Perception and Planning Architecture for Autonomous Ground Vehicles

Bob Touchton, Tom Galluzzo, Danny Kent, and Carl Crane
University of Florida

By combining smart sensors and traversability grids with a JAUS-based component and messaging architecture, DARPA Grand Challenge finalist Team CIMAR quickly developed a robust autonomous ground vehicle platform with advanced sensing and planning capabilities.

Team Gator Nation (http://cimar.mae.ufl.edu/gatornation), formerly Team CIMAR, a finalist in the 2004 and 2005 DARPA Grand Challenge and a competitor in the upcoming 2007 DARPA Urban Challenge, is a collaboration of the University of Florida's Center for Intelligent Machines and Robotics (CIMAR), Machine Intelligence Lab, and Digital Worlds Institute along with the Eigenpoint Company and title sponsor Smiths Aerospace. Autonomous Solutions Inc. was also a member of Team CIMAR.

One of Team CIMAR's major strategies in preparing its entry for the first two driverless competitions was to task a software engineering subteam to design and deploy a standardized software architecture, with accompanying software tools and libraries, that

- complied with the US Department of Defense's JAUS interoperability framework, described in the "Joint Architecture for Unmanned Systems" sidebar;
- unified diverse *smart sensor* output via a smart sensor wrapper; and
- incorporated a multivalued *traversability grid* to manage both positive and negative obstacles as well as assess terrain smoothness.

The team incorporated these architectural features in its NaviGATOR autonomous ground vehicle (AGV), a custom-built off-road vehicle, shown in Figure 1. Key components included six smart sensors for detecting environmental conditions and reporting a priori data, a *smart arbiter* for fusing data from multiple smart sensors, and a *reactive driver* for providing real-time navigation planning and obstacle avoidance.

JAUS-COMPLIANT ARCHITECTURE

The JAUS reference architecture (www.jauswg.org/baseline/refarch.html) provided the framework for developing the NaviGATOR's software components and messaging system. The heterogeneous composition of Team CIMAR, which included multiple organizations and numerous graduate students within the CIMAR lab, dictated the need for such a framework. Having a standardized component interface and messaging system eliminated integration chaos as specific capabilities were introduced and evolved over time. Using JAUS also enabled Team CIMAR to later transfer its DARPA Grand Challenge technologies to the Air Force Research Lab, one of its major sponsors.

Team CIMAR formulated the NaviGATOR's system architecture using existing JAUS-specified components and messages wherever possible, along with a JAUS-compliant messaging infrastructure (the team fielded the only JAUS-compliant vehicle in the event). Tasks for which JAUS specifies no components required the

Joint Architecture for Unmanned Systems

The growing diversity and complexity of robotic systems within the US military during the 1990s motivated the development of a common interoperability framework. In 1998, the Office of the Secretary of Defense chartered the Joint Architecture for Unmanned Systems (JAUS) Working Group (www.jauswg.org) to develop a framework that would

- aid in the procurement of robotic components by ensuring their mutual compatibility,
- promote competition in the marketplace while avoiding dependence on proprietary solutions,
- let developers focus on application needs rather than basic infrastructure, and
- reduce the technology transfer burden.

The JAUS Reference Architecture (currently v3.2) defines a set of reusable components, their associated services and data interfaces, a set of messages, and the transport mechanism for exchanging messages among components. Nodes represent the physical computer hardware units, enabling multiple components to reside on a given node.

Organizations have wide latitude in complying with the JAUS RA, which adheres to four key constraints:

- *Vehicle platform independence.* To promote component interoperability, the JAUS RA makes no assumptions about the underlying vehicle or its means of propulsion.
- *Mission isolation.* JAUS components can be assembled in various ways to support different missions.
- *Computer hardware independence.* To allow for future adaptability and enhancement as new computer hardware becomes available in the future, the JAUS RA does not include any specific hardware dependencies or requirements.
- *Technology independence.* The JAUS RA does not designate any particular technical approaches, devices, or noncomputer hardware, and it minimizes specification of common technologies. For example, it mandates ASCII and TCP/UDP/RS-232 for messaging transport but not any specific language, operating system, algorithm, or sensor.

JAUS is currently undergoing two major transitions. Technically, JAUS is becoming service-oriented rather than component-oriented to allow greater flexibility in specifying and combining services. Also, in line with an early Department of Defense goal to migrate JAUS to a mainstream standards body, JAUS is moving to the Society of Automotive Engineers as the Unmanned Systems Committee (AS-4); the SAE is currently in the process of preparing and approving an initial set of standards.

creation of experimental components with user-defined messages. The JAUS Working Group endorses this approach as the best way to extend and evolve the standard: Researchers can use experimental components and user-defined messages to leverage existing JAUS infrastructure, where practical, and ensure that new components and messages align with JAUS principles should they eventually be formally adopted.

Component architecture

As Figure 2 shows, at the highest level, the NaviGATOR's software architecture consists of four fundamental elements:

- *Planning.* These components act as a repository for a priori data—known roads, trails, or obstacles as well as acceptable vehicle workspace boundaries—and support offline planning using such data.
- *Control.* These components perform closed-loop control to keep the vehicle on a specified path.
- *Perception.* These components perform the sensing tasks required to locate obstacles and evaluate terrain smoothness.
- *Intelligence.* These components determine the best path segment to drive based on sensed information.

Several standard JAUS components guided the AGV's basic operation. The *global position and orientation sensor* (GPOS) provides real-time vehicle position (latitude/

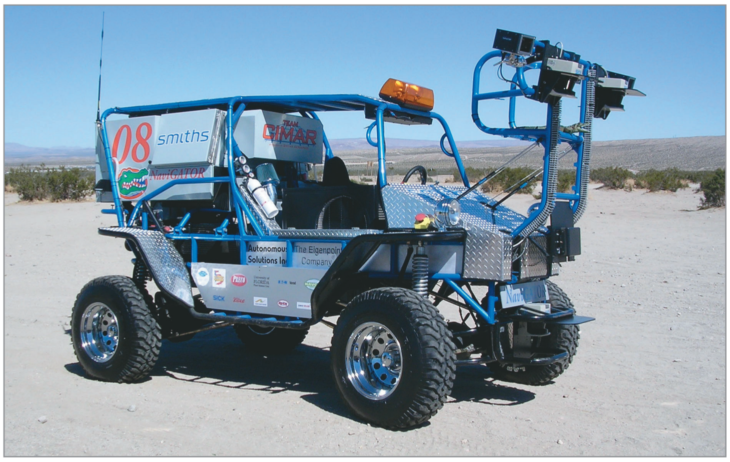

Figure 1. NaviGATOR. Team CIMAR's autonomous ground vehicle was a DARPA Grand Challenge finalist in both 2004 and 2005. This photo shows the 2005 model.

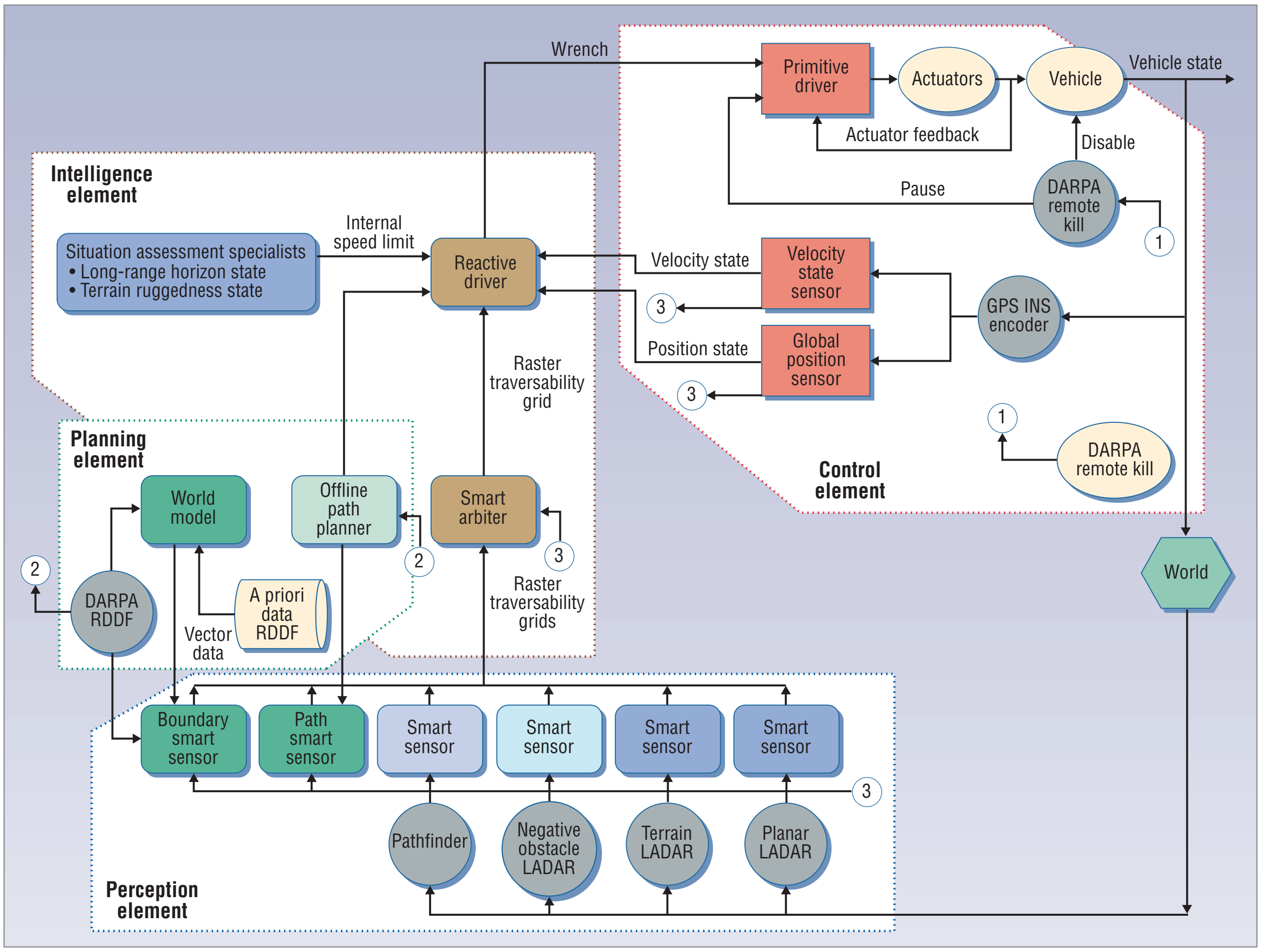

Figure 2. NaviGATOR's JAUS-compliant architecture. Components on the same node share the same color; every distinct node also has a node manager (not shown). Circled numbers indicate to/from jumps.

longitude) and orientation (roll/pitch/yaw) based on a combination of GPS signals, inertial navigation system readings, and a drive shaft encoder. The GPOS uses the standard Report Global Position message. The *velocity state sensor* (VSS) provides instantaneous speed as well as roll, pitch, and yaw rate and uses the standard Report Velocity State message. The *primitive driver* transforms an input "wrench"—the way JAUS describes the desired vehicle motion—into commands to the throttle, brake, and steering actuators. The PD receives a standard Set Wrench message.

To provide the degree of autonomous behavior that the DARPA Grand Challenge routes require, the team added a series of experimental components (differentiated by the rounded corners in Figure 2) and associated user-defined messages. The *situation assessment* and *world model* components are early in their development and play only minor roles in the overall system.

Messaging architecture

JAUS specifies a data/information transport approach that standardizes the addressing and delivery mechanisms via a mandatory message header while providing flexibility in the supported messaging use cases. The message header is required for all JAUS messages and contains component source and destination addresses, the message ID, the attached data's size, and a set of associated properties. JAUS requires that all intercomponent data exchanges occur via JAUS messages; however, data exchanges between processes internal to the component or between a device and the component can use other methods.

The *node manager* is a special component that provides the proper routing of JAUS messages. Each component must exchange information by routing all messages through a node manager residing on its own hardware node, whether the other component involved resides on the same vehicle, a different vehicle, a distinct payload, or an *operator control unit*. Figure 3 schematically depicts the node manager implementation.

Messages are classified by the function they serve: *command*, *query*, or *inform*. This makes it possible to vary use-case behaviors by message function and role. For example, a command message is unidirectional, and the receiving component will ignore it if another

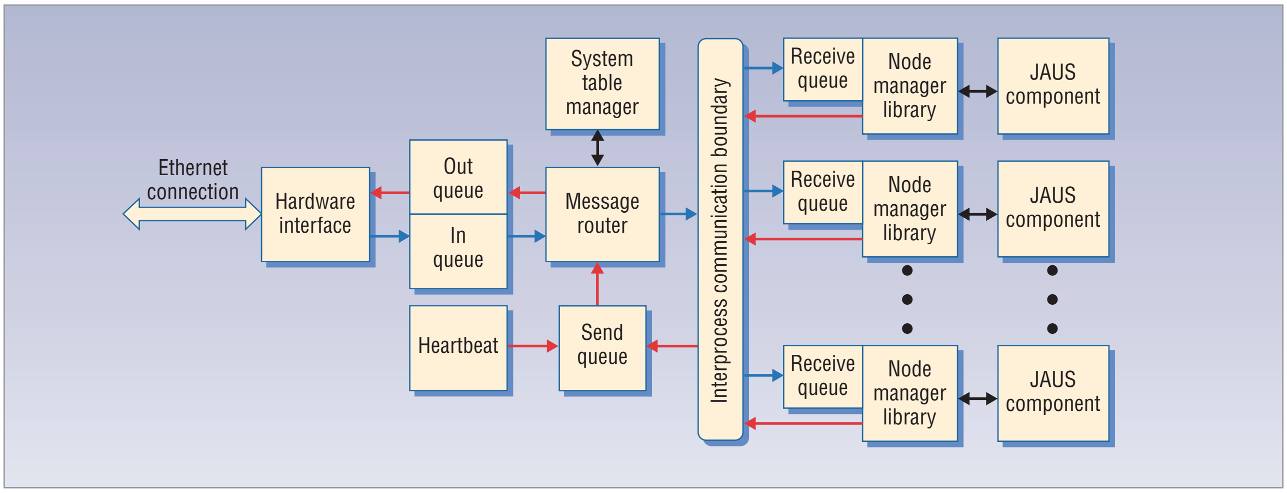

Figure 3. CIMAR implementation of the JAUS node manager. Each component must exchange information by routing all messages through a node manager residing on its own hardware node.

component has requested and received exclusive control of it. Conversely, query and inform messages work in pairs: One component sends a certain query message, and the receiving component responds with the matching inform message, with no concern about controlling components. Numerous other use cases address everything from periodic heartbeats, to component initialization and configuration, to video streaming.

Service connections let a component request the sending of an inform message to another component periodically without the need for additional queries, creating a publish/subscribe relationship between the two components. The system does not queue such inbound inform messages, ensuring that the data that the receiving component uses is always the freshest available. The NaviGATOR uses this capability extensively to gather real-time GPOS and VSS data and to marshal traversability grids.

UNIFYING SENSOR INPUT WITH SMART SENSORS

The smart sensor architecture is based on the idea that each sensor processes its data independently and provides a logically unified interface to other components within the system. This lets system designers create their own technologies and process data in a way that best fits their design. They can then integrate sensors with minimal effort to create a robust perception system.

The primary benefit of this approach is its flexibility. Because each smart sensor consists of a common set of inputs and outputs, system designers can isolate and address an individual sensor's effects on the entire system. Using the accompanying smart arbiter and reactive driver, they can also experiment with sensor suites in a more agile way. They can add and subtract sensors from the system quickly and easily, allowing for evaluation of any combination of sensor or pseudosensor components. The major drawback of this approach is that one sensor cannot exploit another sensor's results when evaluating raw input data.

Smart sensors

Sensors provide the means for an AGV to rationalize its environment and estimate its own state. This in turn enables the vehicle to automatically plan its actions and make decisions. As technology advancements enable sensors to provide more accurate and precise information at lower costs, designers are gradually incorporating more of them into robotic systems.

The multiple types of sensors on a robot can differ greatly in how they perceive information and represent what they sense. Designers and implementers thus face the difficult challenge of interfacing and fusing a proliferation of sensor data in different formats, with varying precision and accuracy and perhaps at different rates.

As Figure 4a shows, the smart sensor unifies heterogeneous sensor designs by combining the sensing, interface, and computational hardware, as well as any software algorithms associated with a particular sensor, into a single unit. Defining a common interface for the associated smart sensors provides a higher level of abstraction. This unification approach standardizes a sensor's abstract notion and essentially allows casting any unique sensor into smart sensor form.

SmartMotor. The smart sensor parallels Animatics Corp.'s SmartMotor motion control system (http://animatics.com). As Figure 4b shows, a SmartMotor integrates all the required elements for effectively controlling an actuator—the controller, amplifier, encoder, and motor itself—into a single package with a common interface. This allows simultaneous control of an array of different actuators, each of unique size, power, and function, over a network via a specified communications bus. The NaviGATOR's throttle, brake, steering, and shifter actuators all use SmartMotor systems.

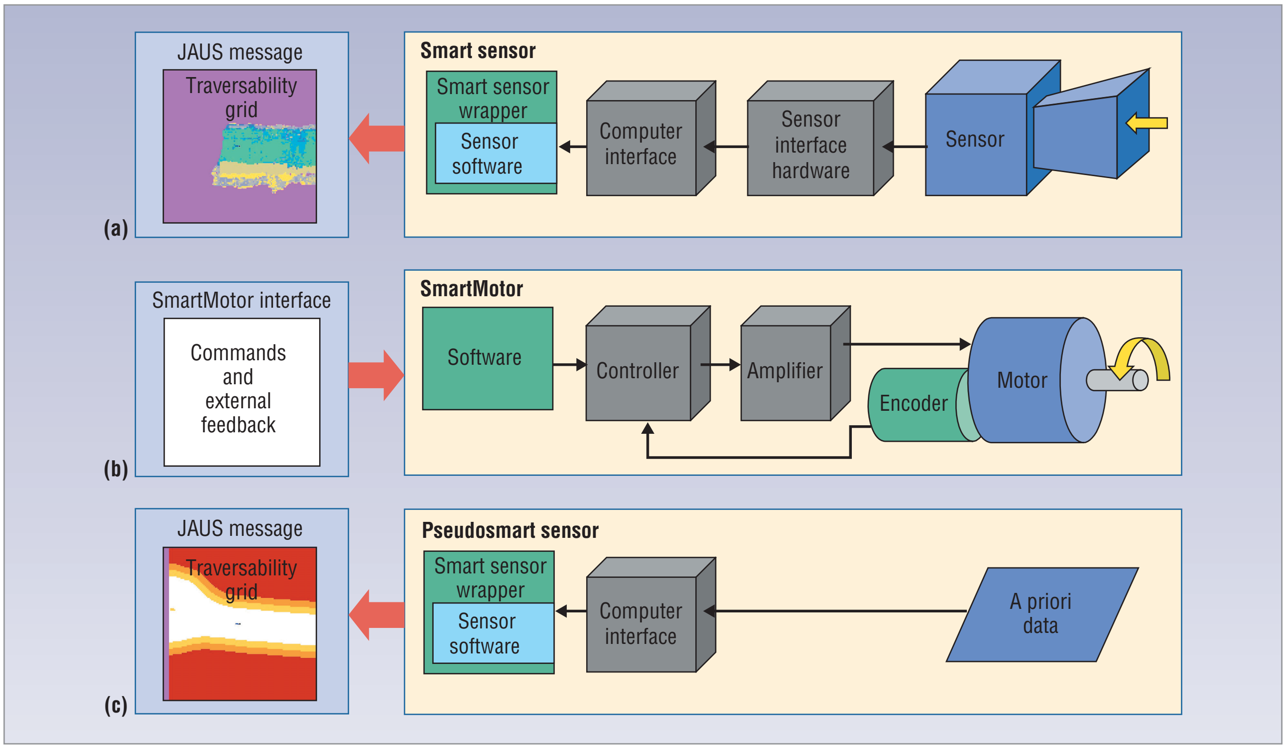

Figure 4. Smart sensors. (a) The smart sensor combines the sensing, interface, and computational hardware, as well as any software algorithms associated with a particular sensor, into a single unit. (b) A SmartMotor integrates all of the required elements for effectively controlling an actuator into a single package with a common interface. (c) Mapping a priori information into a pseudosmart sensor eliminates any need by the intelligence element to receive such information via yet another undefined interface.

In terms of the *sense-plan-act* paradigm,[1] the smart sensor integrates the sensing step the way the SmartMotor integrates the action step. The smart sensor provides sensed information over a network in a common form and delivers it to other parts of the system that use the data for planning and decision making.

Eliminating the need to accommodate multiple sensor-specific interfaces greatly simplifies the process of integrating new sensors into an AGV or any other robotic system. Sensor fusion algorithms need only be tailored to the one particular data interface chosen as the smart sensor interface. It also decreases the amount of time required to bring new sensors into action as they become available. Designers are thus free to give the system plug-and-play-like capability, including or excluding any smart sensor as necessary.

Pseudosmart sensor. Another key discovery that emerged during smart sensor development is that researchers also can map a priori information into an appropriate traversability space. Mapping a priori information into a *pseudosmart sensor*, shown in Figure 4c, eliminates any need by the intelligence element to receive such information via yet another undefined interface. This lets the system utilize just-in-time a priori information via the same interface as an ordinary sensor to obtain a more complete view of the local environment.

Smart sensor wrapper

While the smart sensor unifies the abstract notion of a sensor, numerous additional common internal processes must occur within every sensor component. Therefore, sensor component developers can use a collection of software, the *smart sensor wrapper,* that unifies the storage, localization, formatting, and distribution of perception data among the components that produce or consume it.

GPOS and VSS interface. Every smart sensor must access real-time or near-real-time position and velocity data. The smart sensor wrapper provides a predefined interface for establishing JAUS service connections to the global position and orientation sensor and to the velocity state sensor.

Torus buffer. Team CIMAR also identified the need for a common storage container for traversability values. Prior work in this area necessitated the creation and destruction of large chunks of program memory as well as the manual translation or rotation of that data from frame to frame. Each developer had a unique and sometimes confusing method of handling traversability data storage.

As part of the effort to unify sensor development, the team conceived the notion of a *torus buffer.* Based on linear ring buffers, this is a 2D buffer of traversability values. Each TB contains numerous individual cells that make up the entire traversability grid. As the sensor moves about its environment, the TB automatically cleans cells representing data no longer of interest—that is, cells that have effectively rolled off the grid—and resets them to a default value. It does not move data from cell to cell, but rather maintains an origin position in the grid and references values to that cell. This allows allocating grid memory once at runtime, thereby avoiding costly data copying and reallocation of memory during program execution.

The team incorporated several TB functions into the smart sensor wrapper including the ability to rotate the buffer by the appropriate number of rows and columns, the ability to set and retrieve values in the grid relative to the current origin, and a common method to marshal the grid value data between the TB and the associated JAUS-compliant traversability grid message.

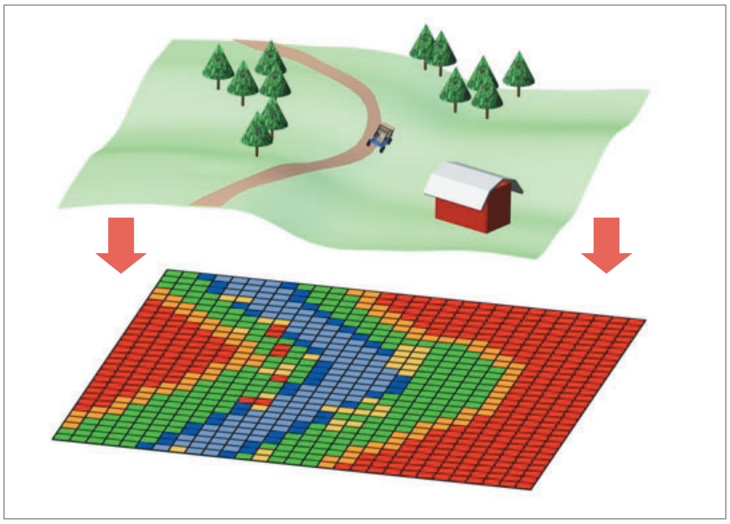
Figure 5. Traversability grid. Sensed data must effectively describe the surrounding local environment before an AGV can plan and execute any motion through it.

Global coordinate system. A significant problem identified during the initial smart sensor implementation was the inability to accurately geolocate a particular grid. The initial approach was to stamp the center of each grid with the appropriate latitude/longitude values, but this led to interpretation differences among the sensor systems, as each smart sensor executes asynchronously on a vehicle in motion. The lack of a common approach in determining the global coordinate system in relation to the local coordinate system created errors during the fusion of data.

To ensure system functionality, a feature registered in a certain cell for one particular sensor needed to be registered in the same cell for all sensors. The team achieved this by implementing a common global coordinate system for sensor readings. Rather than interpret the center position from the associated latitude/longitude values, the system attaches global row/column values for the center cell to each grid message. These values are based on a Universal Transverse Mercator projection and the grid resolution. This action causes small variations in latitude/longitude values to “snap” to the same global row/column values. The smart sensor wrapper enables this transformation of a local point to the global coordinate system for use across all implementations.

TRAVERSABILITY GRIDS FOR INTELLIGENT NAVIGATION

To support the smart sensor framework, Team CIMAR devised a common data structure, the traversability grid, for use by all smart sensors, the smart arbiter, and the reactive driver. This grid is sufficiently specified to let developers work independently and for the smart arbiter to use the same approach for processing input grids regardless of the number at any instant in time.

Traversability grids

To implement a smart sensor system, a common, predetermined data interface must be in place. This interface describes the type of sensed data and its associated format, timing, and transport. These parameters are usually best dictated by the requirements of higher-level planning, decision, and control algorithms. For AGVs, this means the data must effectively describe the surrounding local environment before the vehicle can plan and execute any motion through it, as Figure 5 shows.

The traversability grid is based on Alberto Elfes’ *occupancy grid*, which he defined in 1989 as “a probabilistic tessellated representation of spatial information.”[2] This paradigm has matured over the past two decades,[3] and in recent years the traversability grid emerged as an effort to further expand the occupancy grid’s applicability and utility.[4,5] The primary contribution of the NaviGATOR’s implementation 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 at 20 Hz.

Any type of terrain can be mapped into traversability space without having to convey details about the given topography. Positive obstacles appear the same as negative obstacles or steep slopes. Traversability also allows for the evaluation and scoring of unoccupied space. For example, the system can classify pavement as more traversable than sand, grass, or gravel. This flexibility lets the AGV select the best possible path to navigate when presented with different types of terrain.

Combining different types of sensors, each with a unique perspective on the environment, enables the vehicle to observe the full spectrum of traversability. Some sensors excel at finding hard positive obstacles, while others might only be able to sense terrain smoothness. For example, laser range finders and sonar arrays can effectively determine if space is occupied or free but, depending upon their orientation, might not be able to report anything useful about the terrain's quality. Other types of sensors such as cameras or stereo vision systems can carry out this function, while pseudosensors can transform a priori data of interest into a grid representation. For this reason, each smart sensor's traversability value range can be tuned to the sensor's particular capabilities.

Combining different types of sensors enables the vehicle to observe the full spectrum of traversability.

Traversability grid design

The NaviGATOR's traversability grid currently is 121 rows (0 – 120) × 121 columns (0 – 120), with each grid cell representing a half-meter by half-meter area. The vehicle's reported position always occupies the center cell at location (60, 60). Sensor results are oriented in the global reference frame so that true north is always aligned vertically in the grid. This produces a 60 meter × 60 meter grid capable of accepting data at least 30 m ahead of the vehicle and storing data at least 30 m behind it.

Each cell receives a score ranging from 2 to 12, where 2 means that the cell is absolutely impassable; 12 that the cell is an absolutely desirable, easily traversed surface; and 7 that the sensor has no evidence that the cell's traversability is particularly good or bad. Certain other values indicate out of bounds (0), value unchanged (1), failed/error (13), unknown (14), and vehicle location (15). These discrete values are color coded to help developers visualize the contents of a given traversability grid, from red (2) to gray (7) to green (12).

All of these grid characteristics are identical for every smart sensor, making seamless integration possible with no predetermined number of sensors. All sensors send their grids to the smart arbiter, which is responsible for fusing the data. The arbiter then sends a grid with the same characteristics to the reactive driver, which uses it to dynamically compute the desired vehicle speed and steering.

APPLICATIONS FOR DGC 2005

Team CIMAR applied the smart sensor concept to develop four smart sensors, two pseudosmart sensors, and several intelligent components, including the smart arbiter and reactive driver, for the 2005 DARPA Grand Challenge.

Smart sensors

The *planar LADAR smart sensor* uses a laser detection and range finder mounted at bumper level and aimed to scan a plane horizontal to the ground. Thus, it can only detect positive obstacles and report traversability values from 2 to 7.

The *terrain smart sensor* and *negative obstacle smart sensor* each use a LADAR range finder mounted on the sensor mast (approximately 2 meters above ground) and aimed to scan a plane that intersects with the ground approximately 18 meters and 9 meters, respectively, in front of the vehicle. The TSS assesses terrain smoothness, with the ability to determine some obstacles, while the NOSS assesses negative obstacles, with the ability to determine positive obstacles and terrain smoothness. Both report traversability values from 2 to 12.

The *pathfinder smart sensor* uses a monocular color camera mounted on the sensor mast and direct image assessment to assess terrain smoothness. Thus, it only reports traversability values from 7 to 12.

Pseudosmart sensors

The *boundary pseudosmart sensor* uses a database of allowable corridors, derived from a route data definition file provided by DARPA, and reports traversability in terms of in or out of bounds (0 or 1). The *path pseudosmart sensor* uses a comma-separated value file that represents the path planned by Mobius, an offline deliberative planning tool supplied by team member Autonomous Solutions Inc., and reports traversability values ranging from 8 to 10; these values indicate where the planner would have liked to send the vehicle based on information available during mission-planning efforts.

Intelligent components

The smart arbiter takes as input grids from one or more smart sensors and outputs a single, composite (fused) assessment of traversability. The standardization that the architecture affords makes the smart arbiter very robust with respect to how many and which smart sensors are currently in use. This allowed the team to test the vehicle with whatever sensors were available at a given time.

The reactive driver takes the traversability grid from the smart arbiter and uses it to determine the most appropriate instantaneous steering and speed based on a *receding horizon control* technique. By knowing the current environment blended with a priori data, along with a mathematical model of the AGV, it updates output to the primitive driver 20 times per second to avoid obstacles and seek out the smoothest terrain within the vehicle's capabilities.

The smart sensor architecture also led to the creation of a *smart sensor visualizer* that can be pointed at any component that sends out a traversability grid message to display a color-coded image of that grid.

By combining smart sensors and traversability grids with a JAUS-based component and messaging architecture, Team CIMAR was able to quickly develop a robust AGV platform with advanced sensing and planning capabilities and field a viable finalist in perhaps the most important robotic competition ever held. As the November 2007 DARPA Urban Challenge approaches, Team Gator Nation is extending the scope and features of the technologies that Team CIMAR developed.

The torus buffer now may contain any valid data type or a pointer to a data structure, enabling a component designer to use the TB to track more complex information for each grid cell, such as object height, slope, and variance, or a linked list of historical values. In addition, the team is exploring ways to couple a high-resolution, close-range traversability grid with a low-resolution, long-range one. We are also extending the scoring system for each cell value to address features beyond traversability, such as pavement markings, lane identification, and predicted motion of moving objects. Finally, the situation assessment, decision-making, and world modeling features that were in their infancy in 2005 are playing a major role in addressing an order-of-magnitude increase in the complexity of the behaviors and tasks required for success in the Urban Challenge. ■

Acknowledgments

The authors thank the students and engineers of Team CIMAR, especially title sponsor Smiths Aerospace, and acknowledge the generous financial support of the University of Florida.

References

1. N.J. Nilsson, *Artificial Intelligence: A New Synthesis,* Morgan Kaufmann, 1998.
2. A. Elfes, "Using Occupancy Grids for Mobile Robot Perception and Navigation," *Computer,* June 1989, pp. 46-57.
3. S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous Robots,* vol. 15, no. 2, 2003, pp. 111-127.
4. H. Seraji, "New Traversability Indices and Traversability Grid for Integrated Sensor/Map-Based Navigation," *J. Robotic Systems,* vol. 20, no. 3, 2003, pp. 121-134.
5. C. Ye and J. Borenstein, "T-transformation: Traversability Analysis for Navigation on Rugged Terrain," *Proc. SPIE,* vol. 5422, SPIE, 2004, pp. 473-483.

Bob Touchton *was a PhD candidate and graduate research assistant at the University of Florida's Center for Intelligent Machines and Robotics (CIMAR) and is currently a managing director in the Business Innovation Center of Honeywell Aerospace. His research interests include real-time adaptive planning, situation assessment, and decision-making for intelligent systems. Touchton received an MS in nuclear engineering from Carnegie Mellon University and an MS in computer science from the University of North Florida. He is an active JAUS Working Group participant and a member of the IEEE, the American Association for Artificial Intelligence, the Society of Automotive Engineers, and the Association for Unmanned Vehicle Systems International. Contact him at btouch@ufl.edu.*

Tom Galluzzo *was a PhD candidate and graduate research assistant at the University of Florida's CIMAR and is currently a senior engineer with Harris Corporation. His research interests include planning and control and software engineering for intelligent systems. Galluzzo received a BS in aerospace engineering from Embry-Riddle Aeronautical University. He is a member of the IEEE and the AUVSI. Contact him at galluzzt@ufl.edu.*

Danny Kent *is a PhD candidate and graduate research assistant at the University of Florida's CIMAR. His research interests include dynamic world modeling and software engineering for intelligent systems. Kent received a BS in aerospace engineering from Embry-Riddle Aeronautical University. He is an active JAUS Working Group participant and a member of the IEEE and the AUVSI. Contact him at kentd@ufl.edu.*

Carl Crane *is a professor in the University of Florida's Department of Mechanical and Aerospace Engineering and director of CIMAR. His research interests include autonomous ground vehicles and compliant systems. Crane received a PhD in mechanical engineering from the University of Florida. He is an active JAUS Working Group participant, an American Society of Mechanical Engineers Fellow, and a member of the American Nuclear Society and the AUVSI. Contact him at ccrane@ufl.edu.*