Tactical-level Simulation for Intelligent Transportation Systems

Rahul Sukthankar, John Hancock, Chuck Thorpe
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3891

Fax: 1-412-268-5571
Email: {rahuls|jhancock|cet}@ri.cmu.edu

Abstract

SHIVA (Simulated Highways for Intelligent Vehicle Algorithms) is a kinematic simulation of vehicles moving and interacting on a user-defined stretch of roadway. The vehicles can be equipped with simulated human drivers as well as sensors and algorithms for automated control. These algorithms influence the vehicles' motion through simulated commands to the accelerator, brake and steering wheel. SHIVA's user interface provides facilities for visualizing and influencing the interactions between vehicles.

SHIVA not only models the elements of the domain most useful to tactical driving research but also provides tools to rapidly prototype and test algorithms in challenging traffic situations. Additionally, the scenario control tools allow repeatable testing of different reasoning systems on a consistent set of traffic situations. These features are vital in the development and evaluation of intelligent vehicle technology for ITS applications.

Keywords

Simulation, Intelligent Transportation System (ITS), Automated Highway System, Intelligent Vehicles

1 Introduction

The development of an Intelligent Transportation System (ITS) requires extensive testing, verification and evaluation of new traffic management concepts. The high economic costs and potential safety concerns of prototyping these systems in real traffic have led to the ubiquitous use of simulation in this research community [1, 2, 3, 4]. Unfortunately, most of these tools (developed in the late 1960s and early 1970s) are inflexible and cannot effectively meet the demands of simulating traffic for ITS applications [5]. A new generation of traffic simulators which support a subset of ITS technologies, such as mainline traffic control and real-time route guidance, is emerging [6, 7, 8]. However very few of these simulators address the issue of intelligent vehicles.

Research in intelligent vehicles is motivated by three desires: improved safety, increased highway throughput, and user convenience. Competing concepts for automated highway system designs address each of these desires to a different degree. Concepts which involve minimal modifications to existing highway infrastructure are desirable for many economic and social reasons. Such concepts

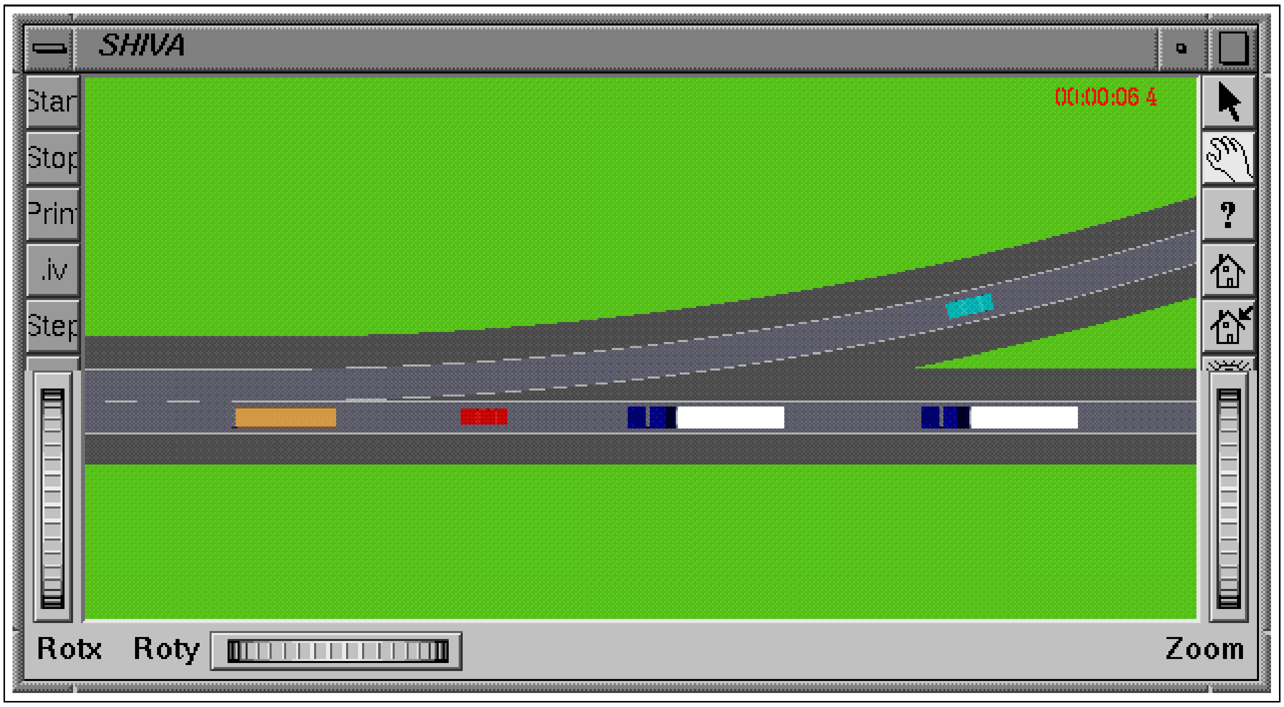

Figure 1: SHIVA's primary view is an interactive camera which shows the simulation from overhead at an aggregate level. Vehicles can be selected from this view for detailed inspection.

typically involve the injection of automated vehicles into manual traffic, and are known as *mixed traffic* concepts.

The mixed traffic environment is especially challenging since intelligent vehicles must make driving decisions based on incomplete information about the environment. Thus, it is critical that automated vehicles in this concept exhibit effective tactical-level [9] reasoning before they are introduced into human traffic. SHIVA (Simulated Highways for Intelligent Vehicle Algorithms) [10, 11] is a simulation and design tool developed at Carnegie Mellon which models the elements of the driving domain most relevant to tactical-level research. Figure 1 shows an overhead view of a typical merge scenario.

2 Tactical-level Simulation

The driving task can be characterized as consisting of three levels: strategic, tactical and operational [12]. At the strategic level, a route is planned and goals are determined; at the tactical level, appropriate maneuvers are selected to achieve short-term objectives; and at the operational level, these maneuvers are translated into control operations. Most traffic simulators use very simple vehicle models allow efficient computation of flow statistics even for large networks. Conversely, vehicle simulators represent sub-systems at a detailed level (engine models, transmission characteristics, braking behavior) to examine car performance but cannot support multiple vehicles. In Michon's classification scheme [12], the former can be considered suitable for strategic-level simulation, and the latter for operational-level simulation. Very few simulators address the needs of tactical-level research.

Three simulators, Pharos [13], SmartPath [14] and SmartAHS [15] possess most of the characteristics necessary for tactical-level simulation: microscopic vehicle modeling, support for different road

Table 1: SHIVA, like all simulators, is a compromise between reality and efficiency. This table summarizes the important design decisions.

Design choice	SHIVA implementation
Modeling level	Microscopic
Kinematic accuracy	Yes, with velocity over dt
Dynamic accuracy	Available in extended versions
Road geometry	Full 2-D geometry.
Highway support	Yes
City streets support	No
Vehicle types	Functionally similar: car, truck, bus
Vehicle models	Multiple, user-defined
Engine models	No
Suspension models	No
Tire/surface models	Available in extended versions
Sensor models	Multiple, detailed, user-defined
Driver models	Multiple, detailed, user-defined
Traffic mix	Heterogeneous, allows mixed traffic
Off-line simulation	Yes, on multiple platforms
Integrated simulation & visualization	Yes
Simulation persistence	Limited: dump & restore to file
Interactive debugging	Yes
Interactive scenario generation	Yes
Can users drive vehicles	Only at the tactical level

geometries and realistic lane changing models. Unfortunately, since none of these tools were designed specifically for tactical-level simulation, each has significant deficiencies in the area. Pharos makes some unrealistic sensor assumptions such as transparent vehicles; SmartPath, though well suited for modeling platoon concepts (where groups of very closely spaced automated vehicles drive along dedicated lanes) [16], cannot support reasoning systems which violate its state machine architecture; SmartAHS has significant potential for further development, but is currently only a simulation framework.

In designing SHIVA, we were faced with a number of difficult design decisions. Since no one simulator can address all of the ITS simulation needs, we elected to create a specialized tool which would concentrate on tactical-level issues, and also provide design and debugging features for intelligent vehicle algorithm designers. Table 1 summarizes many of the important issues that were considered during SHIVA's development.

The choice of motion model proved to be the most difficult design choice. This is because inaccuracies in modeling vehicles at the operational level can create changes in vehicle behavior at the tactical level. The tradeoffs in each choice are shown in Table 2. Since dynamics play only a minor role in tactical-level driving, our current version of SHIVA is restricted to realistic kinematic models. SHIVA is being extended to model dynamics in situations such as emergency obstacle avoidance where skidding and collisions are more common (see Section 11).

Table 2: A summary of the tradeoffs involved with different choices of motion model.

Motion Model	Benefits	Drawbacks
Slot Car	Executes quickly, simple to code	Unrealistic lane changes
Kinematic 2D	Fast enough for interactive simulation and display.	Unrealistic cornering
Dynamic 2D	Allows tire models, engine models, skidding	Unrealistic collisions, too slow for many vehicles
Dynamic 3D	Realistic collisions, banking, suspension	Computationally expensive, very complicated

3 Road Representation

SHIVA represents a highway as a connected set of road segments, where each segment is a stretch of road with an arbitrary shape, a fixed number of lanes and common properties such as speed limits and lane markings (See Figure 2). Segments connect to other segments on a lane-by-lane basis, allowing users to model most given highway topologies. Vehicles typically drive along road segments, smoothly crossing the boundaries between two segments as they take exits or merge. However, vehicles are not *required* to stay on the road — given bad control algorithms (or to avoid obstacles), vehicles may swerve off the road. This is also shown in Figure 2.

Expressing tactical maneuvers in a local *road-coordinate* frame allows vehicle controllers to be invariant to the road geometry. SHIVA represents these coordinate frames explicitly and provides simulator objects with a transparent interface to the road representation. Points represented in road coordinates can be seamlessly expressed in world coordinates as necessary. SHIVA's notion of real-valued lane displacement (rather than integral "lane numbers") is a feature lacking in many existing traffic simulators [17] and allows accurate modeling of lane change maneuvers.

Some simulators [18, 14, 15], while modeling continuous lateral motion, still ignore the impact of lane tracking on vehicle orientation (i.e. they assume that the vehicle is always parallel to the local tangent of the road). Unfortunately these effects are relevant during lane changing. For example, the effects of even a 1 degree yaw at a range of 60 meters corresponds to an lateral error of 1 meter in sensing — possibly causing errors in object-to-lane mapping (i.e. an object in an adjacent lane may appear to be in the current lane). SHIVA's perception modules (See Section 5) correctly account for the vehicle's current heading when mapping obstacles to lanes.

4 Vehicles

SHIVA models vehicles as kinematically accurate, two-axle, front-wheel-steered mechanisms. Three vehicle classes are provided, each with its own physical characteristics (See Figure 3). Functionally, all intelligent vehicles are decomposed into three subsystems: perception, cognition and actuation (See Figure 4). This architecture, given appropriate choices for sensors and algorithms, can describe both human drivers and automated vehicles. Also, since SHIVA is object-oriented, researchers may derive new vehicle types as desired, inheriting the essential attributes and methods from the basic vehicle

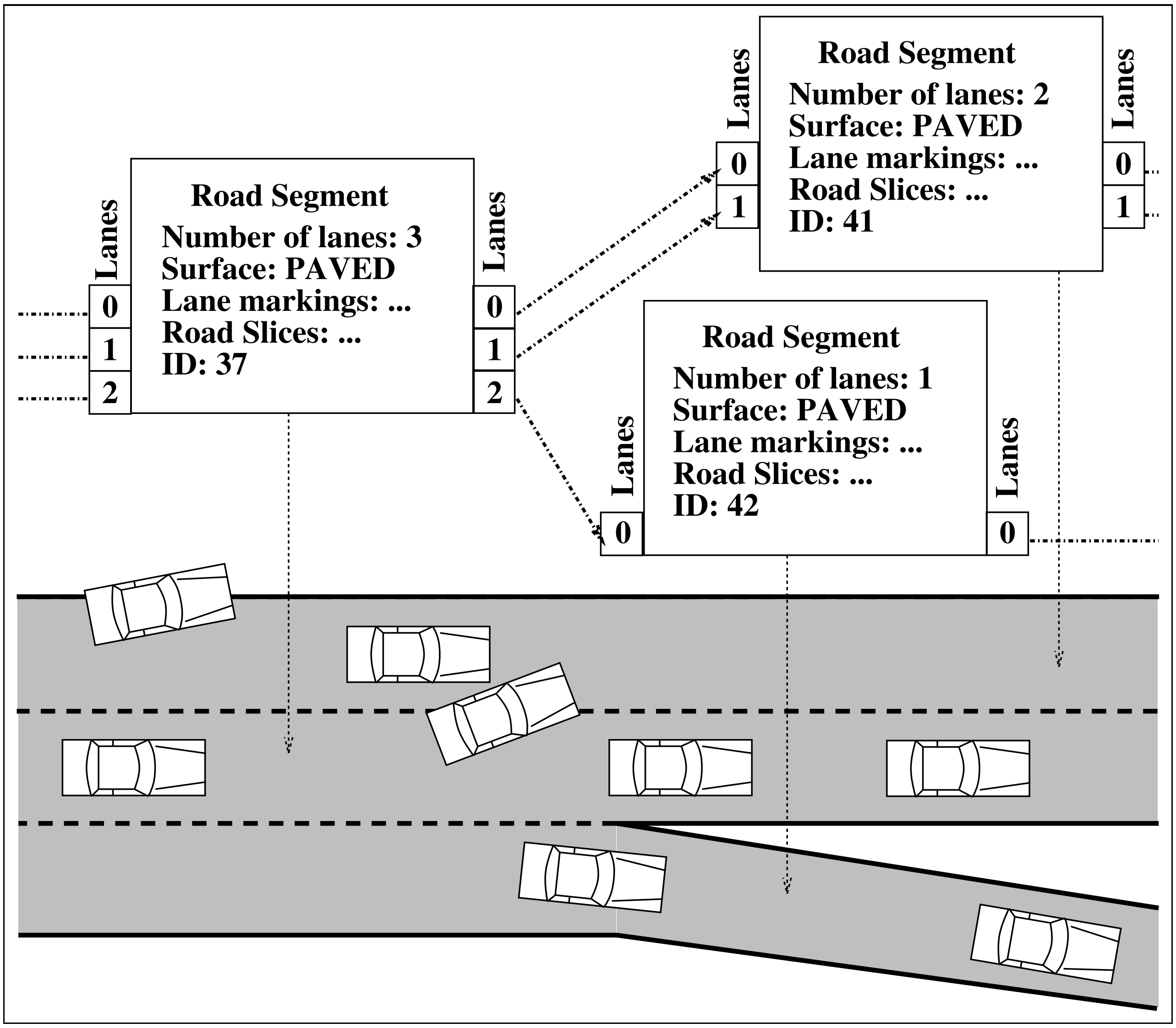

Figure 2: SHIVA's road representation consists of a connected set of Road Segments — stretches of road with an arbitrary shape and common properties. Vehicles are not required to stay on the road.

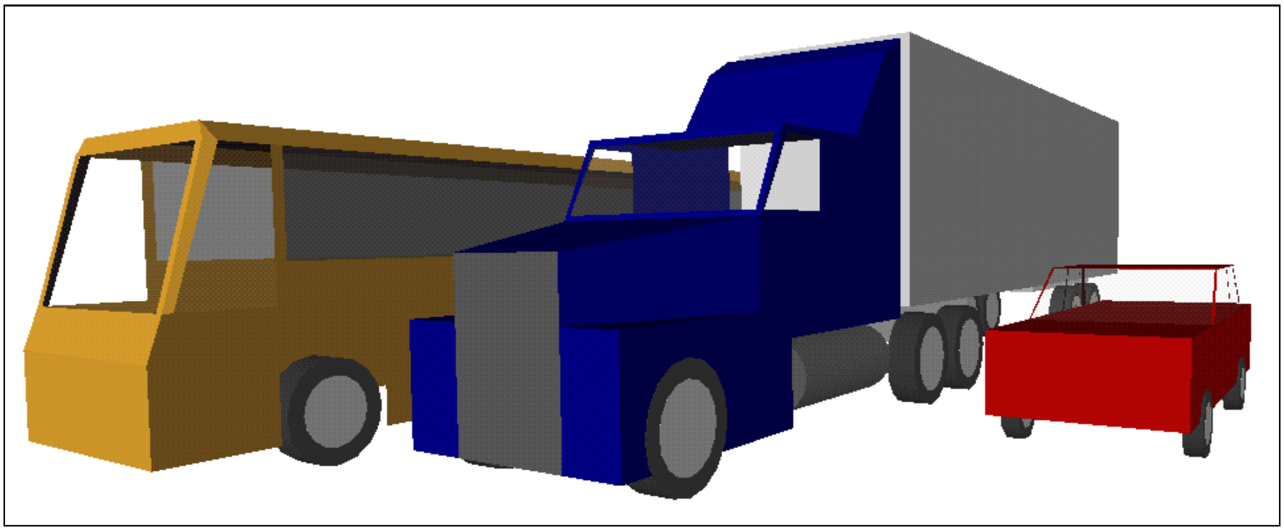

Figure 3: SHIVA models a variety of vehicle classes including cars, trucks and buses. Within each class, vehicles may also be equipped with their own configurations of sensors, reasoning algorithms and actuators.

models.

5 Perception

Largely ignored in most simulators, perception remains one of the most difficult problems in mobile robotics. Control algorithms which make unrealistic perceptual assumptions are destined to remain unimplementable on real systems. On the other hand, modeling realistic sensors perfectly is infeasible since the simulated world cannot match the complexity of real life. Each simulator must therefore select the appropriate level of detail suitable for its task. In tactical driving, issues such as occlusion, ambiguity and obstacle-to-lane mapping are important. The perception subsystem consists of a suite of functional sensors (e.g. GPS, range-sensors, lane-trackers), whose outputs are similar to real perception modules implemented on the Carnegie Mellon Navlab autonomous vehicles (See Figure 5). SHIVA vehicles use these sensors to obtain information about the road geometry and surrounding traffic. Vehicles may control the sensors directly, activating and panning the sensors as needed, encouraging active perception. Perception objects are grouped into an open-ended sensor hierarchy (See Figure 6) which allows designers to create appropriate models and swap them into existing vehicle configurations with minimal modification. Some of these sensor models are detailed below.

5.1 Car Detection and Tracking

Since car tracking is an important perceptual task for tactical driving, SHIVA provides two types of car tracking sensors: realistic range sensors and functional car tracking modules. Both types of car trackers have a limited field of view, and range. Sensors cover different areas of interest around the vehicle, and can be activated (or panned) by the cognition modules as needed during tactical maneuvers to provide relevant information about surrounding traffic. Since perception is expensive, this selective perception enables cognition modules to minimize the computational load [18].

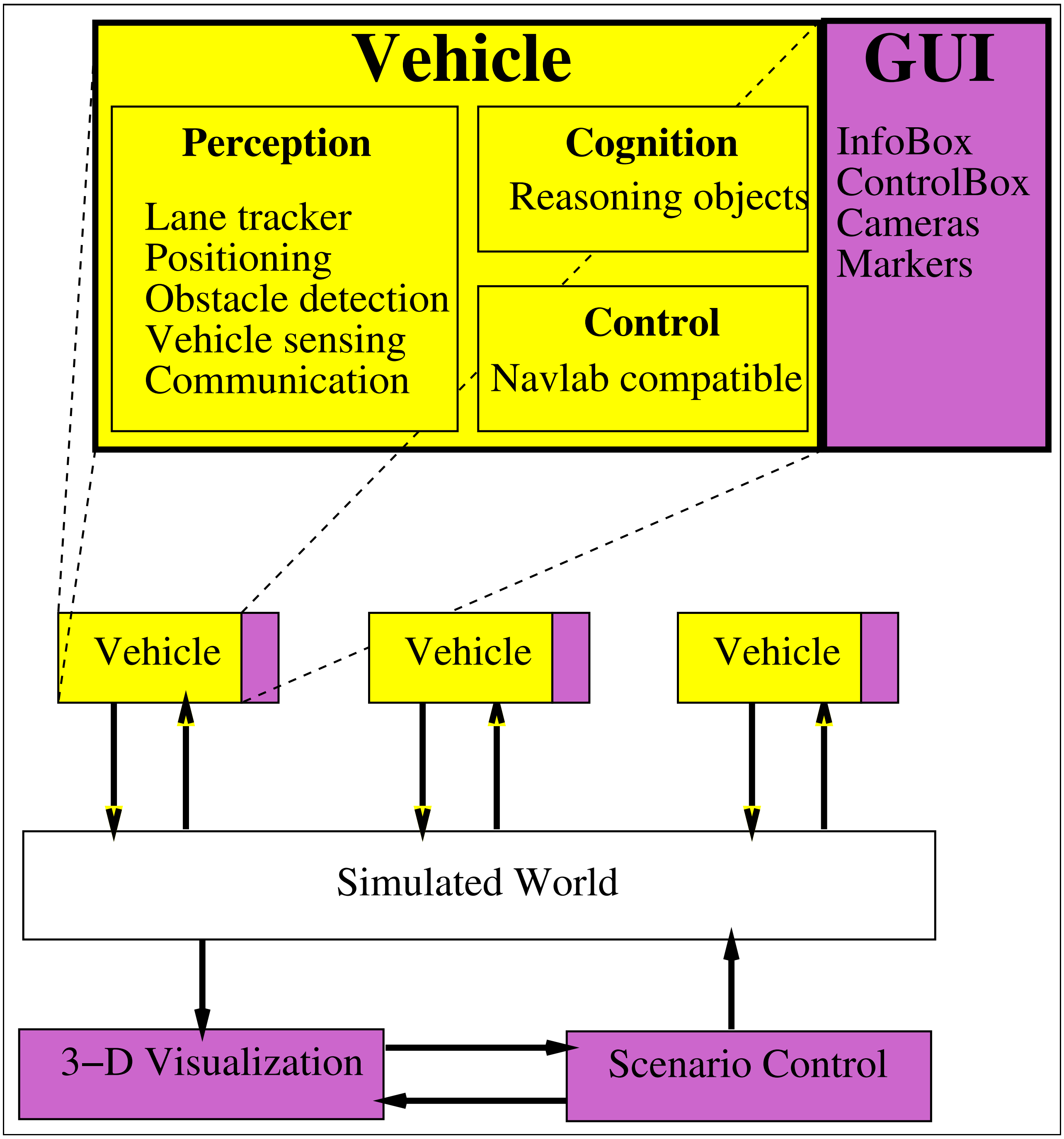

Figure 4: Each vehicle is composed of three subsystems (perception, cognition and actuation) which interact with the simulated world. The design tools automatically adapt to each vehicle's internal configuration and provide access to the various components at an appropriate level of detail.

Figure 5: The Carnegie Mellon Navlab 5 is a testbed for developing autonomous navigation systems. Wherever possible, SHIVA is compatible with existing perception and control modules available on the Navlab.

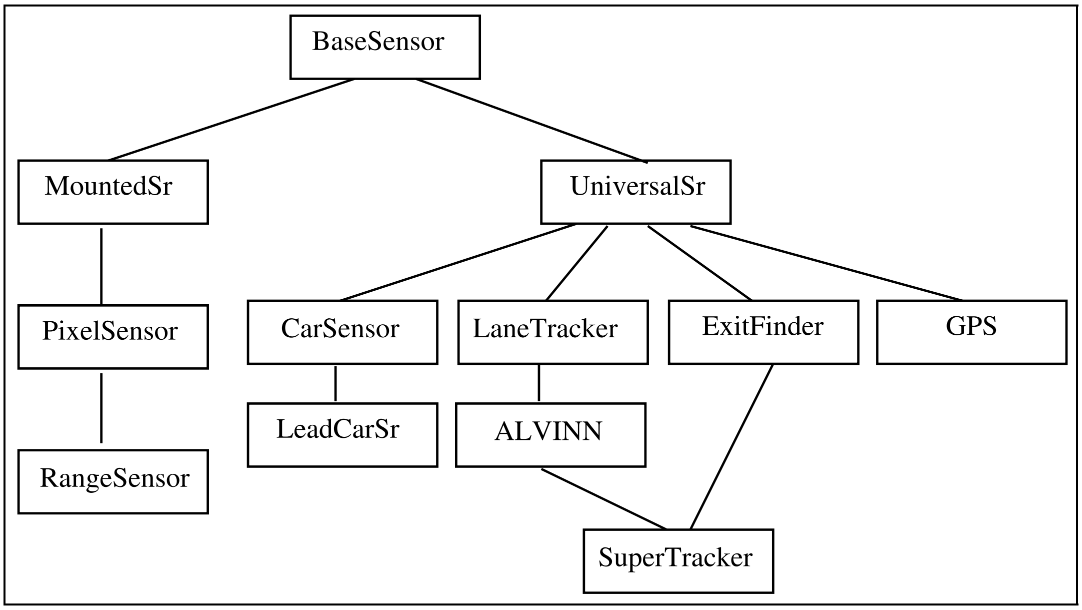

Figure 6: The perception subsystem consists of a suite of sensors whose outputs are similar to modules available on real robots. The hierarchy allows designers to create models at the appropriate level of detail and swap them into existing vehicle configurations with minimal modification.

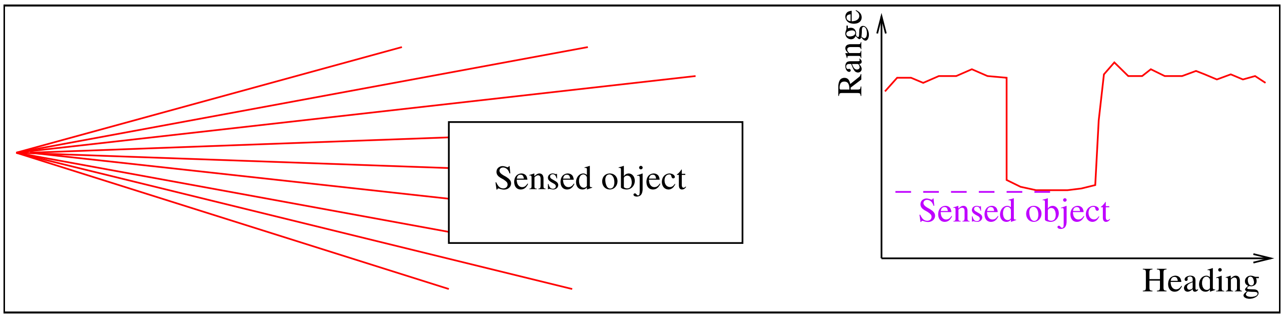

Figure 7: The range sensor model simulates a scanning laser sensor with a limited field of view and range. The detection array returns range values for each ray, corrupted by noise. Segmentation, tracking and velocity calculations need to be performed by the perception module.

5.1.1 Range Sensors

SHIVA's range sensor is a scanning laser sensor with a user-defined range ($\rho_{\max}$), field of view ($\theta_{\max}$) and resolution (given by number of rays, n). Vehicles may have different sensors with individual characteristics, mounted on different locations. See Figure 7 for an example of a particular range sensor's output.

The simulated sensor casts n rays, spaced equally in angle over the field of view, and determines each ray's intersection with objects in the environment. Each ray (r_i) returns the distance (ρ_i) to the closest intersection point (or $\rho_{\max}$ in the case of no intersections). Since each r_i corresponds to a pixel in the sensor's retina, the sensor returns an array of pixels (R):

$$R = [\rho_0, \rho_1, \ldots \rho_i, \ldots \rho_{n-1}]$$

To add realism, each ray's scan angle (θ_i) and returned value (ρ_i) are corrupted by Gaussian noise. Each (θ_i, ρ_i) is subsequently mapped into sensor coordinate frame using: $x = \rho \cos \theta_i$ and $y = \rho \sin \theta_i$. Thus the injected noise translates into errors in the sensed object's position. Note that this method is not equivalent to adding independent Gaussian noise to both x and y — in SHIVA, as in a real range sensor, the errors in x and y are correlated!

Since information about objects in the environment is provided only through discretized range readings, cognition algorithms which maintain explicit models of objects in the surroundings must perform segmentation and tracking on this noisy data. Similarly, since velocity information is not provided directly, it must be inferred from differential range measurements. Furthermore, because all range readings are in sensor coordinates, cognition algorithms must perform their own object-to-lane mappings (i.e. convert observed vehicles into road coordinates).

Because all of this processing is computationally expensive, most tactical-level experiments only use a few vehicles with such sensors, supplemented by larger numbers of vehicles with functional perception modules.

5.1.2 Functional Car Trackers

SHIVA's functional perception modules simplify the processing required by cognition modules. Rather than providing raw range data, functional models also perform the following processing tasks:

- Car tracking: detected vehicles are automatically segmented and correctly tracked in subsequent frames. Optionally, each new vehicle is tagged with a unique ID number to simplify reasoning.

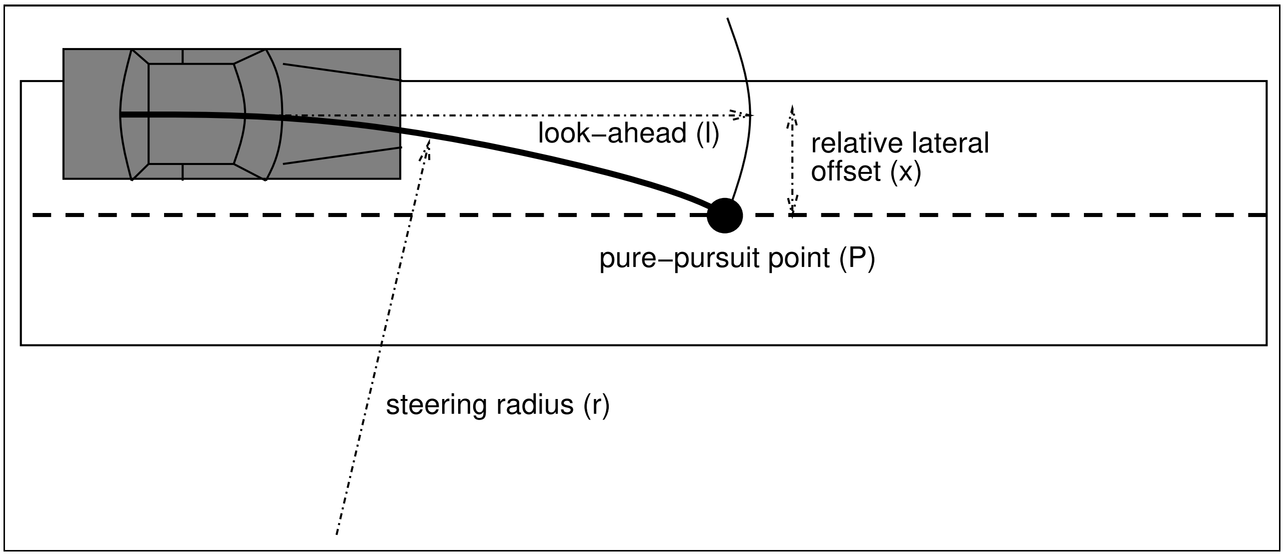

Figure 8: The lane tracker module provides information about the local road geometry in the form of a *pure-pursuit* point (P). This point on the road determines the steering radius that should be used to steer the vehicle onto the road at the specified *look-ahead* distance (l).

- Object-to-lane mapping: all position measurements are converted into road coordinates, enabling cognition modules to largely ignore the effects of the local road geometry.
- Vehicle type detection: the class of vehicle (e.g. car, bus, truck) as well as vehicle size is directly reported, simplifying gap reasoning calculations.
- Direct velocity measurement: the true velocity of the observed object is reported, enabling accurate calculation of time-to-impact or acceleration constraints.

Note that the functional car tracker does not provide higher-order derivatives such as acceleration or jerk, since these (very noisy) measurements cannot be reliably obtained in real life.

A danger with using functional sensor models is that the the simulator may provide unrealistically complete data of the environment. This tendency is common in earlier work in tactical simulation. For example: perfect measurement of sensed vehicles' acceleration [19]; transparent vehicles [18]; and straight road assumptions [20]. In contrast, SHIVA's functional sensor models only return information which could be provided by existing perception technology.

5.2 Lane Trackers

Since realistic lane tracking is an operational-level task (and beyond the scope of this simulation), SHIVA's lane tracking module is a functional model of a complete vision-based lane tracking system. The interface is based on the ALVINN [21] road follower and provides information about the local road geometry through a *pure-pursuit*[22, 23] point (See Figure 8). A pure-pursuit point is defined to be the intersection of the desired lateral offset curve on the road, and a circle of radius (l), centered at the vehicle's rear axle midpoint. Intuitively, it determines the steering arc which would bring the vehicle to the desired lateral offset after traveling a distance of approximately l. The position of the pure-pursuit point maps directly onto a recommended steering curvature: $k = -2x/l^2$ where k is the curvature (reciprocal of steering radius), x is the relative lateral offset to the pure-pursuit point in vehicle coordinates, and l is the look-ahead distance. The look-ahead distance is an empirically determined parameter which corresponds to the gain of the steering controller. When l is too short, vehicle control may become unstable; when l is too long, the controller response is sluggish. Experiments on the Navlab vehicles [23, 21] have shown that good results are obtained when $l = 15$ meters in low speed situations and $l = 25$ meters at highway speeds.

Although lane tracking is a challenging robotics problem [24, 25, 21], most highway simulations [3, 19] still equate lane tracking and lane occupancy. Such simulations model lane changes as instantaneous actions, allowing their cognition modules to completely ignore the difficult decisions needed during lane changing — and fail to capture behaviors such as lane straddling (which affect tactical-level actions). Cognition modules which events which occur during lane changing are also unable to realistically model cases where lane changes need to be aborted.

SHIVA also supports lane trackers that actively steer the vehicle. By varying the desired lateral offset (represented by the pure-pursuit point) smoothly from the center of one lane to the the center of the desired adjacent lane, these road followers are able to implement lane changes [26]. It is important to note that the actual lateral offset of the vehicle always lags the current position of its pure-pursuit point during the lane change.

5.3 Positioning

SHIVA simulates two types of positioning sensors: global positioning systems (GPS) and dead-reckoning. While both return the vehicle's current position in global coordinates, they differ in their noise characteristics: GPS sensors corrupt readings by noise which is relatively invariant over time,[1] while dead-reckoning sensors return measurements that become increasingly inaccurate with distance traveled (since differential errors accumulate with vehicle motion). Positioning information, in conjunction with on-board digital maps, can allow cognition modules to initiate maneuvers (such as changing lanes into the exit lane) well in advance of the desired exit.

6 Cognition

Cognition modules must select appropriate tactical-level actions based upon information provided by the perception subsystems. The operational-level aspects of the driving task are handled by either the actuation subsystem (See Section 7) or by modules such as lane trackers which accept tactical commands (such as the lane tracker described in Section 5.2).

Different cognition strategies require different perceptual inputs. SHIVA enables researchers to create customized configurations of sensors for each cognition module. For example, purely reactive cognition modules can directly accept raw data from the realistic sensors. By contrast, rule-based systems typically reason about higher-level concepts such as gap sizes, velocities and acceleration constraints. The latter configuration requires significant processing of the sensor outputs — such as segmentation, differencing, and obstacle-to-lane mapping.

SHIVA places no constraints on the reasoning of individual cognition modules except that the outputs be compatible with the controller configuration of the vehicle. This also ensures that algorithms implemented in simulation can be later tested on the Navlab with minimal changes. While a detailed

[1] SHIVA's GPS noise model is not completely accurate. For real GPS, errors due to military corruption (selective availability) are not time invariant; however the total errors are bounded.

study of tactical reasoning systems is beyond the scope of this article, an overview of some of the techniques developed in SHIVA is presented in Section 11.

7 Actuation

While kinematics are realistically modeled in SHIVA, dynamics are largely ignored. In the absence of engine, transmission and suspension models, simulating actuation is relatively straightforward. The control interface is compatible with the Navlab controller [27]: cognition modules provide a control tuple, $C = (k, v)$ where k is the desired curvature and v is the desired velocity. Cognition modules provide C to the controller at regular time intervals; in the absence of new information, the controller will servo on the last commanded tuple.

As in the Navlab, steering and velocity control are decoupled. This simple model does not allow vehicles to follow precise trajectories, but is sufficient for specifying coarse behavior. The controllers generate steering wheel position and throttle/brake commands to bring current vehicle curvature and velocity to the desired positions as quickly as possible. The controller also enforces limits on acceleration and rate of change of curvature. Experiments on the Navlab [21, 27] indicate that a bandwidth of 10 Hz is required for highway driving using this controller.

SHIVA's controller also supports low-level steering control through active road-following. In this paradigm, cognition modules specify desired lateral position through manipulation of the pure-pursuit point (See Section 5.2). The lane tracker then computes the appropriate steering curvature needed to bring the vehicle to the correct lateral offset. This allows cognition modules to remain ignorant of vehicle kinematics. Current research in SHIVA is exploring more sophisticated control schemes. Some of this work is discussed in Section 11.

8 Simulation and Design Tools

Most existing simulators are designed to model only a few vehicle configurations and reasoning agents. By contrast, SHIVA's architecture is open-ended — enabling researchers to integrate new sensors, controllers and intelligences into existing vehicle specifications. Researchers can study interactions between different intelligent algorithms by creating scenarios with heterogeneous vehicle configurations. For example, in mixed-mode traffic, manual and automated vehicles use very different sensors and driving strategies. SHIVA allows researchers to examine the repercussions of changing the proportions of different types of vehicles (e.g. manual versus automated[2]) on the highway.

Since reasoning in tactical situations is complex, algorithms are best developed in an iterative manner. To support rapid refinement of reasoning systems, SHIVA provides three main types of tools.

8.1 Visualization and Validation

The first step to validating algorithms is observing them in action. Visualization tools allow the designer to qualitatively evaluate algorithms from different points of view. To verify whether vehicles are behaving reasonably with a given design, the researcher needs to be able to see how the vehicles behave both as individuals and as aggregates.

[2]The validity of such studies depends critically on the driver models that are used for manual traffic. The difficulty of implementing realistic driver models seems no easier than designing intelligent automated vehicles.

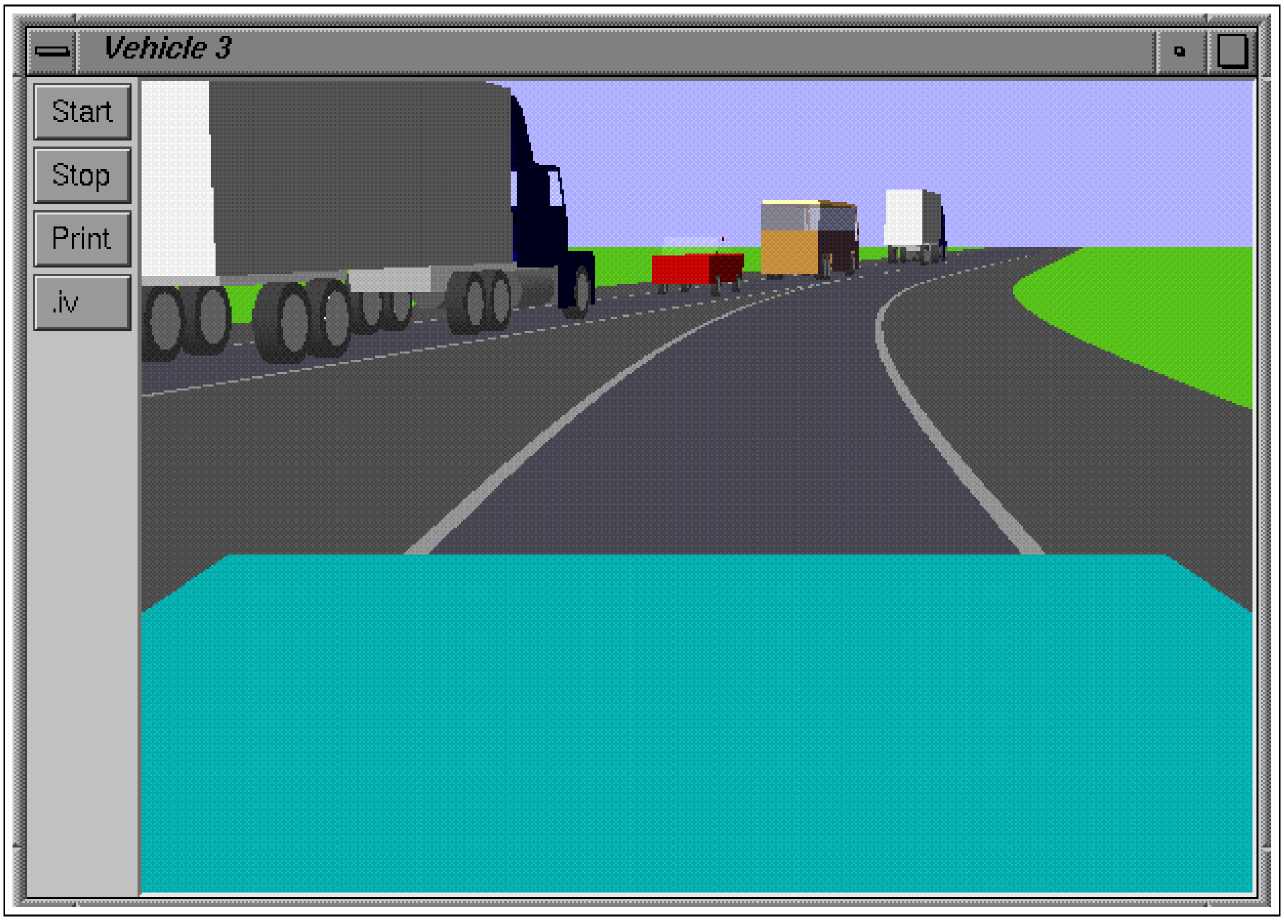

Figure 9: SHIVA allows users to select arbitrary secondary views for detailed examination of the scenario. This figure shows a driver's eye perspective of the scene.

SHIVA provides a flexible suite of visualization tools using Open Inventor (a 3-D graphics library developed by Silicon Graphics)[3]. SHIVA's primary view (See Figure 1) is an interactive camera which shows the simulation from overhead at an aggregate level. Multiple views that track vehicles may be created by selecting the desired vehicle(s) in the primary view. These secondary views can display driver's eye (or other arbitrary) perspectives (See Figure 9). Since humans make tactical decisions from behind the wheel, these views are helpful in judging the quality of decisions made by the AI algorithms. A vehicle selected for monitoring can be ordered to change color or begin dropping markers — "virtual bread-crumbs" — allowing researchers to observe not only the vehicle's current position, but also its previous trajectory.

SHIVA's visualization tools also support a host of features such as printing views to PostScript, 3-D dumps of situations as `VRML` (Virtual Reality Modeling Language) files, and animations of tactical scenarios as `MPEG` or animated `GIF` movies. This allows researchers to demonstrate their results to a wide audience (particularly when such files are made available over the World Wide Web).

[3]Sun and Mac versions of SHIVA currently only provide a limited set of visualization tools.

8.2 Measurement and Analysis

To improve on existing algorithms, users need the ability to analyze what the vehicles are doing correctly, and to identify what they are doing incorrectly. Although visualization tools allow the designer to see if something is wrong with the algorithms, this information is generally insufficient to diagnose the problem. To perform this quantitative analysis on-the-fly, researchers require access to reasoning object internals during the simulation. Current debugging tools are inadequate for this task since they only display a few variables at a time.

SHIVA displays qualitative and quantitative information through `InfoBoxes` which update continually during the simulation. Researchers may request this information on a per-agent basis and focus only on the relevant details. Since simulation and animation is fully integrated, problems spotted using the visualization tools may be immediately investigated. Most importantly, these `InfoBoxes` rapidly customize to the vehicle configuration so that only (and all of) the relevant information is displayed. Without customization, `InfoBoxes` automatically adjust themselves to display whatever information is available for that vehicle configuration through object-oriented inheritance.

In addition to local information about each vehicle, SHIVA also collects statistics about global performance. These include numbers of collisions, throughput, average velocities and exit success rates. Researchers can customize the appropriate aggregate measures and monitor them during the simulation.

8.3 Interactive Exploration and Modification

An important requirement for a simulation and design system is the ability to interact with and modify objects on-the-fly. This provides several benefits:

1. Supports rapid iterative development.
2. Enables better exploration of parameter space.
3. Allows fine-tuned scenario generation.

Iterative development without recompiling is vital for incremental algorithm design. Interactive parameter modification can be used to expose algorithms to sudden changes in environment, encouraging robustness. SHIVA's hierarchical interface tools can be customized for particular vehicle configurations and user needs. Two main types of tools are provided: 1) parameter adjustment tools which allow users to change environmental conditions (such as vehicle injection rates); 2) Vehicle `ControlBoxes` which allow users to generate scenarios and modify cognition algorithm settings. The latter are discussed in greater detail in Section 9.2.

9 Scenario Generation

Many interesting traffic situations occur very rarely, both in real life and in simulation. However these are also the scenarios where tactical-level reasoning systems are most challenged. Rather than forcing researchers to wait patiently for these events to arise, SHIVA provides users with several interactive scenario generation features. There are two aspects to scenario control: creation and manipulation. We examine each in turn.

9.1 Scenario Creation

The primary tool for scenario creation is the *Factory*. Each factory produces vehicle configurations from a specified set (with desired probabilities, at appropriate times) and injects them at the selected places on the highway network. Since factories are specified using text files, they can easily be customized at run-time.

Once vehicles appear on the roadway, users can interactively move them into their desired positions. In addition to an intelligent *pick-and-place* interface, SHIVA supports a *tactical driving interface* which maps user commands into tactical-level actions (interpreted in the current local road coordinate frame). SHIVA automatically adjusts the vehicle's orientation to be parallel to the road in response to changes in position, simplifying the user's control task.

Users may select initial conditions for the scenario, such as starting velocities or vehicle positions and then save the scenario to disk for future use. This is especially valuable when different cognition modules need to be (repeatably) tested on the same set of scenarios.

9.2 Scenario Manipulation

The integrated nature of SHIVA's simulation and animation environment enables users to interactively adjust the behavior of vehicles during a scenario. The primary method of manipulating SHIVA vehicles is through `ControlBoxes`. At a basic level, these tools allow users to modify a given vehicle's physical parameters such as position or velocity. More importantly, they enable run-time adjustments of key parameters in the various cognition modules. SHIVA also supports a number of ways to (indirectly) influence the behaviors of vehicles at run time — for example, the user may command a `breakdown` and force the selected vehicle to slow down and stop. Other functions allow users to simulate various malfunctions in the sensors or actuators.

10 Saving and Restoring the Simulation State

As seen in Section 9, scenario generation requires the ability to recreate simulation states. One way to accomplish this is to build the simulator on top of a persistent database (e.g. SmartAHS [15] is built over Versant). The biggest advantage of this is that state is implicitly saved (allowing rewinding of the simulation on demand). Unfortunately, the price for this convenience is that each time-step of the simulation incurs a substantial overhead. In particular, it is unlikely that such a system could provide integrated and simulation/animation given current computer hardware.

SHIVA compromises on convenience by offering only a subset of this feature — researchers may interactively save the current simulation state to file, and restore it (exactly) as required. The data is saved as a commented text file, allowing users to examine (and edit) any of the elements in the saved state as needed. The primary purpose of this feature is to enable researchers to interactively create traffic scenarios and then efficiently simulate them off-line without graphics. SHIVA can also dump state at regular intervals for later visualization or analysis. This feature also allows researchers to observe the evolution of the system from the same initial conditions given different reasoning object parameters.

11 ITS Applications

SHIVA's flexibility makes it a suitable research platform for developing a variety of ITS applications. We illustrate this by focusing on two successful tactical driving systems built at Carnegie Mellon using SHIVA's design and simulation capabilities.

11.1 Rule-based Driving

Rule-based models have been widely used in the driving domain, both as heuristics for human drivers [28, 29, 30], and as approaches for autonomous vehicle decision-making [31, 14]. The knowledge in such a model is expressed either as a finite state machine or by a set of statements like:

> "Initiate a left lane change if the vehicle ahead is moving slower than $f(v)$ m/s, and is closer than $h(v)$, and if the lane to your left is marked for legal travel, and if there are no vehicles in that lane within $g(v)$ meters, and if the desired right-exit is further than $e(x, y, v)$ meters."

where: $f(v)$ is the desired car following velocity, $h(v)$ is the desired car following distance (headway), $g(v)$ is the required gap size for entering an adjacent lane, and $e(x, y, v)$ is a distance threshold to the exit based on current lane, distance to exit and velocity. As the maneuver is initiated, the vehicle moves from a *lane tracking* to a *lane changing* state. These rules are usually derived by observing human domain experts through a knowledge acquisition process. Our cognition module has hand-crafted rules for car following, gap acceptance, merging and exit maneuvers. In addition to triggering tactical actions, the rules in our cognition module also actively control the vehicle's sensors. This *selective perception* has been shown to significantly reduce perceptual costs in driving [18]. This is possible only because SHIVA allows us to model individual sensors (most traffic simulators do not).

11.2 Distributed Reasoning for Driving

While the rule-based controller described above performs well under most conditions, it scales poorly as the complexity of scenarios increases. As new functionality is added to single-layer finite state machines, the number of states required explores exponentially [17]. Our second driving system, SAPIENT [9] addresses these deficiencies by distributing the driving task over a set of local experts, known as reasoning objects. Each reasoning object monitors an observable entity in the tactical driving situation (such as an exit or a nearby vehicle) and *votes* on the utility of various potential actions. For example, a reasoning object associated with a vehicle directly ahead may vote positively for a lane change, and negatively for an acceleration action. The votes for each action are processed by an arbiter, and the most popular action is executed. Each reasoning object's behavior depends on a number of internal parameters which are automatically learned by an evolutionary algorithm, PBIL [32]. PBIL tests a large number of vehicles with different parameter values on a set of scenarios to derive good parameter settings, which can then be transplanted into cognition modules to be used on a Navlab vehicle. This approach relies heavily on SHIVA's scenario generation facilities, and on the consistent interface between simulated and real test vehicles.

A number of other ITS-related projects are currently being developed using SHIVA's highway environment. These include a k-nearest neighbor adaptive controller [33] (for direct sensor-to-actuation mapping), realistic tire/surface and vehicle models for obstacle-avoidance [34] and human driver models for the Automated Highway System program.

12 Conclusions

Simulation is critical in the development of ITS for economic and safety reasons. SHIVA fills a gap in traffic modeling technology by providing a simulation and design environment targeted for tactical-level analysis. The combination of realistic sensor models, scenario generation tools and support for heterogeneous traffic make SHIVA particularly applicable for the design and testing of intelligent vehicle cognition modules. Since SHIVA's simulated vehicles use the same lane-tracker and controller interface as the Navlab robot vehicles, we expect that algorithms developed in simulation will only require minor modifications before being ready for real-vehicle tests.

SHIVA has been successfully used to develop a variety of ITS-related applications including tactical-level driving systems, vehicle and tire models and Automated Highway concept visualizations. Our planned extensions to SHIVA include:

- Development of realistic human driver models for mixed traffic studies.
- Control algorithms for emergency braking and swerving maneuvers.
- Simulated stereo sensors for obstacle detection.
- Support for human users to drive the simulated vehicles a natural interface.

13 Acknowledgements

The authors wish to thank Dean Pomerleau and Shumeet Baluja for valuable discussions and comments on earlier drafts of this article. This research was partially sponsored by the Department of Transportation, under cooperative agreement "Automated Highway System" (contract number DTFH61-94-X-00001).

References

[1] D. Gibson. *The Application of Traffic Simulation Models*. National Academy of Sciences, 1981.

[2] H. Chin. SIMRO: A model to simulate traffic at roundabouts. *Traffic Engineering and Control*, 26(3):109–113, March 1985.

[3] S. Wong. TRAF-NETSIM: How it works, what it does. *ITE Journal*, 60(4):22–27, April 1990.

[4] M. Van Aerde, S. Yagar, A. Ugge, and E. Case. A review of candidate freeway-arterial corridor traffic models. *Transportation Research Record*, (1132):53–65, 1987.

[5] A. Santiago and A. Kanaan. ATMS laboratories: A requirement for program delivery. In *Proceedings of 3rd Annual Meeting of IVHS America*, April 1993.

[6] D. Codelli, W. Niedringhaus, and P. Wang. User manual for Traffic and Highway Objects for REsearch, Analysis and Understanding (THOREAU) IVHS model. Technical Report MTR 92W208V1, MITRE, 1995.

[7] M. Van Aerde and S. Yagar. INTEGRATION: A model for simulating integrated traffic networks, August 1993.

[8] Q. Yang and H. Koutsopoulos. A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research*, 1995.

[9] R. Sukthankar. *Situational Awareness for Driving in Traffic*. PhD thesis, Carnegie Mellon University, 1996. (in progress).

[10] R. Sukthankar, D. Pomerleau, and C. Thorpe. SHIVA: Simulated highways for intelligent vehicle algorithms. In *Proceedings of IEEE Intelligent Vehicles*, 1995.

[11] R. Sukthankar, J. Hancock, D. Pomerleau, and C. Thorpe. A simulation and design system for tactical driving algorithms. In *Proceedings of AI, Simulation and Planning in High Autonomy Systems*, 1996.

[12] J. Michon. A critical view of driver behavior models: What do we know, what should we do? In L. Evans and R. Schwing, editors, *Human Behavior and Traffic Safety*. Plenum, 1985.

[13] D. Reece and S. Shafer. An overview of the Pharos traffic simulator. In J. Rothengatter and de Bruin R., editors, *Road User Behavior: Theory and Practice*. Van Gorcum, Assen, 1988.

[14] F. Eskafi, D. Khorramabadi, and P. Varaiya. SmartPath: An automated highway system simulator. Technical Report UCB-ITS-94-3, University of California, Berkeley, 1994.

[15] A. Göllü. *Object Management Systems*. PhD thesis, University of California–Berkeley, May 1995.

[16] P. Varaiya. Smart cars on smart roads: Problems of control. *IEEE Transactions on Automatic Control*, 38(2):195–207, February 1993.

[17] J. Cremer, J. Kearney, Y. Papelis, and R. Romano. The software architecture for scenario control in the Iowa driving simulator. In *Proceedings of the 4th Computer Generated Forces and Behavioral Representation*, May 1994.

[18] D. Reece. *Selective Perception for Robot Driving*. PhD thesis, Carnegie Mellon University, May 1992.

[19] A. Niehaus and R. Stengel. Probability-based decision making for automated highway driving. *IEEE Transactions on Vehicular Technology*, 43(3):626–634, August 1994.

[20] F. Eskafi and D. Khorramabadi. SmartPath user's manual. Technical report, University of California, Berkeley, December 1993.

[21] D. Pomerleau. *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, February 1992.

[22] R. Wallace, A. Stentz, C. Thorpe, H. Moravec, W. Whittaker, and T. Kanade. First results in robot road-following. In *Proceedings of the IJCAI*, 1985.

[23] O. Amidi. Integrated mobile robot control. Technical Report CMU-RI-TR-90-17, Carnegie Mellon University, May 1990.

[24] E. Dickmanns and A. Zapp. A curvature-based scheme for improving road vehicle guidance by computer vision. In *Proceedings of the SPIE Conference on Mobile Robots*, 1986.

[25] K. Kluge and C. Thorpe. Explicit models for road following. In *Proceedings of the IEEE Conference on Robotics and Automation*, 1989.

[26] T. Jochem, D. Pomerleau, and C. Thorpe. Vision guided lane transitions. In *Proceedings of IEEE Intelligent Vehicles*, 1995.

[27] T. Jochem, D. Pomerleau, B. Kumar, and J. Armstrong. PANS: A portable navigation platform. In *Proceedings of IEEE Intelligent Vehicles*, 1995.

[28] P. Kearney, editor. *How to Drive Better and Avoid Accidents*. Thomas Y. Crowell Company, New York, 1963.

[29] J. McKnight and B. Adams. Driver education and task analysis volume 1: Task descriptions. Technical report, Department of Transportation, National Highway Safety Bureau, November 1970.

[30] P. Wherrett. *Motoring Skills and Tactics*. Ure Smith, Sydney, 1977.

[31] M. Booth, J. Cremer, and J. Kearney. Scenario control for real-time driving simulation. In *Proceedings of 4th Eurographics Animation and Simulation Workshop*, September 1993.

[32] S. Baluja, R. Sukthankar, and J. Hancock. Prototyping intelligent vehicle modules using evolutionary algorithms. In D. Dasgupta and Z. Michalewicz, editors, To appear in *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, 1996.

[33] F. Dellaert, 1996. Personal Communication.

[34] J. Hancock. Dynamic vehicle simulation. Unpublished report, May 1996.