Planning and modeling extensions to the Joint Architecture for Unmanned Systems (JAUS) for application to unmanned ground vehicles

Robert Touchton[a], Daniel Kent[a], Tom Galluzzo[a], Carl D. Crane III[a], David G. Armstrong II[a], Nick Flann[b], Jeff Wit[c], Phil Adsit[d]

[a]University of Florida ,Center for Intelligent Machines and Robotics, Gainesville, Florida

[b]Autonomous Solutions, Inc., Young Ward, Utah

[c]Wintec, Inc., Tyndall AFB, Florida

[d]Applied Research Associates, Tyndall AFB, Florida

ABSTRACT

This paper describes how the functionality of the DoD Joint Architecture for Unmanned Systems (JAUS) was expanded by developing experimental components related to off-line and reactive path planning and world modeling. JAUS defines a set of reusable components and their interfaces. In order to ensure that the JAUS architecture will be applicable to the entire domain of unmanned mobile systems, the following four characteristics were considered throughout its development: vehicle platform independence, mission isolation, hardware independence, and technology independence. The new experimental components described in this paper have these same characteristics.

Existing JAUS components readily allow for autonomous path tracking. In this work the JAUS Version 3.0 Global Path Segment Driver, Global Pose Sensor, Velocity State Sensor, and Primitive Driver components were utilized for that task. What is described in this paper is the development of the following new experimental components: World Model, Sensor Arbiter, and Path Manager. The functionality and interface messages of each of these components are presented, followed by a discussion of the performance of the overall system.

Keywords: autonomous navigation, path planning, world modeling, autonomous decision-making

1. INTRODUCTION AND JAUS OVERVIEW

The system architecture implemented by the authors is based on the Joint Architecture for Unmanned Systems (JAUS) Reference Architecture, Version 3.0[1]. JAUS defines a set of reusable components and their interfaces. In order to ensure that the architecture will be applicable to the entire domain of unmanned mobile systems, the following four characteristics have been considered:

1. Vehicle platform independence. In order for JAUS components to be interoperable, no assumptions about the underlying vehicle or its means of propulsion are made.
2. Mission isolation. The JAUS components can typically be assembled such that a variety of missions can be supported.
3. Computer hardware independence. No assumption of or requirement of particular computer hardware is made. This allows for future adaptability and enhancement as new computer hardware becomes available in the future.
4. Technology independence. This is similar to the computer hardware independence, but focuses more on the technical approach rather than the computer hardware. For example, there are many approaches that could be used to determine vehicle position and orientation. No one approach, such as for example GPS, inertial dead reckoning, or landmark-based navigation is specified.

2. SYSTEM ARCHITECTURE

The system architecture that forms the framework for this paper was formulated using existing JAUS-specified components wherever possible along with a JAUS-compliant inter-component 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. Experimental Components and User-defined Messages enable researchers to leverage existing JAUS infrastructure where practical and to ensure that these new components and messages are in alignment with the JAUS principles should they eventually be adopted as fully sanctioned JAUS components or messages.

The origination of the new developments and approaches discussed in this paper were performed by the University of Florida and Autonomous Solutions, Inc. authors as part of their efforts in preparing for the 2004 DARPA Grand Challenge. Once that event was over, these ideas were carried over to the Air Force Research Lab for further refinement.

At the highest level, the architecture consists of four fundamental elements, which are depicted in Figure 1:

- Planning Element: The components that act as a repository for *a priori* data. Known roads, trails, or obstacles, as well as acceptable vehicle workspace boundaries. Additionally, these components perform off-line planning based on that data. Advances in off-line planning are discussed in Section 2.1.
- Control Element: The components that perform closed-loop control in order to keep the vehicle on a specified path. Advances in path tracking are discussed in Section 2.2.
- Perception Element: The components that perform the sensing tasks required to locate obstacles and to evaluate the smoothness of terrain. This topic is discussed in a companion paper, "Development of an integrated sensor system for obstacle detection and terrain evaluation for application to unmanned ground vehicles," paper #5804-18.

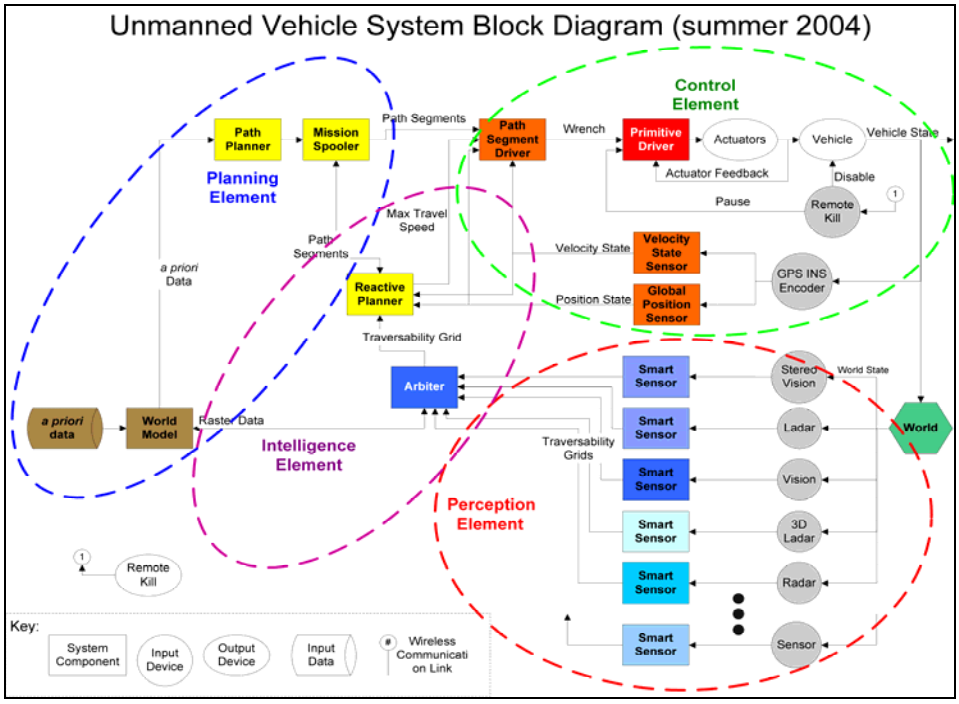

Figure 1: System Architecture

- Intelligence Element: The components that act to determine the 'best' path segment to be driven based on the sensed information. Generation of a single traversability grid that combines the inputs from multiple sensors is discussed in Section 2.3 while on-board decision-making is discussed in Section 2.4.

2.1. Off-line Planning

CIMAR has developed a collection of JAUS components that are categorized under the global umbrella of World Modeling. The primary goals of the World Modeling components are to provide an *a priori* repository for spatial information about the robot's operating environment as well as provide functionality to retain newly discovered spatial data from the environment throughout a robot's mission. The *a priori* knowledge is used to pre-plan mission paths and goals. Information in the form of boundary polygons can also be used to limit the robot's operating environment. The information collected about the environment is used for mission analysis and can be leveraged for planning in subsequent missions.

To accomplish the first goal, a suitable method to both store and provide *a priori* spatial data within the JAUS framework was required. Work towards that goal was done by the World Modeling subcommittee within the JAUS working group and provided a collection of JAUS messages and components suitable to the storage and recall of vector and raster spatial data.[2] The World Model Vector Knowledge Store (WMVKS) is defined as the component responsible for the handling of vectorized spatial data. Each feature defined within the knowledge store includes various metadata about that object which may be of interest to the robotic platform. Metadata may be a high level description, such as obstacle, road or boundary, or more detailed, such as trees, rocks, or sidewalks.

To date, CIMAR has concentrated its efforts on the storage and collection of vector spatial data and is making use of data obtained from the United States Geological Survey's digital data sets. These data sets include information about roadways, railways, lakes, and rivers along with other less robot-sensitive data, such as land-use polygons and county boundaries. Additional *a priori* knowledge in the form of known obstacles and alternate routes has also been employed in various mission tasks. The World Model Vector Knowledge Store is flexible enough to handle a variety of spatial data types and attach metadata to each type.

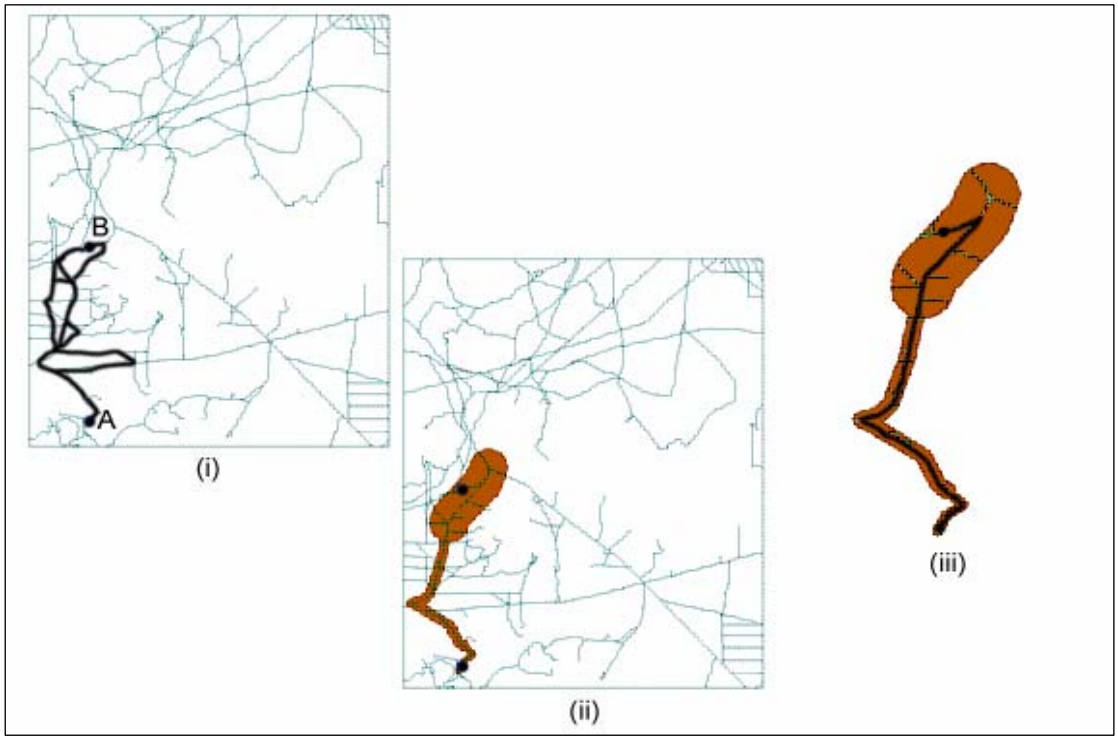

Figure 2: (i) Some possible routes from point A to point B. (ii) Boundary polygon applied to search space. (iii) Optimal solution within search space

The use of *a priori* boundary information enhances the system's ability to plan mission paths. The incorporation of boundaries provides a limit to the search space. As such, the optimal path is more readily determined prior to the mission. For example, given a start point and end point, it is desired to find the most optimal path along a road between them. As shown in Figure 2, if the robot is given solely a network of roads (i), a unique solution cannot be found. However, if the system is further provided with information about where it can and cannot traverse (ii), information outside the traversable area can be ignored. Here (iii), the solution is finite.

Boundary information can also provide the robotic system with valuable real-time information during obstacle avoidance and path re-planning. Knowledge of traversable and non-traversable areas, as defined prior to mission start, assures that the vehicle does not travel outside the given bounds. CIMAR has chosen to provide this real-time boundary information to the vehicle using the Smart Sensor traversability grid format used by its suite of sensors. The traversability information is provided to the Smart Arbiter component in the form of a raster grid centered on the vehicle (further described in Section 2.3). Boundary polygon information is provided to the system by the Boundary Smart Sensor (BSS) component. The BSS is responsible for obtaining the boundary information from the WMVKS and the systems global position from the Global Position and Orientation Sensor (GPOS) component. The BSS uses this data to determine the drivable and non-drivable portions of the traversability grid. Figure 3 shows a typical grid output from the BSS indicating the system's location within the grid and the drivable region around it. Combining this traversability data with detected obstacles allows the system to intelligently plan future paths based on mission goals and sensed data.

Figure 3: Traversability Grid showing boundary data

The final goal of work being performed by CIMAR towards World Modeling is the capture and retention of spatial data from a raster grid. Once the collection of sensor data is fused within the Smart Arbiter component; the output raster grid is returned to the World Modeling suite of components to be analyzed for feature extraction. The World Model component analyzes the traversability grids, ignoring areas outside the previously identified boundary. The collection of resultant points (one point centered in each grid cell) are then ordered by their traversability value; currently a number between 0 and 255 (Figure 4(i)). The raster grid is then encapsulated into polygons using a convex hull approach to determine the edges of the collection of data points (Figure 4(ii)). Due to the nature of most obstacles and their resulting grids, many times there will be a large collection of small clusters of like-valued data. For this reason the encapsulation of all these points is often not a polygon representing the sensed object. To overcome this issue, the World Model component first surveys the data looking for collections of points within a tolerance of each other. "Orphan" points not within the minimum tolerance of any other like-valued points are removed (Figure 4(iii)).

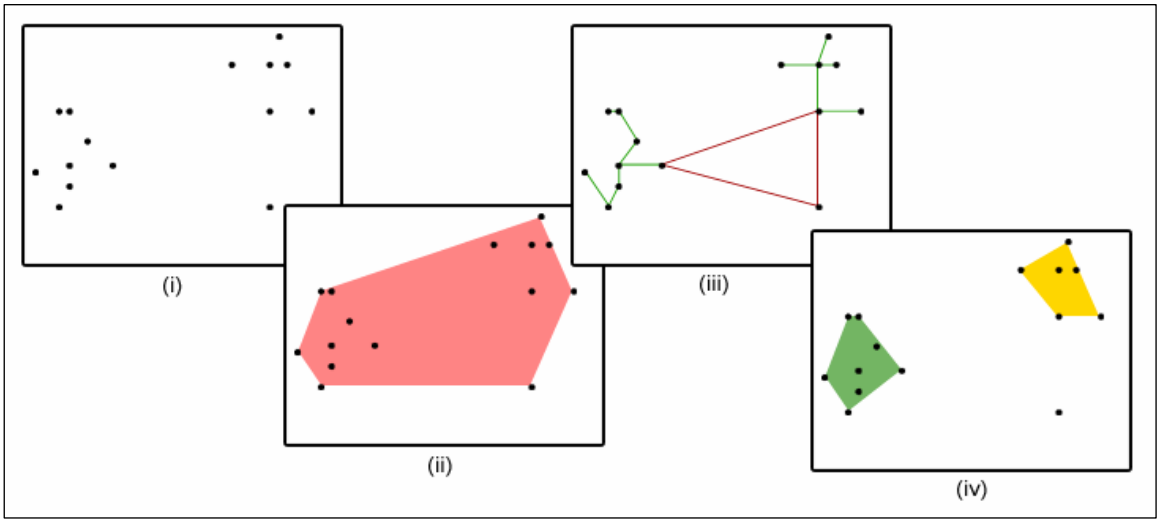

Figure 4: Conversion process from raster grid to vector polygons

The resulting sets of like-valued data points are then analyzed using the convex hull approach to generate the resultant polygonal data (Figure 4(iv)).

With the boundaries, road network and obstacles in place in the World Model, the (off-line) Planner can generate a series of JAUS-compliant global path segments that will take the vehicle from its starting position to the goal position. The path segments for the mission are then stored in the Mission Spooler, another Experimental JAUS Component.

2.2. Path Tracking

The CIMAR autonomous vehicle design requires the navigation of global path segments. These segments describe the desired motion of the vehicle through its environment. A path tracking control system is implemented to achieve this objective. The control element consists of four JAUS components: a Global Position and Orientation Sensor (GPOS), a Velocity State Sensor (VSS), a Primitive Driver (PD), and a Global Path Segment Driver (GPSD).

The GPOS and VSS provide the feedback information necessary to achieve closed loop path tracking. This information is provided from an Inertial Navigation System (INS), which fuses data from a GPS, inertial sensors, and a vehicle odometer. The data are collected by an onboard computer, and transformed into their JAUS form at a rate of 60 Hz. These data are then available to any JAUS component connected to the GPOS and/or VSS.

The low-level vehicle control is achieved by actuating the steering, throttle, brake and shifting mechanisms. Each of theses actuators are controlled simultaneously by the JAUS PD component. The PD is responsible for taking a percent wrench effort command from another onboard component or from a remote Operator Control Unit (OCU) and transforming it into a specific input command to each of the actuators. The wrench describes the linear and rotational propulsive efforts, which translate directly to the throttle and steering input commands respectively. Additionally, a linear resistive effort is provided, which translates to the brake input command. Shifting commands to the PD are separate from wrench effort commands. They are generated and sent from a controlling component on an event basis that depends on the velocity state of the vehicle.

The GPSD is responsible for closing the loop between the PD and the motion feedback components. The control law internal to the component is designed to provide stable and robust path tracking over the full speed range of the platform. It also controls the speed of the vehicle along the path. The input to the path segment driver is a sequence of JAUS global path segments that describe, by latitude and longitude coordinates, the geometric path that the vehicle is desired to follow. Each path segment also contains a recommended traversal speed value. This value is used as an input upper bound to the internal speed controller. In addition, higher level components connected to the GPSD can recommend speed values depending upon information such as how cluttered the environment around the vehicle may be, or what the current weather conditions are like. The actual input to the GPSD speed controller is then the minimum of all recommended speed values.

Obtaining robust path tracking is a challenging problem. Traditional linear compensators such as PID or State Space controllers can be applied, but they will not yield global stability. This means that if the vehicle is substantially off course, it will not converge back onto the path, i.e. the linear controllers will only stabilize the vehicle when it is sufficiently close to the desired path segment. Furthermore, a single set of gains or parameters for these controllers cannot adequately stabilize the vehicle throughout its full speed range. A set of gains tuned for platform operation at high speeds can make the vehicle unstable, or at best, marginally stable at low speeds, and vice versa. For some vehicles is may be possible to find a unique set of gains that make the vehicle stable throughout its speed range, however the transient response will probably not be of satisfactory performance.

To overcome the deficiencies of a linear system CIMAR has developed a nonlinear path-tracking controller. The nonlinear controller is designed with the use of a unique error system, which is a function in terms of the perpendicular distance between the vehicle and the path (cross-track error), and the angular difference between a line tangent to the path and the vehicle heading (heading error):

$$e = \arctan(k_1 e_{xt}) + e_h \tag{1}$$

where the cross-track error is given by e_{xt} and the heading error is e_h. The gain k_1 denotes the multiplicative inverse of an arbitrary reference distance.

This error signal has a unique property in that as the cross track error becomes very large, the total error will go to zero if the heading error is $-\frac{\pi}{2} sign(e_{xt})$. Thus, the vehicle will drive towards the path along a trajectory nearly perpendicular to it when the cross track error is sufficiently large. This solves the problem of obtaining global tracking stability, and has the advantage that the vehicle will always drive along an intuitive stable recovery trajectory.

The controller is formulated with the use of Lyapunov stability theory.[3] The theory states that if we have any function $V(x)$ such that:

$$V(x) > 0 : \forall x \neq 0 \text{ (Positive definite)}$$
$$\dot{V}(x) < 0 \text{ (Negative definite)}$$

then:

$$\lim_{t\to\infty} \|x(t)\| = 0 \text{ (} x(t) \text{ is stable and converges to zero) .} \quad (2)$$

From this, one can then arbitrarily define:

$$V(e) = \tfrac{1}{2} e^2 \quad (3)$$

This is a positive definite function in terms of the new error signal. The time derivative of $V(e)$ is then derived to be negative definite. This proof begins with the analysis of the time derivative of the error signal e:

$$\dot{e} = \frac{k_1 \dot{e}_{xt}}{1 + k_1^2 e_{xt}^2} + \dot{e}_h \quad (4)$$

where

$$\dot{e}_{xt} = v \sin(e_h)$$
$$\dot{e}_h = vu$$

Where v is speed of the vehicle and *u* is the path curvature, which is the input to the system. Since *u* has a direct relationship to the steering angle, which is under computer control, path curvature can be assumed as the plant input in order to simplify the design.

From this a nonlinear inversion of the system is performed, via our control input *u*, by designing it to be:

$$u = \frac{1}{v} k_2 e - \frac{k_1 \sin(e_h)}{1 + k_1^2 e_{xt}^2} \quad (5)$$

Then, by substitution

$$\dot{V}(e) = -k_2 e^2 \text{ (Negative definite)} \quad (6)$$

This proves that the controller design of *u* makes the error signal (*e*) globally exponentially stable.

To support and tune the controller design obtained for our platform, simulations of both a PID and Nonlinear controller were analyzed and compared. Both controllers showed that the vehicle was stable about its operating path. However, a key difference in the performance of the controllers was observed when the position and heading disturbance of the vehicle sufficiently large. The PID control showed an undesirable recovery trajectory, where both the time and distance to path convergence were significantly larger than that of the nonlinear controller (Figure 5 vs. Figure 6). In addition, the nonlinear controller produced a more desirable stabilizing trajectory.

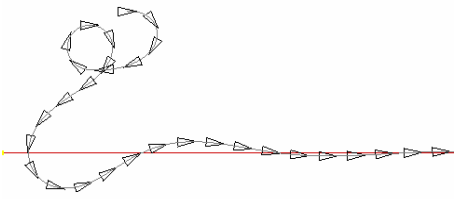

Figure 5: PID controller trajectory with large initial cross-track error

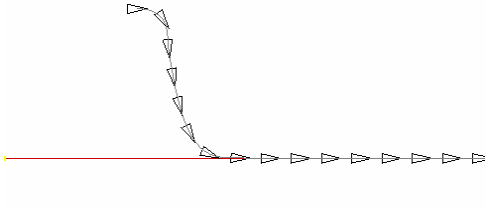

Figure 6: Nonlinear controller trajectory with large initial cross-track error

2.3. Sensor Arbitration

In order to discuss the notions of sensor arbitration and decision-making, a summary of the Perception Element's concept of operations is needed. First, a common data structure, dubbed the Traversability Grid, was introduced for use by all sensors, the arbiter, and the planners (Figure 7). This grid was sufficiently specified to enable developers to work independently for the arbiter to use the same approach for processing input grids no matter how many there were at any instance in time.

The Traversability Grid design called for 121 rows (0 – 120) by 121 columns (0 – 120), with each grid cell representing a half-meter by half-meter area. The vehicle is placed in the center cell (location 60,60) and the sensor results are oriented in a global frame of reference (i.e., North is always the top of the grid). In this fashion, a 60m by 60m grid is produced that is able to accept data 30m ahead of the vehicle and store data 30m behind it.

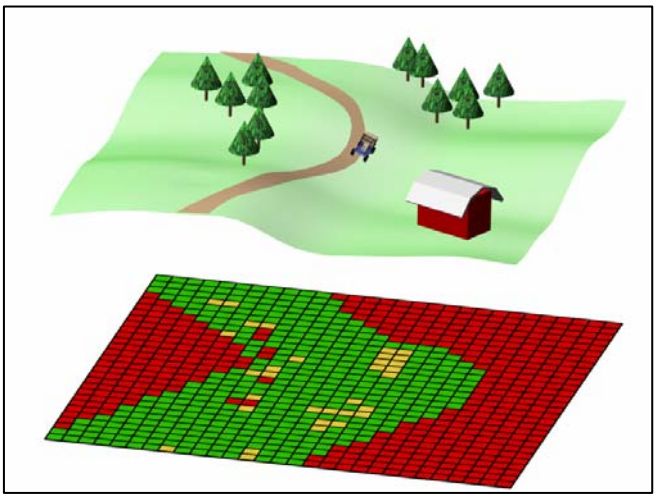

Figure 7: Traversability Grid Portrayal

The scoring of each cell is based on mapping the sensor's assessment of the traversability of that cell into a range of 0 to 255 where 0 means that there is absolutely an insurmountable obstacle detected in that cell, 255 means there is absolutely a desirable, easily traversed surface in that cell, and 127 means that the traversability of that cell is unknown.

The messaging concept for marshalling grid cell data from sensors to the arbiter and from arbiter to planner is to send only those cells that have changed. Thus, for each scan or iteration, the sending component must 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). 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).

With the Traversability Grid 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. To accomplish this, the Arbiter must take in each new message from a given sensor, adjust the center-point of its data set (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) and apply the new data values to the baseline grid for that sensor. This step must be repeated for each sensor that has sent a message. At this point, all input grids are aligned and contain the latest findings from its source sensor. Now the arbiter must traverse the input grids simultaneously, cell-by-cell, and merge the data from each corresponding cell into a single output value for that row/column location. Naturally, this process can be optimized by keeping tabs on which cells experienced a change and, if no change occurred in any of the input cell values, then there is no need to compute a new output value for that cell. In early testing, a simple average of the input cell values was used as the output cell value. Future work will investigate other algorithms, including heuristic ones, to perform the data fusion task. The Arbiter component was designed to make it easy to experiment with varying fusion algorithms in support of on-going research. Once all cells have been treated in this fashion, the Arbiter packs up its output grid message and sends it on the dynamic planner and the World Model for assessment and storage of obstacle polygons.

2.4. Decision-making

The generation of an initial plan, as alluded to in the closing paragraph of Section 2.1, is a complex task beyond the scope of this paper. However, given that such a plan has been generated, one can readily conclude that it will have to be adapted to changes that occur in the environment (e.g., detection of obstacles not previously stored in the World Model), errors in the position and orientation of the vehicle (off-plan drift), the presence of smooth areas not explicitly

represented in the road network, and the like. For reasons such as these, the *a priori* plan must be under constant scrutiny to determine whether and how it must be updated.

The Reactive Planner (RP) experimental component was devised to perform this task. It monitors the global path segments that are in close proximity of the vehicle. It analyzes each Traversability Grid received from the Arbiter component looking for areas that indicate the intersection of a path segment to be driven by the vehicle with a newly-found obstacle. The Reactive Planner design also includes the ability to look for promising paths within the Traversability Grid that would lead to improved performance by the vehicle if that path were followed instead of the path originally planned (as manifested in the current set of *a priori* path segments); however, this aspect of the design has not yet been implemented.

The concept of operation for the Reactive Planner is to continually evaluate the estimated cost of the current path being followed using the contents of the Traversability Grid, which is being continually updated asynchronously. If the estimated cost becomes higher than a threshold, this may indicate that an obstacle has been detected which will interfere with the vehicle's current path. The first action taken is to slow the vehicle, thereby giving the system more time to determine the correct avoidance action if needed, and obtain more reliable sensor data, which may clear the potential obstacle. Since the Traversability Grid only includes data a maximum of 30 meters ahead of the vehicle, the Reactive Planner need only look-ahead by the number of path segments that correspond to that distance (and that number will vary over time since some path segments may be only a fraction of a meter in length, while others may be many meters). Once a potential collision course is detected, the RP then executes a dynamic (re)planning process that attempts to rapidly replace an appropriate number of pre-planned path segments with a new set of path segments that:

1. Avoids the obstacle
2. Is legally drivable by the vehicle (e.g., turning radius constraints)
3. Reconnects with the pre-planned route as soon as practical
4. Never takes the vehicle outside of the current boundary constraints
5. Takes advantage of relevant *a priori* data (e.g., road network)
6. Minimizes the cost of the new path segments

The Reactive Planner maintains a parallel grid, called the Path Grid, containing a modified form of the cost information in the Traversability Grid. The Path Grid takes into account the footprint of the vehicle, so that high cost pixels in the Traversability Grid are posted as contiguous groups of high cost pixels in the Path Grid. This way, paths through the Path grid take into account clearance issues with potential obstacles and other higher cost points in the Traversability Grid. The Reactive Planner posts a set of local sub-goals which will be used to plan a path through the Path Grid. These sub-goals are posted along the current path being followed in front of the vehicle, on the current pre-planned path, at the edge of the Path Grid, around areas of high cost, and within regions of low cost. Then a small set of low-cost routes is computed connecting these sub-goals such that progress is made towards the edge of the Grid. The cost for each route is estimated by considering the cost to traverse the path segments from the grid element costs in the Path Grid, the traversable velocity and the path segment length. Only routes with lower cost compared to the estimated cost of following the current path are considered. A fast, constrained A* search is then performed to identify possible routes that leave the currently followed path, pass through one or more local sub-goals, then reconnect with the pre-planned path within or at the edge of the Path Grid.

An anytime "iterative deepening" algorithm is used to control the Reactive Planner. Initially, the vehicle is slowed and a search for short low-cost paths around obstacles is done. If no path is found, the vehicle speed is further reduced; while a more extensive search is performed (with more local-sub-goals posted) and the acceptable cost threshold increased on plausible paths connecting the sub-goals. Once a path is found it is sent to the Mission Spooler. However, if no path is found the process continues until the vehicle is stationary, and the acceptable cost threshold is at it highest level (implying that the path found will be just traversable at very slow speeds).

Initial studies have shown that the re-planning process can run at approximately 10 Hz, and the vehicle is able to detect and smoothly avoid obstacles using only a Ladar sensor.

3. SYSTEM INTEGRATION AND WORK TO DATE

The most daunting task of all was integrating these components such that an overall mission could be accomplished. Figure 8 portrays the vision of how the key components work together to control the vehicle. Prior to beginning a given mission, the *a priori* Planner uses knowledge stored in the World Model to build the initial path, which it stores in the Mission Spooler component as a series of global path segments. Once the mission is begun, the Path Segment Driver sequentially executes each path segment in the Mission Spooler via the Primitive Driver. Meanwhile, the various sensors begin their search for obstacles and/or smooth surfaces and feed their findings to the Arbiter. The Arbiter performs its data fusion task and sends the results to the Reactive Planner and the World Model. The latter feed is used for persistent storage of any obstacles after conversion from raster to polygon representation. The Reactive Planner peeks at the near-range contents of the Mission Spooler and looks for interferences or opportunities based on the feed from the Arbiter. If a change in one or more path segments is needed, the Reactive Planner dynamically generates a new plan (manifested as some number of global path segments), which overwrites the obviated portion of the original plan in the Mission Spooler. Finally, the vision is to perform this sequence iteratively on a sub-second cycle time (5 to 60 Hz, depending on component).

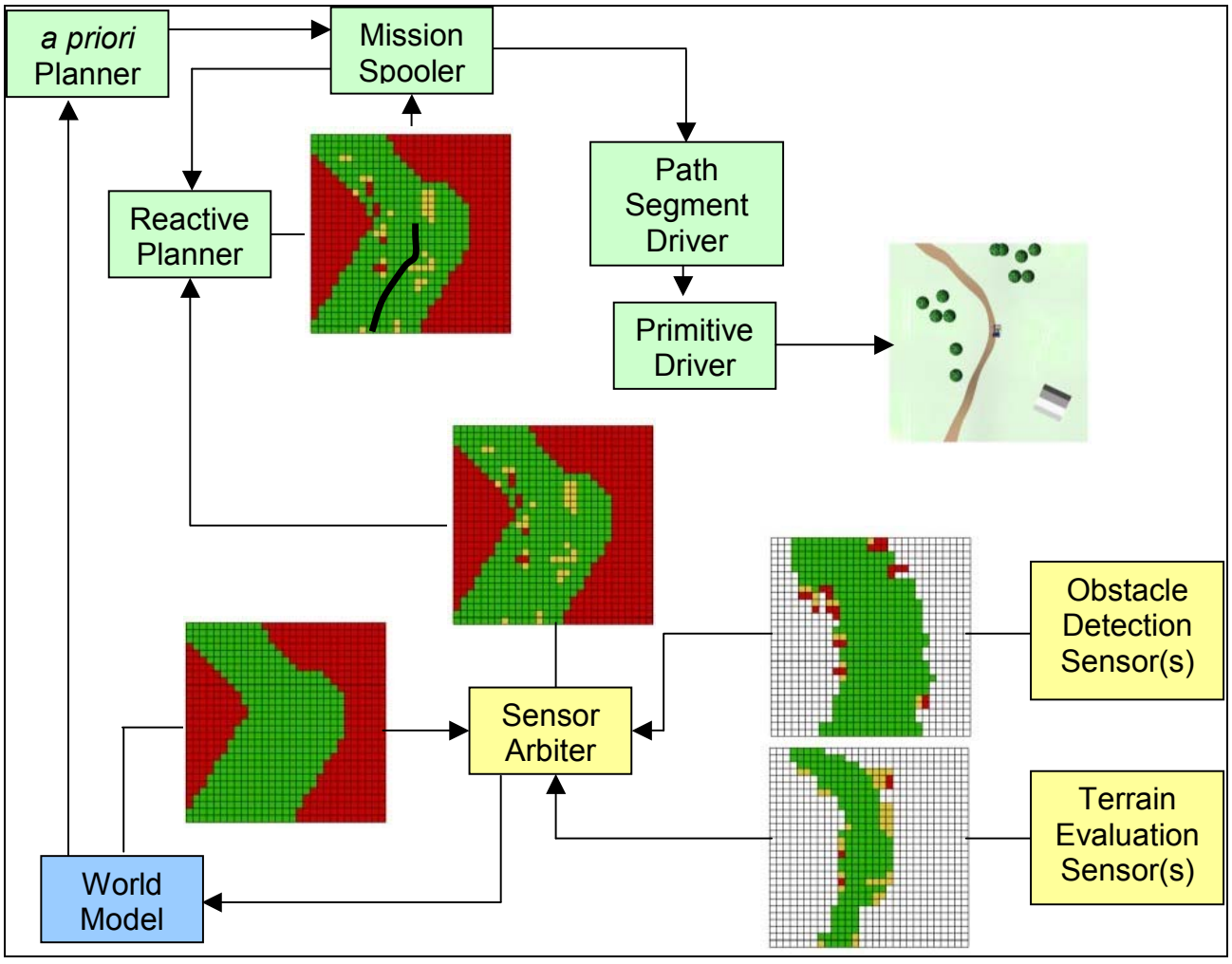

Figure 8: Integrated JAUS and Experimental Components

Much of this vision was accomplished. The process used was to get each component working "on the bench" in a controlled, laboratory environment. Every component discussed was fully operational for this "unit testing" step. Next, various combinations of components were integrated and tested, initially in the lab and later, in the field. Although the envisioned design has not yet been field tested, the results of testing portions of it thus far are encouraging. For example, real-time (sub-second) arbitration of two sensors feeding into the Reactive Planner was achieved during a field test. As might be expected, there were many lessons learned during this integration-testing phase.

Based on the testing performed, the usefulness of having a cell value granularity of 0 to 255 came into question. As a result, it has been decided to simplify this range to be 0 to 15, with 7 being neutral. This range still gives us sufficient differentiation betweens "shades" of good and bad traversability estimates without sacrificing accuracy. It also gives us the option to pack two cells per byte when packing the JAUS message, leading to a 50% savings in message bandwidth.

Another issue discovered was that the changed-cell reporting technique suffered from two shortcomings. First, if the scene is not nominal and a large number of cells must be transmitted, the technique actually increases network traffic and processing load, since each cell position requires three parameters (row, column, and value). Second, since message delivery is not guaranteed, a set of changes could be lost, thus corrupting the target component's baseline view against which it will apply future changed cell messages. To overcome these shortcomings, an alternative messaging technique has been added that addresses these two problems. Now, a component can send a rectangular block of cells (including the entire grid, if desired) by merely specifying the row/column of the opposite corners followed by the values of the encapsulated cells. So, whenever the changed-cell technique begins to approach one-third of the total number of cells in the grid, or if the changes are highly localized, the component can switch to sending the rectangular block of cell values and reduce the number of bytes that must be transmitted over the network and handled by the receiving component. Also, the component can periodically send an entire grid to reset the baseline in order to mitigate the impact of any changed-cell messages that might have been lost.

4. SUMMARY AND CONCLUSION

This paper has presented several new components that have been developed to augment the Joint Architecture for Unmanned Systems (JAUS). The existing Version 3.0 JAUS components allow for vehicle path following and a limited degree of sensor (obstacle avoidance) information to be collected. The new components presented here allow for the storage of *a priori* information, off-line planning, the integration of a variety of obstacle detection and terrain evaluation sensors, and the ability to re-plan a path on-the-fly based on sensed environment information. To date, testing of these new experimental components has been accomplished at slow vehicle speeds (less than 15 mph). Current efforts are focused on the complete integration and deployment of the system on a vehicle capable of traveling at speeds up to thirty-seven miles per hour.

ACKNOWLEDGEMENTS

This work was supported in part by the Air Force Research Laboratory at Tyndall Air Force Base, Florida, contract number F08637-02-C-7022.

REFERENCES

1. Joint Architecture for Unmanned Systems (JAUS) Reference Architecture, Version 3.0, http://www.jauswg.org/.
2. Carl P. Evans, III, "Development of World Modeling Methods for Autonomous Systems Based on the Joint Architecture for Unmanned System," M.S. Thesis, University of Florida, 2004.
3. J.J.E. Slopine, & W. Li,, *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, NJ, 1991.