A Control Architecture for Integrated Cooperative Cruise Control and Collision Warning Systems

Anouck Renée Girard, João Borges de Sousa, James A. Misener and J. Karl Hedrick

Abstract-- In this paper, we propose a hierarchical control architecture for an enhanced variant of Cooperative Adaptive Cruise Control (CACC), which would include some Cooperative Forward Collision Warning (CFCW) functionality. Simply put, a CACC system is a more sophisticated variant of cruise control .

By a control architecture we mean a specific way of organizing the motion control and navigation functions performed by the cars. It is convenient to organize the functions into hierarchical layers. This way, a complex design problem is partitioned into a number of more manageable sub-problems that are addressed in separate layers.

This paper discusses vehicle control requirements and maps them onto a layered control architecture. The formalization of the hierarchy is accomplished in terms of the specific functions accomplished by each layer and of the interfaces between layers. The implementation of the layers is discussed and illustrative examples are provided.

Index Terms—cooperative adaptive cruise control, cooperative forward collision warning systems, control architectures, hybrid systems, supervisory control.

I. INTRODUCTION

Vehicle control technologies are expected to yield many-fold benefits – with fully automated cooperative systems, in congestion relief [1]; and with autonomous driver-assistance systems, in enhanced safety, particularly with respect to rear-end collisions [2]. However, a safety- and congestion-enhancing system that combines features of both may be deployable in the intermediate term and may yield benefit in both congestion relief and increased safety. This is what we seek, and it is such a notional system that we describe along with the accompanying control architecture. In this paper, we propose a hierarchical control architecture for the control and maneuver coordination of multiple cars using integrated cruise control and collision warning systems. To do this we consider several modes of operation for each vehicle: adaptive cruise control (ACC), cooperative adaptive cruise control (CACC), or cooperative forward collision warning (CFCW).

The proposed architecture distributes information and control authority among cars and deals with exceptions and faults. It is organized hierarchically, with the lower layers of the hierarchy consisting of continuous controllers that interact with the sensors and actuators to produce the desired positioning and tracking performance. Higher layers of the hierarchy will be modeled by discrete event systems used for maneuver coordination, fault identification and reconfiguration. These layers sequentially organize the generation of the optimal coordinated trajectories for all vehicles (global control), the optimal trajectories for each car (vehicle control), the optimal trajectory tracking schemes and the optimal engine controls (local control), to name just a few. The de-coupling of optimization problems eventually compromises the overall optimality but makes the problem tractable. Hence, the architecture represents an empirical compromise between tractability and optimality. It will become apparent that the task is formidable, even in a reduced setting. Here we give an overview of what is involved, describe the key design concepts and establish a framework for tackling the problem.

This paper is organized as follows. In section 2, representative requirements for the cruise control and collision warning applications are presented, along with some of the expected maneuvers and scenarios. In section 3, we outline the proposed solution and describe the control architecture that is the starting point of our design. The controller design and implementation for this architecture are described in section 4. Finally in section 5, we highlight some of the subtle issues raised by the design process.

II. AUTOMOTIVE CACC AND CFCW REQUIREMENTS

A. Problem domain and terminology

In this paper, our challenge problem is an enhanced variant of Cooperative Adaptive Cruise Control (CACC), which would include some Cooperative Forward Collision Warning (CFCW) functionality. In short, a CACC is a more sophisticated variant of an Adaptive Cruise Control (ACC) system. The ACC is recently emerging as a marketed option (e.g., Mercedes S-Class, Lexus L430) and the CACC is a vision of extending the autonomous sensing aspect of an ACC by introducing communication of key pieces of information from the lead vehicle.

What do these acronyms really mean? Figure 1 below helps define ACC, CACC and CFCW. With an ACC the following vehicle (Vehicle 2) follows the trajectory of the leader (Vehicle 1), at a driver-requested time gap t, usually between 1 and 2 seconds. As a forward ranging radar in Vehicle 2 senses a change in 1, 2 is commanded to change, though constrained by a maximum deceleration of 2 – 3 m/s^2. When there is no Vehicle 1 present, ACC simply reverts to the driver-requested set speed so it defaults to conventional cruise control.

The authors are with the University of California at Berkeley.

In CACC mode, the preceding vehicle can communicate actively with the following vehicle so that their speed changes can be coordinated with each other. (Because communication is quicker, more reliable and less noisy than autonomous sensing – and because braking rates as well as maximum braking capability can be communicated -- significantly closer safe vehicle following [t=0.5 s] has been postulated as a benefit of CACC.)

In CFCW mode, Vehicle 1 brakes and once Vehicle 2 crosses the automated 2 – 3 m/ s^2 threshold of ACC, a set of warning cues is given to the Vehicle 2 driver. The Cooperative element, again, is communication of the braking rate and maximum braking capability of Vehicle 1; such information can signal the need for emergency intervention by the driver sooner and more reliably than an autonomous FCW. This CFCW mode does not handle other forward collision threats such as stopped vehicles or other stationary objects. This makes our implementation less complex, though still a potentially beneficial system.

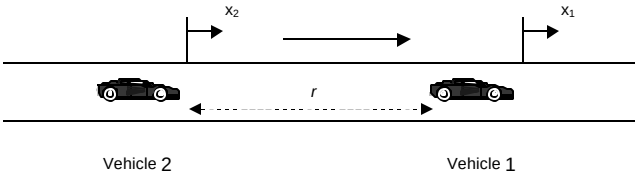

Figure 1. Definition of variables for vehicle following applications.

B. *Modes of Operation*

The integrated CACC + CFCW challenge problem can be regarded as a cooperative vehicle following problem with seven different modes of operation:

1. Off Mode. Cruise control is not activated, and the driver is in control.
2. Conventional Cruise Control Mode (CC). There is no Vehicle 1.
3. ACC Mode. Vehicle 1 is not equipped to communicate with Vehicle 2.
4. CACC Mode. Vehicle 1 and 2 are equipped.
5. CW-Off mode. Collision warning is turned off.
6. FCW Mode. Vehicle 1 is not equipped to communicate with Vehicle 2. Vehicle 1 is interpreted to be a threat, and the driver of Vehicle 2 is alerted to intervene. (Once the driver intervenes, he is in State (1).)
7. CFCW Mode. Vehicle 1 is equipped to communicate to Vehicle 2. Vehicle 1 is interpreted to be a threat, and the driver of Vehicle 2 is alerted to intervene. (Once the driver intervenes, he is in State (1).)

The seven basic modes of operation are insufficient to fully describe the operation of the system: transitioning modes need to be considered. In particular, there are cases where it is not possible to transition from one mode to the other. This is particularly important when the system enters a collision-warning mode. At that point, control must be returned to the driver. It is not possible to transition directly into another cruise control mode.

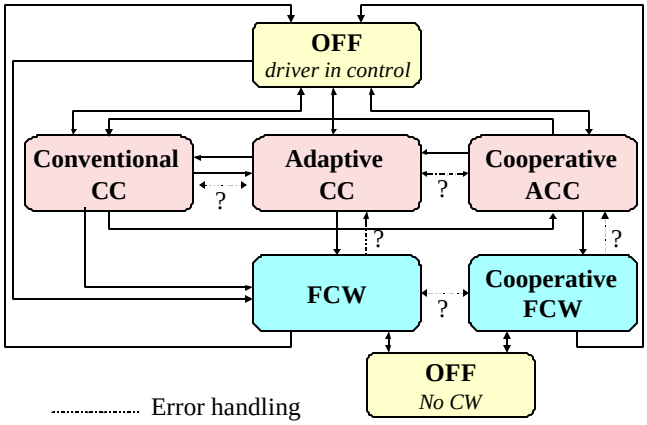

Figure 2. Modes of the system and legal transitions.

The driver has the final say in validating some of the transitions. In particular, he will be able to select which cruise control mode to be in (OFF, CC, ACC, CACC), and whether to turn the collision warning system on or off.

C. *Scenario*

Let us describe a use case for this problem. We will consider the case of two vehicles traveling in the same lane. Vehicle 1 (V1), is traveling at constant speed under CC ahead of vehicle 2 (V2), that is traveling faster and under driver's control. When V2 reaches a pre-specified distance from V1, V2's computer system asks V1's computer system about the corresponding capabilities and the availability for cooperation. Upon reception of an affirmative response, the computer system asks both drivers if they want to engage in CACC. If yes, the computer systems from both cars take control and transition to CACC mode. The transition involves an intermediate phase where V2 is controlled to decelerate in order to reach the speed of V1, and remain at a pre-specified distance from this vehicle until the system is turned off.

III. OUTLINE OF PROPOSED ARCHITECTURE

A. *Assumptions*

We considered the following assumptions in our developments. 1) Cooperative ACC does not involve more than two vehicles at any given time; 2) the collision warning system of a specific vehicle adapts its behavior to the presence of another vehicle when they engage a cooperative maneuver. Thus, it prevents the onset of alarms that would occur otherwise; 3) the maneuvers are specified only for longitudinal control.

B. *Design Process*

The design process consists of the following steps:

1. Deciding on the number of levels, their role, their descriptive language and the way they interact.

2. Identifying the information that is needed at each level to make meaningful decisions.
3. Designing interfaces to the communication and sensing architecture.

The design process leveraged the Berkeley/PATH experience in the motion coordination of multiple vehicles [1,3]. In fact, the control and coordination of cars during CACC/CFCW can organized according to the principles that were used to design and implement the PATH architectural concept for automated highways [1,4]. The PATH architecture design describes individual motions in terms of a small number of prototypical maneuvers that are used as building blocks for more complex maneuvers, such as those involving the coordination of several vehicles. Informally, a software implementation of a prototypical maneuver consists of a reference generation component, an atomic control law (regulation or tracking) and conditions under which the maneuver can take place and terminate. The reference generation component generates the inputs to the control law. Formally, a maneuver is implemented as a hybrid automaton. Prototypical maneuvers can then be verified for safety and consistency as required by the safe operation of the whole system.

We analyzed the integrated CACC + CFCW challenge problem to determine a set of basic maneuvers from which all of the required motions can be assembled. We concluded that each vehicle has to be able to perform the following basic maneuvers:

1. Manual operation (driver in control) (OFF)
2. Drive (CC)
3. Follow (ACC)
4. Smart-Lead (CACC)
5. Smart-Follow (CACC)
6. Join ACC (JACC)
7. Separate ACC (SACC)
8. Join CACC (JCACC)
9. Separate CACC (SCACC)

These maneuvers, in turn, build on a small number of atomic control laws:

a) The "throttle" law allows the vehicle to accelerate or maintain a given acceleration through engine control.
b) The "brake" law allows the vehicle to decelerate.

After identifying the atomic control laws, the requirements in terms of actuator, sensor and communication capabilities were laid out, as shown in figure 3.

A collision avoidance law must be presented in order to make the set complete. However, this maneuver has not been designed, implemented or tested in the scope of this work.

We designed a controller for each of the basic maneuvers.

These controllers share the same structure, as shown in figure 4. Each basic maneuver is paired with a compatible reference generator. Each controller is implemented as a hybrid automaton that coordinates the execution of the

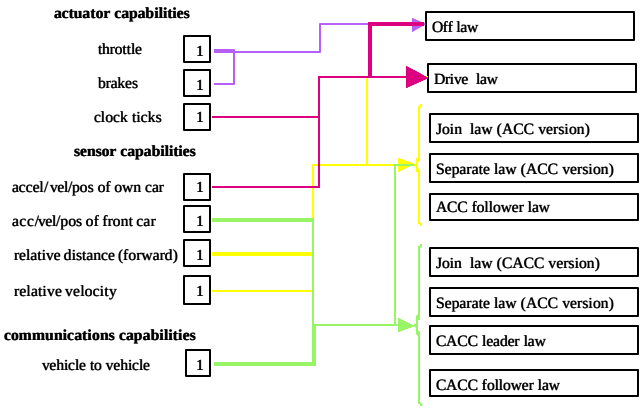

Figure 3.
System requirements for basic maneuver laws.

underlying atomic control laws with that of the associated reference generator, according to the logic encapsulated in the corresponding state machine.

The dependencies for each of the nine aforementioned maneuver controllers can be obtained from figure 3. Identifying the dependencies of each controller on communications, sensors and actuators is particularly useful in the context of fault tolerant architectures. It allows for reconfiguration of the system in the presence of failure. For instance, it provides the logic for cars to switch from CACC to ACC in the event of a communication failure.

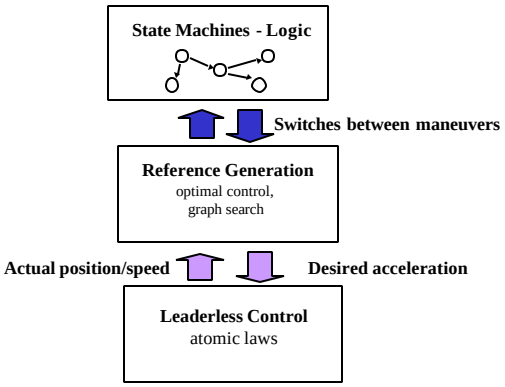

Figure 4.
Organization of basic maneuver control.

C. Overview of Architecture

To deal with mission handling and safety issues, a three-layer architecture (as presented in figure 5) that moves from discrete to continuous signals was used [1,4,5]. It should be noted that our design is not necessarily unique or optimal, but as a preliminary approach is sufficient to prove our point. Different alternatives for hybrid system design can be found in [6,7].

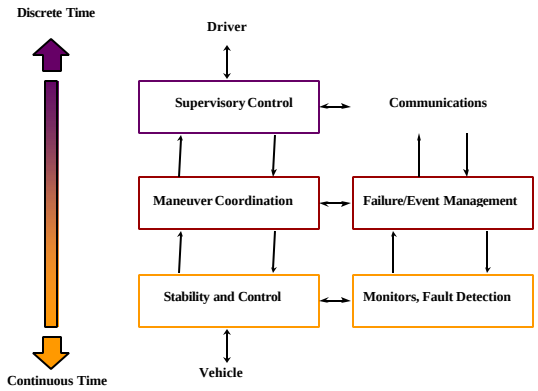

Figure 5. Hierarchical Control Architecture.

1. Vehicle stability and control layer: the automated modules. The vehicle dynamical models are given in terms of nonlinear ordinary differential equations. This level deals with continuous signals, and interfaces directly with the vehicle hardware. It contains several control algorithms and sensor data processing and monitoring for fault detection. Control laws are given as vehicle state or observation feedback policies for controlling the vehicle dynamics. Sensor processing at the stability and control layer includes environmental monitors that are used to signal changes in the environmental conditions. The corresponding events are sent to the maneuver coordination layer that will promote the change to the next preferred mode.
2. Maneuver coordination layer: control and observation subsystems responsible for safe execution of the basic maneuvers such as OFF, CC or CACC. Maneuvers may include several modes according to the lattice of preferred operating modes. Mode changes are triggered by events generated by the stability layer monitors. This feature provides a first level of dynamic reconfiguration that is further extended to accommodate fault handling.
3. Supervisory control layer: control and observation strategies that implement a decentralized scheme for the integrated CACC + CFCW challenge problem. Each vehicle has a supervisor that, upon the occurrence of certain events (including the ones triggered by the driver), starts patterns of interaction with the supervisor of the other vehicle. Depending on the validity of some conditions, this results in a coordinated maneuver of the two vehicles. The patterns of interaction among the two supervisors are implemented as protocols that can be verified for properties such as liveness or absence of deadlocks. Each supervisor, in turn, commands the execution of basic maneuvers of the vehicle under its control.
4. Interfaces between layers:

a) Vehicle: the automobiles input actuator commands (a desired throttle angle and brake master cylinder pressure) from the stability and control layer, and output information on whether the actuators and sensors are functioning properly.

b) Vehicle stability and control layer: the vehicle stability and control layer inputs set points in acceleration from the maneuver coordination layer, and outputs state information to the maneuver coordination layer.

c) Maneuver coordination layer: the maneuver coordination layer inputs specific maneuvers to execute from the supervisory control layer, and outputs state information and information about the status of the current on-going maneuver to the supervisory control layer.

d) Supervisory control layer: the supervisory control layer inputs commands from the driver and outputs all state and on-going maneuver information as well as information regarding whether the maneuvers are being executed properly.

e) Driver: the driver "inputs" state, maneuver and mission information and "outputs" commands for the car to execute.

IV. CONTROLLER DESIGN

In this section we give a brief description of our approach to controller design and implementation within the multi-layered control architecture for integrated CACC and CFCW systems. Each layer will be examined individually. We present a sketch of the controller designs for this control architecture, with special emphasis on longitudinal control techniques. This architectural concept was partly implemented using a real-time hybrid systems language, TEJA [8]. The controller interfaces can be reused to incorporate other longitudinal controllers into the framework, so comparisons can be made under the same set of assumptions.

A. Stability and control layer

The primary focus of the stability and control layer is to ensure smooth operation of a single car and to inform the maneuver coordination layer of whether set points are reached or not.

The stability and control layer contains several different control laws for throttle (or engine) and brake control, and a switching mechanism between throttle and brake control, depending on the desired acceleration, and the actual acceleration and velocity of the car. Hence, the switching law is independent of the basic maneuver and/or coordination mechanism used.

In the following equations, the following variables are used (details in [9]):

- F_a is the aerodynamic drag force
- M_{rr} is the rolling resistance moment
- R_g is the gear ratio (related to engine and vehicle speeds)
- β_{ades} is the desired synthetic acceleration
- τ_{ct} is the control torque
- h is the effective wheel radius

Throttle control is used if:

$$b_{ades} + R_g(M_{rr} + F_a + mg\sin q) - t_{ct} \geq 0$$

For throttle control, the desired torque is computed as:

$$t_{edes} = b_{ades} + R_g (M_{rr} + hF_a + mgh\sin q)$$

Brake control is used if:

$$b_{ades} + R_g (M_{rr} + F_a + mg\sin q) - t_{ct} < 0$$

For brake control, the desired torque is computed as:

$$t_{edes} = -\left[\frac{b_{ades} + t_{ct}}{R_g}\right] - M_{rr} - hF_a - mgh\sin q$$

1) Throttle control

From the desired torque, the desired throttle angle is computed using an engine map.

2) Brake control

From the desired torque, two different brake control strategies have been implemented.

In the first strategy [9], the master cylinder pressure is controlled. A pressure regulator valve controls the pressure applied on the hydraulic actuator. Seal friction exists in the master cylinder and the actuator, and a small amount of hysteresis is present in the pressure regulation valve. The friction is modeled as hyperbolas from various points in the hysteresis loop and can be written as:

$$P_{mc} = g(u)$$

Feed-forward plus proportional feedback control is used. The control law can be written as:

$$u_b = g^{-1}(P_{mc}) + k_b (P_{mc\_des} - P_{mc})$$

Where u_b is the applied command input to the brake solenoid valve, P_{mc_des} the desired master cylinder pressure, P_{mc} the measured master cylinder pressure, and $k_b>0$ a feedback gain.

In the second brake control strategy [10], the wheel brake pressure is controlled, and the brake system is modeled. The control law uses dynamic surface control [11] and can be written as:

$$P_{mc} = P_w - \left[\frac{\dot{P}_{wdes} - l_b (P_w - P_{wdes})}{\frac{\partial P_w}{\partial V} C_q}\right]^2$$

Where P_{w_des} is the desired wheel pressure, V is the volume of displaced brake fluid, P_w the pressure at the wheel, C_q a flow coefficient, and λ_b is a control gain.

B. Maneuver coordination layer

The maneuver coordination layer interfaces high-level supervision (discrete) with the atomic continuous control laws control of the car. To do this we use the basic maneuvers. A reference generator, pre-conditions (that determine the validity of the initial conditions for the maneuver), invariants (that specify conditions under which the maneuver is valid) and termination conditions form a basic maneuver. All these conditions can be enabled or disabled. Only one maneuver is active at any given time.

The nine controllers in the maneuver coordination layer (off, drive, follow, smart-lead, smart-follow, join-acc, separate-acc, join-cacc and separate-cacc) send desired acceleration set points to the controller described in section A.

Each controller is formed of a control law, and a "protocol" that is used to coordinate maneuvers. The current design uses protocols in the form of finite state machines to organize the maneuvers in a systematic way. They receive the commands from the supervisory controller and aggregated information from the individual cars, then use this information to decide on a control policy and issue commands to the stability and control layer.

As an example, the protocol for the join CACC controller is given here. It is formed of two state machines, one for each car involved.

For simplicity, we call the vehicles "lead" and "follow" throughout the following description. The state machine containing the logic for the lead vehicle is presented in figure 6. The state machine containing the logic for the follow vehicle is presented in figure 7.

The lead module gets a request asking it whether it is available for CACC; if not a time out occurs and the maneuver is aborted, and if yes the lead car "blocks" itself so that no other car may join while the maneuver is taking place.

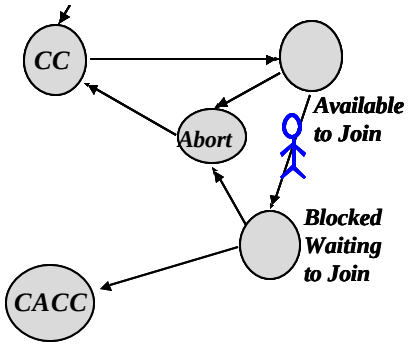

Figure 6.
State machine for lead vehicle.

When the maneuver is complete, the lead vehicle enters CACC operation (smart-lead maneuver).

The follow vehicle asks the lead vehicle for permission to join. If it is not granted, the vehicle goes back to independent operation; otherwise the follow vehicle starts its approach towards the lead vehicle. A trajectory is computed for the join maneuver. This trajectory is communicated to the lower level controller that regulates the follow car.

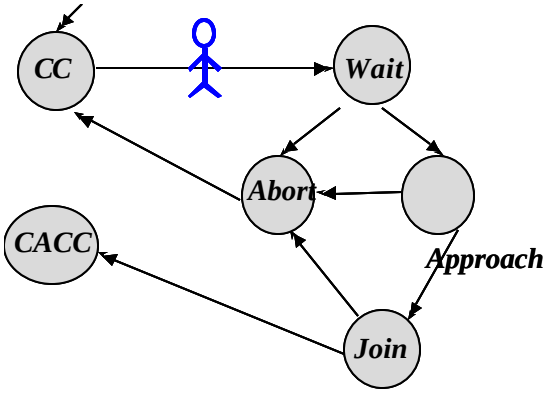

Figure 7.
State machine for follow vehicle.

When the cars have joined, the CACC is activated; the follow car enters CACC operation (smart-follow). The system can be set so, at this point, the follow car makes itself available to join. In this case, platoons of cars may be formed.

C. Supervisory control layer

The supervisory controller encapsulates the logic required to implement the coordinated control of two vehicles in a decentralized fashion. Hence, the integrated CACC + CFCW is achieved without any centralized supervisor, since each of the vehicles is controlled by individual supervisors, that coordinate their operation through discrete protocols. In the current design, finite state machines are used to represent the communication protocols and organize them in a systematic way.

We see that each layer of the control hierarchy involves different entities and algorithms that relate entities at one layer to entities at the adjacent layers. Moreover, there is a theoretical framework at each layer that provides a meaning to its entities, and relations between those frameworks that are operational in terms of the algorithms.

V. CONCLUDING REMARKS

In this paper we have proposed a hierarchical control architecture for integrated CACC and CFCW automotive systems. A hierarchical structure for control and planning has been adopted to provide a systematic design framework for this complex system. This architecture is being implemented on the PATH cars [1] to support the physical validation of the key design issues . We leveraged on our experience in designing control architectures for multi-vehicle systems to define patterns of design and implementation that can be reused across projects. Thus we are distilling from experimentation to derive new concepts for the integrated design and implementation of model-based integrated hybrid systems.

Acknowledgements: The material is based upon work supported by DARPA (MoBIES) under grant number F33615-00-C-1698, and the Link Foundation. The authors would like to thank Prof. Varaiya for helpful discussions providing insight into this problem.

References

[1] P. Varaiya, "Smart Cars on Smart Roads: Problems of Control", IEEE Trans. on AC, Vol 38, No. 2, February 1993.

[2] W. Najm, “A Review of Collision Avoidance Technologies”, Proceedings of the Workshop on Collision Avoidance Systems, Issues and Opportunities, Intelligent Transportation Systems – America, 1995.

[3] A. Girard, J. Borges de Sousa and J.K. Hedrick, “A Hierarchical Control Architecture for Mobile Offshore Bases”, Marine Structures Journal, Special Issue on Very Large Floating Structures, Vol. 13 (2000), pp. 459-476.

[4] F. Eskafi, D. Khorrambadi, and P. Varaiya, 'SmartPath: An Automated Highway System Simulator", Tech. Rep. PATH Tech. Memo 92-3, Institute of Transportation Studies, University of California, Berkeley, CA 94720, October 1992.

[5] A. J. Healey, D.B. Marco and R. B. McGhee, Autonomous Underwater Vehicle Control Coordination Using A Tri-Level Hybrid Software Architecture, Proc. of 1996 IEEE Conference on robotics and Automation, Minneapolis, Minnesota, April 1996, pp 2149-2159.

[6] A. Girard, "A Convenient State Machine Formalism for High-Level Control of Autonomous Underwater Vehicles", Master’s Thesis, Florida Atlantic University, Boca Raton, FL, May 1998.

[7] D. N. Godbole, J. Lygeros and S. Sastry, "Hierarchical Hybrid Control: An IVHS Case Study", in Hybrid Systems II (P. Antsaklis, A. Nerode and S. Sastry, eds.), no 999 in LNCS, Springer Verlag, 1995.

[8] www.teja.com

[9] J.C. Guerdes, “Decoupled Design of Robust Controllers for Nonlinear Systems: as motivated by and applied to Coordinated Throttle and Brake Control for Automated Highways”, Ph.D. Dissertation, Mechanical Engineering Department, University of California at Berkeley, 1996.

[10] J.C. Guerdes and J.K. Hedrick, “Vehicle Speed and Spacing Control via Coordinated Throttle and Brake Actuation”, Proceedings, FAC Conference, San Francisco, CA, 1996.

[11] D. Swaroop, J.C. Guerdes, P.P. Yip and J.K. Hedrick, “Dynamic Surface Control of Nonlinear Systems”, Technical Report, Vehicle Dynamics and Control Laboratory, University of California at Berkeley, 1996.