A Comprehensive Microscopic Autonomous Driver Model for Use in High-Fidelity Driving Simulation Environments

Yiannis Papelis and Omar Ahmad

Yiannis Papelis and Omar Ahmad, National Advanced Driving Simulator and Simulation Center, The University of Iowa, 2401 Oakdale Blvd., Iowa City, Iowa 52242-5003

Driving simulation is an invaluable tool for conducting driving-related research. In any virtual driving environment, autonomous traffic is necessary to complete the driver's immersive experience. The higher the fidelity of a driving simulator, the more stringent are the requirements for the realistic appearance of autonomous traffic. Unfortunately, an increase in the fidelity, or the complexity, of the traffic simulation model does not always enhance the appearance of realism. In our experience with driving simulation applications, the number of behaviors exhibited by the autonomous driver models is the biggest factor in how observers rate the realism of the autonomous traffic; even when compared with a smaller number of high fidelity or validated behaviors. Based on this feedback, we have developed a comprehensive driver model that focuses on breadth of behaviors and is founded on a solid software architecture that supports growth by providing a framework that allows progressive addition and improvement of behaviors.

INTRODUCTION

For many years now, driving simulation has been used as a technique for studying numerous aspects of driving-related subjects. These include driving safety, infrastructure improvements, drugs, and new in-vehicle technologies. Driving simulation, especially in high-fidelity devices, provides several advantages when studying human-centered, environment-centered, or equipment-centered phenomena. A key advantage is that a researcher can put the driver in situations that are either too dangerous to re-create in real life [1], or that require equipment or infrastructure that does not yet exist [2]. In a simulation, situations are repeatable, and the simulator's virtual environment can be fully specified and controlled. Furthermore, with the exception of dependencies on the natural environment—i.e., in-season antihistamine studies [3], environmental factors such as time of day or atmospheric conditions can be completely controlled.

Achieving immersion within a simulator's environment requires a combination of software that models the static and dynamic aspects of the environment and hardware that provides the necessary cues as produced by the software. A key component of any driving simulation virtual environment is the simulation of various active entities such as traffic lights and other traffic participants, generally referred to as traffic. Traffic is a key component because the driver's experience is based largely upon interaction with other drivers. Therefore, providing a realistic simulation of individual traffic elements, often referred to as microscopic traffic simulation, is a necessary

component of high-fidelity driving simulation. A microscopic traffic simulation model, rather than a statistical model, is necessary to ensure that the behavior of individual entities is generated in enough detail that the cueing subsystems, in this case the image generator (IG), can display the motion of each traffic element in real-time.

This paper describes the implementation of a microscopic traffic simulation system that has been designed primarily for driving simulation applications. The design of the system is presented within a framework of requirements raised by users of simulators. This introduces unique aspects to the requirements and the motivation behind decisions on how the system was constructed. The remainder of this paper is organized as follows. The first section provides the motivation for the work described here along with background on traffic simulation systems and how they relate to the system presented in this paper. The second section provides the foundation necessary to fully describe the driver model by describing the mathematical formalism and notational framework used. The third section then describes the model and provides some examples of its operation. The final section provides a conclusion and describes some of the planned future work.

MOTIVATION AND BACKGROUND

This paper presents a comprehensive autonomous driver model that is part of an overall microscopic traffic simulation system used in driving simulator applications. The system described in this paper is currently in its second generation of development and will be used in the National Advanced Driving Simulator (NADS) [4]. Prior generations of this model have been used in other high-fidelity driving simulators with various customizations focused on the target use of the particular simulator [5, 6]. In all cases, however, the motivation was the development of a driver model that would appear realistic to the simulator users under all operating conditions supported by the simulator. Here, the term *simulator user* includes not only the driver, but also any trainers or researchers involved in the use of the simulator. On the surface, that was a rather simple, although comprehensive, requirement. In practice, it turned out to be a very stringent requirement that necessitated numerous customizations and extensions to the basic model. There were two primary reasons for the escalation of complexity. The first is that non-technical people are more likely to be dissatisfied with the traffic model because they do not understand the complexity of building a model. At the same time, they tend to expect all simulator modules to be of consistent fidelity and work. For example, if the steering wheel and dashboard work exactly as in a real car, then oncoming traffic should also behave the same. The second reason for the increased complexity was that the model used for the road network supported realistic, three-dimensional roadways with complex intersections, varying width roads, and

a variety of traffic participants in addition to the basic vehicle. For example, the types of vehicles populating the roadway included anything from cars and trucks to trams, bicycles, and tanks. Such a complex environment within a high-fidelity simulator raises the expectation of all elements operating as in real life. Bicycles particularly provide a challenge because they drive on bicycle lanes but also require special treatment by the driver model used in the cars because cars drive on the left side of bicycles. Similarly, wide trucks posed several problems in narrow rural two-lane roadways because if they simply followed the lane centerline, they would collide with vehicles on adjacent oncoming lanes. The complexity of the seemingly innocuous requirement of "normal" behavior forced us to analyze and decompose it into two related requirements. The first requirement is developing a comprehensive model that exhibits a very rich set of behaviors, and the second is developing a modular model that can be easily extended and improved, even late in the life cycle.

Richness of behaviors refers to the number of distinct behaviors exhibited by the model. Here, we use the term *behavior* to refer to a particular set of observable actions consistently exhibited by the driver model, either continuously or in response to specific situations. Some simple examples of behaviors are lane tracking and vehicle following. The observable action in lane tracking is the generation of steering commands that ensure the vehicle will track the lane centerline. For vehicle following, the observable action is the change in speed to ensure a certain distance when driving behind another vehicle. Another example of a complex behavior is overtaking a slower vehicle by crossing over to a lane of oncoming traffic [7]. Note that in composing an overall autonomous driving model, some behaviors are minimally necessary, whereas other behaviors may be optional. For example, one would generally agree that lane tracking and vehicle following are *de-facto* necessary behaviors. A behavior such as overtaking, described above, may be optional. Naturally, properly classifying behaviors in this manner requires a full definition of each behavior. Our goal in classifying behaviors was to decompose the overall model in modules that can be worked on individually, thus satisfying the modularity requirement.

It is important to note that in talking about individual behaviors, there has been no mention of the fidelity or the validity of that behavior. For example, there is a large body of research that focuses on developing high-fidelity driving following models [8], validating specialized behaviors such as the behavior near toll plazas [9], and measuring the effect of roadways on traffic density [10]. For vertical applications, such depth in analyzing and perfecting a model is necessary, and our initial approach was similar in that it focused on the incorporation of some basic but high fidelity behaviors. In our experience with research studies in high-fidelity simulators, users generally

focus their evaluation of the model realism towards the richness of the behaviors, not their fidelity. When observing an autonomous driver model, a person is more likely to observe that vehicles didn't overtake, when they should have or that they didn't move over to avoid a collision with an oncoming vehicle, than that the following distance didn't vary according to a validated model. To some degree, this can be explained by considering the *observability* of these behaviors. For example, consider a series of vehicles traveling along the same direction on a rural two-lane highway, all controlled by an autonomous driver model with the exception of the lead vehicle that is driven by a human. For the human driver, observing the inter-vehicle distance is harder than observing that the vehicles won't pass, even when the driver slows down to a near-zero speed and moves over to the right. Another plausible explanation is that the relative magnitude of problems caused by the lack of certain behaviors is much higher than the problems caused by using behaviors that are not validated or are of lower fidelity. Consider the example used above with the additional scenario requirement that that the human driver pulls over and lets the other vehicles get ahead. If the human driver pulls to the right but stops with part of the vehicle still on the lane, the rest of the vehicles may still consider that driver to be "on the road" and thus stop. Without an "overtaking" behavior, or some behavior that ensures that vehicles move over from the centerline to bypass an obstacle, such a scenario would fail every time the human driver pulls over without completely removing the vehicle from the road. When comparing this traffic model with another that includes a simple overtaking behavior that has never been validated, it becomes clear that most human observers would rank the latter model as more realistic than the former. One can make numerous similar examples involving behaviors near intersections or the reactions of autonomous driver models to non-standard actions by the human driver. In our experience, many catastrophic problems in traffic simulation systems, such as deadlocks, collisions, or infinite wait situations, can be attributed to lack of behaviors rather than to low fidelity or models that have not been validated.

Note that the above discussion is not meant to imply that developing high-fidelity models of behaviors or validating traffic models is of no use in driving simulator applications. To the contrary, certain types of research necessitate validating at least part of the model. However, it is important to notice that, in general, when looking into traffic models for driving simulation applications, the range of behaviors is equally important, and sometimes more important, than the fidelity of the behaviors. Once a set of minimal behaviors, as dictated by a specific application, has been developed, validation of individual behaviors is the next step in the evolution of the model.

The autonomous model described in this paper has been designed to accommodate a large number of individual behaviors. This model includes the following behaviors: lane tracking, close-coupled object following, decoupled object following, lane change, oncoming traffic collision avoidance, static object collision avoidance, highway entry and exit ramp merging, side obstacle avoidance, slow or stopped vehicle overtaking, and an extensive set of intersection navigation behaviors. The intersection navigation behaviors can negotiate STOP and YIELD signs, uncontrolled intersections, and intersections controlled partially or fully by traffic lights.

The first-generation implementation of the model was customized for driving on roadways in Switzerland. In addition to the behaviors described in the previous paragraph, this model incorporated additional behaviors to negotiate narrow streets, obey Swiss traffic light timing, and interact with trams and tanks on the road network. The second generation of this model, currently being implemented for the NADS, does not include the Swiss driving customizations. Rather, it includes specific behaviors for intersection navigation of traffic lights, including right on red and left on green without a left arrow. In addition, the next-generation software employs an object oriented design and is being implemented in C++. To enhance the realistic appearance of the traffic, the system uses a multi-body vehicle simulation to predict the physical motion of the vehicle. To accommodate vehicle models with different capabilities (i.e., acceleration, turning rate), the behavioral model has been designed to consult a set number of look-up tables that describe the vehicle's response in certain maneuvers. The automatic generation of these look-up tables allows easy incorporation of different vehicle models in the traffic.

AUTONOMOUS DRIVER MODEL FRAMEWORK

The autonomous vehicle model has been implemented using Hierarchical Concurrent State Machines (HCSMs) [11]. Even though HCSMs provide both relationship and execution semantics similar to traditional state machines, they extend the traditional state machine formalism by introducing hierarchy and concurrency. Hierarchy is introduced by allowing one HCSM to contain zero or more HCSMs. This defines a parent-child relationship among HCSMs. Collectively, one refers to a top-level HCSM and all of its children as the HCSM tree. Execution of the model takes place by executing two pieces of code associated with each HCSM in the tree: a pre-activity and a post-activity function. Executing an HCSM takes place by performing an in-order traversal of the HCSM tree. During the first visit to the HCSM, the pre-activity function executes and then the children are visited. During the second visit, the post-activity function executes. Concurrency is introduced by allowing an HCSM to have either concurrent or sequential children. When concurrent, all children are visited and their activities executed. When

sequential, only one child is visited. The children of a sequential HCSM are connected with transitions which determine which determine which one of the children HCSM has control. As in traditional state machines, each transition is associated with a predicate function that executes every frame. When the predicate evaluates to true, the transition fires and control is transferred from the originating to the terminating HCSM.

Several methods facilitate inter- and intra-HCSM communication. These include input and output parameters, which are associated with each HCSM, and dials and buttons, which are associated with an HCSM tree. Each HCSM can have any number of input and output parameters that can only be accessed between HCSMs that have a parent-child relationship. When building a model, a parent HCSM can send information to its child's input parameters and read information from its child's output parameters. Typically, a parent HCSM writes values to its child's inputs during its pre-activity function and reads values from its child's outputs during its post-activity function. Both input and output parameters can be marked as having "no value." Dials and buttons are used primarily for communication among HCSM trees. A dial is a named typed variable that can be set by any HCSM, provided that the HCSM tree and dial name are known. A button is a specialization of a dial, in that a button can only hold a Boolean value, and will reset to false after each execution step.

Figure 1 illustrates the architecture of the overall traffic simulation framework. It displays two major components. The HCSM Runtime System maintains the instances of the HCSM trees and executes the HCSMs. The Correlated Virtual Environment Database (CVED) maintains information about the physical environment of the roadway, including dynamic objects and supports flexible queries that provide real-time access to information about the environment.

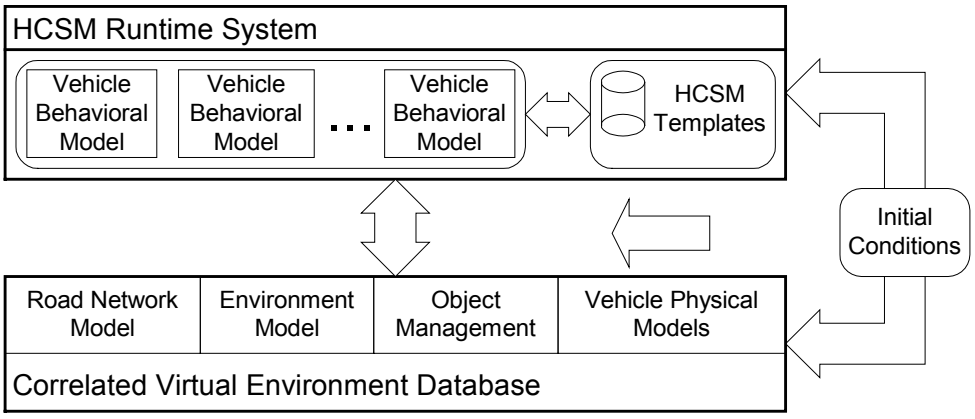

Figure 1 - Overall driver model framework.

The HCSM Runtime System supports dynamic creation and deletion of HCSM trees. To create an HCSM, one provides the name of the HCSM template and a set of initial parameters, and the runtime system then creates the HCSM instance. Deleting HCSMs can also be done dynamically by providing a handle to an existing HCSM's instance. The code associated with the pre-activity or post-activity of an HCSM can invoke the runtime system either to create additional HCSMs or to delete existing HCSMs. An HCSM instance does not necessarily reflect a driver model. The behavior of any object whose state can dynamically change during the course of a simulation can modeled as an HCSM. As an example, consider a high-level manager like the traffic manager HCSM that maintains the position of the simulator driver and creates or deletes other HCSMs to maintain random traffic around the driver. When the number of traffic participants yields a density below a certain threshold, the traffic manager creates a new vehicle HCSM and provides it with the appropriate initial conditions for starting position and initial velocity. If the number of traffic participants yields a density above the threshold, the traffic manager would pick one of the existing HCSMs to delete.

The HCSM runtime system begins execution by reading the contents of an "Initial Conditions" file that lists the HCSM templates that represent HCSM trees to instance during initialization. Generally, the HCSMs listed in this file represent high-level managers, such as the traffic manager, traffic light manager and static object manager. These managers initialize the environment for the simulation being executed by creating the appropriate traffic around the simulator driver, initializing the state and timing of traffic lights and initializing the state of all static objects respectively.

CVED provides the real-time interface to the virtual roadway environment. It provides a rich set of functions to obtain and set the state of the virtual environment. The virtual environment comprises several subcomponents, some static and some dynamic. The road network model, which includes the road pavement description, road markings, intersections, etc., is a static component that cannot be modified at runtime. Dynamic components include environment conditions such as wind and rain, and all objects, including vehicles, traffic signs, and traffic lights. It is important to note that CVED vehicles represent the physical presence of a traffic element, while HCSMs represent the behavioral model of a traffic element. There is a one-to-one correspondence between an HCSM instance representing a driver model and a CVED object representing the vehicle controlled by the driver model. Furthermore, each CVED vehicle object is associated with a dynamic model that is responsible for calculating the motion of the vehicle based on control inputs provided by the associated HCSM.

An external synchronization agent controls execution of the system. It calls functions that execute all HCSMs and then calls functions that execute the various physical models. Usually, the synchronization agent is part of the simulator's real time system. In stand-alone applications, a small program can be used to implement a loop that calls the HCSM and CVED managers in the proper sequence. One of the advantages of decoupling the behavioral and physical models is that they can execute in different frequencies, which allows better coupling to the simulator's environment. For example, it is generally best to run the physical model at the same frequency, such as 30 or 60 Hz, that the Image Generator (IG) renders the scenery. That way, the motion of the vehicles is smooth and coordinated with the visual scene rendering. The behavior execution however can run at a lower frequency, such as 10 or 15 Hz. This introduces a reaction time to all autonomous models that increases the realism. For the remainder of this paper, we use the term *frame* to refer to the execution of all HCSM trees followed by execution of the physical models, independent of the framework under which they have been integrated.

GENERAL STRUCTURE OF THE OVERALL DRIVER MODEL

Figure 2 depicts the high-level model structure with two levels of HCSM structure. The Autonomous HCSM is the root of the HCSM tree; its pre- and post-activity functions are responsible for most of the interaction between the behavioral model and CVED. Note that the remaining HCSMs in Figure 2 are concurrent children of Autonomous, which means that all children execute during each frame. All children have one input and two output parameters. The input parameter receives a composite data structure with information about the environment. Unlike CVED, which provides information about any place in the database, this data is filtered and reflects only data relevant to the actual position of the vehicle controlled by this instance of the model. The pre-activity of the Autonomous HCSM queries CVED and packages the information providing significant time savings by eliminating duplicate queries to CVED by all the children seeking the same information in the same frame. The output parameters of all children are accessible by the parent. Of the two output parameters, one contains a speed control command, and the other contains a position guidance command. The speed control command is expressed as a pair {V, d}, where V represents the desired velocity and d represents the distance ahead at which point that velocity should be achieved. The position guidance command is expressed as a two-dimensional point {x, y} and represents a target point that the vehicle should drive over with an orientation aligned to the vector originating at the current location of the vehicle and terminating at {x,y}. These commands are conveyed to CVED, which uses two controllers to generate a steering angle and gas pedal displacement inputs that are finally fed to the dynamic model

of the vehicle. The controllers utilize a vehicle dynamics model to implement feed forward loops ensuring achievement of the targets within the capabilities of the vehicle being modeled. The internal details of the controllers are beyond the scope of this article, but for purposes of this discussion note that the various HCSMs have explicit knowledge of the capability limits of the vehicle dynamics, and are responsible for maintaining the vehicle operating within a certain state space.

Note that each child of Autonomous, which is generally responsible for a specific behavior or a group of related behaviors, produces one or both of the necessary sets of control inputs that, when passed to CVED, would make the vehicle exhibit the particular behavior.

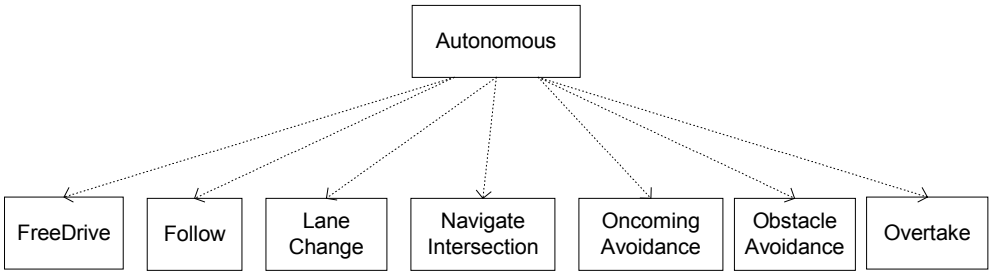

Figure 2 - Autonomous vehicle HCSM.

A key component of the model is how the Autonomous HCSM fuses the outputs of all its children HCSMs in determining the actual control inputs conveyed to CVED, which in turn dictate the behavior exhibited by the overall model. This task is complicated by the fact that the control inputs are not consistent with each other and are often contradictory. Furthermore, due to the nature of the control inputs, numerical averaging cannot be used as a means of fusion. Instead, the fusion algorithm is explicitly designed to take into account not only the control inputs themselves, but also where they come from and the current state of the overall model. Specifically, each state machine is classified as either providing a time-series of inputs or state-less inputs. The idea behind this classification is that certain maneuvers require a single point of control to carry over the whole action because they place the vehicle in situations where another state machine cannot deal with. On the other hand, state machines that provide a continuous set of commands that have no dependencies on the past are state-less. An example of an HCSM providing a time-series of inputs is the Overtake HCSM. Once an overtaking maneuver begins, it must terminate before another HCSM can take over. An example of an HCSM providing stateless inputs is the Follow HCSM, which simply outputs a "desired speed" value during each frame, and whose output during the current frame has no explicit dependency on what it produced during the last frame.

The fusion algorithm obeys the following priority rules.

- The more conservative control inputs have higher priorities. Here, conservative indicates a lower desired speed, or a target point that minimizes the steering angle
- Among time-series HCSMs, the Obstacle Avoidance has higher priority than Overtake
- Once the output of a time-series HCSM has been selected, its output will keep having the higher priority until the maneuver has been completed

Note that the above algorithm explicitly depends on certain mutual exclusion criteria among HCSMs. For example, both the Oncoming Avoidance and Overtake HCSMs are time-series, but can never provide outputs at the same time since Overtake presumes no oncoming traffic. After applying the above algorithm, the Autonomous provides the control inputs to CVED and the associated controllers.

Before explaining each behavior in more detail, it is useful to provide an example that describes the steps involved during execution of a couple of frames of the behaviors. For simplicity, the example does not go into detail on which HCSM performs which action, but simply provides a typical sequence of events that takes place starting with the instancing of an HCSM. For simplicity, we will refer to the HCSM as "executing" as a whole without specifying the internal details. The system starts by executing an iteration of all HCSMs. The appropriate manager HCSM executes and instances our example HCSM. Upon creation, the example HCSM calls CVED functions to create its representative physical object in the virtual environment. The HCSM determines the starting position of the object from its initial parameters. After the object is created, the HCSM uses CVED to obtain information about its surroundings. All the concurrent HCSMs execute, and their cumulative execution determines the appropriate speed based on the posted speed limit and factors such as road curvature, and also determines a direction of travel based on the lane it was placed in. The HCSM then uses CVED to obtain a guide point located ahead of its current position. The distance ahead used to obtain the guide point is such to ensure stability of the control algorithms used for lateral guidance. It then calculates an appropriate steering angle and a desired acceleration and provides these inputs to CVED. Once the CVED execution step takes place, the dynamic model executes one or more iterations using the control inputs associated with the object. As a result of the motion generated by the dynamics model, the object moves along the lane direction. In the next frame, the HCSM interrogates CVED to determine where the object is located, then uses this position to gather detailed data about the speed limit, direction of travel, etc. During this phase, the HCSM can query CVED to obtain any information required by the various behaviors. For example, the HCSM can query CVED to determine if another vehicle is located along the travel path, in which case a driver-follower algorithm is engaged. To implement intersection navigation, the HCSM asks if there are upcoming intersections and, if so, queries the intersection corridors and obtains any controlling signs or traffic lights.

Independent of the complexity of the behavioral model, the final outcome of the HCSM's execution is a fresh set of control inputs that are provided to CVED, which in turn uses them for the dynamics model. This cycle repeats for the duration of the simulation or until the HCSM is deleted by some higher-level manager HCSM.

The remainder of this section describes the function of each HCSM shown in Figure 2. Note that the description is rather qualitative, as space constraints prevent a more detailed and quantitative description of each behavior.

Autonomous

The Autonomous HCSM is the parent of all the HCSMs that encapsulate the various behaviors of a vehicle. Its primary responsibility is to provide information to its children before they execute and then resolve among the conflicting outputs after they finish execution. In its pre-activity function, Autonomous performs the following:

a) Obtain the current position of the vehicle in Cartesian coordinates from the dynamics servers.
b) Use CVED to convert the Cartesian coordinates into a parametric representation (R, L, D), where R represents the current road, L represents the current lane, and D represents the distance along the road.
c) Query CVED for any other relevant information regarding the current state of the vehicle and its surroundings. Using this information, update internal structures such as the current path.
d) Relay this information to its children.

During the post-activity function, Autonomous performs the fusion algorithm to pick the set of outputs to send to the vehicle dynamics model in CVED.

Free Drive

The Free Drive HCSM (FDH) is a state-less HCSM. Its primary responsibility is to provide lateral guidance so the vehicle follows the lane, and a default speed for the vehicle to follow based on roadway conditions and signage. To achieve this, the FDH utilizes the dynamically generated path provided by Autonomous in combination with the information available from CVED.

During each frame, the FDH performs the following actions for determining the new guidance point:

a) Using the current position, obtain the parametric coordinates of a new point located *delta* distance ahead of the current position along the vehicle path. The delta distance is the distance covered in 1.5 seconds of travel at the current speed. When on a road, the parametric coordinates of that point are

specified by (R, L, D+max(delta, MIN_DELTA)). When approaching an intersection, the FDH uses the appropriate corridor for obtaining the parametric point.

b) Convert the parametric point in Cartesian coordinates and provide this as the target point. The conversion functionality is provided by CVED.

In general, the FDH is straightforward, with the only complexity involved in calculating the new target point when the vehicle is either approaching or on an intersection. Most of that complexity is hidden in the implementation of CVED that performs these calculations through a high level interface. The FDH generates a default speed for the vehicle to follow by picking from the lowest of the following: (1) the current speed limit, (2) the speed based on the curvature of the road and (3) any external suggestions for a target speed.

Follow

The Follow HCSM (FH) is responsible for providing a desired speed to ensure following a lead vehicle without collisions. The FH is a state-less HCSM and outputs the following information: desired speed, desired distance to lead vehicle, actual distance to lead vehicle and the CVED id of the lead vehicle. The following controller uses the last 3 values to ensure proper following distance to the lead vehicle.

The FH first identifies the lead vehicle among all vehicles located near its object. Once a lead vehicle has been identified and its velocity and acceleration has been obtained, the FH utilizes a hybrid controller whose primary goal is to match the speed of the two vehicles, and whose secondary goal is to maintain a specific distance behind that vehicle. The desired speed for the two goals is calculated separately, and the final control input is a variable weighted average of the two goals. The weights vary based on the time to collision with the lead vehicle. When the differential speed is high, the speed-matching controller is used almost exclusively, but when the speeds of the lead and the current vehicles match, then the position controller is used almost exclusively.

Lane Change

The LaneChange HCSM (LCH) is responsible for implementing the transfer between same direction lanes. The LCH is a time-series HCSM, and produces three outputs. In addition to the standard inputs received by all HCSMs, the LCH also receives an additional boolean input, called *Initiate* indicating clearance to start a maneuver. The first two outputs are the standard desired velocity and lateral guidance, and the last output, called *Completed* is a boolean indicator of completion of the maneuver.

The LCH goes through a series of states during the implementation of a maneuver. When on the initial state, the LCH monitors a series of conditions for triggering a lane change maneuver. Initiating a maneuver and the subsequent transfer to the implementation state is triggered by the state or the Initiate variable becoming true. Once the maneuver is completed, the final state is reached and the Completed output is set to true. The LCH continuously monitors if the conditions are amenable for a lane change. Note that even if the conditions hold true, unless an Initiate command is set, the LCH will not begin the implementation phase. The Autonomous HCSM will prioritize the output of LCH according to the fusion algorithm described earlier, and when the priority of the LCH is the highest will provide clearance.

While in the implementation state, the LCH computes the lateral guidance points by augmenting a series of pre-calculated later motion commands that are stored in a large lookup table with the current location of the vehicle. The lookup table provides series of inputs for implementing lane changes based on a set of inputs that includes the current speed, the lateral distance to travel while implementing the lane change, and the capabilities of the current dynamics model. Use of static lookup tables provides a significant savings in time, and also automatically takes into account the different capabilities of each vehicle model.

The conditions checked by the LCH, described at a high level, include the following:

a) Current road multiple lanes on the same direction and current lane is not right most and inter-vehicle spacing rules are not violated
b) Lane of the same direction exists to the left and a lead vehicle exists on current lane and difference between desired speed and lead vehicle speed exceeds a threshold
c) The vehicle path through the upcoming intersection requires placement on a lane other than the current lane
d) Current road is a freeway, and an upcoming merge lane merges to current lane and vehicles exist on the merge lane and vehicle spacing rules on the target lane are not violated.

In addition to the above conditions, the LCH maintains a few rules that supercede the above conditions and prevent a lane change from taking place. These include a timer ensuring that a lane change based on conditions (a) and (b) do not take place until after a certain amount of time has elapsed after a lane changed caused by either of these two conditions. This rule eliminates some oscillatory behavior caused by following a lead vehicle whose speed differential is close to the threshold.

Intersection Navigation

The intersection navigation HCSM (INH) is the HCSM responsible for traversing intersections. This includes uncontrolled and controlled intersections with yield signs, stop signs and traffic lights. This also includes on-ramps and off-ramps to and from highways. The INH communicates with the intersection manager HCSM (IMH); a centralized high-level manager responsible for controlling traffic at each intersection. These two HCSMs interact to control when each vehicle crosses the intersection. Based on the rules of the road and the controlling mechanisms at each intersection, the IMH decides which vehicle has priority among the set of vehicles that are present at an intersection. The IMH communicates with each vehicle's INH to control when the vehicle is cleared to traverse the intersection. The INH outputs a desired speed to indicate whether it's stopped at a given point in the intersection waiting for its turn to cross the intersection.

For example, consider the situation at the intersection shown in Figure 3. Vehicles A and D are approaching stop signs so they are required to make a stop. Vehicles B and C are on uncontrolled paths through the intersection. Since vehicle B is making a left turn and vehicle C is traveling straight through the intersection, vehicle B must give the right of way to vehicle C. The IMH tells the INH of vehicles A, C and D that they must stop and tells each one of them exactly where to stop. The INH of each of these vehicles outputs the desired speed to reflect this command. Vehicles A and D must stop at the solid white line next to the stop sign whereas vehicle B must stop at the point indicated on the figure. This point is located right before the intersection of the 2 corridors that vehicles B and C are traveling on. Once vehicle C clears the point where its corridor intersects with other corridors, the IMH informs vehicles A and B that they can proceed thru the intersection since their paths don't intersect. Finally, as vehicle B clears the intersection, the IMH informs vehicle D that it can proceed through the intersection.

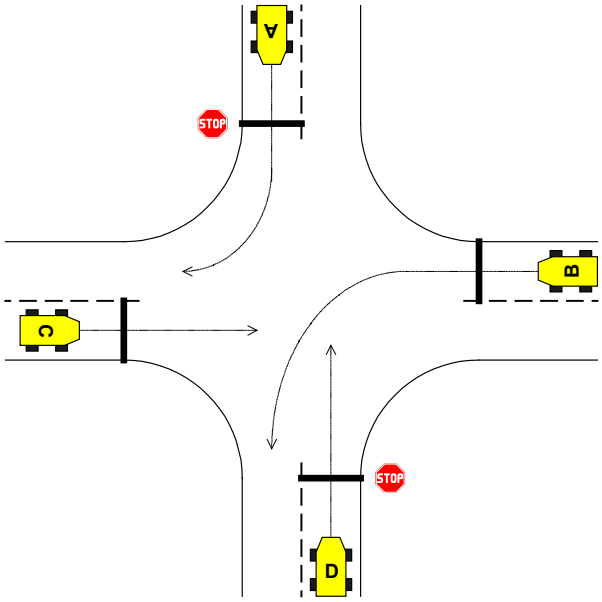

Figure 3 – Intersection example.

Oncoming Avoidance

The Oncoming Avoidance HCSM (OAH) is responsible for moving a vehicle to the right in response to an oncoming vehicle that has veered into the current lane. The OAH is a state-less HCSM that only produces a target point. The need for the OAH was raised by narrow two lane roads that provide a very small gap between vehicles traveling on opposite directions. Whereas the autonomous drivers can be expected to drive perfectly, real drivers have a tendency to veer about the lane and when the oncoming traffic does not react to this motion, collisions registered by the simulator software either terminate the scenario or produce other undesirable results. Interestingly enough, the behavior exhibited by the OAH is often not immediately noticed by drivers because if often involves a slight (less than 0.25 meter) veer away from the oncoming traffic.

The implementation of the OAH follows these steps:

a) Determine if the oncoming lane has any traffic, if none, then the veer is 0

b) Determine the time that the front bumper of the oncoming vehicle and the current vehicle will meet on the line perpendicular to the road centerline

c) Based on their current lateral placement with respect to the road, calculate the lateral distance between the vehicles

d) If the distance is below a threshold, compute a veering factor ensuring the threshold distance is maintained

e) Compute a target point that is translated laterally by an amount equal to the veering factor.

Obstacle Avoidance

The Obstacle Avoidance HCSM (OBAH) is responsible for changing the direction of the vehicle to ensure that any obstacles on the side or on the pavement are avoided. The OBAH is a state-less HCSM. The design of this HCSM is rather simple in that it only uses CVED to query the existence of any obstacles in the road that will be covered over the next few seconds. If any objects exist, the OBAH calculates if the current trajectory would cause the vehicle to collide with the object, and if so, determines a guide point that would place the vehicle either to the left or to the right of the obstacle, whichever yields the smallest deviation angle.

Note that the OBAH, been state-less, continuously outputs a target position when an obstacle is in sight. The actual look-ahead distance used is based on an approximate calculation that takes into account the current speed of the vehicle and the maximum steering angle that the vehicle dynamics can support without excessive slipping.

Overtaking

The Overtake HCSM (OTH) is responsible for implementing the overtaking maneuver. The OTH is a time series HCSM. A detailed description of this maneuver is given in [DSC] and only a short summary of it is provided here.

The OTH utilizes a protocol similar to the LCH, in that it monitors conditions amenable to the maneuver, but requires an explicit clearance from the Autonomous before initiating it. Similar to the LCH, upon initiating a maneuver, the conditions are continuously re-evaluated, and if an abort is necessary, the OTH maintains control until the maneuver is aborted and the vehicle is in a position that other HCSMs can take over. The set of conditions, all of which must be true before a maneuver can take place include:

a) Lane to the left is of opposite direction than current lane
b) A lead vehicle exists, and the speed differential with current vehicle exceeds a threshold
c) No oncoming traffic exists for the predicted duration of the maneuver, including a safety margin
d) Lead vehicle placement within its lane leaves enough room to pass on the left
e) The lead vehicle is not been passed
f) The current vehicle is not been passed

The actual maneuver is implemented as a series of states, each responsible for a distinct phase of the maneuver. Similar to other HCSMs, the OTH re-uses the FDH and FH for tracking the lane and following the lead vehicle during certain phases of the maneuver.

CONCLUSION AND FUTURE WORK

This paper presents a comprehensive autonomous driver model that primarily focuses on exhibiting a wide spectrum of behaviors that ensure the model can operate consistently with user's expectations within a high fidelity virtual roadway environment. The model uses the HCSM formalism, an extension of state machines that is specifically designed to support concurrency and hierarchy. These two features are particularly useful when utilized to model autonomous driver models, because of the inherent concurrency involved in a complex task such as driving. At the time of this writing, the second generation of this model is under implementation for the NADS. Under the current plans, the basic set of behaviors will be operational in early October, 2000. Once in place, we plan to methodically increase the fidelity of individual behaviors by incorporating higher fidelity models, and when possible, tuning the individual modules based on actual observational or experimental data based on real drivers.

REFERENCES

1. Brown, T. L., Lee, J. D., & McGehee, D. V. (2000). Driver Modeling in the Design of Rear-End Collision Warning Systems. Transportation Research Board 79th Annual Meeting (January). Washington D. C. Paper No. 00-1442.
2. Lee, J.D., McGehee, D. V., Dingus, T. A. & Wilson, T. (1997). Collision Avoidance Behavior of Unalerted Drivers' Uinsg a Front-To-Rear-End Collision Warning Display on the Iowa Driving Simulator. Transportation Research Board. National Research Council.
3. Weiler, J.M., Bloomfield, J.R., Woodworth, G.G., Grant, A.R., Layton, T.A., Brown, T.L., McKenzie, D.R., Baker, T.W., & Watson, G.S. (2000). Effects of Fexofenadine, Diphenhydramine, and Alcohol on Driving Performance. Annals of Internal Medicine, 132(5), 354-363.
4. Ranga, Haug, Solis, Papelis
5. Kuhl, J.G., Evans, D., Papelis, Y.E., Romano, R.A., & Watson, G.S. (1995). The Iowa Driving Simulator: An immersive environment for driving-related research and development. IEEE Computer, 28(7): 35-41.
6. Thoeni, U. (1999). Experiences With Practical Driver Training In An Advanced Driving Simulator. International Training Equipment Conference (ITEC), The Hague, April 2000.
7. Ahmad, O., & Papelis, Y. (2000). An Autonomous Driver Model for the Overtaking Maneuver for Use in Microscopic Traffic Simulation. Paper presented at the Driving Simulation Conference 2000. Paris, France.
8. Boer, E.R., & Hoedemaeker, M. (1998). Modeling Driver Behavior with Different Degrees of Automation: A Hierarchical Decision Framework of Interacting Mental Models. In Proceedings of the XVIIth European Annual Conference on Human Decision Making and Manual Control.
9. Zarrillo, M.L., Radwan, A.E., & Al-Deek, H.M. (1997). Modeling Traffic Operations at Electronic Toll Collection and Traffic Management Systems. The International Journal on Computers and Industrial Engineering (special edition, ed. Y.D. Kim), 33(3-4): 857-860.
10. Schmidt, K., & Lind, G. Dynamic Traffic Simulation Testing. Traffic Technology International, Dec 1999/Jan2000, 54-57.
11. Cremer, J., Kearney, J., & Papelis, Y. (1995). HCSM: A Framework for Behavior and Scenario Control in Virtual Environments. ACM Transactions of Modeling and Computer Simulation, 5(3): 252-267.

Author contact information:

Yiannis Papelis

National Advanced Driving Simulator and Simulation Center

The University of Iowa

2401 Oakdale Blvd.

Iowa City, Iowa 52242-5003

(319) 335-4597

(319) 335-4658 FAX

yiannis@nads-sc.uiowa.edu

Omar Ahmad

National Advanced Driving Simulator and Simulation Center

The University of Iowa

2401 Oakdale Blvd.

Iowa City, Iowa 52242-5003

(319) 335-4788

(319) 335-4658 FAX

oahmad@nads-sc.uiowa.edu