Identifying Sensory Processing Requirements for an On-Road Driving Application of 4D/RCS *

John A. Horst[a], Anthony Barbera[a], Craig Schlenoff[a], and David W. Aha[b]

[a]Intelligent Systems Division, The National Institute of Standards and Technology,
U.S. Department of Commerce, Gaithersburg, MD 20899, USA
[b]Intelligent Decision Aids Group, Navy Center for Applied Research in Artificial Intelligence,
Naval Research Laboratory (Code 5515), Washington, DC 20375, USA

ABSTRACT

Sensory processing for real-time, complex, and intelligent control systems is costly, so it is important to perform only the sensory processing required by the task. In this paper, we describe a straightforward metric for precisely defining sensory processing requirements. We then apply that metric to a complex, real-world control problem, autonomous on-road driving. To determine these requirements the system designer must precisely and completely define 1) the system behaviors, 2) the world model situations that the system behaviors require, 3) the world model entities needed to generate all those situations, and 4) the resolutions, accuracy tolerances, detection timing, and detection distances required of all world model entities.

Keywords: On-road driving, hierarchical control architecture, 4D/RCS, task decomposition, sensory processing, world model, world model entities, sensory processing requirements, autonomous on-road driving, behavior generation, intelligent control

1. INTRODUCTION

This paper describes our work identifying sensor requirements on NIST's On-Road Driving project, which is sponsored by DARPA/IPTO's Mobile Autonomous Robot Software (MARS) program. The overall project goal is to design, implement, and evaluate an intelligent real-time control system for an on-road autonomous vehicle that drives under a variety of typical conditions and whose skill approaches human levels of performance. This is an ambitious task, and many researchers have addressed this topic. For example, Dellaert *et al*[1] describe an autonomous vehicle that is designed for highway driving, but it cannot drive in urban conditions, and it is not robust with respect to varied weather, road surface, and other conditions. In addition, many researchers (*e.g.*, Ichise *et al*[2]) have limited their work concerning on-road driving behaviors to simulation, which can significantly reduce focus on identifying real-world sensory processing (SP) requirements for this task. Dickmanns has demonstrated a successful on-road autonomous system and has a detailed architecture, however, connections to sensing requirements through the world model are either not explicit or not complete.[3]

We know of no one who has formally addressed the task of using an autonomous on-road driving system definition to specify sensor system requirements. Of course, anyone building an autonomous on-road system has to do this, albeit informally, otherwise the sensors will not be sufficient for the task. However, one wonders how many sensor systems are more sophisticated than necessary or if system delays and failures could be avoided if an explicit understanding of exact sensor requirements was had at design time. Clearly a formal method for determining these requirements will be useful, as long as the cost of generating such requirements does not outweigh the cost of not generating them. We also have loftier goals to which this sensor requirements identification contributes: design a system that has a toolkit of task capabilities that surpass any that is currently available in a single autonomous on-road driving system.

The function of an SP subsystem is to generate actual measurements of the entities in the environment needed to perform the requisite system behaviors successfully. These entities are represented in the world model (WM) of the system. The

*This work was supported by DARPA's Mobile Autonomous Robotics program (PM: Douglas Gage). We thank Jim Albus, Elena Messina, and our other project colleagues for their continuing support of this work.

Further author information: (Send correspondence to John Horst)
NIST authors: {john.horst, tony.barbera, craig.schlenoff}@nist.gov, phone: (301) 975-{3430, 3460, 3456}, isd.mel.nist.gov
NRL author: David.Aha@nrl.navy.mil, phone: (202)404-4940

types of WM entities can only be discovered if we have defined the system behaviors correctly and completely. System behaviors, in turn, are dependent on situations[†] that trigger transitions from behavior to behavior. Situations that trigger behavior transitions are further dependent on sub-situations. Situations and sub-situations are all dependent on WM entities[‡]. Proceeding in this manner, the designer can correctly identify all the WM entities needed to generate all situations. After identifying the type of WM entities required, the designer can specify the resolutions, accuracy tolerances, detection timing, and detection distances required of all these WM entities. Finally, the designer determines how this required accuracy impacts SP in terms of processing costs and sensor system requirements. For large-scale problems (like on-road driving), the use of the hierarchical control paradigm 4D/RCS[4] helps to keep the total design task manageable. We describe an example of ascertaining SP requirements for an On-Road Driving application built according to the 4D/RCS paradigm. The overall On-Road Driving task is to design, implement, and test a controller that directs an autonomous vehicle to maneuver successfully through common, but complex on-road driving situations. We briefly review the 4D/RCS design methodology and reference architecture, and our design for applying it to this task, including our task decomposition representation format for on-road driving task knowledge. We also detail our approach for two subtasks - passing a vehicle and maneuvering about objects while following a lane. In particular, we describe the planning algorithms, supporting reasoning strategies, required WM entities, the SP tasks required to compute their values, and the tolerances within which they must perform.

The sensory processing requirements of different driving tasks have significantly different resolutions, identification, and classification requirements will allow a performance metric to be defined for each of these tasks. Because these tasks fit into a hierarchy of tasks, performance metrics on these tasks should reveal strikingly valuable detail, allowing resources to be applied appropriately. For example, the task of "maneuvering the vehicle within a single lane" (the task) is at the Elementary Maneuvers System module (see Figure 3). Because the task is at that level in the control hierarchy, the task only requires that the in-lane maneuvering sub-system utilize previously computed items such as in-lane object cost, size, velocity, and acceleration and road surface conditions. At this level there is little requirement for detailed recognition of object types or the need to monitor them at a distance or to sense and read signs alongside or overhead of the road.

However, if our autonomous vehicle decides to pass a vehicle on an undivided two-lane road, then a world representation must be sensed that identifies other WM entities, such as upcoming intersections, railroad crossings, vehicles in the oncoming lane out to very large distances, lane marking types, and roadside signs. Systems satisfying these requirements probably do not exist today.

Our work is to show just how these sensor requirements are connected to the particular driving task that the system is trying to accomplish. In summary, our approach is to exercise the RCS methodology on the on-road driving tasks and employ this to generate our sensor requirements. This methodology is described in (Barbera *et al* 2004).[5] This will provide the set of specifications that allow us to determine if particular sensor systems and sensor processing algorithms are sufficient to support particular driving tasks. Conversely, if the goal is to accomplish a particular set of driving tasks, this specification can be used to select the appropriate sensors and specify the required sensor processing requirements.

By detailing the features, attributes, and classifications of entities required in the world to reason about and generate specific driving tasks, we will have a specification that can be used to

1. identify the SP requirements to the sensory processing researchers and
2. be used as testing performance metrics to evaluate the capabilities of various sensors and sensory processing algorithms.

2. A PARADIGM AND METHODOLOGY FOR BUILDING COMPLEX SYSTEMS

The challenges encountered when designing and building autonomous on-road vehicles are virtually the same as that encountered with any other Complex Electro-Mechanical System (CEMS). We know from experience that general complex systems, *e.g.*, word processing software or a Mars rover, are painstakingly designed, built, and maintained. Nothing can be left to chance and virtually nothing good happens by chance. Careful thought produces success, careless thought produces bugs. No one flips coins to aid in the production of large-scale, complex software. This is our universal experience and we have no evidence to the contrary for artificial systems. Therefore, it is simply foolhardy to think that any truly complex

[†]A "situation" is an estimate of a world state at a particular point in time.

[‡]A "world model entity" is either a primitive (atomic) world state or an aggregate of primitive world states.

system can be anything other than painstakingly built. While it is true that clever algorithms (*e.g.*, ones that learn) will help make the code more efficient and robust, such cleverness requires all the more that the algorithm developer understand the task in *even fuller detail*, than required by designing a system with all tasks explicitly encoded. There may be clever ways to encode and execute the task in the artificial system, but in order to develop these clever techniques, the system designer must comprehend the task knowledge completely.

At least two questions arise:

1. What is the nature of this painstaking work?
2. What is a CEMS design method that make design and implementation efficient?

A simple answer to the first question is in two parts. 1) All the task knowledge must be gleaned from the domain expert and 2) all this task knowledge must be translated into code that the system can successfully execute. Experience teaches that we cannot expect to get the highly functional system we desire by gleaning only some small subset of all the necessary task knowledge from the domain expert, with only a little additional input from the designer. Evidence from successful CEMS universally rejects this hope. So for the time being we need a system design paradigm that expects that all this information needs to be explicitly and painstakingly defined and that there are no short cuts or silver bullets. Furthermore, one must encode that human task knowledge into a computer-controlled system. This is where an architecture, methodology, and tool set become so helpful, as we will argue in the next two sections.

Therefore, the answer to the second question is tied into the answer for the first: selecting the type of design method will dictate just how the expert's task knowledge will be realized in the artificial system. We surely want a paradigm that has the following characteristics.

- A conceptual framework allowing the explicit entry of all the knowledge relevant to the system task
- A conceptual framework that employs appropriate division of labor between the system components that avoids combinatorial explosion of knowledge complexity
- A precisely defined design method that consists of explicit design steps that the designer can follow and that works for any CEMS system
- A software design and implementation tool that hides unnecessary complexity, allowing the designer/implementer to focus on realizing the task knowledge in the artificial system

A conceptual framework,[4] design method,[5] and tool set[6] exist, which to some degree satisfy these requirements.

In the next few sections we will describe the 4D/RCS architecture and methodology. A description of an appropriate tool set will not be described in this paper, but can be found in (Horst, 2000).[6]

3. THE 4D/RCS ARCHITECTURE SUMMARIZED

A design architecture is a framework for system design with terms, definitions, and guidelines. A methodology embodies the design process. So, the architecture is a "framework" and the methodology is a "process". In terms of a common food recipe, the architecture would define the ingredients, tools, terms, and meanings of terms, and the methodology are the steps by which the food is prepared. We describe the architecture in this section and the methodology is described in Section 4.

4D/RCS has been used for conceptualizing, designing, engineering, integrating, and testing intelligent software for vehicle systems with any degree of autonomy. It integrates NIST's Real-time Control System (RCS)[7] and Universität der Bundeswehr München's VaMoRs 4-D approach to dynamic machine vision.[8] Their integration can be used to define a multi-resolution hierarchy of feedback control loops between sensing and acting that integrates reactive behavior with perception, world modelling, and planning. 4D/RCS has been used to develop several real-time control systems, including ones for an inspection workstation,[6] groups of unmanned air vehicles, and an experimental off-road vehicle. Albus and Meystel (2001)[9] and Meystel and Albus (2002)[10] review several 4D/RCS applications and describe how 4D/RCS relates to other intelligent system architectures.

Figure 1 displays an overview of the control loop supported by 4D/RCS. It characterizes complex real-time control systems as involving the interaction of three components, which each need performance metrics to judge their quality and correctness:

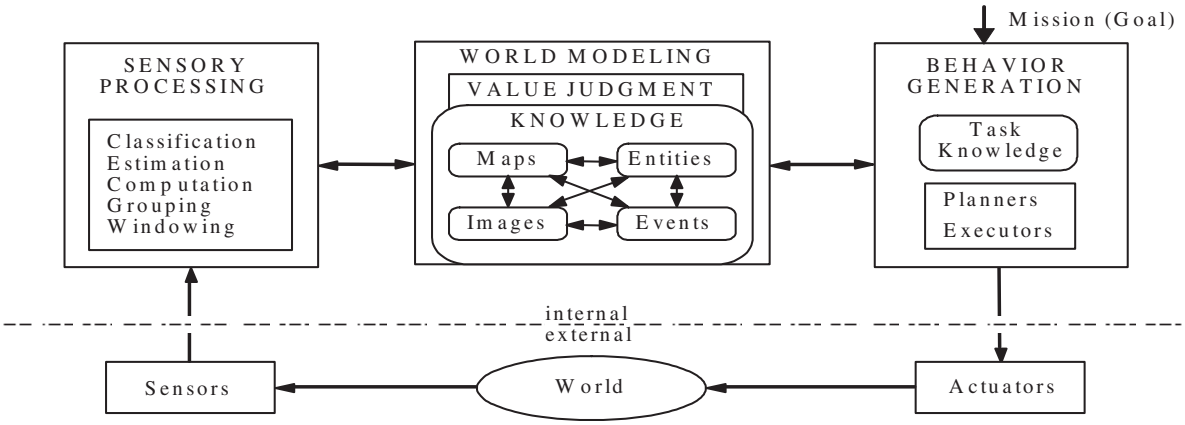

Figure 1. The basic internal structure of a 4D/RCS control loop. Sensory processing performs the functions of windowing, grouping, computation, estimation, and classification on input from sensors. World modelling maintains knowledge in the form of images, maps, entities, and events with states, attributes, and values. Relationships (*e.g.*, class membership, inheritance, pointers to situations) between images, maps, entities, and events are defined by ontologies. Value judgment provides criteria for decision making. Behavior generation is responsible for planning

1. Behavior generation (BG) processing: BG develops, selects, and executes plans for accomplishing the given goal tasks.
2. Sensory processing (SP): SP measures entities of interest in the environment and internal to the system.
3. World model (WM) processing: WM derives internal representations from sensory processing, and its value judgment logic makes them available for behavior generation.

In more detail, the world modelling component is responsible to maintain up-to-date estimates of task-relevant models of the environment and the system. These task-relevant models can be, for example, maps, object shapes, object types, costs, histories, events, or value judgments. For example, we may make a judgment that it is safe to straddle an object between the wheels of the vehicle, based on details about the object's measured shape and cost. This kind of computation is all done in the world model.

Behavior generation involves reasoning from real-time world model representations to conduct strategic and tactical behaviors (*e.g.*, for on-road civilian driving tasks). This includes planning alternate courses of action and alternate paths, evaluating these plans, and selecting the most appropriate action through some type of value judgment. Some challenges with applying 4D/RCS involve identifying appropriate sensors, developing sensory processing algorithms to generate accurate, registered world maps, and recognizing and classifying entities at sufficient resolution to populate a world model representation for use by the behavior generation component.

The sensory processing components perform sensor fusion, feature and attribute detection, object classification, etc. - all in the context of the current task activities. We will describe how to develop these sensory processing components given the world model data specification.

4. THE RCS METHODOLOGY

As we mentioned in the previous section, a methodology is the process by which an architecture is realized for a particular problem. The following sections describe the steps in creating a 4D/RCS design. Examples throughout will be gleaned from the on-road driving application.

The 4D/RCS methodology has the following steps. 1) Define the total system task with help from domain experts and decompose that task into logical subtasks as in Figure 6. 2) Group subtasks into agent control modules by functionality as in Figure 3. 3) Define a finite state machine (FSM) for each subtask, with state transitions (world/system conditions) and state transition functions (commands to next lower level, responses to upper level, or other functions) as in Figure 5.

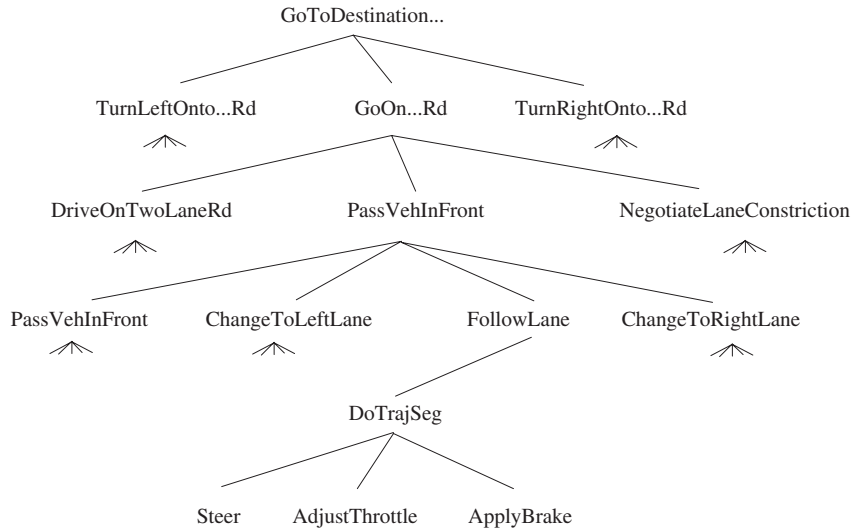

Figure 2. An example representation of a hierarchical task decomposition for the on-road driving task

4) Determine the world model entities that are needed to generate the state transition conditions as in Figure 7. 5) Determine the sensory processing system requirements dictated by the world model entities as in Figure 7. The following two sections describe these steps in more detail.

4.1. Task Decomposition Knowledge

One of our first tasks in this project was to provide a task analysis for autonomous on-road driving that can serve, among other things, as the basis for developing a number of performance metrics. This task analysis is based on earlier work performed by the Department of Transportation[11] although our context required many modifications, additions, and, in general, a focus on tasks relevant to autonomous driving.

The 4D/RCS methodology uses a hierarchical task decomposition format[9] for representing domain knowledge. Hierarchies are the architectural mechanisms used to "chunk" and abstract systems into manageable layers of complexity. The scenario descriptions of intelligent control system activities naturally evolve into a task decomposition representation because the scenarios are task sequences and can easily be discussed at many levels of abstraction, leading to well-defined levels within the task hierarchy. This provides a convenient framework for system designers and knowledge engineers to organize the information from the expert within an architecture that preserves the narrative character of the expert's scenarios, thus allowing the expert to easily review this representational format. A hierarchical task decomposition representation is clearly well suited for this. Figure 2 shows an example of how a task decomposition hierarchy can be used to represent expert knowledge for the on-road driving task.

This task decomposition hierarchy also acts as a convenient structure in which to encode semantic knowledge from the expert. In the on-road driving task, semantic knowledge includes such knowledge items as the rules of the road, the rules that require the vehicle to drive more slowly on wet or icy roads or to allow larger following distances on wet roads, etc. Because each layer in the task decomposition represents a different abstraction level of the tasks, each layer also delineates levels of detailed task context for incorporating semantic knowledge relevant to that level of detail within a particular task's activities. We will exploit this very organized layering of the task knowledge into different levels of abstraction and task responsibility to aid us in performing a detailed analysis of the knowledge associated with finely partitioned task activities for the on-road driving activities.

Given that the 4D/RCS methodology uses a task decomposition decision hierarchy to capture knowledge from the expert's narratives, it is straightforward to instantiate this into an implementation of a hierarchical architecture for agent control modules executing this task decomposition in a one-to-one fashion (Figure 3). This 4D/RCS implementation technique represents the knowledge in the implemented system in a manner that continues to be easily recognized by the

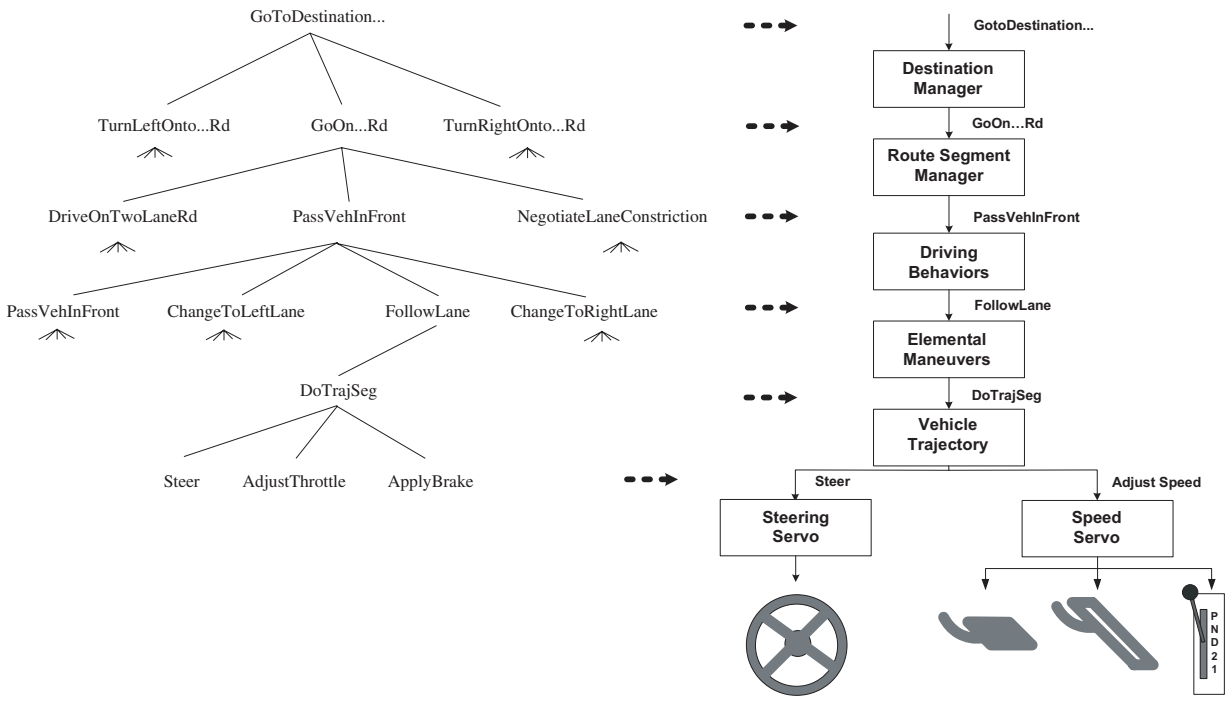

Figure 3. The RCS implementation creates a hierarchical organization of agent control modules (right side of figure) that will be the execution engine for the task decomposition (left side of figure). An agent control module is assigned to each actuator system to be controlled and an organizational structure is built up that mimics the same number of layers in the task decomposition representation. Each corresponding agent control module will accept the appropriate subtask command at the equivalent level in the task hierarchy and will determine which subgoal command will next be given to its subordinate, based on the rules encoded in the corresponding state table. For example, the subgoal command PassVehInFront to the Driving Behaviors agent control module will select the state table that contains all of the rules necessary to evaluate the present world state at this level of abstraction and in the context of passing the vehicle in front. It will send the appropriate subgoal command (i.e., FollowLane, ChangeToRightLane, or ChangeToLeftLane) for this present state to the Elemental Maneuvers agent control module.

domain expert. It maintains the layered partitioning of the task to create levels of abstraction, task responsibility, execution authority, and knowledge representation in a manner so as to greatly enhance the designer's ability to think about each of these layers separately. Each layer totally encapsulates the problem domain at one level of abstraction so all aspects of the task at this one layer can be analyzed without overwhelming the designer. All of the system's interactions and co-ordinations within the context of this abstraction layer are contained here so that modifications and enhancements to it can be evaluated with respect to their completeness and potential interaction with other task activities at that same abstraction level. At each layer, all of the relevant sensory processing, world modelling, and behavior generation processing for that level of responsibility and authority is encapsulated. As such, the 4D/RCS approach provides a very well ordered representation of the tasks at various levels of finer and finer detail, clustered at each level in a task sensitive context. This is ideal for the manner in which we want to identify the performance metrics.

A generic agent control module (Figure 4) is used as the unit building block in our hierarchical implementation system. Tasks are grouped into modules based on similarity of function and purview. The task decomposition process is not executable until explicit and complete process plans are defined for each task. These process plans are generally expressed as Finite State Machines (FSMs).

A task is decomposed at one hierarchical level into a sequence of constituent subtasks at a subordinate level, which may perform some processing at that level and/or send a command to the next subordinate level. Each task at each level can be realized as an FSM. An FSM consists of states, state transition situations (STS), and state transition functions (STF). STFs are generally tasks executed by FSMs at a lower hierarchical level. The STSs are snapshots of world states that generally come from sensory processing (SP). STSs are mediated through the generation of world model (WM) entities and

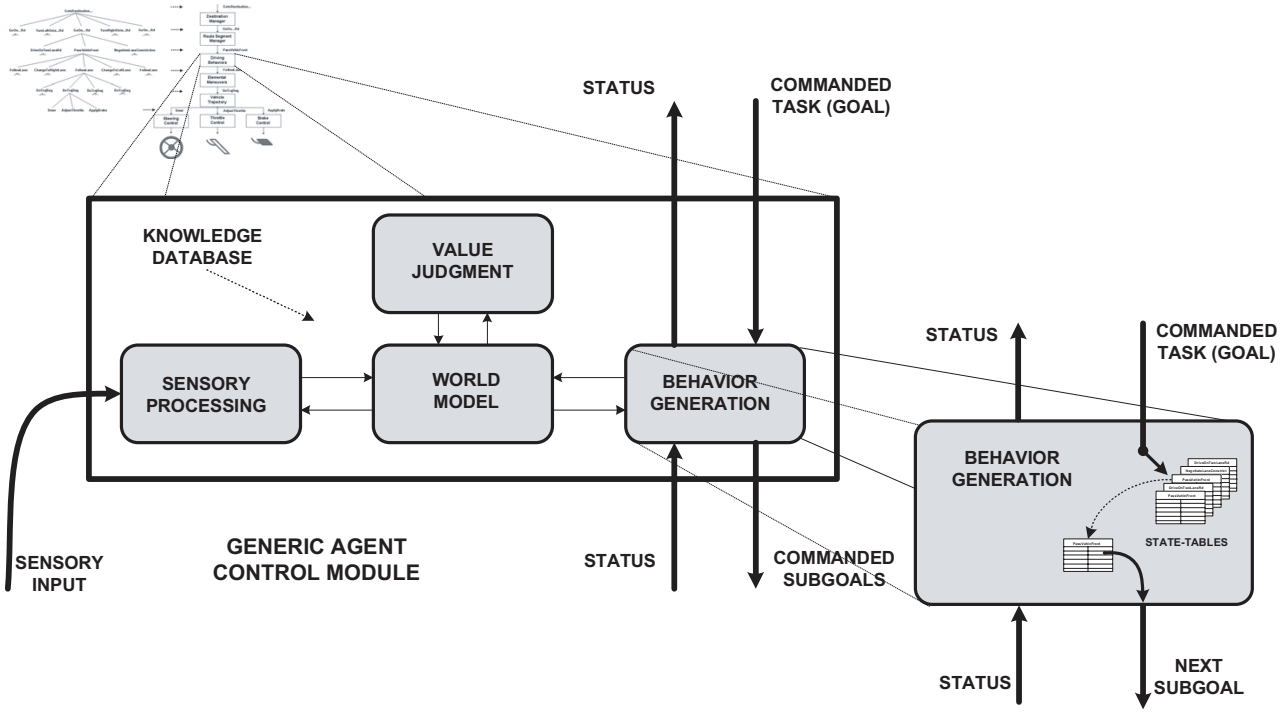

Figure 4. Every agent control module in the 4D/RCS hierarchy has the same processing structure of the generic agent control module. A module receives a commanded task (goal) that represents the present activity to be done at this level in the hierarchy at this instant. The Behavior Generation (BG) function uses this commanded task to look up and retrieve the state-table that contains the rules relevant to this activity. This sets the context for all of the processing at this module. Sensory Processing (SP) fills in world model data from the environment that is important to this particular task. If the situation requires planning activity, then the Value Judgment (VJ) function projects possible courses of action and performs some cost based analysis to determine a plan. As the situation creates matches to the rules in the BG's state table, the corresponding action part of the rule generates the next subgoal command to the subordinate agent control module.

sub-situations derived from those WM entities.

FSMs, which can be represented as state tables, are also an extremely convenient representational format for the developer. They capture the relevant task sequencing and state knowledge at each control module for every task activity. As the need arises to modify the system, the state table that contains the knowledge rule set that concerns the activity to be modified can be easily identified and retrieved. Potential conflicts that might arise in the execution are easily detected through inspection (because this is such a small set of rules) and avoided by ordering the rules using additional state variables. In this manner, the expert can provide additional task knowledge to resolve potential conflicts in specific task activities rather than require the system designer to devise some arbitrary and general conflict resolution mechanism. Figure 6 displays an example mapping of task decomposition knowledge into a state table.

4.2. World Model Knowledge

The FSMs described above are used to encode task decomposition knowledge. Each line of each state table uses some symbolic value to describe the present situation that must be matched to execute the corresponding output action of that rule. The processing required to determine whether a given situation is true can be thought of as a knowledge tree lying on its side, funnelling left to right, from the detailed sensory processing branches until all of the values have been reduced to an appropriate situation identification encoded in a symbolic value such as "ConditionsAreGoodToPass" (see Figure 6) or "Circ Obj1 AND Circ Obj2" (see Figure 7). This lateral tree represents the layers of refinement processing made on the present set of world model data to conclude that a particular situation now exists (*e.g.*, "ConditionsAreGoodToPass").

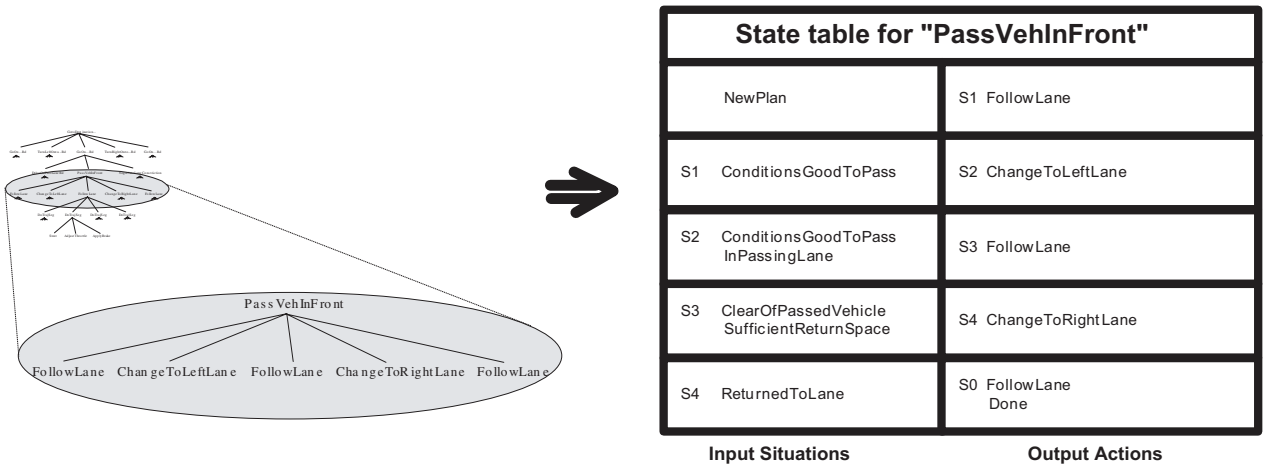

Figure 5. The task to "Pass a vehicle in front" is shown in both a task tree representation on the left and as a state table (FSM) on the right. The state transition situations, such as "ConditionsGoodToPass", "InPassingLane, and "ClearOfPassedVehicle", are in the left column of the state table, along with the states, S1, S2, etc. The state transition functions, such as "FollowLane", "ChangeToLeftLane", and "ChangeToRightLane", are in the right column of the state table, and, in this case, consist only of tasks at the subordinate level. The task knowledge for this particular on-road driving task is the set of subgoals, their sequence, and the conditions (i.e., current world situations) that cause each of these subgoals to be commanded.

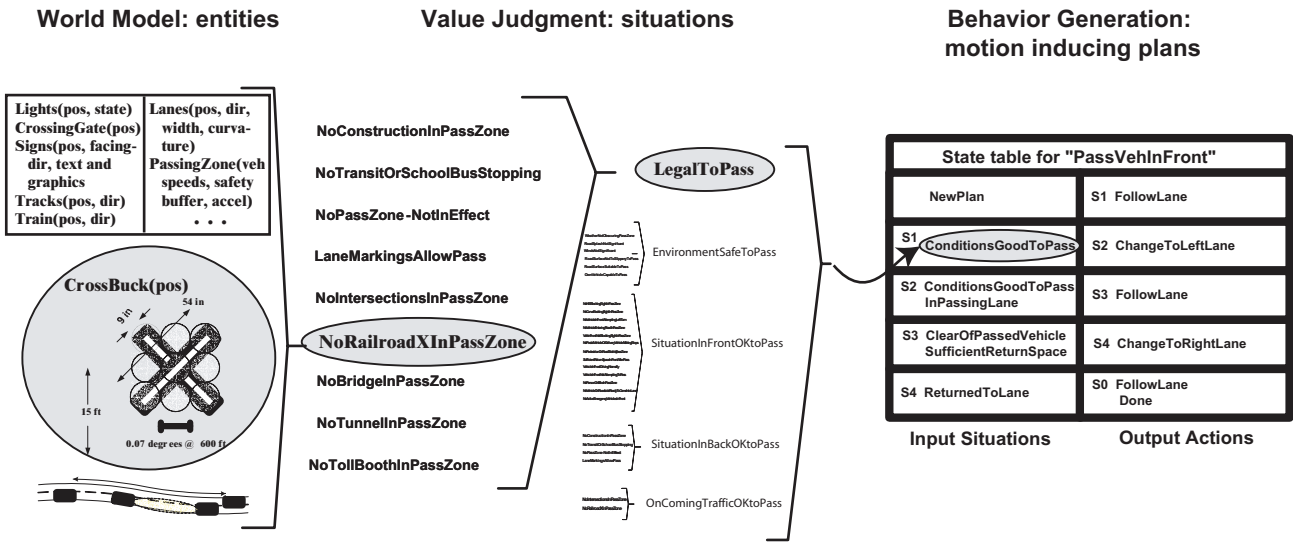

Figure 6. The "PassVehInFront" Plan State Table encodes the task decomposition representation of its input conditions and corresponding output action subgoals. In this example, the next subgoal "ChangeToLeftLane" is chosen as the output action when the input condition "ConditionsGoodToPass" is recognized. This figure illustrates how all of the dependencies on the world model data are derived. The high level group of situations that must be true for "ConditionsGoodToPass" to be true are identified. Here, one of these (LegalToPass) is further refined to identify all of the world model states that help define this situation. Similarly, we display detail on one of these world model states (NoRailroadXInPassZone) and the world entities, attributes, features, dimensions, and resolutions that help determine whether this state is true. One of these entities (CrossBuck sign) is further detailed in terms of the features, dimensions, and sensor resolutions required to recognize it within the distances required for the passing vehicle task.

The identification of these layers of knowledge processing to evaluate to the situation value is done in reverse. We know that we cannot change into the oncoming traffic lane (the "ChangeToLeftLane" action) during the passing operation until "ConditionsAreGoodToPass". Now we have to determine what must be considered for this to be true. To do this, we review many different example scenarios to determine all of the pieces of knowledge required for all of these variations. The results are grouped by category into (in this example) five major evaluation areas. Thus, to be able to say that the "Conditions-AreGoodToPass", we first had to evaluate that each of the five sub-groups were true (i.e., the conditions "LegalToPass", "EnvironmentSafeToPass", "SituationInFrontOKtoPass", "SituationInBackOKtoPass", and "OncomingTrafficOKtoPass"). In this example, we have clustered all of the rules of the road that pertain to the passing operation at this level of task detail into the "LegalToPass" subgroup evaluation. We have itemized nine world states to be evaluated and named them with the identifiers such as "NoConstructionInPassZone", "NoTransitOrSchoolBusStopping", and "NoPassZone-NotInEffect". These world states can now be further decomposed into the primitive world model elements we need to measure (*e.g.*, vehicles, their speed, direction, location, lane markings, signs, railroad tracks) to determine whether these world states exist. These primitive world model elements then set the requirements for the sensory processing system we need to build to support these control tasks. Everything has been determined in the context of the individual tasks that we want the system to support.

5. APPLICATION EXAMPLES

In this section, we exemplify the use of the RCS methodology through two different on-road driving tasks, passing and maneuvering around obstacles in-lane. Section 5.1, focuses on the example mentioned throughout this paper pertaining to passing another vehicle on a two lane undivided road. Section 5.2 concerns a related task pertaining to following a lane and doing in lane maneuvers. Maneuvers are any change in own vehicle (OV) dynamics. Whereas in Section 5.1 we emphasize the RCS methodology, in Section 5.2 we emphasize a broader coverage of sensory processing requirements.

5.1. Passing

5.1.1. Applying the 4D/RCS methodology

Domain experts are consulted and play an integral part throughout this entire process. In the case of on-road driving, we are all domain experts, though many of the conditions we examine and the actions we perform are determined subconsciously.

1. Scenario development with a domain expert: For any task in on-road driving, we walk through detailed scenarios with domain experts to deeply understand the actions they take in certain situations, what conditions spawned those actions, and why they believed the actions were most appropriate in that situation. If possible, we try to immerse the domain expert in similar situations and have them talk through their behaviors. In the case of passing on a two lane undivided road, it is often beneficial to drive in a vehicle with the domain expert and to have them describe their process of determining when it was appropriate to pass. Specific conditions that spawn behaviors often change slightly depending on the driver's personality and aggressiveness level, but we try to generalize the behavior to its fundamental components when encoding it in the control system.

2. Develop the task decomposition hierarchy: Before we can encode the knowledge needed to pass on a two lane undivided road, we must understand and build an initial, overall task decomposition hierarchy for on-road driving. This is an iterative process, and the task decomposition hierarchy often changes as new on-road driving scenarios are explored. Changes in the task decomposition hierarchy are much more frequent in the beginning, and gradually reduce in frequency as more scenarios are explored. This passing scenario is one of many scenarios that has been used to develop this task decomposition hierarchy.

3. Determine the conditions that cause you to perform an action and the sub-actions that are needed to perform that action: In the case of passing, the actions that need to be performed are fairly straightforward: change to left lane, follow left lane for some period of time, and change to right lane. This is shown in Figure 5. However, the conditions of when the vehicle should start this sequence of actions and when it should progress from one action to the next is much more difficult to understand.

   Let's examine the conditions when one would initiate a passing operation. In speaking with domain experts, one could decompose the conditions (that must be true to pass) into two categories: namely, that our autonomous vehicle

desires to pass and that the conditions are good to pass. Only when both of these conditions are true will it initiate the passing operation. Through continued interrogation and "what-if" scenarios, we determined five conditions that must be true when it is "good to pass":

(a) it is legal to pass,

(b) the environmental weather and visibility conditions are conducive to passing (often related to weather conditions),

(c) the situation in front of our vehicle is OK to pass (other vehicles, pedestrians, and objects in front of us do not hinder our ability to pass),

(d) the situation behind our vehicle is OK to pass (the vehicle behind us is not passing or tailgating us), and

(e) oncoming traffic allows us to pass safely (we have time to get around the vehicle in front of us).

Each of these five sub-conditions would be recursively decomposed until we identify the objects in the environment, and their pertinent attributes, that impact the decision of whether to perform this passing action.

4. Use the previous step to define the concepts that must be captured in the system's underlying knowledge base, and structure the knowledge base to ensure maximum efficiency for the application: The objects and attributes discovered in the previous step sets the requirements for the knowledge base that underlies the system. Following through with the scenario of passing on a two lane undivided road, in order to evaluate the conditions mentioned in the previous step, the knowledge base must contain concepts such as:

   - other vehicles, including their speed, dimensions, direction, location, and possibly intention;
   - pedestrians, including their speed, orientation, direction, location, and possibly intention;
   - lane markings, along with the type of lane marking;
   - weather conditions and visibility; and
   - signs, including the text on each sign.

   Once these concepts are captured in the knowledge base, they can be structured in such a way to ensure maximum system efficiency.

5. Carefully evaluate all of the above objects and attributes in the context of the appropriate tasks to define resolutions, distances, and timing of the measurement of these items by the sensory processing system: As shown in Figure 6 with the identification of the railroad crossing buck sign, we must define the sizes, shapes, relative locations, and angles to the road, distances at which they have to be identified (thereby setting resolution requirements), etc. This will yield the sensory processing specifications in terms of the world model elements that must be measured and generated. These same specifications become the performance requirements on sensory processing during test and evaluation.

5.1.2. Defining sensory processing requirements for the passing task

In this section, we will examine some detailed examples of requirements for sensory processing, following through with our passing example. In particular, we will determine what it requires of the vehicle sensors to decide, at any given time and speed, if it is legal to pass.

As shown in Figure 6, for a passing operation to be legal, there cannot be:

- any construction in the passing zone
- a transit or school bus stopping in the passing zone
- a no-passing-zone sign in the passing zone
- lane markings that prohibit passing,
- intersections in the passing zone

- a railroad crossing in the passing zone
- a bridge in the passing zone
- a tunnel in the passing zone
- a toll booth in the passing zone

Therefore, the sensory processing system must detect these items, or indicators that these items are approaching, at a distance that allows the vehicle to pass safely. In this analysis we make a few assumptions:

- the vehicle can accelerate comfortably at 1.65 m/s^2
- our vehicle is positioned approximately one second behind the vehicle in front of it (i.e., our vehicle will be at the preceding vehicle's current position in one second travelling at constant velocity)
- our vehicle will begin merging back into its original lane when it is one car length in front of the vehicle it is passing
- the merging operation that brings our vehicle back into its original lane will take one second
- each vehicle is five meters in length

All of these values are variables, and can easily be changed depending on the exact situation. With these assumptions, we calculated the distance that our vehicle would travel during a passing operation, how long it would take to travel that distance, and its final velocity assuming both vehicles have initial speeds of 13.4 m/s (30 m/h), 17.9 m/s (40 m/h), and 26.8 m/s (60 m/h). Table 1 displays the results (we assume un-occluded visibility).

For the "no railroad crossing in passing zone" requirement, there are multiple markings that can indicate a railroad crossing is upcoming, such as a crossbuck just before the railroad crossing, or railroad signs at pre-defined distances before the railroad crossing. Table 2 displays the specification of how far before a railroad crossing a warning sign should be placed, what size the sign must be, and what size the letter on the signs must be, according to the Manual of Uniform Traffic Control Devices[12] (MUTCD).

Speed (m/s)	**Time to Complete Pass (s)**	**Distance Travelled in Pass (m)**	**Velocity at End of Pass (m/s)**
13.4	6.32	117.8	23.9
17.9	6.81	159.3	29.1
26.8	7.68	253.9	39.5

Table 1. Pertinent values for passing operation at various speeds

Considering that the railroad warning sign is a pre-defined distance before the railroad crossing, we can subtract that distance from the full passing distance shown in Table 1 to identify the forward distance our sensors must to be able to sense. These distances are shown in Table 3. This sets the specification for how far a sensor must be able to "see" to determine if there is a railroad crossing sign in the passing zone. However, we can take this one step further and determine what the resolution of the sensors must be to read the sign. The following paragraphs examine the requirements of the sensor itself. We ignore the software that performs the character and object recognition task in this discussion, although we recognize that it is at least as important as the specifications for the sensors.

Speed (m/s)	**Distance from Railroad Crossing (m)**	**Sign Dimensions (m$\times$m)**	**Letter height (m)**
13.4	99	0.450 $\times$ 0.450	0.125
17.9	145	0.450 $\times$ 0.450	0.125
26.8	236	0.450 $\times$ 0.450	0.125

Table 2. Specifications for railroad crossing signs

For a sign that needs to be read (i.e., where its shape and/or color do not convey its meaning), we assume that a 20$\times$20 array of pixels hits on each letter is required to recognize it. Using simple trigonometry based on the distance to the sign and the size of its letters as shown in Table 2, we can determine that a camera with resolutions of about 0.3491×10^{-3} rad (0.02°) is needed for all three cases above.

In some cases, a warning sign is not present and the sensors must rely on recognizing a crossbuck that is immediately before the railroad crossing. In this case, we assume that we need an array of 5×5 pixel hits on the crossbuck to recognize it by shape, and that the size of the crossbuck is the standard 900×900 mm in total dimensions, as specified by the MUTCD manual. Based on this information, we would need a sensor with a resolution as shown in Table 4. Similar calculations could be performed for all other items the sensor would need to sense when determining if it is legal to pass at any given time and speed.

5.2. Lane following

5.2.1. Applying the 4D/RCS methodology

We now describe how we can generate sensory processing (SP) requirements merely by performing the 4D/RCS methodology for a particular command, namely, the FollowLane command. FollowLane is one of the commands processed at the Elemental Maneuvers (EM) level of the 4D/RCS hierarchy. Sensing system requirements are a natural byproduct of performing the steps in the methodology.

The EM level is responsible for executing real-time maneuvers based on the presence of "objects" in the road, any weather-related condition that may change OV motion parameters, and any road surface type that may change OV motion parameters. Road surface types are things like concrete, macadam, gravel, grass, dirt, etc. Objects are defined as any physical item on the road of finite size with respect to the whole road that may require OV to change motion parameters. An object can be part of the road like a pothole. It can be moving like another vehicle or a bicyclist. It can be a large block of ice or a deep puddle of water. However, the EM level knows nothing about the type of the object. What it does have is a boolean indication of whether the generic object can be run over, what is its safe clearance distance (called its "offset"), and what is its maximum safe passing speed. These distillations of higher level information are computed at the Driving Behaviors (DB) level,[5] the next highest level in the 4D/RCS hierarchy, see Figure 3. This distillation of information substantially simplifies the logic and math computations of the EM level. This reduction of computation is the goal of the distribution of labor in a hierarchical control system like 4D/RCS.

The EM level can execute the following commands: FollowLane, PassLeft, ChangeToLeftLane, ChangeToRightLane, TurnRightToLane, TurnLeftToLane, ChangeLanes, StopAt, and a few other commands. In the process of executing these commands EM looks to see if there is any local entity that may require trajectory path adjustment maneuvers such as, slow down, speed up, stop, circumvent object, straddle object, or run over object. These maneuvers permit the autonomous vehicle to travel safely in the presence of other objects (including vehicles, of course), in the presence of motion-affecting local weather conditions like rain or snow, and in the presence of different road surface types such as concrete or macadam. The commands that FollowLane can give to the next lower level, namely the Vehicle Trajectory (VT) level,[5] are DoTrajSeg, DoTrajClockWiseCircularArc, and DoTrajCounterClockWiseCircularArc, see Figure 3.

Speed (m/s)	Passing Distance (m)	Warning Sign Distance (m)	Sensor Sign Distance (m)
13.4	117.8	99	18.8
17.9	159.3	145	14.3
26.8	253.9	236	17.9

Table 3. Sight distance for a railroad warning sign

As is true for the other levels, the EM level can access information that was prepared for it by nodes at higher levels. This includes three tables, one pertaining to information on the autonomous vehicle itself (the Own Vehicle (OV) table – Table 5), another concerning other objects (the Active Objects (AO) table – Table 6) that are being tracked by EM's supervisors; this may include other vehicles, bicycles, objects lying in the road, etc., and another called lane segments (LS) data that is a data base of the lane segments and their properties appropriate to the task to be performed. We assume the availability of a Road Network Database (RND), which provides detailed information on a scenario's road network (*e.g.*, lanes, lane segments, intersections). The LS data is derived from the RND by the Drive Behavior (DB) level, just above the EM level. Tables 5 and 6 display some (not all) of the information that we expect to locate in the OV and AO tables, respectively. For the OV, this includes information such as the autonomous vehicle's current lane segment, position, size, and displacement

Speed (m/s)	Sensor Resolution
13.4	1.819×10^{-3} rad (0.1042°)
17.9	1.241×10^{-3} rad (0.0711°)
26.8	0.7086×10^{-3} rad (0.0406°)

Table 4. Sensor resolution required by a standard crossbuck sign

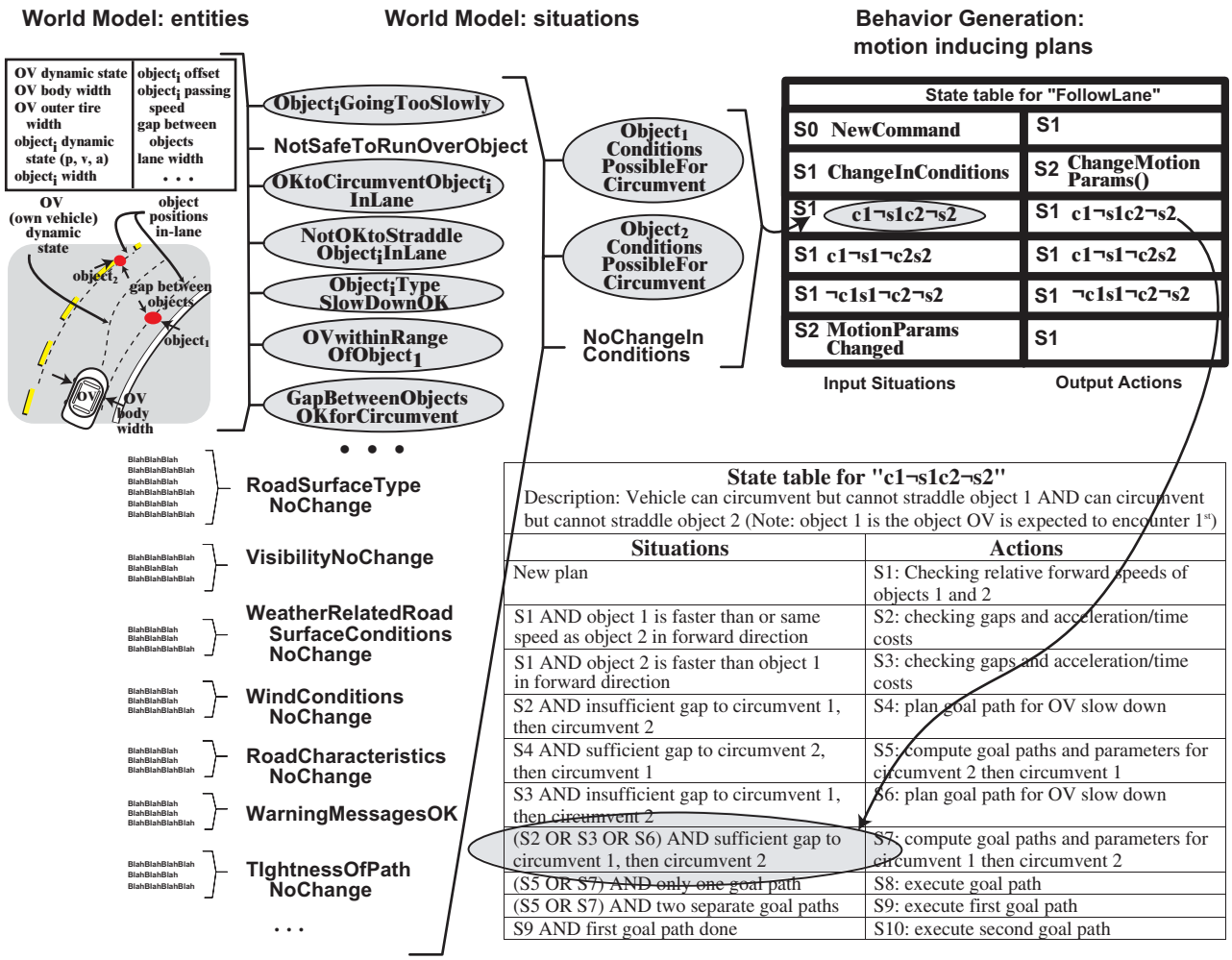

Figure 7. Here is a snapshot of the connection between world model primitives, conditions, and behaviors for EM level activity. The conditions for circumvent object 1 in lane, circumvent object 2 in lane, and NoChangeInConditions have to be true for the followLane state table to run the plan c1¬s1c2¬s2 for the command of the Elemental Maneuvers level in the 4D/RCS hierarchy for the on-road driving task. Shown also are the details of the sub-plan table c1¬s1c2¬s2. This subplan negotiates the relative speeds and positions of the two objects to determine an approximate maneuvering trajectory for OV. The subplan then computes acceleration, velocity, and position values for the OV and sends these at the correct times as command parameters to the vehicle trajectory level (see Figure 3) until the plan is completed or another command is sent to the EM control module from the Drive Behaviors module. To the left of the FollowLane state table along with the value judgment logic for one of this table's conditions. To the left of this logic lie its leaves, which denote the world modelling primitives that it depends on. For example, determination of the gap between objects 1 and 2 is shown as a situation and it feeds into the subplan table c1¬s1c2¬s2 as do object speeds.

from the center of its lane segment. For each object being monitored, its AO table will include information such as the cost of damaging it if comes into contact with the OV, its jerk vector, and its type.

Figure 7 displays a portion of the state transition table for FollowLane, shown on the right. This figure also displays, at the bottom, the subplan to be executed for one of its conditions, namely that there are exactly two active objects in front of the OV in its lane segment, and that, taken independently, both can be circumvented but neither can be straddled. Also shown is the value judgment (VJ) logic for this condition, shown to the left of the condition's position in the state transition table. The far left "leaves" of this logic are the WM entities whose values are required to compute the truth value for this condition. For example, these include such variables as the OV's body width, an object's dynamic state, and the width of the

Table 5. Some parameters maintained in the Own Vehicle table

Name	Description
body width	Width of Own Vehicle (OV)
cost	Cost for damaging OV via contact with another object
desired speed	Desired tangential velocity
displacement	Distance ($\pm$) of OV from center of lane in a direction normal to OV's "forward direction" in the lane segment
dynamic state	position, v_{tan}, a_{tan}, j_{tan}, v_{norm}, a_{norm}, j_{norm}
headlights	Contribution to visibility conditions
inner_tire_width	Width of space between inner right and left tires
LS_id	OV's current lane segment
min_desired_forward_speed	OV's maximum desired speed
noise	Contribution to "conditions around the road"
orientation	OV's current orientation
outer_length	Length of OV from front to rear bumpers
position	Current x, y location of OV
tire_width	Width of OV's tires
underbody_height	Distance between OV's underbody and the road
v_{norm}, a_{norm}, j_{norm}	Current motion parameters in direction normal to OV's "forward direction" in the lane segment
v_{tan}, a_{tan}	Current motion parameters in direction tangent to OV's "forward direction" in the lane segment

current lane segment. The complexity of this figure suggests that the number of nodes in the VJ logic could be quite large, as well as the number of logical functions that must be defined that connect a parent node in the VJ logic to its children, each providing a functional definition for computing the parent's truth value.

For WM entities that are in the OV table, one of the AO tables, or the RND, their values can be directly looked up without requiring any additional sensing. This is because this information was previously sensed, most often by one of EM's supervisors (i.e., an ancestor node in the hierarchy). However, some of these WM entities have not yet been sensed. For example, in Figure 7, examples of WM entities that must be sensed at the EM level include road surface temperature, the road surface normal and tangential spatial derivatives, and the position of an object at the time when the OV will contact it, assuming a constant velocity, acceleration, and direction model for both the OV and object. We focus on this subset of the WM entities in Section 5.2.2, fleshing out their sensory processing needs.

5.2.2. Elementary maneuver level world model entities and conditions

We have determined most of the sensory processing requirements for all the WM entities at the EM level. A few of these entities are shown in Figure 7. The tables in the Appendix B detail these requirements. For each entity, we provide the following information:

- Type: These are either entities or conditions. So far, we have identified only one situation, Water Spray, which we consider to be a situation, due to its transitory existence and the subjective evaluation of its nature.
- Motivation: This summarizes how each entity (or situation) is necessary for deciding conditions critical to behavior. For example, in Figure 7, there are exactly two active objects in front of the OV in its lane. OV can determine the situation, "both are circumventable independently and neither can be straddled" only if individual object positions, velocities, sizes, etc. are known at some frequency and to sufficient accuracy.
- Attributes: These describe details for the entities that need to be sensed. For each attribute, we include its name, possible values, and sensing requirements. In some cases, notes on implementation priorities are also given.

Here is our current list of entities or conditions that need to be sensed at the EM level in our 4D/RCS hierarchy. In Appendix B, we include tables for all of them.

1. Water spray: Composed of a set of sprays, these come from other vehicles, and could possibly impact driving behavior (*e.g.*, slowing down and/or veering) by reducing OV's sensing capabilities. This is the only situation sensed at the EM level; the others are all entities.
2. Air: Attributes include Ambient Temperature, Wind Speed, and Wind Direction. As an example motivation for studying these, OV's behaviors may be impacted by high winds. Ambient temperature may affect the coefficients of friction for different road surface types.
3. Predicted object position: This is the predicted position of the two other objects in the OV's lane at the time that the OV will encounter (each of) them. This information could indicate whether there is sufficient room for OV to safely maneuver about these objects. Currently, we use a simple linear model and constant velocity assumptions using only OV's current dynamic state to predict the positions of encounter. This could be expanded, but probably would be done by a higher level agent module. This item may be computed by the DB module and be placed in the AO table (Table 6).
4. Road surface: This has the attributes Just Wet, Salt, Grit, and Temperature. As an example motivation for sensing, the amount of salt on the road, particularly when its surface contains snow, could impact the maximum safe speed for the OV to traverse it. We may want to use surface freezing temperature instead of amount and type of salt.
5. Spatial derivatives: The derivatives of the lane segment's normal and tangential slope could constrain OV movement and desired speeds.
6. Ice Cover: Composed of the attribute Pattern and a set of Ice Objects, the values of these entities could obviously impact safe speed and acceleration maximums on the OV's driving behavior.
7. Snow cover: Similar to Ice Cover, these entities could likewise constrain safe OV dynamics.
8. Water cover: Similar to Ice Cover, these entities could likewise constrain safe OV dynamics.
9. Wet leaves: Composed of Position, Extent, and Height attributes, these could reduce OV traction and, thus, could impact OV safe speeds and selected path (as can several of the preceding entities).
10. Ambient light: The level and amount of ambient light could affect OV's sensing capabilities. We group this together with three other visibility-affecting conditions, namely Direct celestial light, Shadow, and Street light, which have similar attributes.

The tables in Appendix B describe detailed sensing requirements for a few of these situations and entities, and also some of their sub-entities (*e.g.*, Ice Object, Snow Object, and Water Object). An example of the impact of these entities and conditions on the follow lane behavior is shown in Figure 7.

We have analyzed the sensory processing requirements for each of the WM entities shown in Figure 7 that must be sensed at the EM level. The tables in the Appendix detail these requirements. For each primitive, we provide the following information:

- Type: The primitives are either entities or situations. In this case, we have only one situation (Water Spray).
- Motivation: This summarizes the logic denoted by the value judgment that motivates the role of that primitive for contributing to the decision making for the highlighted condition (i.e., that there are exactly two active objects in front of the OV in its lane, both are circumventable, and neither can be straddled).
- Attributes: These describe details for the primitives that need to be sensed. For each attribute, we include its name, possible values, and sensing requirements. In some cases, notes on implementation priorities are also given.

In the Appendix we include tables for each of the following primitives sensed at the EM level in our 4D/RCS hierarchy:

1. Water spray: Composed of a set of sprays, these come from other vehicles, and could possibly impact driving behavior (*e.g.*, slowing down, veering) by reducing the OV's sensing capabilities. This is the only situation sensed at the EM level; the others are all entities.

2. Air: Attributes include Ambient Temperature, Wind Speed, and Wind Direction. As an example motivation for studying these, OV's behaviors may be impacted by high winds.

3. Predicted object position: This is the predicted position of the two other objects in the OV's lane at the time that the OV will encounter (each of) them. This information could indicate whether there is sufficient room to permit the OV to circumvent these objects. Currently, we use a simple linear model and constant velocity assumptions to predict the positions of encounter.

4. Road surface: This has the attributes Just Wet, Salted, and Temperature. As an example motivation for sensing, the amount of salt on the road, particularly when its surface contains snow, could impact the maximum safe speed for the OV to traverse it.

5. Spatial derivatives: The derivatives of the lane segment's normal and tangential slope could constrain OV movement and desired speeds.

6. Ice Cover: Composed of the attribute Pattern and a set of Ice Objects, the values of these entities could obviously impact maximum safe speed limits on the OV's driving behavior.

7. Snow cover: Similar to Ice Cover, these entities could likewise constrain safe OV speeds.

8. Water cover: Similar to Ice Cover, these entities could likewise constrain safe OV speeds.

9. Wet leaves: Composed of Position, Extent, and Height attributes, these could reduce OV traction and, thus, could impact OV safe speeds and selected path (as can several of the preceding entities).

10. Ambient light: The level and amount of ambient light could affect OV's sensing capabilities. We group this together with three other visibility-affecting conditions, namely Direct celestial light, Shadow, and Street light, which have similar attributes.

In summary, the tables in the Appendix describe detailed sensing requirements for these situations and entities, and also some of their sub-entities (*e.g.*, Ice Object, Snow Object, and Water Object). All are expected to impact decision making for the condition highlighted in Figure 7 for the FollowLane state transition table.

Table 6. Some parameters maintained in the Active Objects table

Parameter Name	**Description**
Acceleration vector	Object's absolute world acceleration coordinates
Behavior type	Object's behavior type *e.g.*, consistent, erratic
Cost	Cost of damaging object through contact with OV
Cost of violating minimum following distance	Cost to OV and object for violating minimum following distance
Cost of exceeding height offset	Cost for damaging an object via contact with OV
Cost to exceed maximum passing speed	Cost to OV and object violating Min following distance
Cost to violate minimum offset	When passing in less distance than Minimum offset
Displacement	Object's displacement from center of its lane segment
Distance OV to object	Distance from object to OV
Dynamic state	Object position, vel, acc, and jerk vectors, orientation
Estimated time past object	Estimated time for OV to completely pass object
Estimated time to object	Estimated time for OV nose to reach object
Headlights	Contribution to visibility conditions
Height	Object's height

ID	Object's unique identifier
Jerk vector	Lane motion
Left side object offset	Distance of object's left side from the center of its lane
Length	Length of object
LS ID	Lane segment within which the object resides
Maximum passing speed	Maximum speed at which the OV is permitted to pass this object
Min following distance	Min tangential distance permitted from OV to this object
Minimum offset	Min value for the desired offset
Noise	Affect of object noise on relevant conditions
Normal acceleration	Object's acceleration along it lane segment
Normal extent	Object's width normal to its lane segment
Offset dynamics	Dynamics required when OV moves past object
Offset normal	Max normal safe distance needed from object for OV to clear it
Offset tangential	Min tangential safe distance needed from object for OV to clear it
Offset vertical	Min height safe distance needed from object for OV to clear it
Orientation	Object's orientation
Passing speed	Desired speed for OV when passing object
Position	Object's position in world coordinates
Relative lane position	Object's normal position relative to OV's position
Relative motion	Object's motion relative to its lane segment
Right side object displacement	Distance of Object's right side from the center of its lane segment
Straddlable	Indicates that ability of OV to straddle object
Straddle flag	Indicates whether supervisor permits OV to straddle object
Subtype	Adult, box, teenager, child, debris dropping, Jersey barrier, etc.
Tangential acceleration	Object's acceleration along its forward direction
Tangential extent	Object's length along its forward direction
Type	Bicycle, obstacle, pedestrian, car, truck, barrier
Velocity lane normal component	Object's velocity perpendicular to its lane segment
Velocity lane tangential component	Object's velocity along its lane segment
Velocity vector	Object's absolute world velocity coordinates
Width	Object's width

6. SUMMARY

Our goal is to produce a taxonomy of on-road driving behaviors that can be further analyzed to produce the specifications for identifying the world model entities, features, attributes, resolutions, recognition distances, and locations for each separate driving task. These specifications can be used as the basis of performance metrics for sensory processing and world model building. This requires representing two sets of domain knowledge. One is the task decomposition knowledge that defines the sequences of subtask activities for every aspect of every type of driving task. This task decomposition knowledge is encoded into ordered sets of production rules clustered by the context of the individual driving tasks. These rules consist of input conditions (present world situations) that, when matched, cause the output of the appropriate sub-task goals. The second set of domain knowledge is the detailed world state descriptions and evaluation functions required to produce the world situation symbolic values that are used as the input transition conditions by the task decomposition rules.

We described how the 4D/RCS methodology and reference architecture was used to define the task decomposition and the resulting state tables of production rules. We then described how the input conditions of these rules were further evaluated to derive all of their dependencies on all of the corresponding world model states and world entities, features, and attributes. Still using the context of the individual driving tasks, the appropriate recognition distances were factored in to attain a specification of the requirements for the sensory and world model processing necessary for each separate driving task behavior. These requirements now serve as both a requirements list for the development of the sensory and world model processing for different on-road driving tasks as well as the performance metrics against which they can be measured to assess the correctness of their operations.

The main need for future research is to spend the huge effort and painstaking work of defining FSMs, world model entities, and sensing requirements for all the tasks in the system, and then continuously reforming the design based on feedback received from implementation and testing. Such an effort will only be taken when there is either the will to accomplish the effort, or there is an broadly accepted persuasion that no short cut or silver bullet is available to accomplish the tasks, only techniques to make the design task more efficient.

REFERENCES

1. F. Dellaert, D. Pomerleau, and C. Thorpe, "Model-based car tracking integrated with a road-follower," in *International Conference on Robotics and Automation*, 1998.
2. R. Ichise, D. G. Shapiro, and P. Langley, "Learning hierarchical skills from observation," in *Proceedings of the Fifth International Conference on Discovery Science*, pp. 247–258, Springer-Verlag, (Lübeck, Germany), 2002.
3. E. D. Dickmanns, "An Expectation-based Multi-focal, Saccadic (EMS) Vision System for Vehicle Guidance," in *Proceeding 9-th International Symposium on Robotics Research (ISRR'99) , address=*,
4. J. S. Albus, *et al*, "4D/RCS version 2.0: A reference model architecture for unmanned vehicle systems," (NISTIR 6910), (Gaithersburg, MD: National Institute of Standards and Technology, Intelligent Systems Division), 2002.
5. A. Barbera, J. Horst, C. Schlenoff, and D. Aha, "Task analysis of autonomous on-road driving," in *Proceedings of the SPIE*, D. Gage, ed., **5609**, SPIE Mobile Robots XVII, (Optics East, Philadelphia, PA USA), 2004.
6. J. Horst, "Architecture, design methodology, and component-based tools for a real-time inspection system," in *Proceedings of the 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, 2000.
7. 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* **1**, p. 15, 1996.
8. E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen, "The seeing passenger car 'VaMoRs-P'," in *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pp. 68–73, IEEE Press, (Paris, France), 1994.
9. J. S. Albus and A. M. Meystel, *Engineering of mind*, John Wiley & Sons, New York, 2001.
10. A. M. Meystel and J. S. Albus, *Intelligent systems: Architecture, design, and control*, John Wiley & Sons, New York, 2002.
11. J. McKnight and B. Adams, "Driver Education Task Analysis. Volume 1: Task Descriptions," (Washington, D.C.: Department of Transportation, National Highway Safety Bureau, Human Resource Research Organization), 1970.
12. "Manual on uniform traffic control devices (MUTCD 2000) millennium edition." U.S. Department of Transportation, Federal Highway Administration, 2000.

APPENDIX A. STATE TRANSITION TABLE AND VALUE JUDGMENT LOGIC FOR FOLLOWLANE

Figure 7 presents sample of the entire state transition table for FollowLane, a task within the Elemental Maneuvers Level in the 4D/RCS hierarchy for NIST's on-road driving project. A complete listing of this state table is in Table 7

Table 7. This table displays the full FollowLane state table associated with the action of the highlighted condition-action pair in Figure 7. We next display the value judgment logic required by this condition.

State: Conditions	State: Actions
S0: newCommand	S1
S1: changeInConditions	S2: pl_changeMotionParams
S2: motionParamsChanged	S1
S1: noObjects	S1: pl_noObject
S1: can circumvent object 1 AND can straddle object 1 AND exactly one object	S1: pl_c1s1
S1: can circumvent object 1 AND cannot straddle object 1 AND exactly one object	S1: pl_c1s1
S1: cannot circumvent object 1 AND can straddle object 1 AND exactly one object	S1: pl_c1s1
S1: cannot circumvent object 1 AND cannot straddle object 1 AND exactly one object	S1: pl_c1s1

S1: can circumvent object 1 AND can straddle object 1 AND cannot circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND can straddle object 1 AND cannot circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND can straddle object 1 AND can circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND can straddle object 1 AND can circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND cannot straddle object 1 AND cannot circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND cannot straddle object 1 AND cannot circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND cannot straddle object 1 AND can circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: can circumvent object 1 AND cannot straddle object 1 AND can circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND can straddle object 1 AND cannot circumvent OR cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND can straddle object 1 AND cannot circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND can straddle object 1 AND can circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND can straddle object 1 AND can circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND cannot straddle object 1 AND cannot circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND cannot straddle object 1 AND cannot circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND cannot straddle object 1 AND can circumvent object 2 AND cannot straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2
S1: cannot circumvent object 1 AND cannot straddle object 1 AND can circumvent object 2 AND can straddle object 2 AND exactly two objects	S1: pl_c1s1c2s2

APPENDIX B. TABLES OF SENSORY PROCESSING REQUIREMENTS

Below we present 12 tables concerning WM entities and SP requirements for the FollowLane command that were introduced in Section 5.2.2. These tables describe WM entities, their attributes, the attribute name, the required accuracy and precision in the values of the attributes, and finally the sensing needed to generate that attribute at the required accuracy. FollowLane is a command performed at the Elementary Maneuvers level of the 4D/RCS control hierarchy for NIST's on-road driving task as sketched in Figure 7. Absent more precise detail on a given task, we chose to approximate intuitively the required accuracy for a sensing parameter.

This is still a work in progress. For example, the assignment of appropriate priorities of implementation are not yet complete.

Situation	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Water Spray	Sub-Situations	Water Spray per Other Vehicle			
	Motivation	This type of visibility condition could impact motion-affecting conditions (e.g., either by constraining some OV motion dynamics parameter maximums or represent a sufficient change in conditions so as to affect driving behavior).			
Water Spray per Other Vehicle	Attributes	Vehicle	Available in AO Table		
		Current height & width	{h, w} to nearest 0.5m	Need water spray object ID and image segmentation capability	Transparency is important and is strongly dependent upon the current amount of rain precipitation. Height, width, and location important for any planned maneuvering. However, priority is lower because there is usually a safe option, namely, slow down to place larger distance between OV and spraying vehicles
		Current location	Center of mass {x, y, z} m		
		Predicted max height & width	{h, w} to nearest 0.5m		
		Current transparency	Percent	Requires knowledge of depth of water attribute or water object	
		Vehicle (that caused this spray)	{Motorcycle, car, truck, large truck}	Detection of the amount of anything NOT water spray within the field of view	
		Start time	Estimated Greenwich mean time, nearest second	Pointer to the Active Objects Table	
		Expected Duration	Time in seconds	+/- 0.25 sec	
	Parent	Water Spray			

Table 8. Sensing requirements for water spray attributes

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Air	Motivation	Temperature is a weather-related road surface condition, which is a type of road surface condition, and it can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing a sufficient change in conditions so as to affect driving behavior). Wind direction and speed are wind conditions that could impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing sufficient changes in conditions so as to affect driving behavior).			
	Attributes	Ambient Temperature	Temperature T in degrees Celsius; accurate to ±0.5˚ C generally and ±0.25˚ C for a range of roughly ±10˚ C around the freezing point at the road surface	Precision: ±0.25˚C (±0.1˚C about road surface freezing point)	
		Wind direction	Scalar angle in radians	Precision: ±π/8 radians	This precision allows for directions {north, north-northwest, west-northwest, west-southwest, etc.}
		Wind speed	Scalar in m/s	Precision: ±5 m/s	

Table 9. Sensing requirements for air-related attributes

Entity	Frame			Sensing Requirements	Motivation
	Component	Name	Values		
Predicted Object Position	Motivation	The predicted position of each of the two objects, at the time of encountering OV, along with the lane width at that location, will help to determine whether the OV can circumvent that object while staying within its lane. These variables are also consulted when considering whether it is unsafe to go around an object that is moving too slowly and will soon be encountered.			
	Attributes	Lane Segment Parameters	Predicted <x,y> position of object for each of next 10s	Improvement of dynamic values in AO table based on EM's sensors (e.g., LADAR)	
		Lane Width	scalar double	None: Look up from RND	

Table 10. Sensing requirements for predicted object positions

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Road Surface	Motivation	These weather-related road surface conditions can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing a sufficient change in conditions so as to affect driving behavior).			
	Attributes	Just Wet	Time in minutes since most recent precipitation situation began, if precipitating	History of current precipitation situation? Some other analysis of oil slickness and/or traction?	Low
		Salt	Amount in grams per square meter; Type of salt with melting properties and friction props	Three possible methods: 1) road surface image processing, 2) active tests of traction on road, and 3) explicit reports of salt	Medium; most freezing-temperature-reducing elements are based on some type of salt (potassium, magnesium, or sodium chloride)
		Grit	A possible value: coefficients of dynamic and static friction as a function of surface position	Three possible methods: 1) road surface image processing, 2) active tests of traction on road, and 3) explicit reports of grit application	Medium; may simply depend on local and real-time measurements of friction
		Tem-pera-ture	Temperature **T** in degrees Celsius; accurate to ±0.5° C generally and ±0.25° C for a range of roughly ±10° C around the freezing point at the road surface	Precision to satisfy **T**	Medium

Table 11. Sensing requirements for road surface conditions

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Spatial Derivatives	Motivation	These are a type of road characteristic that could impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing sufficient changes in conditions that affect driving behavior).			
	Attributes	Tangential Spatial Derivative (to detect hills, dips)	[m≥10˚, 10˚≥m≥5˚, 5˚≥m≥0˚, 0˚≥m≥-5˚, -5˚≥m≥-10˚, -10˚≥m], for 11 intervals of 0-10 seconds in front of OV assuming constant dynamic state	Slope detector on OV for local slope and LADAR for surrounding slopes	medium
		Normal Spatial Derivative	Same as for Tangential, bounded by LS width	Slope detector, LADAR	medium

Table 12. Sensing requirements for road spatial derivatives

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Ice Cover (on current LS)	Motivation	Ice coverage is a weather-related road surface condition that can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing a sufficient change in conditions so as to affect driving behavior).			
	Attributes	Pattern	{Uniform, Left, Right, Spotty}	-	
	Sub-entity	Ice Object			
Ice Object (on current LS)	Attributes	Relative Tangential Position in LS	Relative distance in feet from front tires	Precision: ±0.1 m	
		Relative Normal Position in LS	Relative location in feet from center of OV	Precision: ±0.1 m	
		Extent	Length and width in feet, centered on the Relative Position	Precision: ±0.1 m	
		Orientation	Degrees	Precision: ± π/8 radians	
		Average Thickness	Thickness of this ice object	Precision: ± 0.5in	
		Type	{Cracked, Smooth, Rough}		

Table 13. Sensing requirements for ice cover

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Snow Cover (on current or local LS)	Motivation	Snow coverage is a weather-related road surface condition that can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing a sufficient change in conditions so as to affect driving behavior).			
	Attributes	Moisture Level	{Very wet, moderately wet, dry}	Temperature measurement plus 'white' color detector	This is a total characteristic of the road
		Pattern	{Uniform, left, right, spotty}	Image segmentation plus 'white' color detector	Need to distinguish between dry and snow-covered road surfaces for determining kinematics maximums. Color detection may be sufficient. Left and right are important since finding a dry spot can greatly increase coefficient of friction for tires.
		Depth	{Shallow, deep, very deep}	May be difficult to detect generally: combination of precipitation history (precipitation/state/duration) integrated with precipitation/attribute/rate	Shallow means tires are still touching some pavement; deep means tires contacting snow only; very deep means not traversable at any speed
	Sub-component	Snow Object			

Table 14. Sensing requirements for snow cover

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Snow Object (on current or local LS)	Attributes	Relative Tangential Position in LS	Relative distance in feet from front tires	Precision: ±0.1 m	A snow object is some finite snow entity that needs to be analyzed further for possible impact on driving maneuver (not just change in speeds and accelerations), i.e. OV will most likely have to avoid running over this entity
		Relative Normal Position in LS	Relative location in feet from center of OV	Precision: ±0.1 m	
		Extent	Length and width in feet	Precision: ± 0.1 m	
		Orientation	Degrees	Precision: ± $\pi/8$ radians	
		Type	{Pile, Drift, Untouched, Flattened, Slush}		Piles and drifts have above-average height/depth vs. coverage, but drifts are otherwise Untouched. Flattened results in below-average but relatively consistent depth, while slush has a high ("Wet") moisture level and highly variable depth
		Moisture Level	{Wet, Dry}	Function of air, road surface temperatures, and amount of solar radiation	
		Average Depth	Centimeters	Precision: ±2 cm	
	Parent	Snow Cover			

Table 15. Sensing requirements for snow object

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	**Name**	**Values**		
Water Cover (on current or local LS)	Motivation	Water coverage is a weather-related road surface condition that can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing a sufficient change in conditions so as to affect driving behavior).			
	Attributes	Moisture Level	{Very wet, moderately wet, dry}	Mere detection of specularity may be sufficient	This is a total characteristic of the road and can be used to describe wetness levels at the onset and the tail end of a rainfall situation, for example
		Pattern	{Uniform, left, right, spotty}	Mere detection of specularity may be sufficient, but also need image segmentation capability	Need to distinguish between dry and wet road surfaces for determining kinematics maximums. Left and right are important since finding a dry spot can greatly increase coefficient of friction for tires.
		Depth	{Shallow, deep, very deep}	May be difficult to detect generally: might require 3D analysis of the intersection of water surface plane with predicted road surface under the water (LADAR)	Deep means too deep to traverse at speed limit without hydroplaning; very deep means not traversable at any speed; in this latter case, the water cover should be considered a water object
	Subcomponent	Water Object			

Table 16. Sensing requirements for water cover

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Water Object (on current or local LS)	Attributes	Relative Tangential Position in LS	Relative distances in feet from front tires	Precision: ±0.1 m	medium Water on the road typically becomes an object when water depth is "deep" or "very deep" (see definitions above)…affecting traverability
		Relative Normal Position in LS	Relative location in feet from center of OV	Precision: ±0.1 m	
		Extent	Length and width in feet	Precision: ±0.1 m	
		Orientation	Degrees	Precision: ±π/8 radians	low
		Depth	{Thin layer, Deep enough to hydroplane at speed limit, Not traversable}	There are several methods to detect water depth, and most of them are challenging. Options: 1) static image analysis (specularity), 2) situation analysis (rain type, precipitation history, knowledge of road contour, spray emanating from other vehicles), 3) dynamic image analysis (waves)	medium
	State	Velocity	m/sec	Precision: ±10 m/s	low
		Flow Direction	Angle in radians	Precision: ± π/8 rad	low
	Parent	Water Cover			

Table 17. Sensing requirements for water object

Entity	**Frame**			**Sensing Requirements**	**Implementation Priority (high, medium, low); Comments**
	Component	**Name**	**Values**		
Wet Leaves	Motivation	This type of road surface condition can impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing sufficient changes in conditions so as to affect driving behavior).			
	Attributes	Relative Position	<x,y> offset from OV		low
		Extent	Length and width in feet		low
		Height	Max Height		low

Table 18. Sensing requirements for wet leaves

Entity	Frame			Sensing Requirements	Implementation Priority (high, medium, low); Comments
	Component	Name	Values		
Entities Related to Visibility Conditions (see below)	Motivation	These time-of-day visibility conditions could impact motion-affecting conditions (e.g., by constraining some OV motion dynamics parameter maximums or representing sufficient changes in conditions so as to affect driving behavior).			
Ambient Light	Attributes	Level	Candela per square meter (cd/m^2)		
		Type	{sun , moon}		
Direct Celestial light		Amount	{in Candela per square meter (cd/m^2)}		
		Amount of obstruction of forward view	{serious, minor, none}		
		Amount of obstruction of rear view	{serious, minor, none}		
Shadow		Relative Position	<x,y>		
		Gradient between shadow region and lighted region {in cd/m^2}			
Street light		Relative Position	<x,y>		
		Level	Candela per square meter (cd/m^2)		

Table 19. Sensing requirements for visibility conditions relating to the time of day