A Modular, Hybrid System Architecture for Autonomous, Urban Driving

Dave Wooden, Matt Powers, Magnus Egerstedt, Henrik Christensen, and Tucker Balch
Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, GA 30332

September 26, 2007

Abstract

Autonomous navigation in urban environments inevitably leads to having to switch between various, sometimes conflicting control tasks. Sting Racing, a collaboration between Georgia Tech and SAIC, has developed a modular control architecture for this purpose and this paper describes the operation and definition of this architecture through so-called nested hybrid automata. We show how to map the requirements associated with the DARPA Urban Grand Challenge onto these nested automata and illustrate their operation through a number of experimental results.

1 Introduction

In this paper we describe Sting Racing's design and implementation of an unmanned system for entry into the DARPA Urban Challenge. In particular, the focus of this paper is on the system architecture and we present an extensively field tested architecture that is based on a modular structure comprised of so-called Nested Hybrid Automata (NHA).

The vehicle that we use for this is a Porsche Cayenne (as shown in Figure 1), retrofitted for complete computer control, and use a combination of GPS/IMU, camera, radar and LADAR data to generate *situational awareness*, which is, arguably, the major challenge separating the Urban Grand Challenge 07 from the previous two Grand Challenges [1, 2].

Figure 1: The Sting Racing retrofitted Porsche Cayenne entry to the DARPA Urban Challenge.

Situational awareness constitutes one of the main sources increased difficulty from Grand Challenge 05 to the Urban Grand Challenge 07, which can be understood as the ability to operate in multiple complex scenarios - from driving on multi-lane roads, to navigating intersections while obeying precedence rules, to overtaking stopped vehicles. In this paper, we argue that the standard, hybrid architecture in which a behavior-based reactive layer controls actuation, while deliberative path planning provides

intermediate waypoints in the configuration space, fails to cover the capabilities required by the Urban Grand Challenge. Instead, what is required is a system capable of switching both control strategy and sensing priorities based on the perceived state of the robot. For example, the derived perception needs and appropriate control regimes of a robot at an intersection differ greatly from that of one driving at high speeds on a highway.

In this paper we describe an extension of hybrid automata models (e.g. [3]) for the purpose of situational awareness, and include a description of the installation of this approach to Sting Racing's Urban Grand Challenge robot. The overall arrangement of the control architecture will be divided into a planning block and a control block. At the highest level in the control block is an arbiter which combines the outputs of a multitude of behaviors. We chose to let the arbiter vote on the control output (desired steering angle and desired translational velocity) in the fashion based on the DAMN architecture [4]. Each behavior's output is assigned a weight by the arbiter before all outputs are combined and the steering and velocity command are executed. These weights are prescribed by the arbiter based on its current action, and the current action is an output of the planning block.

To summarize: The contribution of this paper is the Sting Racing modular architecture, as well as a detailed description of how it is used for solving the Urban Challenge. Moreover, a number of experimental test results are given, illustrating how the requirements for urban driving can be mapped onto a finite set of distinct modes-of-operation.

2 Coverage of the Required Capabilities

2.1 The Urban Grand Challenge

The two previous grand challenges organized by DARPA emphasized autonomy and robust operation in cross-country off-road environments [1, 2]. The environment was largely assumed to be static, with few or no moving objects. If other vehicles were encountered, one of them would be paused while the other vehicle continued its route towards the goal. The desired route to be followed was defined by a relatively dense list of waypoints rather than by perceptual features (i.e., roads) in the environment. The objectives of the previous challenges therefore focused on endurance, robustness to local sensory dropouts, and trajectory following within a corridor defined by waypoints, with local deviations to accommodate static obstacles. As witnessed by the number of finishers in the last Grand Challenge, the lower level sensing, control, and vehicle reliability required to drive between waypoints while avoiding sparse static obstacles are now largely solved problems [1, 2, 5].

The Urban Challenge (UC) poses a number of very different higher-level cognition challenges for the design of a system. First of all, navigation must be performed with respect to locally defined structures such as lane-markings, stop lines, etc. Driving is required to perform lane keeping in situations with widely spaced waypoints. The vehicle is required to come to a stop at a stop line. Navigation must be performed relative to these markings, not with respect to global coordinate frames as defined by GPS. In addition, global position estimation methods such as GPS might have limited availability. In short, instead of being told where it is relative to a detailed path to follow, the vehicle must reason as to its location and the associated appropriate control responses.

In contrast to earlier Grand Challenges, the vehicle is required to show situational awareness of dynamic as well as stationary vehicles and structures within changing areas around the vehicle. Situational awareness is required to allow the vehicle to plan its actions in response to the context. For example, if a slow moving vehicle is in front of the car, and the lane marking is a double yellow line, then following at an appropriate distance is the correct action. But the same situation with a slow moving vehicle alone in a lane with dashed lane dividers might allow an overtake maneuver, provided there are no vehicles in front and there are no oncoming vehicles with the segment needed for passage. For the overtake maneuver, there is a need for long-range detection of vehicles in other lanes to ensure safe passage. At intersections there is a need to detect vehicles that are waiting or approaching, which calls for long-range lateral coverage.

Our architecture to address these challenges is based upon the assumption that the required capabil-

ities can be broken down into a small (enumerable) number of operating modes, each mode consists of a collection of parameterized behaviors and a behavior arbitration mechanism. This modularization makes design and development tractable, as well as provides a mechanism for structured, incremental testing. Traffic laws and conventions structure the world dynamics into this small set, though robust behavior within an operating mode requires being robust with respect to large variety of possibilities *relevant to that mode*.

2.2 Driving in Urban Environments

The novel, modular architecture employed by Team Sting was arrived at by observing that the sensing, planning, and control capabilities needed to drive down a road are fundamentally different than those needed to park the vehicle. As such, rather than choosing a single, sense-plan-act solution in which a unified planner produces references for a trajectory tracker, a number of distinctly different environments were identified, based on the unique challenges posed by the Urban Grand Challenge. In fact, the operation of the system is modeled as a finite set of "modes of operation" that each capture a nominal situation to be handled by the vehicle. Within each of these modes of operation, a dedicated set of controllers is used to handle both the nominal situation and unexpected variations.

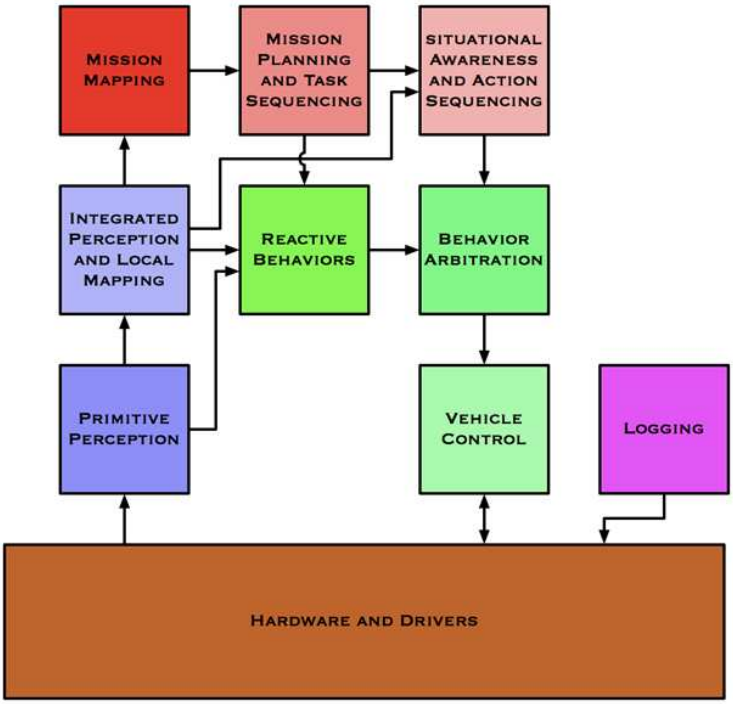

Figure 2: Sting Racing Software Architecture

Each mode of operation is represented as a hybrid automaton, as seen in Figure 3. An automaton is composed of states and transitions among the states. For example, consider a state Follow-Lanes which represents the behavior of driving along lanes on a road while obeying speed limits and recognizing the speed of nearby traffic. This state would have transitions to another state, Handle-Intersection, where the transition occurs based on a combination of the distance from the robot to the stop point (from GPS information) and other visual cues, such as the detection of a stop line.

In the left automaton in Figure 3, nodes at the highest level of abstraction are shown. These correspond to the high-level modes of operation Follow Lanes, Overtake Static Obstacle, U-Turn, Handle Intersection, Park, and Unpark. Based on the specifications of the Urban Grand Challenge mission, these are the six modes of operation that are selected by Team Sting as the minimal set of modes needed to successfully

complete the mission. An important additional benefit, however, associated with the modular design is that new modes can be added, whenever the need arises further down the development cycle. Indeed, encapsulation and ease of extendibility are a key features of this software architecture and important to the short development cycles required for the Urban Grand Challenge.

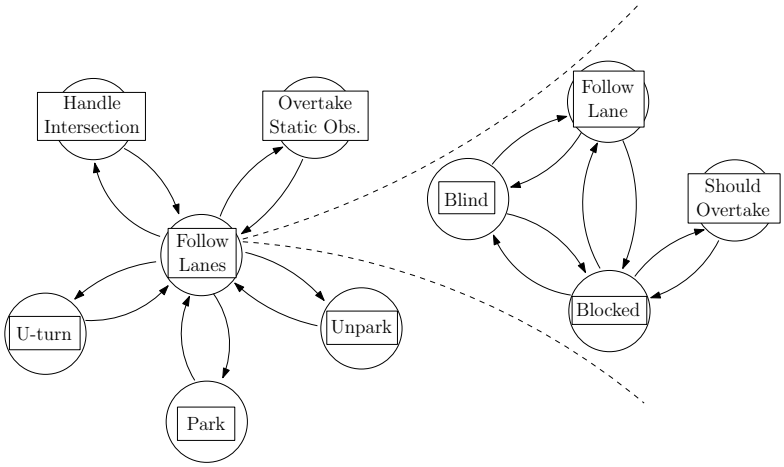

Figure 3: Modes of operation modeled as a Nested Hybrid Automaton

The transitions between modes are guarded in the sense that environmental conditions trigger the transitions. As such, the situational awareness component of the novel Team Sting architecture can be thought of as the guard conditions (or transition conditions associated with the different edges in Figure 3), and the cognition component is encoded by the underlying state machine dynamics. And, for the sake of easy reference, each of the modes of operation are roughly described. A more detailed description is given to the Follow Lanes mode of operation. The remaining modes are discussed only cursively.

Follow Lanes

In the right figure of Figure 3, the Follow Lanes mode of operation is given. Here, each node corresponds to a particular set of behavioral controllers as well as to a particular arbitration mechanism. In fact, the modes that make up this high-level mode are

- Follow Lane: This mode corresponds to a set of behaviors that use visual perception to track lane striping and that use fused LIDAR and radar data to track nearby traffic, thereby adjusting speed and avoiding collisions.
- Overtake: Typically, transitions between the states are based on environmental or perceptual information. Overtake mode, however, is a state-based signal to switch from the larger Follow-Lanes model into the Overtake-Static-Obstacle model. This mode corresponds to a command to the behavior arbiter to stand still until the Overtake-Static-Obstacle model is enabled.
- Blocked: This mode uses the same behavior arbiter as the Follow-Lane mode. However, it has a transition based primarily on time. If this mode is active for a parameterized amount of time, it transitions to Overtake, which then signals the robot to overtake a static obstacle.
- Blind: This mode corresponds to a behavior arbiter that uses GPS and laser information to drive in the lane because the lane detector has failed in some way. Fused laser and radar data is used to avoid collisions and maintain speed in the lane.

Overtake Static Obstacle

This mode of operation governs the control of vehicle during a maneuver to overtake a static obstacle. The four modes comprising this high-level mode include the following: Init-State, Change-Left, Change-Right, and Done. Init-State establishes a fixed coordinate frame to govern the transitions through the subsequent modes. Change-Left and Change-Right correspond to the tracking of lane markings one lane to the left or right, respectively of the current lane being tracked. That is, the lane change maneuver is achieved primarily by shifting visual perceptual attention on the road. The lane change commands are triggered based on a combination of distance travel (relative to the coordinate frame established in Init-State) and the presence/absence of obstacles from fused LIDAR/radar data.

U-Turn

The states of this high-level mode encode a mapping from vehicle orientation (i.e., position and heading) to output primitive (e.g., drive forward, hard left; drive in reverse, hard right). This mapping stabilizes the vehicle (in the presence of imperfect vehicle control) to the desired final position and heading.

Handle Intersection

Intersections are handled by cycling through a string of simple modes: Approach, Find Queue Position, Wait For Turn, Go, and Done. Approach smoothly brings the vehicle to a stop at the stop line based on visual perception of the lane markings, while queueing behind other vehicles. Once stopped, Find Queue Position establishes the robot's precedence order based on fused LIDAR/radar data. Wait For Turn checks the interior of the intersection for traversals by the adjacent vehicles with higher precedence. Once its turn has come, Go is triggered, and the robot traverses a path through the intersection towards the entry point back onto the lane segment.

Park and Unpark

This pair of high-level modes guides the robot through RNDF zones and in and out of parking spots. These states govern the path of the robot (e.g., to drive it to a parking spot) given the constraints of Ackermann steering and encode the rules of driving in the unstructured RNDF zones (e.g., pass to the right of oncoming traffic).

3 A Modular, Hybrid Architecture

Figure 4 provides a more detailed view of the processes comprising the Sting software architecture shown in Figure 2. The Planning Group consists of the Mission Mapping, Mission Planning, and Situational Awareness and Action Sequencing blocks. Similarly, the Control Group consists of the Reactive Behaviors, Behavior Arbitration, and Vehicle Control blocks. This section describes the operation of these blocks in detail and outlines their functionality with respect to the key software and architectural challenges associated with the Urban Challenge.

3.1 Primitive and Integrated Perception

In order for the vehicle to estimate its own state as well as relevant environmental conditions, sensing and estimation are needed at different levels of abstraction, frequency, and fidelity. The primitive perception part of the software architecture collects and processes single scans/images/measurements from individual sensory sources. In order to arrive at a comprehensive list of perception primitives, Team Sting relied on the mission scenarios to be expected in the Urban Challenge. In particular, as safety is going to be a critically important issue, static and dynamic obstacle detection are needed as well as scan matching algorithms for obstacle classification. The dynamic obstacle detection is necessary also from a traffic management point-of-view. Moreover, as the vehicle will be operating in environments in which GPS signals may or may not be readily available, an integrated GPS/IMU primitive is needed in combination

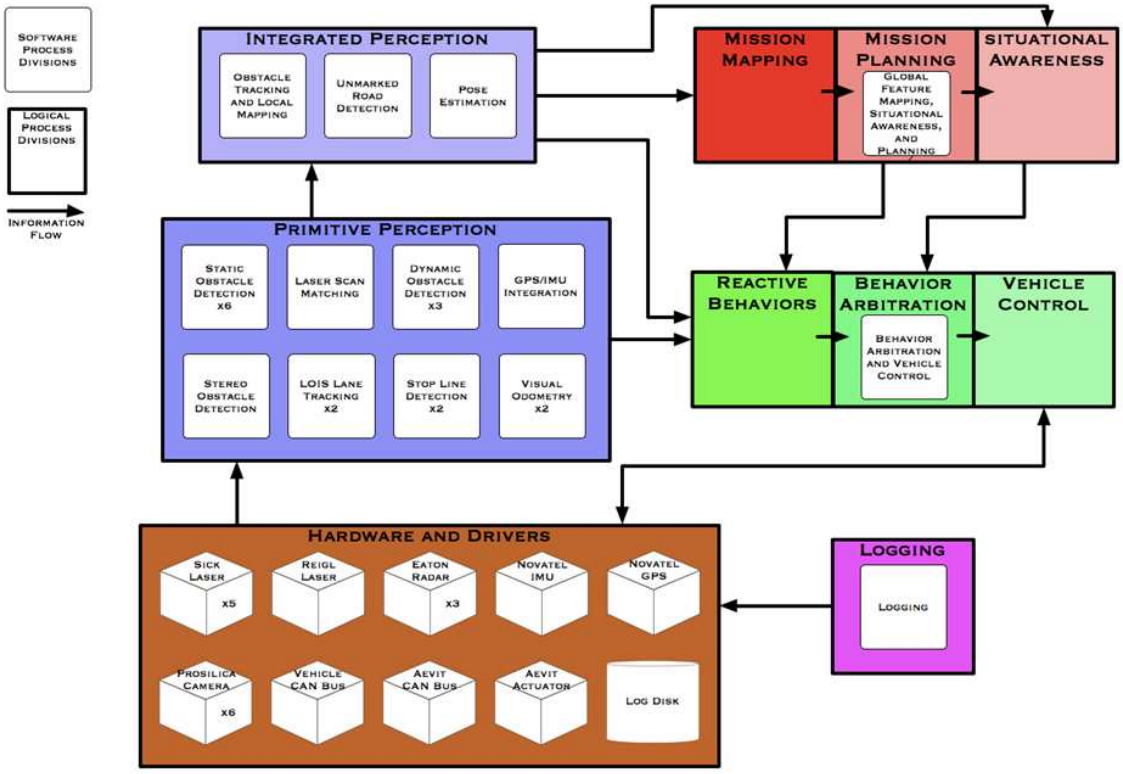

Figure 4: Software processes used within the Sting software architecture and their relationship to the conceptual architecture presented previously. Smaller boxes represent divisions of labor between software processes (e.g. Static Obstacle Detection). Larger boxes represent divisions of labor within the conceptual architecture (e.g. Primitive Perception).

with a vision-based method for local pose estimation, i.e., visual odometry. Finally, lane and stop line tracking capabilities will also be needed in order to place the vehicle correctly in its local environment. Note that these primitives are not providing all of the perceptual skills needed, but the remaining, more complex perception tasks will be handled at the integrated perception level.

To summarize, the derived set of required primitive perception capabilities are:

- Static Obstacle Detection
- Laser Scan Matching
- Dynamic Obstacle Detection
- GPS/IMU Integration
- Stereo Obstacle Detection
- Lane and Stop Line Tracking
- Visual Odometry

The Integrated Perception functional group deals with sensor fusion, in which the data from the primitive perception group is used in an integrated fashion to achieve higher-level perceptual tasks. These tasks are Pose Estimation, Unmarked Road Detection, and Obstacle Tracking and Local Mapping. Two of the key problems associated with the Urban Challenge are driving on a road network without detailed, high accuracy information about the road location, and detecting and tracking other moving entities in the world.

3.2 Planning and Control

Planning and Control tasks span a number of processes in our software architecture, due to its multilayered hybrid continuous/discrete control strategy. Figure 5 shows the structure of these processes. At the top of this hierarchical structure is the Mission Level Mapping block. At the beginning of a mission, a map is produced that consists of a graph structure based on the provided RNDF. As the mission progresses, this graph structure is augmented with information about the routes it represents. Experiences of traffic congestion, dangerous obstacles, and impassible lanes are noted in the graph for future reference.

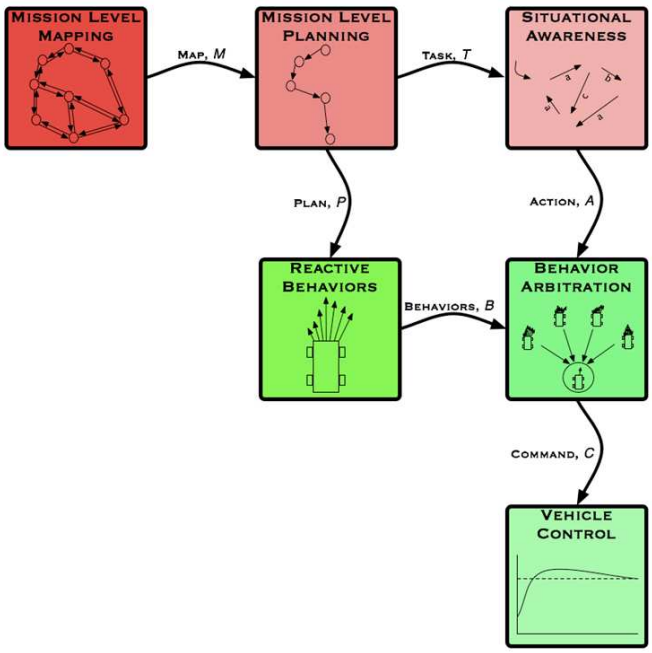

Figure 5: A detailed view of the planning and control architecture, presented as part of the full architecture. Arrows indicating information flow are labeled with the type of information communicated.

The map produced by the Mission Level Mapping block is passed on to the Mission Level Planning block. This block incorporates the MDF and plans a route through the graph-based map to achieve the specified checkpoints. Information stored in the map is used to weight edges of the graph, allowing the planner to find a route that optimizes the expected time-to-complete, rather than simply distance. The plan is passed on to the Reactive Behaviors block. A representation of the robot's current task (e.g., PARK, UNPARK, DRIVE TO CHECKPOINT) is passed on to the Situational Awareness block. The Situational Awareness block implements a nested hybrid automaton (NHA), which is driven by the robot's current task and perception. The NHA implements an a priori representation of the structure of the robot's environment and task. The nested structure allows for asynchronous transitions at different levels of functionality. Each state in the NHA maps, in a many-to-one fashion, to actions such as FOLLOW-LANE, DRIVE-TO-POINT, and STAND-STILL. Selected actions are passed on to the Behavior Arbitration block. The Behavior Arbitration block maps an action to a set of weights (which may be zero) which is applied to the output of the behaviors provided by the Reactive Behavior block. Each behavior provides a set of votes over discrete values of curvature within the vehicle's drive capabilities, and provides a maximum allowable velocity for each evaluated curvature. The Behavior Arbitration block chooses a commanded steering angle according to the input provided by the behaviors and their respective weights, and a commanded velocity according to the minimum of the maximum allowable velocities provided by the behaviors for the selected curvature. This commanded curvature and velocity is passed on to the Vehicle Control block, which runs in a tight loop, controlling the actuation of the vehicle to achieve the commanded set points.

3.3 Atypical and Unexpected Situations

Within the Team Sting planning and control architecture, atypical and unexpected events and situations are addressed in two different ways. First, the transitions between states at a given level of the nested hybrid automaton are asynchronous with respect to the state/transitions of lower levels. This reduces the possibility for deadlock. Moreover, by using the hybrid automaton structure, existing and well known tools for analyzing the design (e.g., assessing the reachability of bad states, finding the possibility of deadlock) are readily available. By dividing the complexity of the larger Situational Awareness problem into separable components - the various high-level modes described below - the standard software principles of modularity and encapsulation are employed. This planning architecture thus lends itself to quickly determining the fault in the existing design as well as allowing for a revision of that component with minimal impact on other components.

The second major way for handling unexpected situations comes from the use of a behavior-based arbitration mechanism based on the DAMN architecture [4], as shown at the Arbitration Level in Figure 5. A number of active behaviors express appropriate commands for their respective interests (such as avoiding obstacles or following the lane) by voting for or against values in a set of steering angles. Because each behavior can express multiple preferences across the set of steering angles, the behavior arbiter is less likely to arrive at a local minima or an oscillatory state. For example, a behavior dedicated to avoiding obstacles can express that turning either left or right is appropriate for avoiding an obstacle in front of the vehicle, and let the arbiter evaluate the other behaviors before deciding to turn left or right, as shown in [5].

4 Nested Hybrid Automata

A hybrid automaton is a model that captures both the continuous and the discrete aspects of a dynamic system. In particular, a continuous state (typically the position and velocities of the car) evolves concurrently with a discrete state (the current mode of operation), and we follow the standard definition of a hybrid automaton (e.g. [3]) as a tuple $HA = (Q, X, E, U, f, G, R, x_0, q_0)$, where

- Q – the set of discrete states
- X – the continuous state space
- E – the set of events that can trigger transitions between different discrete states
- U – the input space
- $f : Q \times X \times U \rightarrow TX$ – encodes the evolution of the continuous state x as $\dot{x} = f(q, x, u)$
- $G : Q \times Q \times X \times (E \cup \epsilon) \rightarrow \{0, 1\}$ – gives the guard conditions that triggers transitions between discrete states. In particular, a transition occurs between q to q' if the continuous state is x, the external event is $e \in E$ or possible the "empty event" ϵ (no event happened) if $G(q, q', x, e) = 1$
- $R : Q \times Q \times X \times E \rightarrow X$ – encodes the reset condition, in that the continuous state is reset to $R(q, q', x, e)$ when the system transitions from q to q' at continuous state x under event e
- $q_0 \in Q$ – initial discrete state
- $x_0 \in X$ – initial continuous state

An example of this is seen in Figure 6. In the figure, the discrete state starts out at q_0 and the continuous state evolves from x_0 according to $\dot{x} = f(q_0, x, u)$ until time τ. At that time, the continuous state is at $x(\tau_-)$ and event e happens. The guard condition $G(q_0, q', x(\tau_-), e)$ becomes 1 and the discrete state transitions from q_0 to q'. The continuous state is reset to $x(\tau_+) = R(q_0, q', x(\tau_-), e)$, from which it evolves as $\dot{x} = f(q', x, u)$. Different definitions of such hybrid dynamics have been given, but they all share these basic building blocks in terms of continuous and discrete dynamics, guards, and resets.

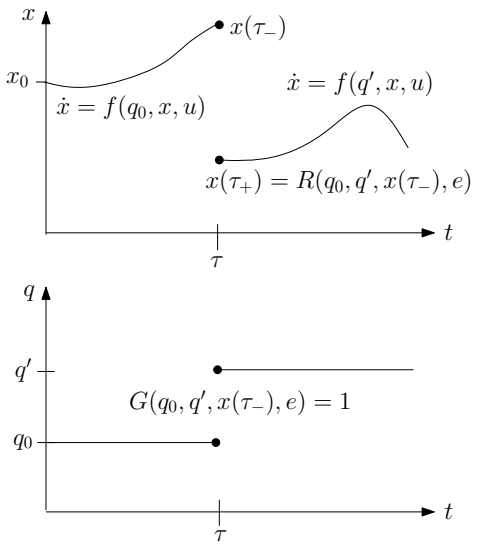

Figure 6: Evolution of a hybrid automaton.

Now, the modular architecture proposed in this paper can certainly be cast as a hybrid automaton, albeit an overly complex one. Instead, we have designed a Nested Hybrid Automaton (NHA) that operates at different levels of abstraction.

At the highest level is HA_0 composed of the operator assigned states

- `operator-run`
- `operator-pause`
- `operator-stand-by`

The only non-trivial of these states is `operator-run` that corresponds to the operator/user putting the vehicle in an autonomous run-mode. Formally, we define the top level of a NHA as a standard hybrid automaton $HA_0 = (Q^0, X^0, E^0, U^0, f^0, G^0, R^0, x_0^0, q_0^0)$, where the superscript 0 denotes level 0. The way in which the nesting works is that each discrete state at level $k-1$ in the hierarchy induces its own hybrid automaton at level k, as $HA_k(q^{k-1}) = (Q^k(q^{k-1}), X^k(q^{k-1}), E^k(q^{k-1}), U^k(q^{k-1}), f^k(q^{k-1}), G^k(q^{k-1}), R^k(q^{k-1}), x_0^k(q^{k-1}), q_0^k(q^{k-1}))$, $k = 1, 2, \ldots$ The interpretation here is that the hybrid automaton evolves as a regular automaton at each level. However, as a transition occurs higher up in the hierarchy, a new automaton is instantiated at the lower level, initiated at its corresponding initial condition. Moreover, events at higher levels can be triggered by transitions occurring at lower levels.

For instance, as seen in Figure 3 the hybrid automaton HA_1(`operator-run`) that corresponds to the `operator-run` mode at level 0 has the discrete states

- `follow-lanes`
- `handle-intersection`
- `overtake-static-obstacle`
- `execute-u-turn`
- `park`
- `unpark`

Each of these nodes in turn contain their own hybrid automata. The `follow-lanes` modes is a hybrid automaton HA_2(`follow-lanes`) whose discrete states correspond directly to an action in the sense that they define an arbiter selection. In other words, no lower automata are defined here.

As an example of a discrete state that corresponds to a further nested structure is the `handle-intersection` mode in that HA_3(`handle-intersection`) consists of the following discrete states

- `approach-intersection`
- `establish-precedence`
- `wait-for-precedence`
- `wait-for-oncoming-traffic`
- `traverse-intersection`

If one wants to dig even deeper, HA_4(`traverse-intersection`) in turn consists of the following discrete states

- `go`
- `resest-lane-tracker`
- `follow-points`
- `request-lane-tracker-lane-change`
- `follow-lanes-in-intersection`

Rather than enumerate all of these, we show, in Figure 7 a screen shot of the different modes engaged.

5 Testing

5.1 Testing Methodology

Due to the complexity and integrated nature of the system, it is vitally important that a testing strategy is devised that allows the designers to test different aspects of the system, the validity of design modifications and additions, as well as the entire, integrated system. In order to accommodate these requirements, Team Sting's testing strategy is based on a combination of carefully engineered unit tests, integrated mission and scenario-level tests, open-loop tests in which no autonomous control of the vehicle is allowed, and simulated tests in synthetic environments.

Unit testing

Unit tests are tests designed to capture a targeted, isolated part of the system. Such tests have been conducted extensively at the early stages of development by Team Sting and they are important for capturing the basic behavior of the system from both sensing, actuation, and planning points-of-view.

Integrated system testing

One aspect of the Urban Challenge that sets it apart from previous Grand Challenges is the fact the system is forced to switch between many different modes of operation in response to environmental conditions. The high-level modes of operation (Follow Lanes, Overtake Static Obstacle, U-Turn, Handle Intersection, Park, and Unpark) identified by Team Sting as critical to a successful completion of the race are. These high-level modes of operation must be tested in an integrated fashion, i.e. with all low-level functionality engaged, and all transitions enabled. That is, unit tests are used to test individual perceptual and behavioral components while integrated tests are those that test the situational awareness modes that depend on these lower-level components. The hierarchical layering of the software system lends itself to translation into testing strategies at different levels of abstraction and integration.

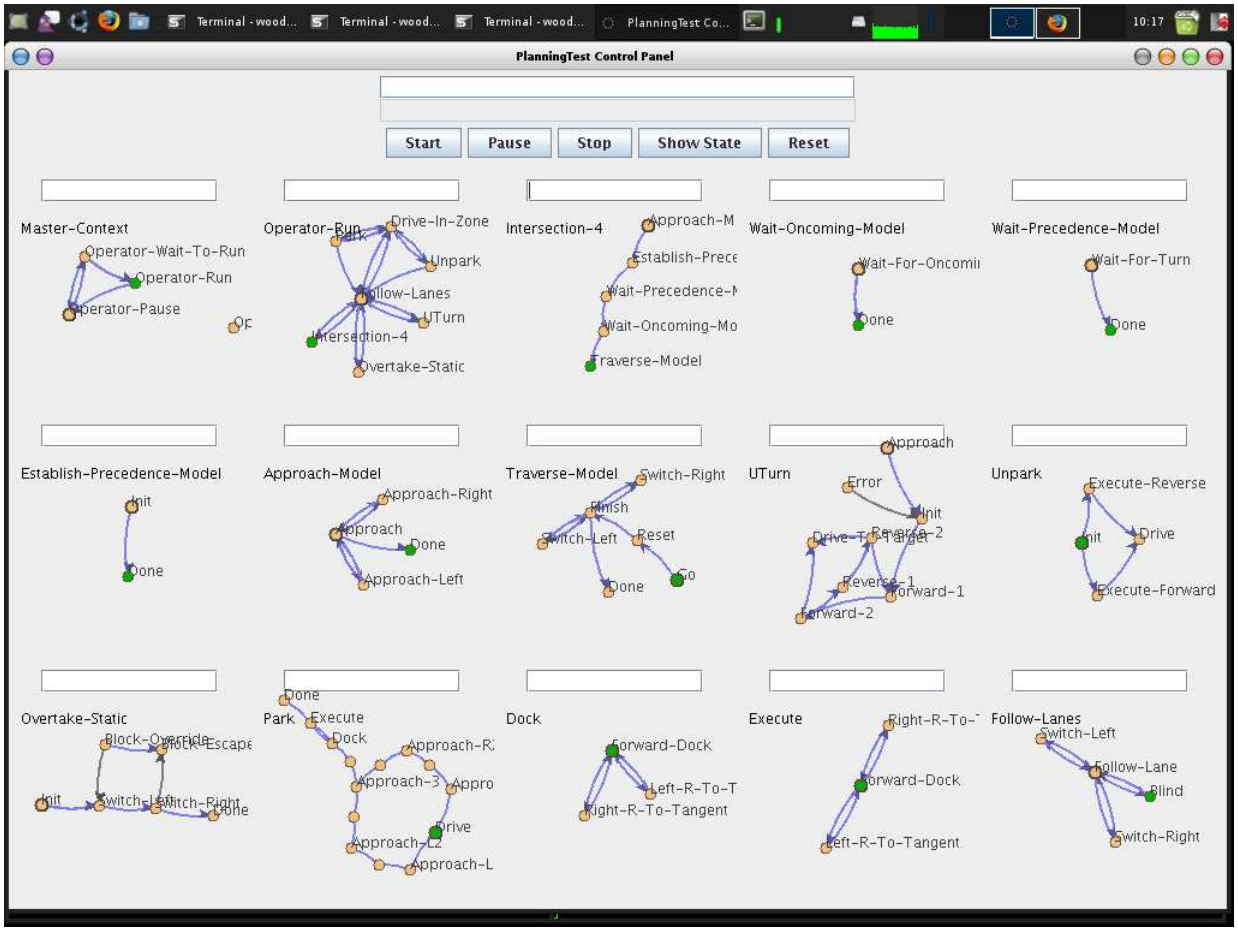

Figure 7: Screen shot from the execution of the Sting Racing software architecture, involving the full functionality needed to cover the requirements for the Urban Challenge.

Open loop testing in real urban environments

As safety is a key issue that must be addressed when testing the system, Team Sting is conducting so-called Open Loop Tests, in which the vehicle is deployed in an actual, urban environment with the software system running. The only difference is that the proposed control signals are not allowed to actually control the vehicle. Instead the vehicle is controlled by a human driver. This mode of operation has proved to be very useful for evaluating the perception modules in truly complex environments. Moreover, rough, qualitative estimates of the validity of the proposed control signals have been obtained in this manner. In the future, Team Sting will continue to employ this strategy in combination with a formal assessment of the proposed control signals as compared to that of the behavior of a human driver.

5.2 Experimental Results

Some examples are given in the following figures (Figures 8 - 10) of Sting Racing's entry to the DARPA Urban Challenge.

(a) (b)

Figure 8: The figure shows the operation of the Sting Racing vehcile during an overtake maneuver in which the nested hybrid automaton is going through a number of modes, including slowing down to the car ahead, changing lanes, and overtaking.

(a) (b)

Figure 9: An example is given in which the vehicle is executing a parking maneuver.

(a) (b)

Figure 10: An intersection is traversed by first establishing the correct precedence and then waiting for precedence, as part of the traverse intersection mode of operation.

6 Conclusions

In this paper, we discuss the modular software and control architecture employed by Sting Racing, the joint Georgia Tech, SAIC entry into the DARPA Urban Challenge. In particular, we discuss methods

for switching between different modes of operation by employing a nested hybrid automata formalism. We discuss how to map the requirements of the Urban Challenge onto this formalism, and give some preliminary, experimental results showcasing the operation of the software system.

References

[1] Multiple authors, "Special Issue on the DARPA Grand Challenge 2005 (Part 1)," *Journal of Field Robotics*, 23(8), 2006.

[2] Multiple authors, "Special Issue on the DARPA Grand Challenge 2005 (Part 2)," *Journal of Field Robotics*, 23(9), 2006.

[3] T.A. Henzinger. The theory of Hybrid Automata. *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, IEEE Computer Society Press, pp. 278-292, 1996.

[4] Julio K. Rosenblatt, "DAMN: a distributed architecture for mobile navigation". In *Journal of Experimental & Theoretical Artificial Intelligence*, 9:2, p. 339-360, 1997.

[5] J. Sun, T. Mehta, D. Wooden, M. Powers, J. Regh, T. Balch, and M. Egerstedt. Learning from Examples in Unstructured, Outdoor Environments. *Journal of Field Robotics*, Vol 23, No. 11/12, pp. 1019-1036, Nov/Dec. 2006.