between Barstow, California and Las Vegas, Nevada. On the day of the event, DARPA supplied a Route Data Definition File (RDDF) with waypoint coordinates, corridor segment width, and velocity data. In order to process the *a priori* environment data and generate a route through DARPA's waypoint file, Team CIMAR used Mobius, an easy to use graphical user interface developed by Autonomous Solutions Inc. for controlling and monitoring unmanned vehicles. Mobius was used to plan the initial path for the NaviGATOR in both the National Qualification Event and the final Grand Challenge Event.

The route pre-planning is done in three steps: Generate corridor data, import and optimize the DARPA path, and modify path speeds. A World Model component generates the corridor data by parsing DARPA's RDDF and clipping all other environment information with the corridor such that only elevation data inside the corridor are used in the planning process (see Figure 10.8). The RDDF corridor (now transformed into an ESRI shapefile) is then imported into Mobius and displayed to the operator for verification.

In the next step, Mobius imports the original RDDF file for use in path generation. Maximum velocities are assigned to each path segment based on the DARPA assigned velocities at each waypoint. From here, the path is optimized using the NaviGATOR's kinematics constraints and a desired maximum deviation from the initial path. The resultant path is a smooth drivable path, from the start node to the finish node, that stays inside the RDDF corridor generated specifically for the NaviGATOR (see Figure 10.9). Mobius is then used to make minor path modifications where necessary to create a more desirable path.

The final step of the pre-planning process is to modify path velocities based on *a priori* environment data and velocity constraints of the NaviGATOR itself. Sections of the path are selected and reassigned velocities. Mobius assigns the minimum of the newly desired velocity and the RDDF-assigned velocity to the sections in order to ensure that the RDDF-assigned velocities are never exceeded. During the DARPA events, the maximum controlled velocity of the NaviGATOR

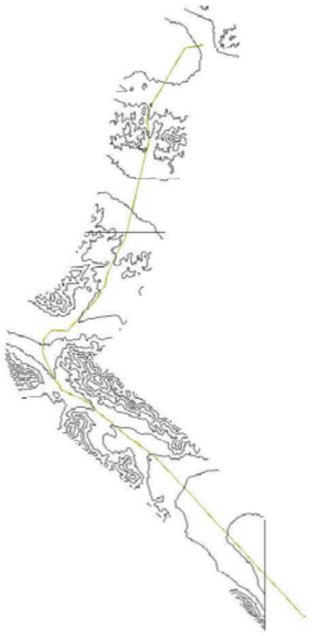


Fig. 10.8. RDDF corridor (parsed with elevation data)

was 25 miles per hour so, in the first pass, the entire path was set to a conservative 18 mph except in path segments where the RDDF speed limit was lower. From there, the path is inspected from start to finish and velocities are increased or decreased based on changes in curvature of the path, open environment (dry lake beds), elevation changes, and known hazards in the path (e.g., over/under passes). After all velocity changes are made, the time required to complete the entire path can be calculated. For the race, it was estimated that it would take the NaviGATOR approximately 8 hours and 45 minutes to complete the course. Finally, the path is saved as a comma-separated Path File and transferred to the NaviGATOR for autonomous navigation.

10.5 Localization

The NaviGATOR determines its geolocation by filtering and fusing a combination of sensor data. The processing of all navigation data is done by a Smiths Aerospace North-finding Module (NFM), which is an inertial navigation system. This module maintains Kalman filter estimates of the vehicle's global position and orientation, as well as linear and angular velocities. It fuses internal accelerometer and gyroscope data, with data from an external NMEA GPS, and external odometer. The GPS signal provided to the NFM comes from one of the two onboard sensors. These include a NavCom Technologies Starfire 2050, and a Garmin WAAS Enabled GPS 16. An onboard computer simultaneously

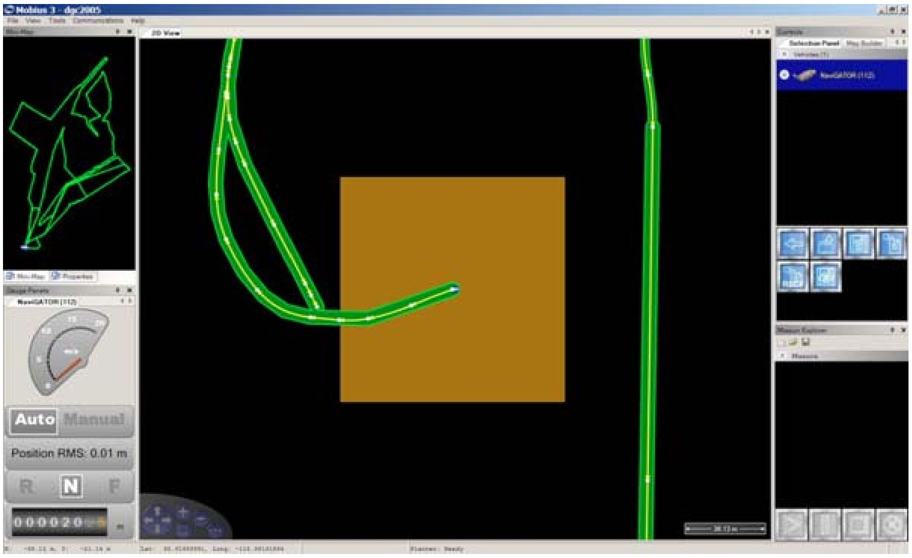


Fig. 10.9. Mobius screen shot with path optimized for the NaviGATOR. The race RDDF is shown in the upper left corner and the start/finish area is centered on the screen.

parses data from the two GPS units and routes the best-determined signal to the NFM. This is done to maintain valid information to the NFM at times when only one sensor is tracking GPS satellites. During valid tracking, the precision of the NavCom data is better than the Garmin, and thus the system is biased to always use the NavCom when possible.

In the event that both units lose track of satellites, as seen during GPS outages, which occurs when the vehicle is in a tunnel, the NFM will maintain localization estimates based on inertial and odometry data. This allows the vehicle to continue on course for a period of time; however, the solution will gradually drift and the accuracy of the position system will steadily decrease as long as the GPS outage continues. After a distance of a few hundred meters, the error in the system will build up to the point where the vehicle can no longer continue on course with confidence. At this point, the vehicle will stop and wait for a GPS reacquisition. Once the GPS units begin tracking satellites and provide a valid solution, the system corrects for any off-course drift and continues autonomous operation.

The Smith's NFM is programmed to robustly detect and respond to a wide range of sensor errors or faults. The known faults of both GPS systems, which generate invalid data, are automatically rejected by the module, and do not impact the performance of the system, as long as the faults do not persist for an extended period of time. If they do persist, then the NFM will indicate to a control computer what the problem is, and the system can correct it accordingly. The same is true for any odometer encoder error, or inertial sensor errors. The NFM will automatically respond to the faults and relay the relevant information to control computers, so the system can decide the best course of action to correct the problem.

10.6 Perception

This section of the paper discusses how the NaviGATOR collects, processes and combines sensor data. Each of the sensor components is presented, organized by type: LADAR, camera, or "pseudo" (a component that produces an output as if it were a sensor, but based on data from a file or database). Finally, the Smart Arbiter sensor fusion component is discussed.

10.6.1 LADAR-Based Smart Sensors

There are three Smart Sensors that rely on LADAR range data to produce their results: the Terrain Smart Sensor (TSS), the Negative Obstacle Smart Sensor (NOSS) and the Planar LADAR Smart Sensor (PLSS). All three components use the LMS291-S05 from Sick Inc. for range measurement. The TSS will be described in detail and then the remaining two will be discussed only in terms of how they are different than the TSS.

A laser range finder operates on the principle of time of flight. The sensor emits an eye-safe infrared laser beam in a single-line sweep of either 180° or 100°, detects the returns at each point of resolution, and then computes single line range image. Although three range resolutions are possible (1°, 0.5°, or 0.25°) the resolution of 0.25° can only be achieved with a 100° range scan. The accuracy of the laser measurement is +/- 50 mm for a range of 1 to 20 m, while its maximum range is ~80 m. A high-speed serial interface card is used to achieve the needed high-speed baud rate of 500 kB.

10.6.1.1 Terrain Smart Sensor

The sensor is mounted facing forward at an angle of 6° toward the ground. For the implementation of the TSS, the 100° range with a 0.25° resolution is used. With this configuration and for nominal conditions (flat ground surface, vehicle level), the laser scans at a distance of ~20 m ahead of the vehicle and ~32 m wide. The TSS converts the range data reported by the laser in polar coordinates into Cartesian coordinates local to the sensor, with the Z axis vertically downward and the X axis in the direction of vehicle travel. The height for each data point (Z component) is computed based on the known geometry of the system and the range distance being reported by the sensor. The data is then transformed into the global coordinate system required by the Traversability Grid, where the origin is the centerline of the vehicle at ground level below the rear axle (i.e., the projection of the GPS antenna onto the ground), based on the instantaneous roll, pitch, and yaw of the vehicle.

Each cell in the Traversability Grid is evaluated individually and classified for its traversability value. The criteria used for classification are:

1. The mean elevation (height) of the data point(s) within the cell.
2. The slope of the best fitting plane through the data points in each cell.
3. The variance of the elevation of the data points within the cell.

The first criterion is a measure of the mean height of a given cell with respect to the vehicle plane. Keep in mind that positive obstacles are reported as negative elevations since the Z-axis points down. The mean height is given as

$$\mu = \frac{\Sigma Z_i}{n}, \tag{10.1}$$

where μ is the mean height, ΣZ_i is the sum of the elevation of the data points within the cell, and n is the number of data points.

The second criterion is a measure of the slope of the data points. The equation for the best fitting plane, derived using the least squares solution technique, is given as

$$S_{\text{optimum}} = (G^T G)^{-1} G^T b, \tag{10.2}$$

where:

S_{optimum} is the vector perpendicular to the best fitting plane,

G is an $n \times 3$ matrix given by

$$G = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ - & - & - \\ x_n & y_n & z_n \end{bmatrix}, \tag{10.3}$$

b is a vector of length "n" given by

$$b = \begin{bmatrix} -D_{01} \\ -D_{02} \\ - \\ -D_{0n} \end{bmatrix}. \tag{10.4}$$

Assuming that D_{0i} is equal to 1, the above equation is used to find $S_{optimum}$ for the data points within each cell. Once the vector perpendicular to the best-fitting plane is known, the slope of this plane in the "x" and "y" directions can be computed. Chapter 5 of Solanki (2003) provides a thorough proof of this technique for finding the perpendicular to a plane.

The variance of the data points within each cell is computed as

$$\text{Variance} = \frac{\Sigma(Z_i - \mu)^2}{n}. \tag{10.5}$$

A traversability value between 2 and 12 is assigned to each cell, depending on the severity values of the mean height, slope, and variance information. A cell must contain a minimum of three data points or else that cell is flagged as unknown. This also helps in eliminating noise. Each of the parameters is individually mapped to a corresponding traversability value for a given cell. This mapping is entirely empirical and non-linear. A weighted average of these three resulting traversability values is used to assign the final traversability value.

10.6.1.2 Negative Obstacle Smart Sensor

The NOSS was specifically implemented to detect negative obstacles (although it can also provide information on positive obstacles and surface smoothness like the TSS). The sensor is configured like the TSS, but at an angle of 12° toward the ground. With this configuration and for nominal conditions, the laser scans the ground at a distance of ~10 m ahead of the vehicle. To detect a negative obstacle, the component analyzes the cases where it receives a range value greater than would be expected for level ground. In such cases, the cell where one would expect to receive a hit is found by assuming a perfectly horizontal imaginary plane. As shown in Figure 10.10, this cell is found by solving for the intersection of the imaginary horizontal plane and the line formed by the laser beam. A

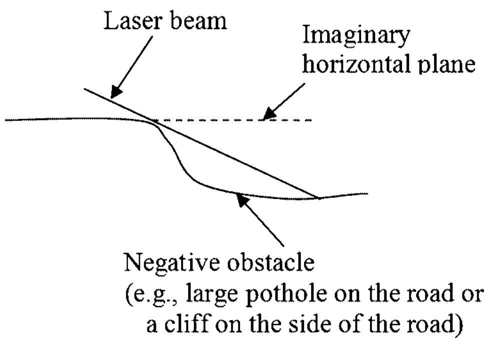

Laser beam
Imaginary horizontal plane
Negative obstacle (e.g., large pothole on the road or a cliff on the side of the road)

Fig. 10.10. NOSS implementation (side view)

traversability value is assigned to that cell based on the value of the range distance and other configurable parameters. Thus, a negative obstacle is reported for any cell whose associated range data are greater than that expected for an assumed horizontal surface. The remaining cells for which range value data is received are evaluated on a basis similar to the TSS.

10.6.1.3 Planar LADAR Smart Sensor

The sensor is mounted 0.6 m above the ground, scanning in a plane horizontal to the ground. Accordingly, the PLSS only identifies positive obstacles and renders no opinion regarding the smoothness or traversability of areas where no positive obstacle is reported. For the PLSS, the 180° range with a 0.5° resolution is used. The range data from the laser is converted into the Global coordinate system and the cell from which each hit is received is identified. Accordingly, the "number of hits" in that cell is incremented by one and then, for all the cells between the hit cell and the sensor, the "number of missed hits" is incremented by one. Bresenham's line algorithm is used to efficiently determine the indices of the intervening cells.

A traversability value between 2 and 7 is assigned to each cell based on the total number of hits and misses accumulated for that cell. The mapping algorithm first computes a score, which is the difference between the total number of hits and a discounted number of misses in a cell (a discount weight of 1/6 was used for the event). This score is then mapped to a traversability value using an exponential scale of 2. For example, a score of 2 or below is mapped to a traversability value of "7," a score of 4 and below is mapped to a "6," and so on, with a score greater than 32 mapped to a "2." The discounting of missed hits provides conservatism in identifying obstacles, but does allow gradual recovery from false positives (e.g., ground effects) and moving obstacles.

10.6.1.4 Field Testing

The parameters of the algorithm that affect the output of the component are placed in a configuration file so as to enable rapid testing and tuning of those parameters. Examples of these tunable parameters for the TSS and NOSS components are the threshold values for the slope, variance, and mean height for mapping to a particular traversability value. For the PLSS, these parameters include the relative importance of the number of hits versus misses in a given cell, and a weighting factor to control how much any one scan affects the final output.

By making these parameters easy to adjust, it was possible to empirically tune and validate these components for a wide variety of surroundings such as a steep slopes, cliffs, rugged roads, small bushes, and large obstacles, like cars. This approach also helped to configure each component to work in the most optimum way across all the different surroundings. Finally, it helped in deciding on the amount of data/time the component required to build confidence about an obstacle or when an obstacle that was detected earlier has now disappeared from view (e.g., a moving obstacle).

10.6.2 Camera-Based Smart Sensor

The Pathfinder Smart Sensor (PFSS) consists of a single color camera mounted in the sensor cage and aimed at the terrain in front of the vehicle. Its purpose is to assess the area in the cameras scene for terrain that is similar to that on which the vehicle is currently traveling, and then translate that scene information into traversability information. The PFSS component uses a high-speed frame-grabber to store camera images at 30 Hertz.

Note that the primary feature used for analytical processing is the red, green and blue (RGB) color space. This is the standard representation in the world of computers and digital cameras, and is therefore often a natural choice for color representation. Also, RGB is the standard output from a CCD-camera. Since roads typically have a different color than nondrivable terrain, color is a highly relevant feature for segmentation. The following paragraphs describe the scene assessment procedure applied to each image for rendering the Traversability Grid that is sent to the Smart Arbiter.

10.6.2.1 Preprocess Image

To reduce the computational expense of processing large images, the dimensions of the scene are reduced from the original digital input of 720×480 pixels to a 320×240 reduced image. Then, the image is further preprocessed to eliminate the portion of the scene that most likely corresponds to the sky. The segmentation of the image is based simply on the physical location within the scene (tuned based on field testing), adjusted by the instantaneous vehicle pitch. This very simplistic approach is viable because the consequences of inadvertently eliminating ground are minimal due to the fact that ground areas near the horizon will likely be beyond the 30 m planning distance of the system. The motivation for this step in the procedure is that the sky portion of the image hinders the classification procedure in two ways. First, we considered that the sky portion slows down the image processing speed by spending resources evaluating pixels that could never be drivable by a ground vehicle. Second, there could be situations where parts of the sky image could be misclassified as road.

10.6.2.2 Produce Training and Background Data Sets

Next, a 100×80 sub-image is used to define the drivable area, and two 35×50 sub-images are used to define the background. The drivable sub-image is placed in the bottom-center of the image, while the background sub-images are placed at the middle-right and middle-left of the image, which is normally where the background area will be found, based on experience (Lee, 2004) (see Figure 10.11). When the vehicle turns, the background area that is in the direction of the turn will be reclassified as a drivable area. In this case, that background area information is treated as road area by the classification algorithm.

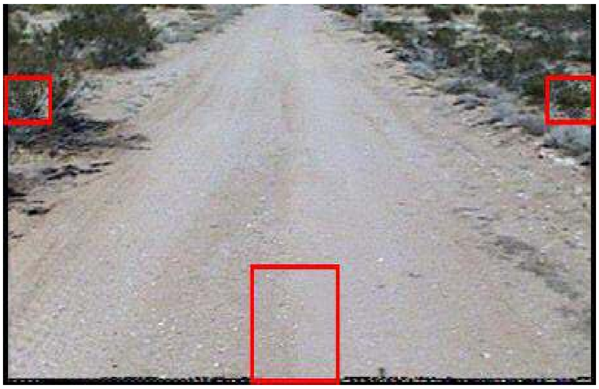


Fig. 10.11. Scene segmentation scheme

10.6.2.3 Apply Classification Algorithm

A Bayesian decision theory approach was selected for use, as this is a fundamental statistical approach to the problem of pattern classification associated with applications such as this. It makes the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known. The basic idea underlying Bayesian decision theory is very simple. However, this is the optimal decision theory under Gaussian distribution assumption (Morris, 1997).

The decision boundary that was used is given by

$$\frac{1}{(2\pi)^{d/2}|\Sigma_1|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)\right] = \frac{1}{(2\pi)^{d/2}|\Sigma_2|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)\right], \tag{10.6}$$

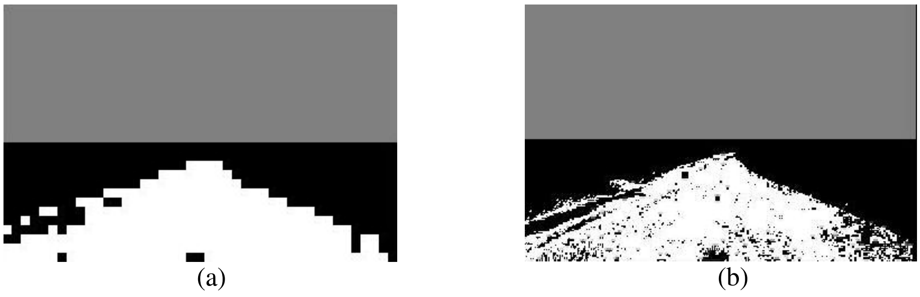


Fig. 10.12. Processed images

where μ_1 and $\sum_1$ are the mean vector and covariance matrix of the drivable-area RGB pixels in the training data, μ_2 and $\sum_2$ are those of the background pixels, and $\mathbf{x}$ contains RGB values of the entire image.

The decision boundary formula can be simplified as

$$(\mathbf{x}-\boldsymbol{\mu}_1)^T\Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)=(\mathbf{x}-\boldsymbol{\mu}_2)^T\Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2). \tag{10.7}$$

A block-based segmentation method is used to reduce the segmentation processing time. 4×4 pixel regions are clustered together and replaced by their RGB mean value, as follows:

$$\mu^L_{(x,y)}=\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}P^L_{(i,j)}, \tag{10.8}$$

where $\mu(x,y)$ is the new pixel mean value for the 4×4 block, P is the raw pixel data, (i,j) is the raw pixel orientation, (x,y) is the new pixel orientation, L$\epsilon\{1,2,3\}$ for RGB, and N is the block size.

The clusters, or blocks, are then segmented, and the result, as shown in Figure 10.12(a), has less noise compared with pixel-based approaches, Figure 10.12(b). Also, the segmentation process is accomplished faster than pixel-based classification. A disadvantage, however, is that edges are jagged and not as distinct.

10.6.2.4 Transform to Global Coordinate System

After processing the image, the areas classified as drivable road are converted by perspective transformation estimation into the global coordinates used for the Traversability Grid (Criminisi, 1997). The perspective transformation matrix is calculated based on camera calibration parameters and the instantaneous vehicle heading. Finally, the PFSS assigns a value of 12 (highly traversability) to those cells that correspond to an area that has been classified as drivable. All other cells are given a value of 7 (neutral). Figure 10.13 depicts the PFSS Traversability Grid data after transformation into global coordinates.

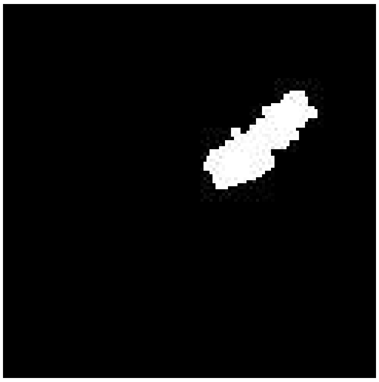


Fig. 10.13. Transformed image

10.6.3 Pseudo Smart Sensors

There are two Smart Sensors that produce Traversability Grids based on stored data: The Boundary Smart Sensor (BSS) and the Path Smart Sensor (PSS).

The BSS translates boundary knowledge, defined as boundary polygons prior to mission start, into real-time Traversability Grids, which assures that the vehicle does not travel outside the given bounds. The BSS is responsible for obtaining the boundary information from a local spatial database. The BSS uses these data to determine the in-bounds and out-of-bounds portions of the traversability grid for the instantaneous location of the vehicle. The BSS also has a configurable "feathering" capability that allows the edge of the boundary to be softened, creating a buffer area along the edges. This feature provides resilience to uncertainties in the position data reported by the GPOS component. Figure 10.14

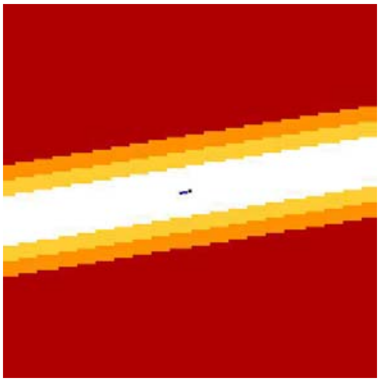


Fig. 10.14. Traversability Grid showing boundary data

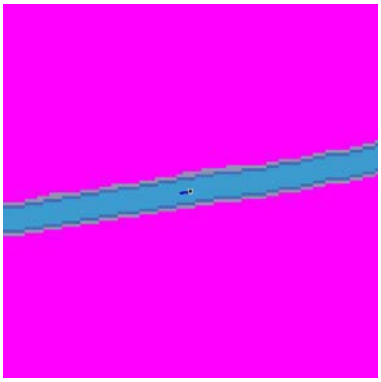


Fig. 10.15. Traversability Grid showing *a priori* path data

shows a typical grid output from the BSS indicating the vehicle's location within the grid, and the drivable region around it. By clearly demarking areas of the grid as out-of-bounds, the BSS allows the Smart Arbiter to summarily dismiss computation of out-ofbounds grid cells and the Reactive Driver to prune its search tree of potential plans.

The PSS translates the *a priori* path plan, stored as a "path file" prior to mission start, into real-time Traversability Grids. The PSS uses these path data to superimpose the originally planned path onto the traversability grid based on the instantaneous location of the vehicle. The PSS has a configurable "feathering" capability that allows the width of the path to be adjusted and the edges of the path to be softened. This feature also allows the engineer to select how strongly the originally planned path should be weighted by setting the grid value for the centerline. A 12 would cause the Arbiter and Planner to lean toward following the original plan even if the sensors were detecting a better path, while a 10, which is what was used at runtime, would make the original plan more like a suggestion that could be more easily overridden by sensor findings. Figure 10.15 shows a typical grid output from the PSS indicating the vehicle's location within the grid and the feathered *a priori* planned path flowing through the inbounds corridor.

10.6.4 Sensor Fusion

With the Traversability Grid concept in place to normalize the outputs of a wide variety of sensors, the data fusion task becomes one of arbitrating the matching cells into a single-output finding for that cell for every in-bounds cell location in the grid.

10.6.4.1 Grid Alignment

First, the Smart Arbiter must receive and unpack the newest message from a given sensor and then adjust its center-point to match that of the Arbiter (assuming that the vehicle has moved between the instant in time when the sensor's message was built and the instant in time when the arbitrated output message is being built). This step must be repeated for each sensor that has sent a message. The pseudo-code for this process is:

```
Determine current location of vehicle,
Adjust centerpoint of Smart Arbiter Grid to match current location
For each active Smart Sensor:
    Adjust centerpoint of Smart Sensor Grid to match current location.
```

At this point, all input grids are aligned and contain the latest findings from its source sensor. To support the alignment of Traversability Grids with current vehicle position, a so-called "torus buffer" object was introduced. This allows the system to use pointer arithmetic to "roll the grid" (i.e., change the row and column values of its center-point) without copying data.

10.6.4.2 Data Arbitration

Now the Smart Arbiter must simultaneously traverse the input grids, cell-by-cell, and merge the data from each corresponding cell into a singleoutput value for that row/column location. Once all cells have been treated in this fashion, the Smart Arbiter packs up its output grid message and sends it on the Reactive Driver.

For early testing, a simple average of the input cell values was used as the output cell value. Later work investigated other algorithms, including heuristic ones, to perform the data fusion task. The Smart Arbiter component was designed to make it easy to experiment with varying fusion algorithms in support of on-going research. The algorithm that was used for the DGC event entailed a two-stage heuristic approach. Stage 1 is an "auction" for severe obstacles for the cell position under consideration. Stage 2 then depends on the results of the "auction". If no sensor "wins" the auction, then all of the input cells at that position are averaged, including the arbiter's previous output value for that cell. The pseudo-code for this algorithm is:

For each cell location:

```
IF any sensor reported a "2,"
    THEN decrement the Arbiter's output cell by decr (min=2),
ELSE, IF any sensor reported a "3,"
    THEN decrement the Arbiter's output cell by decr/2 (min=3),
ELSE
    Arbiter's output cell= Average(input cells+ Arbiter's prior output cell),
```

where *decr* is a configurable parameter (= 2 for the Grand Challenge Event).

Thus, a sensor must report a severe obstacle for several iterations in order for the arbiter to lower its output value, thus providing a dampening effect to help circumvent thrashing due to a sensor's output values. The averaging of input values along with the arbiter's previous output value also provides a dampening effect.

The Smart Arbiter, using the algorithm described here, was able to achieve its specified processing cycle speed of 20 Hz. The premise used for all algorithms that were explored was to keep the arithmetic very simple and in-place since the data fusion task demands can reach 2 million operations per second just to process the algorithm (7 grids/cycle * 14,641 cells/grid * 20 cycles/second). Thus, complex probabilistic-based and belief-based approaches were not explored. However, adding highly traversable cells to the auction (i.e., 11's and 12's) and post-processing the output grid to provide proximity smoothing and/or obstacle dilation were explored, but none of these alternatives provided any better performance (in the sense of speed or accuracy) than the one used for the event.

10.7 Real-Time Planning and Vehicle Control

The purpose of online planning and control is to autonomously drive the NaviGATOR through its sensed environment along a path that will yield the greatest chance of successful traversal. This functionality is compartmentalized into the Reactive Driver (RD) component of the NaviGATOR. The data input to this component include the sensed cumulative traversability grid, assembled by the Smart Arbiter component, vehicle state information, such as position and velocity, and finally the *a priori* path plan, which expresses the desired path for the vehicle to follow sans sensor input. Given this information, the online real-time planning and control component, seeks to generate low-level actuator commands, which will guide the vehicle along the best available path, while avoiding any areas sensed as poorly traversable.

10.7.1 Receding Horizon Controller

The objective of the RD component is to generate an optimized set of the actuator commands (referred to as a "wrench" in JAUS), which drives the vehicle through the traversability grid and brings the vehicle to a desired goal state. The NaviGATOR accomplishes this real-time planning and control simultaneously, with the application of an innovative heuristic-based receding horizon controller.

Receding horizon is a form of model predictive control (MPC), an advanced control technique, used to solve complex and constrained optimization problems. In this case, the problem is to optimize a trajectory through the localized traversability space, while adhering to the nonholonomic dynamics constraints of the NaviGATOR. An in-depth explanation and analysis of the technique is provided in (Mayne, 2000), and the application of suboptimal MPC to nonlinear systems is given in (Scokaert, 1999). This method was selected because it unifies the higher-level planning problem with the lower-level steering control of the vehicle. Separate components are not needed to plan the geometry of a desired path, and then regulate the vehicle onto that path.

The controller attempts to optimize the cost of the trajectory by employing an A* search algorithm (Hart, 1968). The goal of the search is to find a set of open-loop actuator commands that minimizes the cost of the trajectory through the traversability space, and also bring the vehicle to within a given proximity of a desired goal state. The goal state is estimated as the intersection of the *a priori* path with the boundary of the traversability grid. As the vehicle nears the end of the path, and there is no longer an intersection with the grid boundary, the desired goal state is simply the endpoint of the last *a priori* path segment.

Special consideration was given to formulating the units of cost (c) for the search. An exponential transformation of the traversability grid value (t), multiplied by distance traveled (d) was found to work best. The cost equation is given here, where the exponent base is represented by (b):

$$c = db^t. \tag{10.9}$$

Thus, the cost of traversing a grid cell scales nonlinearly with its corresponding traversability value. An intuitive comparison is best to describe the effect of this transformation and why it works well: With a linear transformation, the cost of a path traveling through a traversable value of two is only twice as high as the same path through a value of one. (Note, these values are just given for the purpose of an example and are not actually encountered in the NaviGATOR system.) Therefore, the search would possibly choose a path driving through up to twice as much distance in the value of one, rather than a much shorter path driving through a value of two. Whereas, an exponential transformation ensures that there is always a fixed ratio between neighboring integer traversability values. Thus, this ratio can be used as a tuning parameter to allow the algorithm to achieve the desired tradeoff between the length and cumulative traversability cost of a selected path. Conveniently, the ratio used for tuning is equal to the base of the exponent given in the cost equation.

Closed-loop control with the receding horizon controller is achieved by repeating the optimization algorithm as new traversability data are sensed and vehicle state information is updated. Thus, disturbances, such as unanticipated changes in traversability or vehicle state, are rejected by continually reproducing a set of near optimal open-loop commands at 20 Hz, or higher.

The search calculates different trajectories by generating input commands and extrapolating them through a vehicle kinematics model. The cost of the resulting trajectory is then calculated by integrating the transformed traversability value along the geometric path that is produced through the grid. The search continues until a solution set of open-loop commands is found that produce a near-optimal trajectory. The first command in the set is then sent to the actuators, and the process is repeated. A typical result of the planning optimization is shown in (Figure 10.16, where the dark line is the final instantaneous solution).

Rather than plan through the multidimensional vector of inputs, i.e., steering, throttle, and brake actuators, the search attempts to optimize a one dimensional set of steering commands at a fixed travel speed; the control of the desired speed is handled separately by a simple proportional integral differential (PID) controller. Since it may be necessary to change the vehicle's desired speed in order to optimize the planned trajectory through the search space, extra logic is

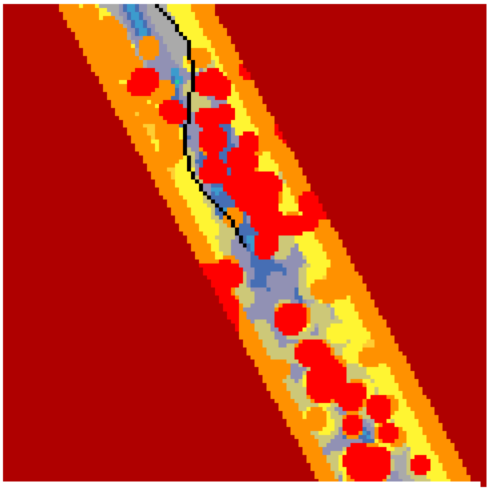


Fig. 10.16. Sample planning result through traversability grid

included in the search to either speed up or slow down the vehicle according to the encountered data.

10.7.2 Vehicle Model

The kinematics model used for response prediction of an input command to the system is that of a front-wheel steered rear-wheel-drive vehicle. The input signals to this model are the desired steering rate (u), and linear vehicle velocity (v). The model states include the vehicle Cartesian position and orientation (x, y, θ) and the angle of the front steering wheels (φ) with respect to the vehicle local coordinate frame. The kinematics equation is given here, where (b) represents the wheel base of the vehicle:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ \frac{1}{b}\tan\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix}. \tag{10.10}$$

In the algorithm implementation, additional constraints were added to the model to limit the maximum steering rate, and also the maximum steering angle. These values were determined experimentally and then incorporated into the software as configurable parameters. Also, due to the complex nature these system dynamics, obtaining a solution to the differential equations is not feasible; therefore, a series of simplifications and assumptions were made to allow for fast computation of future state prediction. The underlying assumption is that, since the resolution of the traversability grid is relatively low (0.5 m), very accurate estimates for the vehicle's predicted motion are not necessary. The assumptions made were that for a short period of time or traveled distance, the curvature of the path that the vehicle followed was near constant, and this constant curvature could be taken as an average over the predicted interval. Thus, the predicted path trajectory was simply a piece-wise set of short circular arcs. An implementation of Bresenham's algorithm (Bresenham, 1977) was used to integrate the traversability grid value along the determined arcs.

As an additional measure for vehicle stability, a steering constraint was added to limit the maximum steering angle as a function of speed (v) and roll angle (Φ) (due to uneven terrain). The goal of this constraint was to limit the maximum lateral acceleration (n_y) incurred by the vehicle due to centripetal acceleration and acceleration due to gravity (g). Thus, if the vehicle were traveling on a gradient that caused it to roll toward any one direction, the steering wheels would be limited in how much they could turn in the opposite direction. Additionally, as the vehicle increased in speed, this constraint would restrict turns that could potentially cause the NaviGATOR to roll over. This constraint is given by the following equation:

$$k_{\text{max}} = \frac{\pm n_{y\text{max}} + g\,\sin(\Phi)}{v^2}. \tag{10.11}$$

The value for maximum lateral acceleration was determined experimentally with the following procedure. A person driving the NaviGATOR would turn the wheels completely to one direction, and then proceed to drive the vehicle in a tight circle slowly increasing in speed. The speed in which the driver felt a lateral acceleration that was reasonably safe or borderline comfortable was recorded, and the acceleration value was calculated. This was done for both left and right turns, and the minimum of the two values were taken for conservatism. The value found to be a reasonable maximum was, 4 mps^2, and was hard coded as a constraint to the vehicle model.

10.7.3 Desired Speed Logic

The determination of the commanded and planned vehicle speed is derived from many factors. The RD receives several sources of requested speed, calculates its own maximum speed, and then chooses the minimum of these to compute the throttle and brake commands to the vehicle. Each of the input speed commands was calculated or originated from a unique factor of the vehicles desired behavior. At the highest level, vehicle speed was limited to a maximum value that was determined experimentally based on the physical constraints of the NaviGATOR platform. The next level of speed limiting came from the *a priori* path data, which itself was limited to the speeds provided by the RDDF corridor. Additionally, the speed provided by the path file can be assessed to see if a slower desired speed is approaching ahead of the vehicle. Therefore, if it is about to become necessary to slow down the vehicle, the RD can allow for natural deceleration time. Also, desired speed as a function of pitch was added to slow down the vehicle on steep ascents or descents. This ensures that the NaviGATOR does not drive too fast while encountering hilly terrain.

Another important speed consideration resides with the planning search. Embedded in the search itself is the ability to slow the vehicle down in the cases where the desired trajectory comes within proximity to poorly traversable grid cells. The planner uses a look-up table that enumerates all of the maximum speeds that the vehicle is allowed to drive while traveling through areas of low traversability. Thus, if the vehicle attempts to avoid an obstacle or travel down a narrow road with hazardous terrain to either side, it is commanded to slow down, thus providing a lower risk while allowing for a more comprehensive search to find the best course of action. Also, if the search was unable to find a reasonable solution (i.e., only a solution that goes through very poor areas was found), then the desired speed is lowered. In its next iteration, the RD attempts to find a better solution at that slower speed. This approach is reasonable because the vehicle has greater maneuverability at low speed, and therefore the planner has a better chance of finding a less costly route to its goal.

Additional speed control is provided by a Situation Assessment component consisting of a Long Range Obstacle Specialist and a Terrain Ruggedness Specialist. The Long Range Obstacle Specialist uses a data feed from the PLSS LADAR to determine whether the space directly in front of the vehicle is free of obstacles beyond the 30 m planning horizon (i.e., 30 m out to the 80 m range limit of the LADAR). The Terrain Ruggedness Specialist uses the instantaneous pitch rate and roll rate of the vehicle (provided by the Velocity State Sensor) to classify the current terrain as "Smooth," "Rugged," or "Very Rugged." Based on the Long Range Obstacle State and Terrain Ruggedness State, with appropriate hysteresis control and dampening, the permitted speed of the vehicle is selected and sent to the RD. For example, if the terrain is Smooth and no Long Range Obstacles are detected, then the RD is permitted to drive the vehicle up to its highest allowable speed and thus faster than an empirically derived Obstacle Avoidance speed of 7.2 mps (16 mph).

10.7.4 Controller Fault Detection

There are four faults that the RD is capable of detecting during normal operation of the vehicle. They are the cases where the NaviGATOR has: become blocked, become stuck, collided with an obstacle, or gone out of the bounds of the RDDF. In each of these scenarios, it is possible for the system to take corrective action. The most commonly found of these errors is the blocked condition. In this case, there is no viable path planning solution, even when the search is attempting to plan a trajectory at the vehicle's most maneuverable speeds. It was determined through analysis of the collected data that this case was most often occurring due to sensor misclassifications. The corrective action in these scenarios is to simply wait a short period of time for the sensor data to correct itself, allowing the planner to find a solution. Sometimes, the data will not correct without the vehicle changing its position, therefore an active correction is taken to automatically "nudge" the vehicle forward after a brief wait, and continue with the planned path once the blockage is clear.

10.8 Testing and Performance

This section of the paper summarizes the testing and performance that occurred at each of several key venues. This section is supplemented by a video depicting the NaviGATOR operating in each of these venues (see multimedia).

10.8.1 The CIMAR Lab

Testing began with the JAUS messaging system on the ten computers that would drive the NaviGATOR. The JAUS messaging would need to be capable of sending up to 500 messages per second per node for over 14 hours. On race day, over 20 million JAUS messages were actually sent and received. Next, initial testing of the individual JAUS components, discussed in this paper, took place in the spring of 2005 primarily in the CIMAR lab at the University of Florida. The goal was to get each component working by itself, "on the bench" in a controlled laboratory environment. To support bench testing, a simple vehicle simulator component was devised that sends out position- and velocity-related JAUS messages as if the vehicle were moving through an RDDF corridor. Once each individual component had been successfully tested and declared operational, then various combinations of components were integrated and tested together as the system began to take shape. The base vehicle platform had been assembled during the same period of time as the various JAUS components were being bench tested. With both the vehicle assembled and the JAUS components operational, the various JAUS components were then mounted in the NaviGATOR.

10.8.2 The Citra Test Site

Next, it was time to take the system to the field. On 20 April 2005, a test site was designed and constructed at the University of Florida's Plant Science Unit located in Citra, Florida. The course was laid out in an open field and consisted mainly of a figure eight, an oval, and several left and right sharp turns (see Figure 10.17). Various segments were added to this course to replicate terrain that was expected in the desert. While this course had a few tough obstacles, it was basically the "safest" place to test. This was Team CIMAR's main test site and was used for extensive development of the system as well as the location where the DARPA site visit took place on 6 May 2005. On 20 May 2005, the NaviGATOR was put into a 1/2 mile loop, and it ran for 12 miles before stopping due to a minor problem. This was the furthest it would run prior to heading west in September, as it spent the next three months undergoing major upgrades to both hardware and software.

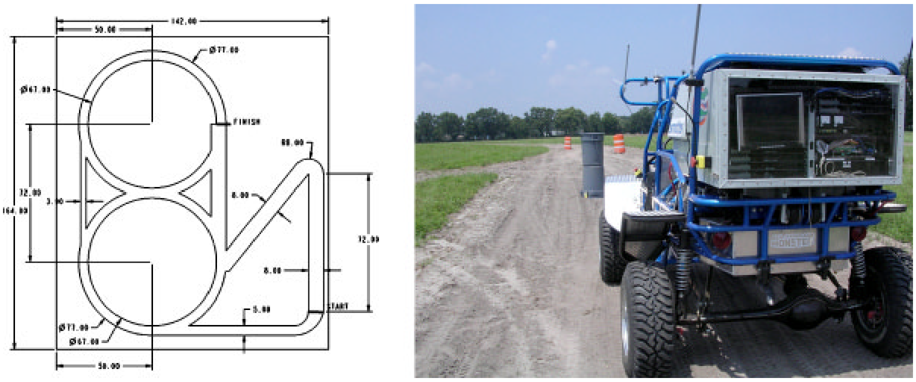


Fig. 10.17. The Citra test site

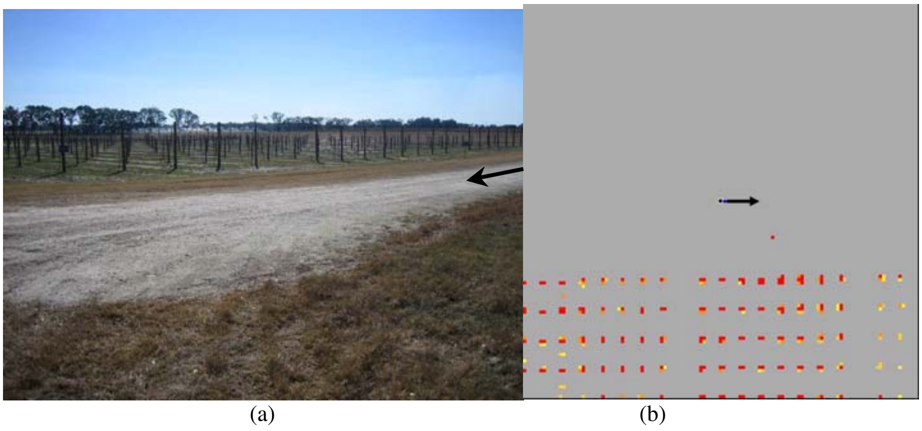


Fig. 10.18. PLSS testing images

Part of the Citra testing effort was devoted to the initial tuning of the sensors. Figures 10.18 and 10.19 depict scenes of the terrain at Citra and the accompanying Smart Sensor output, as captured during the sensor tuning process. Figure 10.18(a) shows evenly spaced orchard poles, while Figure 10.18(b) shows a snapshot of the PLSS Traversability Grid while traveling on the graded road in which the poles have been clearly detected and scored as impassable obstacles. This area was specifically chosen to assure that the output of the PLSS accurately maps obstacles onto the grid. Note that the PLSS algorithm has been tuned to accurately locate the poles, even though most of them are occluded for periods of time as the vehicle moves past them. Figure 10.19(a) shows a roadway with rough terrain appearing to the left of the vehicle when traveling in the indicated direction. Figure 10.19(b) shows a snapshot of the TSS Traversability

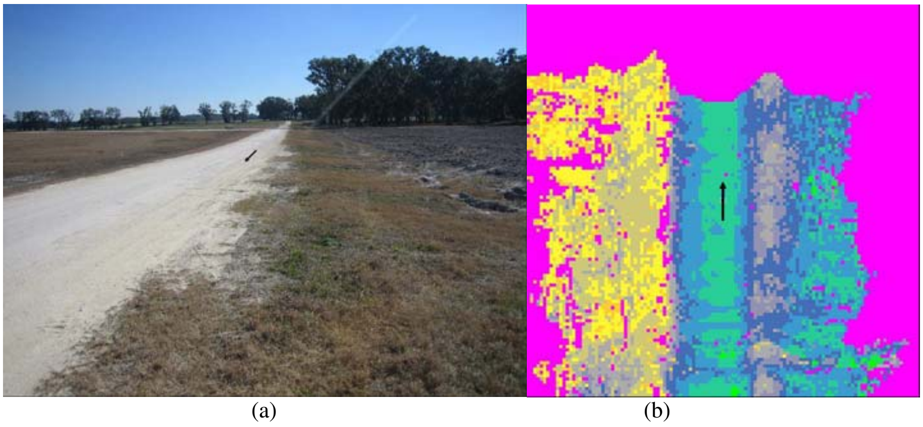


Fig. 10.19. TSS testing images

Grid for that same section of road, having scored the rough area as somewhat undesirable, but not absolutely blocked (i.e., 4's, 5's and 6's).

By 20 August 2005, the major hardware and software upgrades were complete and the system was ready for one last round of testing at the Citra site prior to heading west. On 25 August 2005, however, while performing a high-speed radar test, the vehicle suffered a serious failure. One of the rear shocks snapped and the engine and frame dropped onto the rear drive shaft and odometer gear. The sudden stop also caused the front sensor cage struts to snap and the sensor cage collapsed forward. The causes of the failures were determined and the system was re-designed and re-built in approximately one week. With the damage repaired, the NaviGATOR returned to Citra for several days to verify that the system was operational and ready to graduate to the desert for a more serious round of testing.

10.8.3 The Stoddard Valley OHV Area

On 11 September 2005, the NaviGATOR headed west to the Stoddard Valley Off Highway Vehicle (OHV) Area near Barstow, California (see Figure 10.20). The team first attempted some short test runs to ensure system operation. This also marked the first times the team had run the NaviGATOR with a chase vehicle setup (see Figure 10.21(a)). These system tests were done in the OHV area of Stoddard Valley (marked 1 in Figure 10.20). This test route is approximately 4 miles long and included the first serious autonomous uphill and downhill climbs, allowing the team to evaluate the performance of the system during both accent and decent maneuvers. Speeds during these tests stayed in the 10 mph range. Overall, the system showed an almost surprising ability to handle the terrain, prompting the team to accelerate their efforts in finding more challenging test paths. Following these successful tests, the team moved the vehicle on to Slash X.

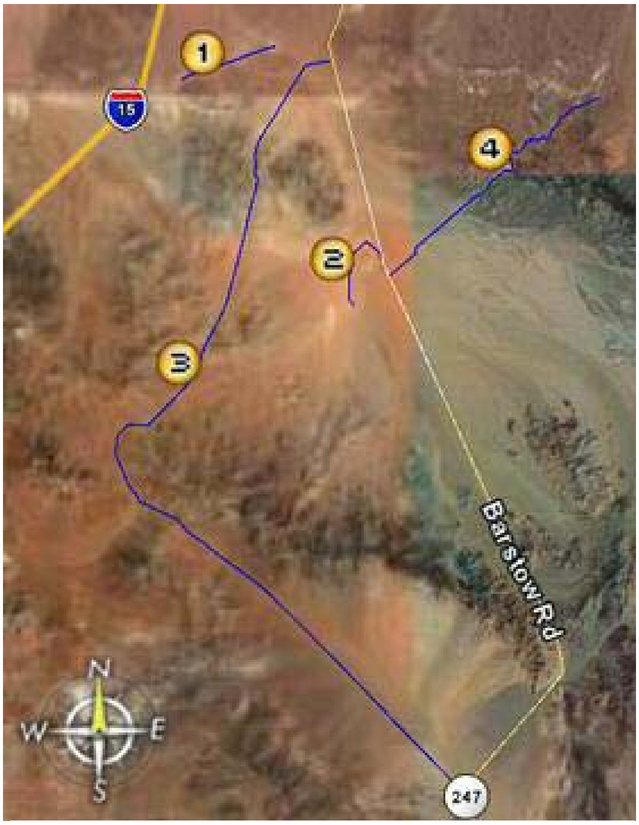


Fig. 10.20. Stoddard Valley OHV test sites

Fig. 10.21. Testing in the Stoddard Valley OHV area

Slash X was the site of the start of the DARPA Grand Challenge 2004 (DGC04) event and during their time there, Team CIMAR shared the area with several other DARPA Grand Challenge 2005 teams. First, the team ran the start and first mile of the DGC04 route (site 2 in Figure 10.20). This allowed the team to test and tune the sensors specifically against the barbed wire fence that was the downfall of the 2004 NaviGATOR. This path also provided a good place to test higher speed navigation. Between path 2 and an open area behind Slash X, the team was able to test and tune the NaviGATOR up to 30 mph, with an empirically determined obstacle avoidance speed of 16 mph.

Sunday 18 September 2005 turned out to be a historic day for Team CIMAR. Team TerraMax graciously gave us one of their RDDFs through the desert (labeled site 3 in Figure 10.20). The team took the file and after several false starts finally launched the vehicle at 4 pm. Path 3 is approximately 20 miles each way (with a built-in turnaround). The speed testing had not yet been completed and the first test was done at a cap of 10 mph. The team had never seen nor traversed this path prior to this first test. Not knowing exactly where they were going, the NaviGATOR led the way (see Figure 10.21(b)). Surprising even the team members, the NaviGATOR successfully navigated the entire 20 mile distance on the first try, stopping only to give its human handlers time to drink and rest.

Over the following week, the team tested the NaviGATOR several more times on this course, reaching speeds of 25 mph and completing the entire 40-mile course several times. The path included long straight roads, a mountain climb, and areas covered by power lines; all terrain the team expected to encounter during the DARPA Grand Challenge event.

The last significant area of testing in Stoddard Valley (marked 4 in Figure 10.20) was another portion of the DGC04 event. Known as Daggett Ridge, this was the area that the farthest teams had reached during the previous event and consisted of very dangerous mountain switchbacks and drop-offs of hundreds of feet. The sensor team made several trips with the vehicle to tune and test the sensor suite on the path during manual drive, especially focusing on detecting negative obstacles (in the form of cliffs and washouts).

During two weeks of dawn-to-dusk testing in the Stoddard OHV area, the NaviGATOR went from a personal best of 12 miles in a 1/2-mile circuit to 40-mile runs across miles of desert terrain. The team was able to scale the system quickly, going from 10 mph runs to 25 mph with reliable obstacle avoidance at speeds up to 16 mph, along with tuning and validating the software that dynamically determines which speed should be used. That time in the desert was perhaps the best time spent testing during the entire DARPA Grand Challenge project, both in progress for the vehicle and the team members. While more testing time would have been very useful, on 27 September 2005 the team left for the California Speedway and the National Qualification Event.

10.8.4 The National Qualification Event

Immediately following the opening ceremony, the NaviGATOR was the fourth team in line for the first qualification run. The qualification course is shown in Figure 10.22. It consisted of a 2.3 mile long path with three parked cars, a rough

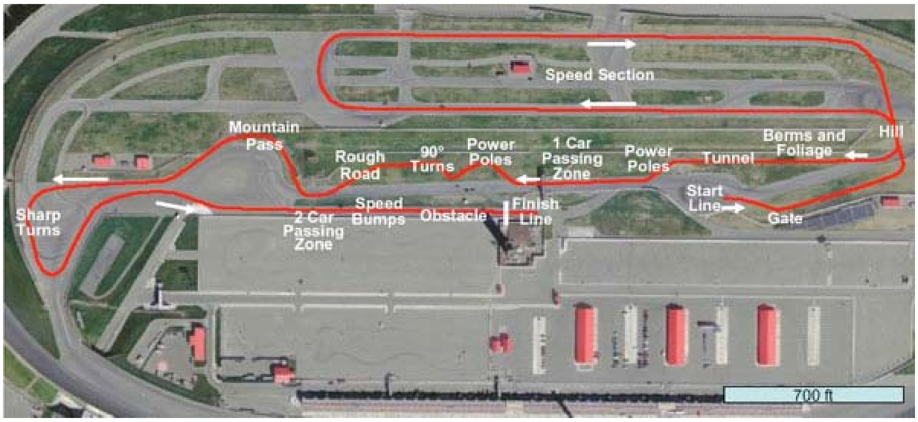


Fig. 10.22. Qualification course at the California Speedway

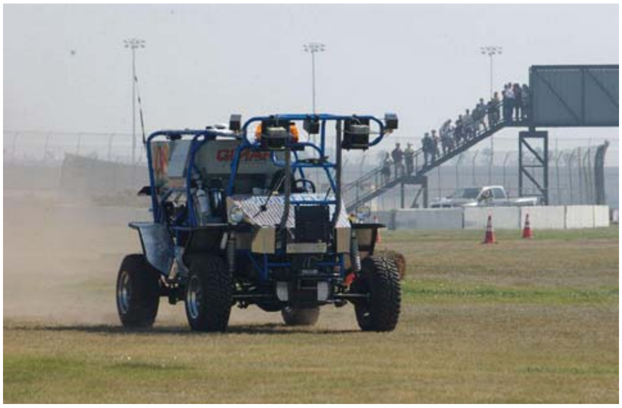


Fig. 10.23. NaviGATOR at NQE

terrain section, a simulated mountain pass, a tunnel, and finally a wooden "tank trap" obstacle.

The NaviGATOR completed the entire course on the first attempt. Figure 10.23 depicts the NaviGATOR on the NQE course. However, three lane-marking cones had been hit and the tank trap obstacle at the end of the course had been slightly brushed. Two changes were made to the NaviGATOR for the second run. The desired speed on the high-speed section of the course was increased from 16 mph to 20 mph and the dilated size of the perceived obstacles was increased in an attempt to completely miss the tank trap obstacle. During the second run, the NaviGATOR began oscillating and became unstable on the high-speed section and the run was aborted. The problem was that the high-speed section of the qualification course was on pavement whereas all high-speed testing had been conducted off-road. The disturbances caused by the constant four-wheel drive on pavement were responsible for the oscillation.

For the third run on the qualification course, all parameters were reset to those used during the first run. All went well until the vehicle scraped the concrete wall in the mountain pass section of the course, snapping the front steering linkage. The vehicle was quickly repaired. For future runs, the path centerline as reported by the PSS was shifted 12 in. away from the wall in the mountain pass section. After this, the qualification course was successfully completed two more times. In summary, the NaviGATOR completed the entire qualification course three out of five times, and the team was selected by DARPA to compete in the desert race.

10.8.5 The Race

The team received the RDDF containing the course waypoints in the early morning of 8 October 2005. Two hours were allocated for processing the data, which primarily consisted of setting desired speeds for each section of the course. The path file was then uploaded to the vehicle and by 9:30 a.m. the NaviGATOR was off. After leaving the start gate, the NaviGATOR headed off into the desert and then circled around past the crowd at about the eight-mile mark. The NaviGATOR headed past the spectators at approximately 24 mph, performing very well at this point in the race (see Figure 10.24). After following the dirt road a bit further, the NaviGATOR encountered a paved section of the course and started to oscillate. It stopped and did a couple of turns, criss-crossed the road, and then regained its composure

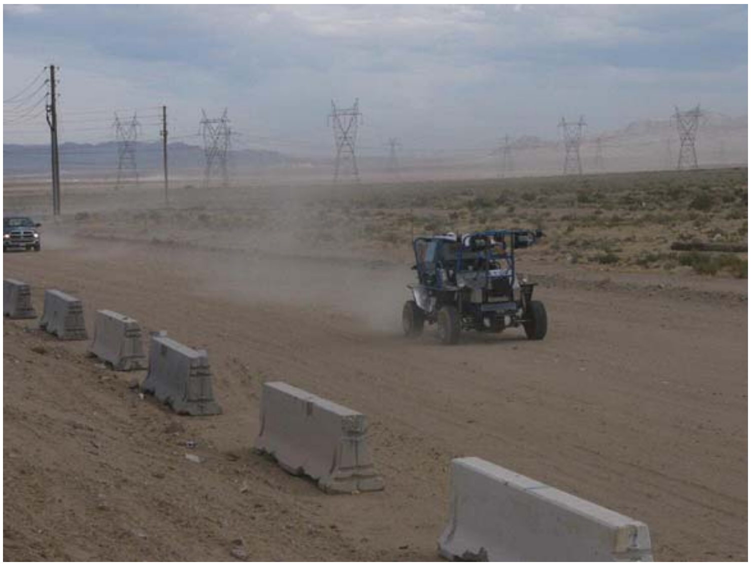


Fig. 10.24. NaviGATOR passing the stands at the 2005 DARPA Grand Challenge Event

and headed back in the right direction. As during the second qualification run, the desired speed was set too high for operation on pavement.

The NaviGATOR next flawlessly traversed a bridge over a railroad track and disappeared into the brown desert haze. Shortly before 11 a.m., the team received word from the chase truck that was following NaviGATOR that the vehicle had inexplicably run off the road and stopped. NaviGATOR appeared reluctant to move forward into and out of low brush in front of it, although its off-road capabilities would have easily carried it through. After several attempts to pause and restart the NaviGATOR, the driver called back to say the vehicle was moving, but slowly and still off the road. After about one-half of a mile of starting, stopping, and driving very slowly over brush, it regained the road and took off again at high speed following the road perfectly. However, after about another mile, the vehicle again went off the road and this time stopped in front of a bush. This time, DARPA officials quickly declared the NaviGATOR dead. The time was shortly before noon, and NaviGATOR had traveled past the 24-mile marker. NaviGATOR placed 18$^{\text{th}}$ among the 23 finalists. A total of five teams actually completed the entire course, with Stanford's Stanley taking the $2 million prize for the shortest time of six hours, 53 minutes and 58 seconds.

10.8.6 What Stopped the NaviGATOR?

Team members went out on the course the day after the race and found the NaviGATOR tire tracks at the two locations where the vehicle went off the right side of the road. From this information and data that were logged on the vehicle, it appears that the calculated GPS position drifted by approximately twenty feet causing the vehicle to want to move to the right of the actual road. From the tire tracks and from the traversability grid (see Figure 10.25), it was apparent that the vehicle wanted to move to the right, but the obstacle avoidance sensors were

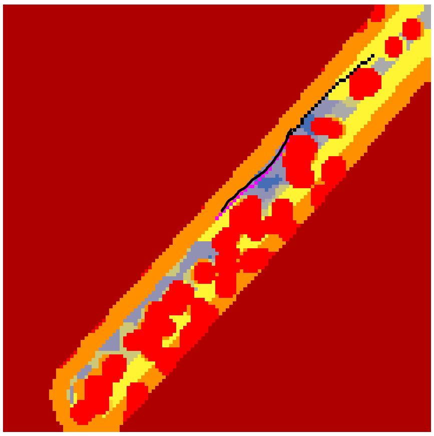


Fig. 10.25. Traversability Grid (during time of position system drift)

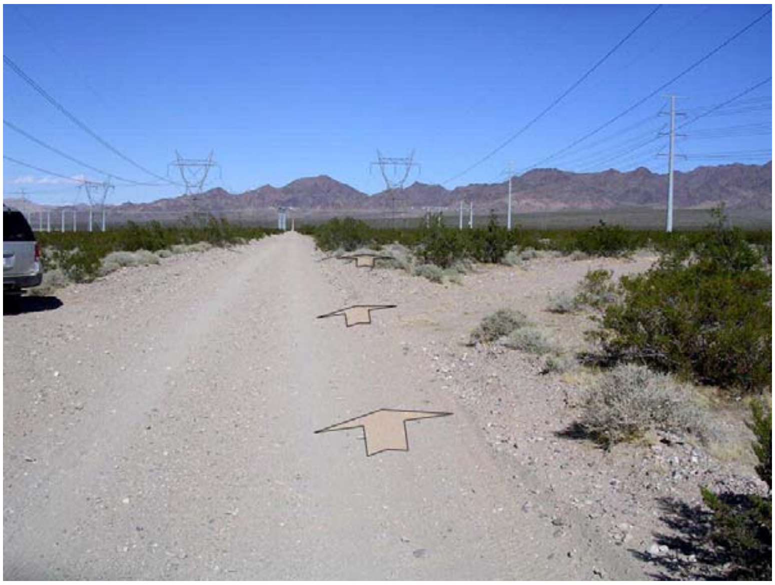


Fig. 10.26. Location where NaviGATOR veered off the course and was stopped

detecting the bushes and berms on the right side of the road. From the vehicle's perspective (see Figure 10.25) it appeared that the corridor was littered with objects and the best it could do was to travel along the left side of the corridor on the verge of going out of bounds on the left. In reality, the vehicle was hugging the right side of a very navigable dirt road, however most of the open road was being classified as out of bounds.

Both times that the vehicle went off course were due to the fact that the right side became free of obstacles and the vehicle attempted to move to the center of its incorrect corridor. Figure 10.26 shows the location where the NaviGATOR moved off the course for the second time whereupon DARPA officials stopped it. In summary, a twenty-foot position error caused a corresponding shift of the boundary smart sensor that eliminated the actual sensed road as an option to the planner.

10.9 Conclusion

Overall the team was very pleased with the NaviGATOR system. The base vehicle is very capable and has excellent mobility in very rough terrain. The obstacle and terrain detection sensors, and sensor integration approach, worked very well as did the reactive planner module. Overall, the control loop (from sensed objects to determination of vehicle actuation parameters) operated at a rate of over 20 Hz. Also, a significant contribution of the effort was to show that JAUS could be used successfully in a situation such as this, and that the standardized messaging system defined by JAUS could greatly simplify the overall integration effort.

There are four key areas that are currently being pursued by the team. The first two of these focus on resolving specific issues encountered while competing at the Grand Challenge event. The other two are improvements that will make the NAVIGATOR system more resilient to such problems when they occur.

1. **Stability.** The stability of the controller can be improved simply by putting additional time into getting the control parameters properly tuned. The goal is to achieve stable control at 25 mph on pavement and 30 mph on dirt in the near future.
2. **Position System.** We are currently improving the accuracy of the position system's estimate of error so that when the output of the system is degraded, it can inform the rest of the system appropriately. A better version of the GPS switching code is being implemented that will allow the system to decide which GPS to use as the input to the NFM, the NavCom or the Garmin, based on which is better at the time. At the same time, NavCom and Smiths Aerospace are working together to further improve the overall accuracy of the system.
3. **Dynamic BSS and PSS.** As discussed earlier, the reason the NaviGATOR got stuck off the road in the race was due to a position error causing the Boundary Smart Sensor to shift the drivable corridor off the road. To prevent this from happening in the future, the width of the corridor created by the BSS will be made a function of the position system root mean square error (RMS). For example, if the position RMS is good then the BSS corridor in the grid will be correspondingly tight; but when the position RMS degrades, then the BSS will stroke a correspondingly wide corridor through its traversability grid. In this way, the BSS will no longer eliminate the road as an option, thus allowing the sensors to find the road off to the side. Similarly, the weight of the PSS can be adjusted such that its recommended path is painted with tens when the position RMS is good, but only sevens or eights when the position RMS degrades, thus reducing its influence accordingly.
4. **Adaptive Planning Framework.** A more extensive implementation of the situation assessment specialists and high-level decision-making capabilities is currently underway. This will allow the NaviGATOR to do such things as determine when it has become blocked and decide how to best fix the problem, such as backing up and re-planning. Other examples include altering the aggressiveness of the plan (risk) based on mission parameters and altering the contribution of a given sensor based on the environmental situation.

The first three items on this list are relatively short term and should be completed before this paper is published. With a tuned controller, the position system upgraded, and the BSS and PSS dynamically adjusting to the position RMS, the NaviGATOR should be capable of completing the 2005 DARPA course in under 10 hours. The maturation of the Adaptive Planning Framework will likely continue into the future for some time.

In retrospect, the team would have benefited from more testing time in the California desert. The issues associated with the positioning system and the high-speed control on pavement could have been resolved. However, the project was very successful in that an entirely new vehicle system was designed, fabricated, and automated in a nine-month period, ready to compete in the 2005 DARPA Grand Challenge. This was a monumental effort put on an aggressive time and resource schedule.

Acknowledgments

The authors would like to acknowledge the work of all the DARPA staff members who organized and operated a very successful event. Also, thanks are given to the members of the JAUS Working Group whose efforts are leading to an open architecture that will promote interoperability of military robotic systems. Finally, the authors gratefully acknowledge the technical and financial support of the team's Title Sponsor, Smiths Aerospace.

References

Bresenham, J. (1977). A Linear Algorithm for Incremental Digital Display of Circular Arcs. *Communications of the ACM*, 22(2), 100–106.

Criminisi, A., Reid, I., and Zisserman, A. (1997). A Plane Measuring Device [Electronic Version]. Retrieved April, 2006 from `http://www.robots.ox.ac.uk/~vgg/presentations/bmvc97/criminispaper/`

Elfes, A. (1989). Using Occupancy Grids for Mobile Robot Perception and Navigation. *IEEEComputer Magazine*, pp 46–57.

Hart, P. E., Nilsson, N.J., Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, SSC4(2), 100–107.

JAUS. (2005). *Joint Archtiecture for Unmanned Systems Reference Architecture, version 3.2*: JAUS Working Group (`http://www.jauswg.org/`).

Lee, J., Crane, C., Kim, S., and Kim J. (2004). *Road Following in an Unstructured Desert Environment using Monocular Color Vision as Applied to the DARPA Grand Challenge*. Paper presented at the ICCAS2005, Seoul.

Mayne, D., Rawlings, J., Rao, C., Scokaert, P. (2000). Constrained Model Predictive Control: Stability and Optimallity. *Automatica*, 36(6), 789–814.

Morris, R. D., Descombes, X., and Zerubia, J. (1997). Fully Bayesian image segmentation - an engineering perspective. *Proceedings of IEEE International Conference on Image Processing.*

Scokaert, P., Mayne, D., Rawlings, J. (1999). Suboptimal Model Predictive Control (Feasibility Implies Stability). *IEEE Trans. Automatic Control*, 44, 648–652.

Seraji, H. (2003). New Traversability Indices and Traversability Grid for Integrated Sensor/Map- Based Navigation. *Journal of Robotic Systems*, 20(3), 121–134.

Solanki, S. (2003). *Implementation of laser range sensor and dielectric laser mirrors for 3D scanning of glove box environment.* Unpublished Master's Thesis, University of Florida, Gainesville.

Thrun, S. (2003). Learning Occupancy Grid Maps with Forward Sensor Models. *Autonomous Robots*, 15, 111–127.

Ye, C., and Borenstein, J. (2004). T-transformation: Traversability Analysis for Navigation on Rugged Terrain. *Proceedings of the Defense and Security Symposium, Unmanned Ground Vehicle Technology VI (OR54).*