Proceedings of SPIE Vol. 3693
AeroSense Session on Unmanned Ground Vehicle Technology
April 7-8, 1999

4-D/RCS Reference Model Architecture for Unmanned Ground Vehicles

James S. Albus

Intelligent Systems Division
Manufacturing Engineering Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899
[3693-02]

ABSTRACT

4-D/RCS is the reference model architecture currently being developed for the Demo III Experimental Unmanned Vehicle program. 4-D/RCS integrates the NIST (National Institute of Standards and Technology) RCS (Real-time Control System) with the German (Universitat der Bundeswehr Munchen) VaMoRs 4-D approach to dynamic machine vision. The 4-D/RCS architecture consists of a hierarchy of computational nodes each of which contains behavior generation (BG), world modeling (WM), sensory processing (SP), and value judgment (VJ) processes. Each node also contains a knowledge database (KD) and an operator interface. These computational nodes are arranged such that the BG processes represent organizational units within a command and control hierarchy.

Keywords: Intelligent control, unmanned ground vehicles, real-time control architecture, RCS, 4-D/RCS

1. INTRODUCTION

The 4-D/RCS architecture being developed for the Demo III Experimental Unmanned Vehicle program [1] consists of a hierarchy of computational nodes each of which contains behavior generation (BG), world modeling (WM), sensory processing (SP), and value judgment (VJ) processes. [2, 3] Each node also contains a knowledge database (KD) and an operator interface. These computational nodes are arranged such that the BG processes represent organizational units within a command and control hierarchy. A typical node is illustrated in Figure 1. Each BG process includes a planner module that accepts task command inputs from its supervisor and generates coordinated plans for subordinate BG processes. The BG planner hypothesizes tentative plans, WM predicts the probable results, and VJ evaluates the results of each tentative plan. The BG planner then selects the tentative plan with the best evaluation to be placed in the plan buffers in the BG Executors. There is an Executor that services each subordinate BG unit, issuing subtask commands, monitoring progress, compensating for errors and differences between planned and observed situations in the world, and reacting quickly to emergency conditions with appropriate actions. Feedback from a real-time knowledge database KD enables the executors to generate reactive behavior. SP and WM processes update the KD with images, maps, entities, events, attributes, and states necessary for both deliberative and reactive behavior. Coordination between subordinate BG processes is achieved by cross-coupling among plans and sharing of information among Executors through the KD.

Commands into each BG module consist of six elements:

1) CommandedAction (ac1) describes the action to be performed and may include a set of modifiers such as priorities, mode, path constraints, acceptable cost, and required conditions.
2) CommandGoal (gc1) describes the desired state (or goal state) to be achieved by the action. Mobility system state typically includes the position, heading, velocity, and turning rate of the system being controlled. The goal may include the name of a target or object that is to be acted upon. It also may include a set of modifiers such as tolerance.
3) GoalTime (gt1) defines the timing constraint on achieving the goal plus modifiers such as tolerance.
4) NextCommandedAction (ac2) describes the planned next action to be performed plus modifiers.
5) NextCommandGoal (gc2) describes the planned next goal state to be achieved plus modifiers.
6) NextGoalTime (gt2) describes the timing constraint on achieving the next goal plus modifiers.

The planner in each BG process decomposes commands into plans for each of its subordinate BG processes. Each plan is designed to have about ten steps. For each plan, an Executor cycles through the plan issuing commands, monitoring progress, compensating for errors, and reacting to surprises and emergencies. For example, a command into the Vehicle level (4) for the first vehicle in a scout Section would have the form:

CommandedAction = $ac1^4_1$ CommandGoal = $gc1^4_1$ GoalTime = $gt1^4_1$ ~ t + 1 min
NCAction = $ac2^4_1$ NCGoal = $gc2^4_1$ NextGoalTime = $gt1^4_1$ ~ t + 2 min

This command would be decomposed into three plans for the Subsystem level of the form:

Autonomous Mobility Plan	RSTA Plan	Communications Plan
$ap1^3_1$, $gp1^3_1$, $gt1^3_1$= t+5 sec	$ap1^3_2$, $gp1^3_2$, $gt1^3_2$	$ap1^3_3$, $gp1^3_3$, $gt1^3_3$
$ap2^3_1$, $gp2^3_1$, $gt2^3_1$= t+10 sec	$ap2^3_2$, $gp2^3_2$, $gt2^3_2$	$ap2^3_3$, $gp2^3_3$, $gt2^3_3$
$ap3^3_1$, $gp3^3_1$, $gt3^3_1$= t+15 sec	$ap3^3_2$, $gp3^3_2$, $gt3^3_2$	$ap3^3_3$, $gp3^3_3$, $gt3^3_3$
$ap4^3_1$, $gp4^3_1$, $gt4^3_1$= t+20 sec	$ap4^3_2$, $gp4^3_2$, $gt4^3_2$	$ap4^3_3$, $gp4^3_3$, $gt4^3_3$
$ap5^3_1$, $gp5^3_1$, $gt5^3_1$= t+25 sec	$ap5^3_2$, $gp5^3_2$, $gt5^3_2$	$ap5^3_3$, $gp5^3_3$, $gt5^3_3$
$ap6^3_1$, $gp6^3_1$, $gt6^3_1$= t+30 sec	$ap6^3_2$, $gp6^3_2$, $gt6^3_2$	$ap6^3_3$, $gp6^3_3$, $gt6^3_3$
$ap7^3_1$, $gp7^3_1$, $gt7^3_1$= t+35 sec	$ap7^3_2$, $gp7^3_2$, $gt7^3_2$	$ap7^3_3$, $gp7^3_3$, $gt7^3_3$
$ap8^3_1$, $gp8^3_1$, $gt8^3_1$= t+40 sec	$ap8^3_2$, $gp8^3_2$, $gt8^3_2$	$ap8^3_3$, $gp8^3_3$, $gt8^3_3$
$ap9^3_1$, $gp9^3_1$, $gt9^3_1$= t+50 sec	$ap9^3_2$, $gp9^3_2$, $gt9^3_2$	$ap9^3_3$, $gp9^3_3$, $gt9^3_3$
$ap10^3_1$, $gp10^3_1$, $gt10^3_1$= t+1 min	$ap10^3_2$, $gp10^3_2$, $gt10^3_2$	$ap10^3_3$, $gp10^3_3$, $gt10^3_3$

where ap is action planned, gp is goal planned, and gt is planned goal time
and api^j_k is the i-th planned action for the k-th subordinate BG module at the j-th level

The Vehicle level Executor for the Autonomous Mobility (AM) Subsystem would then transform the first and second steps in the AM plan into a command to the AM BG module at the Subsystem level (3).

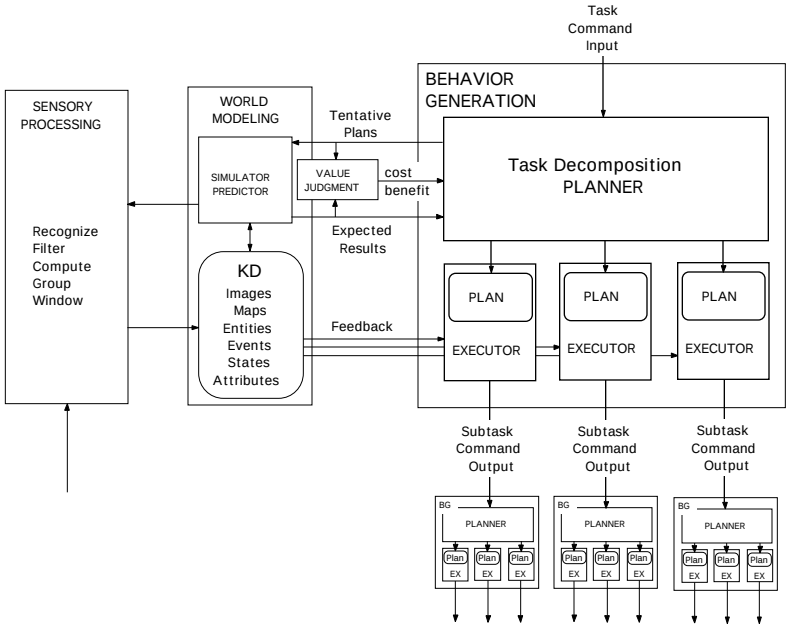

Figure 1. A typical 4-D/RCS computational node. Each task command input to a behavior generation (BG) process is decomposed into plans that become subtasks for subordinate BG processes. A world modeling (WM) process maintains a knowledge database (KD) that is the BG unit's best estimate of the external world. A sensory processing (SP) system operates on input from sensors by focusing attention (i.e., windowing), grouping, computing

attributes, filtering, and recognizing entities, events, and situations. A value judgment (VJ) process evaluates expected results of tentative plans. VJ also evaluates entities, events, and situations in the KD (not shown here.) The BG command and control hierarchy for the first five levels of the Demo III Experimental Unmanned Vehicle (XUV) is shown in Figure 2.

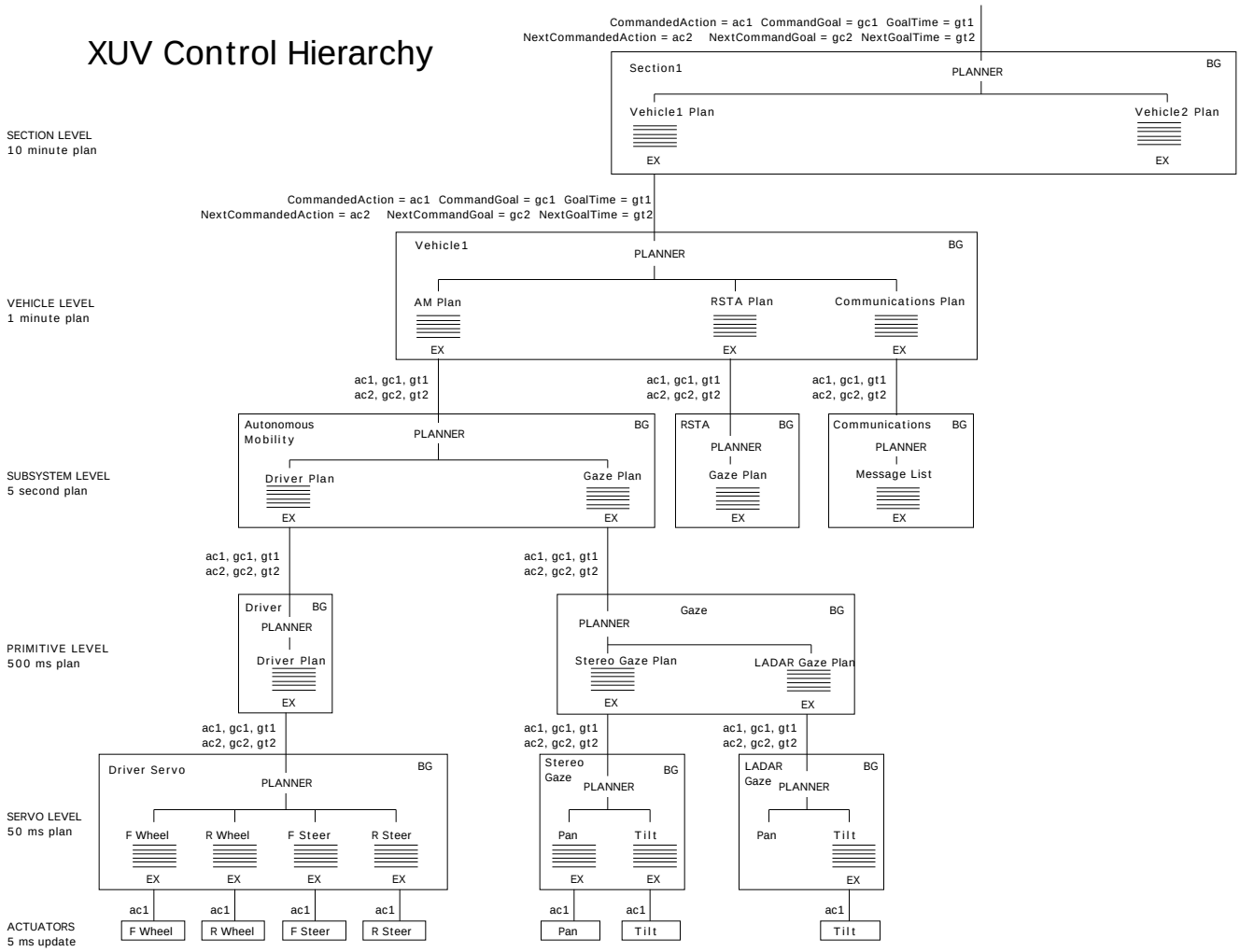

Figure 2. The command and plan structure for Demo III In the BG modules at each level there is a planner that produces one or more Plans for one or more subordinate BG modules. There is an Executor for each plan that communicates with the subordinate about how to integrate the lower level plan into the higher level plan.

2. INTERACTION BETWEEN DELIBERATIVE AND REACTIVE EXECUTION

In most of the robotics literature, the interface between deliberative planning and reactive reflex is poorly understood. This is because it is not a single interface. To achieve optimum performance, there must be a multiplicity of interfaces distributed over multiple levels. The 4-D/RCS architecture has an interface between deliberative and reactive execution in every node at every hierarchical level. This enables 4-D/RCS to fully realize the desirable traits of both deliberative and reactive control in a practical system. Multiple levels of deliberative planning ensure that plans can be recomputed frequently enough that they never become obsolete. Multiple levels of representation cause the planning search space to be limited in range and

resolution, and plans to be limited in the number of steps and amount of detail. Multiple levels of sensory information from the environment ensure that reactive behavior can be generated with a minimum of feedback time delay.

The 4-D/RCS architecture has seven levels of distributed, hybrid, deliberative/reactive control. It is designed to enable long-range big-picture plans for complex problems at higher levels while producing high-speed high-precision control at lower levels. The following are specifications for the planning horizon and reaction latency at all seven levels:

Level	Planning horizon	Reaction latency	Function performed
Level 1	50 milliseconds	5 milliseconds	Actuator servo
Level 2	500 milliseconds	20 milliseconds	Vehicle heading, speed
Level 3	5 seconds	100 milliseconds	Obstacle avoidance
Level 4	50 seconds	500 milliseconds	Single vehicle tactical behaviors
Level 5	10 minutes	2 seconds	Section level tactical behaviors
Level 6	2 hours	5 seconds	Platoon level tactical behaviors
Level 7	24 hours	20 seconds	Company level tactical behaviors

Table 1. Planning Horizon and Executor Reaction Latency at each level of the 4-D/RCS hierarchy.

The planning horizon refers to the future time interval over which each level plans. Plans at each level typically have about five to ten steps between the anticipated starting state and a planned goal state at the planning horizon. Replanning is normally done cyclically about every one-tenth of the planning horizon (i.e., level 3 replans about every 500 milliseconds.) The reaction latencies refer to the time lapse through the executor feedback loop between sensing and acting. Reaction latency is the minimum delay through the reactive feedback loop at each level. Reaction can interrupt cyclic replanning to immediately select an emergency plan, or to simply begin a new replanning cycle based on new information. Reaction latencies at each level are determined by computational delays in updating the world model as well as the sampling frequency and computation cycle rate of the Executors.

4-D/RCS planners are designed to generate new plans well before current plans become obsolete. Thus, action can always take place in the context of a recent plan, and feedback through the executor can close a reactive control loop using recently selected control parameters. To meet the demands of Demo III, the 4-D/RCS architecture specifies that replanning should occur within about one-tenth of the planning horizon at each level (e.g., replanning at level 3 will occur about every 500 milliseconds.) Executors react to sensory feedback considerably faster[1] (e.g., reaction at level 3 will occur within 100 milliseconds).

To achieve this desired rate of replanning, it is necessary to limit the amount of data in the world model that needs to be refreshed between each planning cycle. Multilevel representation of space makes it possible to limit the number of resolution elements in maps and the amount of detail in symbolic data structures at each level. Information can be chunked so that only what is necessary for making decisions need be represented in each node. Multilevel representation of time makes it possible to limit the number of events and amount of detail stored at each level. Multilevel representation enables the world model in any computational node to be rich and detailed at the point of interest, yet contain only a modest amount of information. This allows the world model in each node to be updated in real-time.

To replan frequently, it is also necessary to limit the amount of search required to generate new plans. There are several ways to limit the search. One is to pre-compute and store plans that can be selected by a rule-based planner in response to the recognition of an object, event, or situation. A second approach is to limit the range and resolution of the state space that needs to be searched and evaluated. There are various combinations of these approaches, such as partially developed plans that must be instantiated with parameters at execution time, or schemas that define the general forms of behavior that are appropriate to various situations. All of these options can be supported by 4-D/RCS.

Maps at each level provide information to planners about the position, attributes, and class of entities. For example, maps at various levels may indicate the shape, size, class, and motion of objects such as obstacles and vehicles, and the location of roads, intersections, bridges, streams, woods, lakes, buildings, and towns.

[1] Except at level 1 where replanning and reaction times are the same. At levels 2 and above, the difference between replanning and reacting becomes more significant with each successively higher level.

For the Demo III program, the range and resolution of maps is limited to about 128x128 (~16,000) resolution elements at each level. The range and resolution of maps at all levels of the Demo III 4-D/RCS hierarchy are shown in the following table:

Level	Map resolution	Map range	Function performed
Level 1	n/a	n/a	Actuator servo
Level 2	4 cm	5 m	Vehicle heading, speed
Level 3	40 cm	50 m	Obstacle avoidance
Level 4	4 m	500 m	Single vehicle tactical behaviors
Level 5	40 m	5 km	Section level tactical behaviors
Level 6	400 m	50 km	Platoon level tactical behaviors
Level 7	4 km	500 km	Company level tactical behaviors

Table 2. Range and resolution of maps at all levels in the Demo III 4-D/RCS architecture.

For different vehicle speeds, the map resolution required for planning at various levels may be different. The numbers in Table 2 are for a ground vehicle traveling about 10 meters per second. A helicopter skimming over the ground at 100 meters/second would require planning maps with an order of magnitude greater range and an order of magnitude lower resolution than that shown above. At some time in the future, we intend to incorporate map representations that are velocity dependent.

Figure 3 is a high-level block diagram of the first five levels in the 4-D/RCS architecture for Demo III. On the right, Behavior Generation modules decompose high level mission commands into low level actions. The text beside the Planner and Executor at each level indicates the planning horizon, replanning rate, and reaction latency of commands at each level. Each planner has a world model simulator that is appropriate for the problems encountered at its level. In the center, each map as a range and resolution that is appropriate for path planning at its level. At each level, there are symbolic data structures and segmented images with labeled regions that describe entities, events, and situations that are relevant to decisions that must be made at that level. On the left is a sensory processing hierarchy that extracts information from the sensory data stream that is needed to keep the world model knowledge database current and accurate.

At the bottom are actuators that act on the world and sensors that measure phenomena in the world. The Demo III vehicles will have a variety of sensors including a laser range imager (LADAR), stereo CCD (charge coupled device) cameras, stereo forward looking infra red (FLIR) devices, a color CCD, a vegetation penetrating radar, GPS (Global Positioning System), an inertial navigation package, actuator feedback sensors, and a variety if internal sensors for measuring parameters such as engine temperature, speed, vibration, oil pressure, and fuel level. The vehicle also will carry a Reconnaissance, Surveillance, and Target Acquisition (RSTA) mission package that will include long-range cameras and FLIRs, a laser range finder, and an acoustic package.

In Figure 3, the bottom (Servo) level has no map representation. The Servo level deals with actuator dynamics and reacts to sensory feedback from actuator sensors. The Primitive level map has range of 5 meters with resolution of 4 centimeters. This enables the vehicle to make small path corrections to avoid bumps and ruts during the 500 millisecond planning horizon of the Primitive level. The Primitive level also uses accelerometer data to compensate for vehicle dynamics.

The Subsystem level map has range of 50 meters with resolution of 40 centimeters. This map is used to plan about 5 seconds into the future to find a path that avoids obstacles and provides a smooth and efficient ride. The Vehicle level map has a range of 500 meters with resolution of 4 meters. This map is used to plan paths about one minute into the future taking into account terrain features such as roads, bushes, gullies, or tree lines. The Section level map has a range of 5000 meters with resolution of 40 meters. This map is used to plan about 10 minutes into the future to accomplish tactical behaviors. Higher level maps (not shown in Figure 3) are used to plan section and platoon missions lasting about 2 and 24 hours respectively. These are derived from a priori military maps.

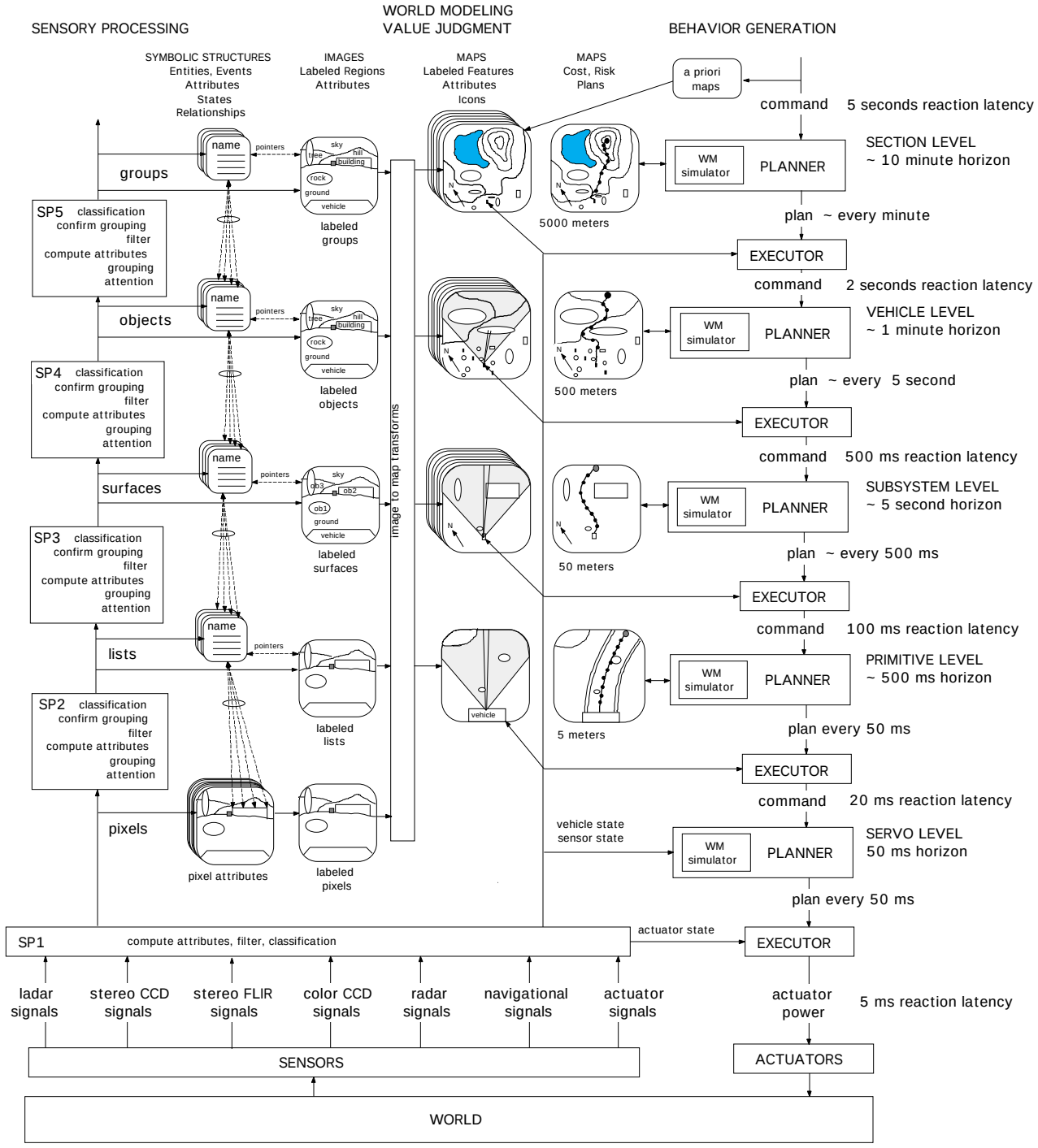

Figure 3. Five levels of the 4-D/RCS architecture. On the right are Planner and Executor modules. In the middle are maps for representing terrain features, road, bridges, vehicles, friendly/enemy positions, and the cost and risk of traversing various regions. On the left are Sensory Processing functions, symbolic representations of entities and events, and segmented images with labeled regions.

3. TWO KINDS OF PLANS

There are two kinds of plans that will be required by the Demo III vehicles: 1) path plans for locomotion, and 2) task plans for other types of behavior. A typical path plan consists of a series of waypoints on a map. A typical task plan consists of a set of instructions or rules that describes a sequence of actions and subgoals required to complete the task. Both path plans and task plans can be represented in the form of augmented state graphs, or state tables, which define a list (or graph) of planned actions (subtasks) with the desired state (subgoal) to be achieved by each action in the plan. Typically states are represented by nodes, and actions by arcs that connect the nodes. Both types of plans can be executed by the same executor mechanism.

In principle, both types of planning can be performed by searching the space of possible futures to find a desirable solution. Usually, however, only path plans are generated by searching on a map for the path with the lowest cost and risk between start and goal points. Task plans are typically generated from schema or recipes that have been developed off-line and stored in a library where they can be accessed based on a rule or case statement when conditions arise. If there is more than one recipe or schema that is appropriate to a task, each may be submitted to the world model for simulation and the predicted results evaluated by the value judgment process. The planner then selects the best recipe or schema from a limited list for execution.

In 4-D/RCS, path planners use cost maps that represent the estimated cost or risk of being in, or traversing, regions on the map. Values represented in cost maps depend on mission priorities and knowledge of the tactical situation. Path planners search the cost maps for routes that have the lowest cost under a given situation. Task planners use rules of engagement and military doctrine to select modes of operation and schema for tactical behaviors. State variables such as mission priorities and situational awareness determine which type of behavior is selected.

For example, if enemy contact is likely or has occurred, cost maps of open regions and roads will carry a high cost and regions near tree lines and under tree cover will have lower cost. In this case, path planners will plan cautious routes near tree lines or through wooded areas, and task planners will plan behaviors designed to search for evidence of enemy activity in likely places. However, if enemy contact is unlikely, roads will have a very low cost and open regions will carry a lower cost than wooded areas. This will cause path planners to plan higher speed routes on the road or through open regions, and task planners to focus on issues such as avoiding local traffic. Thus, a very small amount of information, such as enemy contact is likely or unlikely, can completely change the tactical behavior of the vehicle in a very logical, intuitive, and meaningful way.

4. SENSORY PROCESSING

Sensory processing (SP) is the set of computational operations performed on sensory signals to extract the state-variables, vectors, images, entities, events, symbols, strings, maps, and data structures necessary to generate and maintain useful internal representations of the world in the world model knowledge database (KD). As shown in Figure 3, there is a hierarchy of SP processes that maintain a hierarchy of images, maps, and symbolic representations. Within each level of SP, there are five basic functions: 1) focusing attention, 2) grouping, 3) computing group attributes, 4) filtering group attributes and confirming grouping hypotheses, and 5) classifying, recognizing, or identifying grouped entities and events.

1) Focusing attention selects (e.g., windows) the regions of space and the intervals of time over which specific SP processes operate on sensory inputs. The remainder of the input can be masked out or ignored. Windowing allocates the available computational resources to the entities and events that are most important for success in achieving behavioral goals. Windowed regions can be assigned priorities and be allocated computational resources in proportion to their relative importance.

2) Grouping aggregates or clusters lower-level entities and events into higher-level entities and events and assigns them labels or names. In image processing, spatial grouping segments images into regions that can be labeled with an entity name. Each pixel in the labeled region carries the name of the entity to which it belongs. This creates an entity image consisting of pixels labeled with entity names. For each region in an entity image, there is a named entity frame that contains entity attributes computed over the region. Entity frames also contain pointers that define relationships with other

entities such as *belongs-to*, or *is-part-of*. Grouping also aggregates temporal sequences into events or strings that can be assigned labels or names. Event frames contain event names, event attributes, and relationship pointers to other events and entities.

Any particular grouping of entities or events is a hypothesis based on a gestalt heuristic such as: proximity (subentities are close together in the image, or subevents are sequential along the time axis); similarity (subentities have similar attributes such as color, texture, range, or motion, or subevents have similar spectral properties or temporal patterns); continuity (subentities have directional attributes that line up, or lie on a straight line or smooth curve); or symmetry (subentities are evenly spaced or are symetrical about a point, line, or surface.) Grouping hypotheses need to be tested, and be confirmed or rejected, by observing how well predictions based on each grouping hypothesis match subsequent observations of sensory data over time under a variety of circumstances. Hypothesis testing may be done by filtering and recursive estimation, or by comparisons between observations and predictions. This may be accomplished locally through filtering and correlation with immediate expectations, or globally through reasoning about consistency between perception and physical laws, logical rules, or mathematical principles.

3) Computing attributes for hypothesized entities can be accomplished by integrating subentity attributes over the region in an image covered by the hypothesized entity. Entity attributes may include position, velocity, orientation, area, shape, and color. For example, the brightness of a pixel entity is computed by integrating the photons within a spectral energy band falling on a photodetector in the image plane during an interval of time. The color of a pixel entity is computed from the ratio of brightness of registered arrays of pixels in three different spectral bands. The spatial or temporal gradient of brightness or color at a pixel can be computed from spatial or temporal differences between adjacent pixels in space or time. The length of an edge entity in an image can be computed by counting the number of pixels along the edge. The area of a surface entity can be computed by counting the number of pixels contained in it. The cross sectional area of an object entity can be computed by multiplying the area of its projection in the image by the square of the ratio of its range to the focal length of the camera. The lateral velocity of an object entity can be computed by computing the angular motion of its center of gravity in the image multiplied by its estimated range. Radial velocity can be computed from range-rate measurements.

4) Filtering is a process that reduces noise, enhances signal quality, and eliminates ambiguity. The computed values of entity attributes can be filtered over intervals of space by averaging, or by convolution with spatial filters. The computed value of entity or event attributes can be filtered over intervals of time by phase-lock loops, by correlation with Fourier components or wavelets, or by recursive estimation such as Kalman filtering. Recursive estimation operates on each new sensory observation as it occurs to compute a new "best estimate" (over a window of space and time) of entity attributes and states. Each sensory measurement adds new information to what was previously known about entities and events in the world. Recursive estimation operates by comparing a prediction based on the current "best estimate" with an observation based on sensory input. Variance between observations and predictions are used to update the "best estimate" and to compute a confidence value for the "best estimate."

Computed confidence can be used to confirm or deny the grouping hypothesis that created the entity. When variance between observed and predicted attributes is small, confidence in the grouping hypothesis is increased. When the variance between the observed and predicted attributes is large, confidence is reduced. When the confidence value for entity attributes rises above a confirmation threshold, the grouping hypothesis that generated the entity is confirmed. When the confidence falls below a denial threshold, the grouping hypothesis is rejected and a new grouping hypothesis must be selected.

5) Classification, recognition, detection, or identification is based on similarity between the attributes of a confirmed entity and attributes of a known entity class. Entity classes are categories into which confirmed entities can be sorted. For each entity class there exists a set of attributes (i.e., an attribute vector) that is characteristic of that class. This characteristic attribute vector (or template) can be compared with the attribute vectors of entities observed in the world. Similarity between the attribute vector of a confirmed entity and the attribute vector of a stored class can be computed by taking the normalized dot product between the two attribute vectors. If a confirmed entity has an attribute vector that matches the characteristic attribute vector of a particular class, then the confirmed entity can be classified as a member of that class. Recognition occurs when the degree of similarity between the entity attribute vector and a class attribute vector exceeds a recognition threshold. At this point, the confirmed entity is recognized as belonging to the entity class whose attribute vector it matches.

Figure 4 illustrates the five basic SP functions and the interactions that take place between SP and WM at each level of the 4-D/RCS architecture.

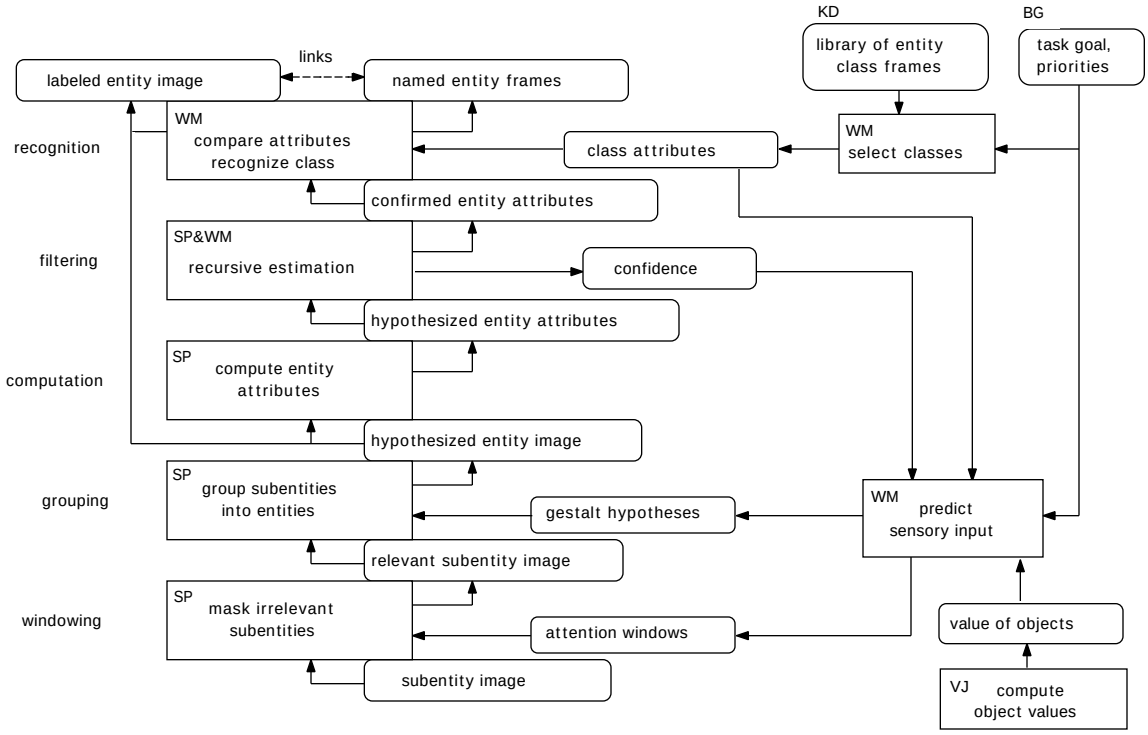

Figure 4. The set of five basic processing functions that make up SP functionality for image processing and their interaction with WM processes. The boxes with rounded corners represent information. Boxes with square corners represent functional processes.

At each level, WM predictions are driven by task goals and priorities, by VJ value estimates, by what is known about objects, and by the level of confidence that they exist. Value based predictions allow SP to focus attention on what is important and mask out what is irrelevant. Predictions also guide gestalt hypotheses for grouping things together into entities. Once entities have been hypothesized their attributes can be computed, filtered, and tested for consistency over time and space. Once entity hypotheses have been confirmed, their attributes can be compared with class attributes of known classes, especially those of goal or target classes. Recognized entities are labeled with class names and forwarded to the next level of SP processing.

5. DISCLAIMER

It should be noted that only parts of the 4-D/RCS architecture have been implemented as of this March 1999. The most complete implementation is in the BG hierarchy. BG modules have been implemented at the Servo, Primitive, Subsystem, and Vehicle levels. WM maps have been implemented at the Subsystem and Vehicle levels, and SP functions have been implemented at the pixel level. All regions detected as obstacles, clear space, or regions providing cover are represented only at the pixel level in LADAR and stereo images.

This relatively primitive implementation has enabled the NIST HMMWV (Highly Mobile Multipurpose Wheeled Vehicle) to drive cross-country in mostly open terrain avoiding obstacles such as trees and fences at speeds of more than 25 km per hour. The 4-D approach developed by the Universitat der Bundeswehr Munchen (UBM) for the 'VaMoRs-P' system [4] includes Road Detection and Tracking (RDT) and the Obstacle Detection and Tracking (ODT) algorithms. These use relatively simple object models and object recognition algorithms, and function primarily in on-road situations. The RDT and ODT systems have not yet been integrated into the 4-D/RCS architecture. However, the stand-alone results have been impressive. On-road driving by the UBM vehicle has achieved speeds up to 150 km per hour on the Autobaun in traffic with automatic lane changing. [4]

It is anticipated that all of the features described in this paper will be operational by the end of the Demo III program in the year 2001. We are confident that the 4-D/RCS architecture is sufficiently general and powerful and provides a sufficiently rich representation of the world to successfully perform all of the navigation, driving, and tactical behavior requirements of the Demo III program.

6. REFERENCES

1. J. S. Albus, "4-D/RCS: A Reference Model Architecture for Demo III, Version 1.0," *NISTIR 5994*, National Institute of Standards and Technology, Gaithersburg, MD, March 1997.

2. J. S. Albus and A. M. Meystel, "A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems," *International Journal of Intelligent Control and Systems*, Vol.1, No. 1, 1996, pp. 15-30.

3. J. S. Albus, "The NIST Real-time Control System (RCS): An Application Survey," *Proceedings of the AAAI 1995 Spring Symposium Series*, Stanford University, Menlo Park, CA, March 27-29, 1995.

4. E. D. Dickmanns, et. al., "The Seeing Passenger Car 'VaMoRs-P'," *International Symposium on Intelligent Vehicles'94*, Paris, October 24-26, 1994.