Appears in: Proceedings of Intelligent Systems and Advanced Manufacturing, 1997

A distributed tactical reasoning framework for intelligent vehicles

Rahul Sukthankar[ab], Dean Pomerleau[b], and Chuck Thorpe[b]

[a]Justsystem Pittsburgh Research Center
4616 Henry Street
Pittsburgh, PA 15213

[b]The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3891

ABSTRACT

In independent vehicle concepts for the Automated Highway System (AHS), the ability to make competent tactical-level decisions in real-time is crucial. Traditional approaches to tactical reasoning typically involve the implementation of large monolithic systems, such as decision trees or finite state machines. However, as the complexity of the environment grows, the unforeseen interactions between components can make modifications to such systems very challenging. For example, changing an overtaking behavior may require several, non-local changes to car-following, lane changing and gap acceptance rules.

This paper presents a distributed solution to the problem. PolySAPIENT consists of a collection of autonomous modules (termed "reasoning objects"), each specializing in a particular aspect of the driving task — classified by traffic entities rather than tactical behavior. Thus, the influence of the vehicle ahead on the available actions is managed by one reasoning object, while the implications of an approaching exit are managed by another. The independent recommendations from these reasoning objects are expressed in the form of votes and vetos over a "tactical action space", and are resolved by a voting arbiter. This local independence enables PolySAPIENT reasoning objects to be developed independently, using a heterogeneous implementation (e.g., obstacle avoidance modules can use potential fields while exit reasoning is performed using fuzzy rules). PolySAPIENT vehicles are implemented in the SHIVA tactical highway simulator, whose vehicles are based on the Carnegie Mellon Navlab robots.

Keywords: distributed AI; intelligent vehicles; tactical driving; traffic simulation; Automated Highway System

1. INTRODUCTION

An intelligent vehicle is an automobile equipped with sensors, computers, and a control system. The sensors enable the vehicle to perceive the road, potential hazards, and other vehicles; the computers process this information and determine actions (such as steering or braking); the control system executes the chosen actions. In designs where the control system is not present (or is disengaged), the intelligent vehicle passively observes the traffic situation to issue warnings or recommendations for the human driver.

The idea of a car that drives itself is not new. Research in the 1960s focused on the problem of semi-autonomous vehicle control with promising results.[1] Significant progress was made when vision-based lane-trackers were integrated with vehicle control systems, enabling robot cars to drive on clear highways under controlled circumstances.[2,3] Simultaneously, research in automatic headway control[4] and convoying[5] led to vehicles capable of autonomous car

Other author information:
RS: rahuls@cs.cmu.edu; Telephone: 1-412-683-8046; Fax: 1-412-683-4175
DP: pomerlea@cs.cmu.edu; Telephone: 1-412-268-3210; Fax: 1-412-268-5571
CT: cet@cs.cmu.edu; Telephone: 1-412-268-5571; Fax: 1-412-268-5571

following.[6,7] In 1995, an intelligent vehicle, the Carnegie Mellon Navlab 5, steered 98% of the distance between Washington D.C. and San Diego (a distance of 2800 miles), demonstrating the maturity of this technology. The *Automated Highway System* (AHS) is a proposed large-scale application of this technology. One concept for the AHS is to deploy large numbers of independent, automated vehicles into existing traffic with the goal of reducing traffic accidents and improving highway capacity.[8]

Driving in highway traffic is challenging for intelligent vehicles because good decisions need to be made given only incomplete information, in real time. Standard AI techniques such as search-based planning[9] are infeasible for several reasons. First, the effects of the robot's actions on the environment (especially other vehicles) cannot be determined. Second, most of these methods cannot function under noisy, uncertain conditions. Third, the state-space is extremely large, if realistic maneuvers such as aborted lane changes are taken into account.

The problem of navigating in traffic should be examined in the context of the entire driving task. Driving may be characterized as consisting of three levels[10]: strategic, tactical and operational. At the highest (strategic) level, a route is planned and goals are determined. At the intermediate (tactical) level, maneuvers are selected to achieve short-term objectives — passing slower traffic, changing lanes in preparation for an exit, or maintaining a space cushion. At the lowest (operational) level, the maneuvers recommended at the tactical level are translated into steering, throttle, and brake commands.

Mobile robot research has addressed these three levels to different degrees. Strategic-level route-guidance has been successfully tackled using standard AI search techniques on digital maps using GPS positioning information.[11–14] Operational-level systems have become increasingly robust, allowing intelligent vehicles to be reliably controlled on real highways.[15–17] However, the tactical level, critical to the deployment of intelligent vehicles, has not been rigorously addressed.

This paper is organized as follows. Section 2 provides an overview of the simulation environment, focusing on the perceptual and control models used by the systems described later. Section 3 presents an overview of earlier (monolithic) approaches to the tactical-level driving problem, describes one such implementation (MonoSAPIENT), and discusses its shortcomings. Section 4 develops a distributed solution to the problem (PolySAPIENT) and provides details about this architecture. Section 5 presents some results of tests involving both of these systems on micro- and macro-level scenarios. Section 6 summarizes the paper and outlines directions for future research.

2. SIMULATION ENVIRONMENT

Tactical-level driving research is generally conducted in simulation for two main reasons. First, testing new intelligent vehicle control algorithms in close proximity to other vehicles is risky. Second, tactical-level algorithms tend to require accurate information about the locations and speeds of the surrounding vehicles. Current sensor technology does not reliably provide this in all directions. Microsimulators (traffic simulators which model individual vehicles) are thus popular testbeds for these reasoning systems. The research described in this paper was conducted in the SHIVA environment.[18,19]

Intelligent vehicles consist of three subsystems: perception, cognition, and control. While this paper focuses on cognition, a brief overview of the other subsystems is warranted since they greatly impact the requirements of the cognition subsystem. In all of the implementations described in this paper, the sensor configuration shown in Figure 1 is assumed. The perception modules which process the sensor outputs are summarized in Table 2. The vehicle-tracking sensors are assumed to be reliable within 100 meters (sufficient to cover the stopping distance of the intelligent vehicle). The lane-tracker is intended to functionally model ALVINN,[3] a vision-based road follower used on the Carnegie Mellon Navlab[15] robots. The simulated controller is also modeled on the Navlab[15] robot vehicles. Cognition modules are expected to provide control tuples of the form (v, k), where v is the desired velocity, and k is the desired steering curvature setting. Until the next control tuple is received, the controller attempts to servo the vehicle's velocity and position at the previously specified settings.

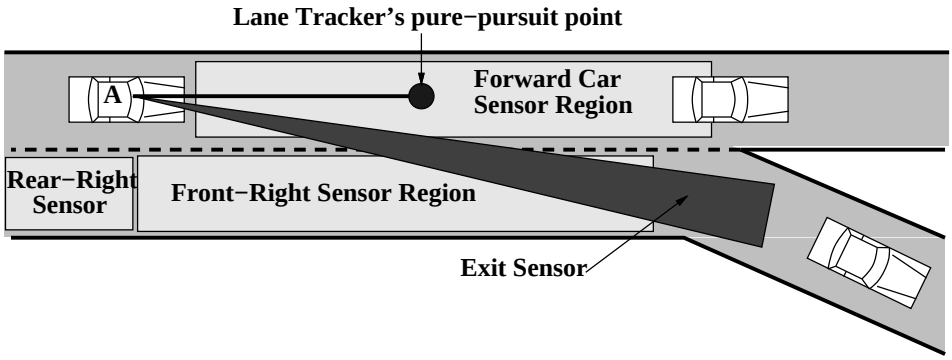

Figure 1. Functional sensor configuration used for driving in traffic.

Sensor Type	Sensed attributes
Lane Tracker	lane width, current position of pure-pursuit point, current lateral displacement, current road curvature, lane type (travel, on-ramp, etc.)
Exit Finder (range = 500m)	distance to chosen exit, lane delta to chosen exit, current exit number
Odometer	total distance traveled
Speedometer	current velocity
Positioning (e.g., GPS)	current global position (for strategic-level map)
Vehicle Sensor (range = 100m)	longitudinal distance to vehicle, lateral offset to vehicle, current velocity of vehicle, vehicle length, width, and class

Table 1. A summary of the perception modules used for driving in traffic, and the functions that they support.

3. MONOLITHIC APPROACHES TO TACTICAL-LEVEL DRIVING

The traditional approach to tactical-level reasoning is the rule-based system[20–22] (often implemented as a monolithic decision-tree or finite state machine). Such an approach is encouraged by defensive driving literature, where knowledge is often expressed in the form of high-level rules.[23,24] For example:

> "Before lane changing, look for traffic approaching from the rear in the new lane. Also, check cars that are about to enter the new lane from the far lanes.[23]"

Such rules are useful in driver education because their triggers are clear (e.g., "before lane changing") and the recommended actions are also explicit (e.g., "check cars that are about to enter the new lane"). Simple versions of such rules can be used as a starting point for a rule-based tactical driving system.

3.1. MonoSAPIENT: a decision-tree for tactical reasoning

MonoSAPIENT[19] is a hand-crafted decision-tree that determines an intelligent vehicle's tactical-level actions on the basis of sensor inputs and the vehicle's current *cognitive state*. The states, which correspond to global modes of operation, are summarized below:

lane tracking: The intelligent vehicle drives at its desired speed in the current lane, as steered by the road follower.

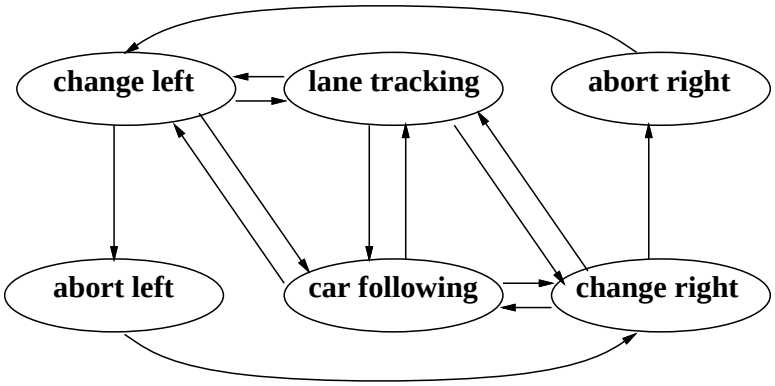

Figure 2. A summary of MonoSAPIENT's cognitive states, and their connections.

The vehicle may transition to either of the two *lane changing* states on the basis of a short-term plan, or to the *car following* state if a vehicle appears in the forward sensor.

car following: The intelligent vehicle maintains a constant, safe headway to the vehicle in front. A *lane change* may be initiated by a short-term plan, or the vehicle may move to the *lane tracking* state if the lead vehicle changes lanes or accelerates beyond the preferred velocity.

changing left/right: These states are an indication that the intelligent vehicle is involved in a lane transition (typically initiated by a short-term plan). At the conclusion of the lane change, the vehicle will move into either a *lane tracking* or *car following* state, as dictated by vehicle sensors. In response to unforeseen events (such as a pinch condition), a lane change in progress may need to be *aborted*.

aborting change left/right: These intermediate states indicate that the previously executing lane change became unsafe, and that a new lane change (recovery) should be initiated. The cognitive state transitions to one of the *lane changing* states once the recovery maneuver has been initiated.

These states and the transitions are summarized in Figure 2. It should be noted that the clarity of the state diagram is a little misleading for several reasons. First, the complex decisions involved in selecting transitions between the states and executing the associated actions are not shown. Second, the number of states (and transitions) does not scale well as the driving model becomes more complex. Third, the rules for transitions between states are brittle and may require many thresholds to be carefully tuned before acceptable driving behavior is generated. These issues are explored in greater detail in the next section.

3.2. Shortcomings of monolithic tactical driving systems

Although the monolithic reasoning system outlined above is straightforward to implement, extending it beyond toy scenarios proves to be difficult. For instance, consider the development of a rule for lane changing. A reasonable first version of such a rule may look something like this*:

> "Change left if the vehicle ahead is moving slower than $f(v)$ m/s, is closer than $h(v)$, and the target lane is clear (no other vehicles within $g(v)$ meters)."

where: $f(v)$ is the desired car following velocity, $h(v)$ is the desired car following distance (headway), and $g(v)$ is the required gap size for entering an adjacent lane. Although this rule correctly handles many lane changing scenarios, it fails in two important situations. First, a vehicle using this rule will change lanes out of an exit lane to pass a

*This rule was used in the earliest implementation of MonoSAPIENT.

blocker, even if this maneuver would cause it to miss its desired exit. Similarly, the vehicle will attempt to pass a blocker by accelerating through an exit lane, even if this would force it to exit prematurely. Second, since the rule implicitly assumes that lane changes are *atomic*, the MonoSAPIENT vehicle is unable to react to unexpected changes in the desired gap's status (such as pinch conditions).

The obvious solution is to make the rule more complex by adding a clause to the precondition: "... and if the desired exit is further than $e(x, y, v)$ meters ...", where $e(x, y, v)$ is a threshold based on the current lane, distance to exit, and velocity. Unfortunately, simple changes to rules cannot address the second complication. This is because the implicit assumption of atomic lane changes is also tied to an implicit assumption of integral lanes (i.e., that vehicles always fully occupy their (single) lane at any instant in time). By adding explicit states for lane changing (as shown in Figure 2), MonoSAPIENT can be adapted to non-atomic lane changes. This requires rules for determining transitions into the *change-left/right* states, as well as rules for determining when the lane changes are complete. As shown in Ref. 19, these rules can become very complicated.

The following disadvantages can be identified with the monolithic approach. First, as shown above, creating realistic rules requires the designer to account for many complex factors. Second, a small change in the desired system behavior generally requires many non-local modifications. Third, the brittle hand-coded rules may perform poorly in unanticipated situations. And finally, implementing new features requires one to consider a large number of interactions with existing rules. While these are not problematic in a small domain, they do not scale easily towards large applications such as the Automated Highway System. Thus, while monolithic cognition systems such as MonoSAPIENT are *theoretically capable* of solving tactical driving problems, they fare poorly in large applications. The next section describes our alternate approach.

4. A DISTRIBUTED TACTICAL REASONING FRAMEWORK

PolySAPIENT is a distributed framework for tactical reasoning that addresses these problems through the use of independent local experts, each specializing in a small aspect of the tactical driving task. Each expert, termed a *reasoning object* (see below), recommends actions for the current situation based solely upon its area of specialization. These recommendations, expressed in terms of votes and vetos, are sent to an action arbiter (See Section 4.4) which selects from these, the next action to be taken by the intelligent vehicle.

The PolySAPIENT architecture is shown in Figure 3. The *perception modules* (depicted as ellipses) are connected to the intelligent vehicle's sensors, and perform functions such as lane tracking or vehicle detection. Wherever possible, they correspond to existing systems available on real robots (e.g., the lane tracker is based on Carnegie Mellon's ALVINN[3]). Each *reasoning object* (shown as a dark rectangle) obtains information about the situation from one or two perception modules and independently calculates the utility of various courses of action. This information is sent to the *voting arbiter*, which integrates the recommendations and selects an appropriate response. Finally, the tactical action is translated into steering and velocity commands, and executed by the operational-level controller.

Conceptually, each reasoning object is associated with a single, relevant, *traffic entity* in the intelligent vehicle's tactical-level surroundings, and is responsible for assessing the impact of this entity on upcoming tactical actions. This is illustrated in Figure 4, where the scenario from Figure 1 is shown with entities and reasoning objects. For example, the reasoning object associated with the vehicle ahead monitors the motion of that vehicle and determines whether to continue car following, initiate a lane change or begin braking. Similarly, a reasoning object associated with an upcoming exit is concerned with recommending the lane changes and speed reductions needed to successfully maneuver the intelligent vehicle to the off-ramp. Obviously, for such an approach to succeed, the tactical-level task must be decomposable into several simultaneous subtasks.

4.1. Decomposing the tactical driving task

As seen in Figure 3, there is no communication between reasoning objects — the reasoning objects operate in parallel without sharing intermediate results. Thus, central to the PolySAPIENT architecture is an Assumption of Local

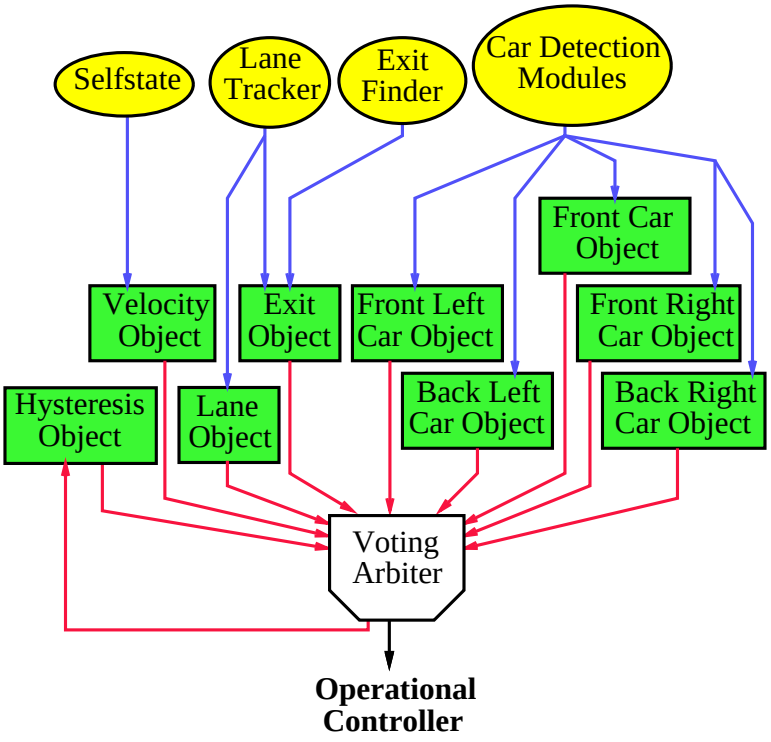

Figure 3. PolySAPIENT consists of a collection of reasoning objects which recommend actions based upon local considerations. Each reasoning object monitors a subset of the vehicle's sensors and votes upon a set of possible actions. Action fusion is performed by a domain-independent, voting arbiter.

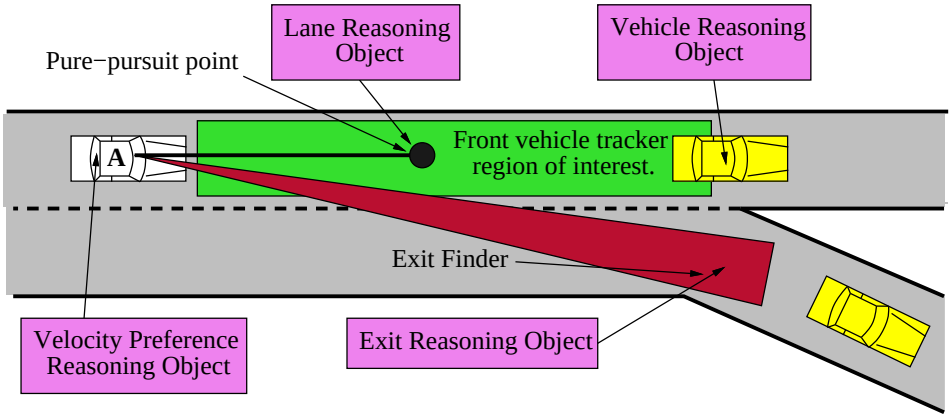

Figure 4. PolySAPIENT reasoning objects are associated with relevant physical entities in the environment. In this situation, the intelligent vehicle (A) is following a slow blocker as it approaches its desired exit.

Independence:

> When determining the utility of a given tactical-level maneuver, each relevant traffic entity in the environment may be treated *independently*.

At first glance, this statement seems very restrictive since it prevents reasoning objects from calculating the effects of potentially important interactions (e.g., an exit reasoning object, when determining the utility of a lane change, is unable to factor in the presence of a vehicle in the target lane). In practice however, the voting scheme presented in Section 4.4 correctly resolves most such interactions. The limits of the Local Independence Assumption are discussed in greater detail in Ref. 19.

By mapping each relevant traffic entity in the environment to a single reasoning object, PolySAPIENT gains a number of significant advantages. First, the tactical reasoning task is partitioned in a very clear manner, allowing designers to check at a glance whether PolySAPIENT addresses a particular aspect of driving. For example, by observing the collection of reasoning objects, it is clear that the implementation shown in Figure 3 cannot reason about traffic lights; verifying this hypothesis in a monolithic system, such as MonoSAPIENT, potentially requires the researcher to examine conditions in many scattered rules. Second, PolySAPIENT can be modified for new environments by adding appropriate reasoning objects. For instance, adding traffic light reasoning to the current implementation of MonoSAPIENT would be very difficult; by contrast, adding similar capabilities to PolySAPIENT would require very little beside designing an appropriate reasoning object (one that needed to know only about traffic lights). Third, PolySAPIENT reasoning objects are not forced to share a single world representation; each reasoning object can be implemented by a different researcher, each selecting algorithms and representations appropriate to the specific task.

4.2. The structure of a reasoning object

Externally, all reasoning objects share a similar structure — each object accepts inputs from a subset of the intelligent vehicle's perception modules and sends outputs to the voting arbiter as a set of votes over the entire *action space* (See Section 4.4). Internally, however, PolySAPIENT's reasoning objects are heterogenous, maintaining local state and using whichever representations are most applicable to the assigned subtask. To illustrate this point, two important reasoning objects are described in greater detail below.

4.2.1. Internals of the obstacle reasoning object

Each obstacle reasoning object is responsible for monitoring obstacles (including both stationary hazards as well as other vehicles) in the intelligent vehicle's surroundings. Whenever a sensor detects a new obsacle on the roadway, an instance of this object is created. While the obstacle is within range, its location, velocity and size are monitored. At every time-step, a time-to-impact potential field is generated around the relevant obstacle (similar to the generalized potential fields used by Krogh & Thorpe[25]). Actions which increase the time-to-collision are preferred, while actions that reduce the time-to-collision are discouraged. For instance, this reasoning object will be in favor of braking or swerving when faced with slow-moving obstacles ahead, and will strongly protest swerving into the path of a high-speed vehicle, or accelerating towards a hazard. Additionally, in the spirit of defensive driving, the reasoning object tries to maintain a *space cushion*[24] around the intelligent vehicle. This provides several potential benefits. First, it increases the number of useful actions in a given situation. Second, it reduces the chance that a mistake on the part of an adjacent vehicle will precipitate a collision. Third, it partially addresses the current lack of explicit planning in PolySAPIENT by breaking vehicle configuration deadlocks (such as two vehicles driving side-by-side for long periods of time). When the obstacle leaves the sensor range, the obstacle reasoning object is deleted.

4.2.2. Internals of the exit reasoning object

This reasoning object has two main responsibilities. First, it attempts to guide the intelligent vehicle into the appropriate lane when the vehicle is approaching its desired exit. Second, it prevents the vehicle from prematurely

accelerate + shift-left	accelerate + straight	accelerate + shift-right
coast + shift-left	coast + straight	coast + shift-right
decelerate + shift-left	decelerate + straight	decelerate + shift-right

Table 2. The Myopic Action Space (MAS) is a 3×3 discretization of the lateral/longitudinal space. The labels are translated at the operational level into specific numbers. Thus, "left" and "right" map to lateral positions (e.g., 0.1 of a lane) while "accelerate" and "decelerate" map to changes in velocity (e.g., 0.1 m/s).

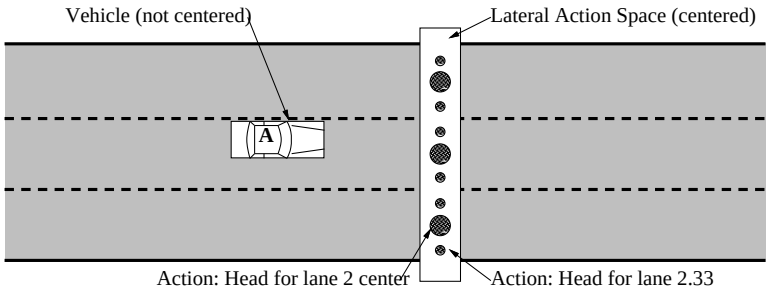

Figure 5. In the Global Action Space (GAS), a rich lateral representation is provided. Each lane is discretized into a number of sub-lanes, representing possible lateral positions for the intelligent vehicle. Reasoning objects are required to vote on the utility of driving in each of the sub-lanes.

exiting by driving in the exit lane. Unlike the mode-less obstacle reasoning object presented above, the exit reasoning object has two modes: passive (when the desired exit is distant) and active (when the desired exit is nearby). A set of fuzzy rules are used to determine the utility of the available actions. For instance, in the active state, changing lanes towards the correct exit lane is scored favorably.

4.3. Actions and action spaces

The *action space* defines the scope of tactical-level actions — unless a maneuver can be expressed by the action space, no reasoning object (or combination of reasoning objects) can express an opinion (vote) on the maneuver. While an action space could be an arbitrary set of tactical actions, the action spaces discussed in this paper are discretizations of the 2-D lateral/longitudinal space. Two types of action spaces have been explored in PolySAPIENT (see below). More details on action spaces may be found in Ref. 19.

The Myopic Action Space (MAS) is a mode-less 3×3 discretization of the lateral/longitudinal space, centered on the *current* position of the intelligent vehicle (See Table 2). This simple action space has several clear benefits: 1) MAS' semantics are unambiguous (Compare with GAS below); 2) MAS is efficient, since reasoning objects only have to consider nine possible actions. To alleviate the potential problem of jerky control (due to the coarse discretization of the action space), MAS can be extended to use a finer grid (e.g., 5×7), or the output from the tactical level could be sent through a low-pass, smoothing filter at the operational level.

The Global Action Space (GAS) provides a richer representation than the MAS, at the expense of additional complexity in the arbiter. The motivation is that MAS does not differentiate between small lateral motions (to adjust position within a lane) and larger motions corresponding to lane changes. In GAS, each lane is discretized into a number of sub-lanes, and reasoning objects are required to vote on the utility of the vehicle driving in that sub-lane (See Figure 5). While GAS is also vehicle-centric (only lateral positions in the neighborhood are considered), it is not vehicle *centered*: the position of the origin in GAS is always centered in the middle of the current lane. Whenever the vehicle changes lanes[†], GAS re-centers the coordinate frame appropriately. The advantages of this representation include: 1) the lane centers are explicitly represented — allowing lane-trackers to vote without interpolation; 2) votes

[†]When vehicles are changing lanes, GAS is centered in the old lane until the center of the vehicle crosses the dividing line — at which point, the representation shifts to the new lane.

on semantically different actions are not combined — thus, the vote for "Swerve Left" (to avoid an obstacle) should not be added to "Change lanes to the left" (since driver prefers to drive in the fast lane).

In practice, there is a trade-off between expressiveness and semantic ambiguity (not to mention efficiency) in these action spaces [‡], and in most circumstances, MAS performs adequately. Consequently, the PolySAPIENT results presented below used the MAS action space representation.

4.4. Action Selection in PolySAPIENT

In any decentralized system, the issue of command fusion is critical. The recommendations from different modules are often contradictory, and these contradictions need to be resolved since the robot can only perform a single action at any given time. Previous work in command fusion includes: subsumption,[26] potential field techniques,[25] fuzzy control,[27] and schema-based methods.[28] PolySAPIENT uses a voting arbiter, largely motivated by the Distributed Architecture for Mobile Navigation[29] (DAMN), where modules vote independently for or against discrete actions. The biggest difference is that PolySAPIENT adds *vetos* as a possible vote, to allow a allow a single reasoning object to override positive votes exerted by a group of (less knowledgeable) reasoning objects.

The simplest voting strategy is for each reasoning object to select the action which it believes best suits the current situation. Given a large number of reasoning objects and a small number of possible actions, it is possible to create a knowledge-free arbitration scheme that would select the most popular action for execution. Unfortunately, this straightforward scheme has the drawback that reasoning modules are restricted to choosing only one favorite action. Neither does this scheme allow reasoning objects to express the strength with which they favor a specific candidate action.

Thus, most voting arbiters extend the idea in two ways. First, each module is required to provide a vote (either supporting or opposing) for *every* candidate action. Second, the magnitude of each vote reflects the strength with which the module favors the given action. The total number of votes available to a module is generally unrestricted — allowing a module to support multiple candidate actions, if desired. The incoming votes from different modules are scaled according to the reasoning object type (e.g., obstacle avoidance votes are amplified). As observed above, the PolySAPIENT arbiter accepts vetos in addition to votes. Thus, the vehicle reasoning object responsible for monitoring the car ahead can prevent the *accelerate* action even if other reasoning objects are in favor of the action.

Re-examine the scenario shown in Figure 4. In this example, the reasoning object associated with monitoring the blocker strongly votes against speeding up in the current lane (since this would create an unsafe headway) while also voting against staying in the current (tailgating) position. The exit reasoning object, recognizing that the desired exit is approaching, votes for right lane changes while opposing left lane changes. And the desired velocity object, oblivious to the blocker ahead, votes to speed up. The votes submitted by a reasoning object for each action are scaled by weights corresponding to the importance of that object before being added together. Thus, the negative votes for speeding up are amplified because of the vehicle reasoning object's importance, while the positive votes for speeding up are diminished. Once all the votes associated with each action have been tallied, the action with the highest accumulated vote (which has not been vetoed) is selected. In this case, the vehicle elects to initiate a right lane change.

Decentralized vote-based reasoning introduces several complications which must be considered. First, the semantics of a vote need to be consistent across all reasoning objects. Second, the amount of domain-specific knowledge to be included in the arbiter must be determined: too much domain knowledge in the arbiter resurrects the problems of a monolithic system, but too little leads to a lack of consistency. Third, the synchronization issue must be addressed: if various reasoning objects are operating asynchronously, how can one guarantee that all of the votes were generated for the same world state? Finally, the problem of silent reasoning objects must be handled: can one safely ignore a reasoning object in any given time-step? PolySAPIENT assumes that perception modules are responsible for guaranteeing consistency in the observed world state across reasoning objects, and that all reasoning objects will

[‡]See Ref. 19 for details on action space tradeoffs.

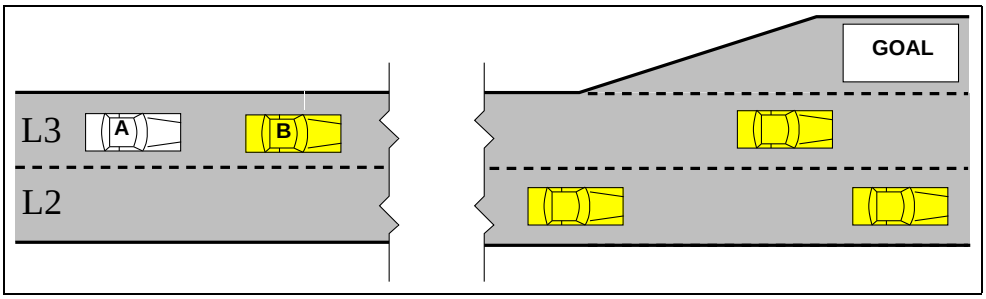

Figure 6. In this difficult scenario, the intelligent vehicle (A) must decide whether to attempt overtaking Car B at the risk of missing its desired exit.

respond within their alloted time. Also see Ref. 19 for details about vote-interpolation and hysteresis.

4.5. Automated parameter tuning

The PolySAPIENT architecture contains many tunable parameters, both within individual reasoning objects (such as thresholds and gains), and vote scaling factors between the reasoning objects and the arbiter. Manually adjusting these parameters, while possible, is tedious. Accordingly, PolySAPIENT employs an automated optimization technique that allows these parameters to be learned in an unsupervised manner using a set of training scenarios and a user-specified evaluation metric. Ref. 30 provides details of the evolutionary algorithm, and presents results showing the robustness of this strategy.

5. RESULTS

Tactical reasoning systems should be examined both in terms of individual vehicle performance and in aggregate traffic characteristics. Several experiments were performed using MonoSAPIENT and PolySAPIENT in these settings. This section presents a single representative scenario of each type. More details are available in Ref. 19.

Consider the scenario shown in Figure 6, where the intelligent vehicle (A) must decide whether to attempt overtaking Car B at the risk of missing its desired exit. Figure 7 highlights the differences in behavior between MonoSAPIENT and PolySAPIENT. The former, employing hand-coded rules, is unable to perform the overtaking maneuver because the exit-taking rule inhibits overtaking. PolySAPIENT's success on this particular scenario can be attributed to two factors. First, the aggressive driving style resulting from the use of time-to-impact potential fields enables the PolySAPIENT vehicle to safely merge into a smaller gap. Second, the distributed reasoning system is better at making tradeoffs (in this case between travelling at the desired speed versus missing the exit). Thus, PolySAPIENT is likely to take risks in favorable circumstances.

In the second set of experiments, intelligent vehicles were injected onto a test track every few seconds from a single on-ramp. We examined three cases: traffic consisting only of MonoSAPIENT vehicles; traffic consisting only of PolySAPIENT vehicles; and a mixed traffic setting with equal numbers of each type. The results showed that while none of the three cases led to an unsafe roadway, they resulted in significantly different highway capacities. The presence of PolySAPIENT vehicles increased the number of vehicles that could drive on the test track because their headway rules were not brittle. PolySAPIENT vehicles, in their attempts to maintain space cushions, also served to spread the traffic more evenly. More details on these experiments is available in Ref. 19.

6. SUMMARY AND FUTURE DIRECTIONS

Monolithic approaches to the tactical driving problem (MonoSAPIENT), while theoretically capable of solving the task, are difficult to implement and maintain. Decomposing the problem into a set of simultaneous, relatively

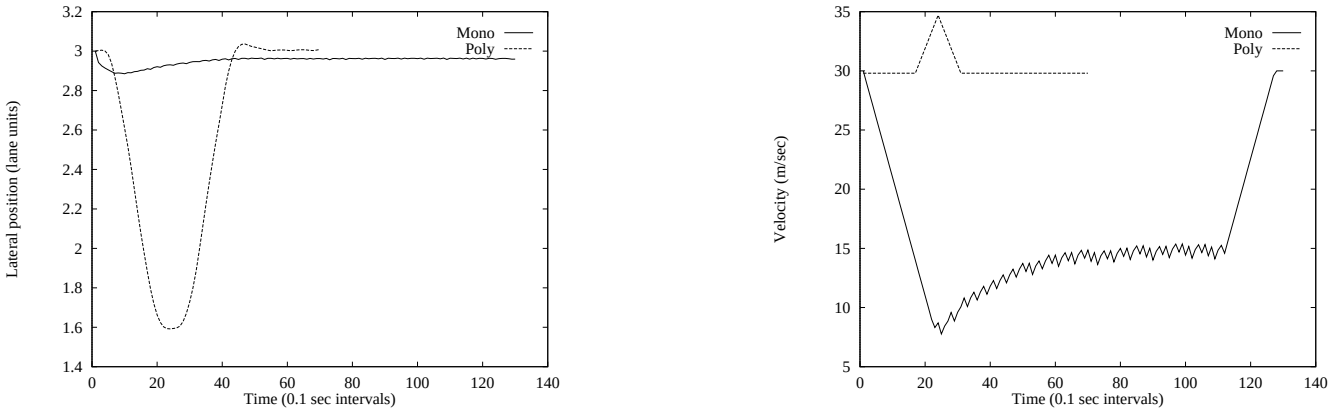

Figure 7. Lateral displacement (left) and velocity (right) as a function of time, for rule-based and SAPIENT vehicles on the more difficult exit scenario (See Figure 6).

independent subtasks allows designers to rapidly build distributed solutions (PolySAPIENT). Each of these subtasks, associated by a traffic entity in the environment, is managed by a local expert, known as a reasoning object. Although different reasoning objects implement their processing using a variety of algorithms, all reasoning objects speak a common output language: votes and vetos over a shared action space. A knowledge-free arbiter performs action selection by selecting the most popular (non-vetoed) action. The loosely coupled structure of the reasoning objects allows different reasoning objects to be implemented by different researchers, using different methods.

To date, this reasoning system has only been tested in simulation. Our primary direction for future research is to implement such a tactical reasoning system on real robot vehicles, such as the Carnegie Mellon Navlabs. This will involve overcoming significant challenges in perception such as reliable obstacle and vehicle tracking, as well as integrating the reasoning system with the existing operational-level controllers.

ACKNOWLEDGEMENTS

The microsimulator, SHIVA, was developed in collaboration with John Hancock. The automated parameter tuning for PolySAPIENT reasoning objects was in collaboration with Shumeet Baluja and John Hancock. The data processing scripts were developed by Gita Sukthankar. This research was partially sponsored by the Department of Transportation, under cooperative agreement "Automated Highway System" (contract number DTFH61-94-X-00001).

REFERENCES

1. K. Gardels, "Automatic car controls for electronic highways," Tech. Rep. GMR-276, General Motors Research Labs, June 1960.
2. 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.
3. D. Pomerleau, *Neural Network Perception for Mobile Robot Guidance*. PhD thesis, Carnegie Mellon University, February 1992.
4. J. Bender and R. Fenton, "A study of automatic car following," *IEEE Transactions on Vehicular Technology* **18**, pp. 124–140, 1969.
5. D. Gage and B. Pletta, "Ground vehicle convoying," *SPIE Mobile Robots II* **852**, pp. 319–328, 1987.
6. N. Kehternavaz, N. Griswold, and S. Lee, "Visual control of an autonomous vehicle (BART) — the vehicle following problem," *IEEE Transactions on Vehicular Technology* **40**(3), 1991.

7. R. Sukthankar, "RACCOON: A Real-time Autonomous Car Chaser Operating Optimally at Night," in *Proceedings of IEEE Intelligent Vehicles*, 1993.
8. M. Bayouth and C. Thorpe, "An AHS concept based on an autonomous vehicle architecture," in *Proceedings of the Third Annual World Congress on Intelligent Transportation Systems*, 1996.
9. R. Fikes, P. Hart, and N. Nilsson, "Learning and executing generalized robot plans," *Artificial Intelligence* **3**(4), 1972.
10. J. Michon, "A critical view of driver behavior models: What do we know, what should we do?," in *Human Behavior and Traffic Safety*, L. Evans and R. Schwing, eds., Plenum, 1985.
11. R. Elliot and M. Leak, "Route finding in street maps by computers and people," in *Proceedings of AAAI*, 1982.
12. M. Sugie, O. Menzilcioglu, and H. Kung, "CARGuide — on-board computer for automobile route guidance," Tech. Rep. CMU-CS-84-144, Carnegie Mellon University, 1984.
13. J. Rillings and R. Betsold, "Advanced driver information systems," *IEEE Transactions on Vehicular Technology* **40**(1), 1991.
14. R. von Tomkewitsch, "Dynamic route guidance and interactive transport management with ALI-Scout," *IEEE Transactions on Vehicular Technology* **40**(1), 1991.
15. C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the Carnegie Mellon Navlab," *IEEE Transactions on PAMI* **10**(3), 1988.
16. I. Masaki, ed., *Vision-Based Vehicle Guidance*, Springer-Verlag, 1992.
17. R. Larsen, "AVCS: An overview of current applications and technology," in *Proceedings of IEEE Intelligent Vehicles*, 1995.
18. R. Sukthankar, D. Pomerleau, and C. Thorpe, "SHIVA: Simulated highways for intelligent vehicle algorithms," in *Proceedings of IEEE Intelligent Vehicles*, 1995.
19. R. Sukthankar, *Situation Awareness for Tactical Driving*. PhD thesis, Carnegie Mellon University, January 1997. Also available as CMU Tech Report CMU-RI-TR-97-08.
20. D. Reece, *Selective Perception for Robot Driving*. PhD thesis, Carnegie Mellon University, May 1992.
21. A. Niehaus and R. Stengel, "Probability-based decision making for automated highway driving," *IEEE Transactions on Vehicular Technology* **43**(3), 1994.
22. 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*, 1994.
23. J. McKnight and B. Adams, "Driver education and task analysis volume 1: Task descriptions," tech. rep., Department of Transportation, National Highway Safety Bureau, November 1970.
24. P. Wherrett, *Motoring Skills and Tactics*, Ure Smith, Sydney, 1977.
25. B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proceedings of IEEE Conference on Robotics and Automation*, 1986.
26. R. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation* **2**(1), 1986.
27. L. Zadeh, "Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence," 1991. Introduction in Bart Kosko's book.
28. R. Arkin, "Motor schema-based mobile robot navigation," *The International Journal of Robotics Research* **8**(4), pp. 92–112, 1989.
29. J. Rosenblatt, *DAMN: A Distributed Architecture for Mobile Navigation*. PhD thesis, Carnegie Mellon University, 1996.
30. R. Sukthankar, S. Baluja, and J. Hancock, "Evolving an intelligent vehicle for tactical reasoning in traffic," in *Proceedings of the International Conference on Robotics and Automation*, 1997.