A Reasoning Framework for Autonomous Urban Driving

Dave Ferguson, Christopher Baker, Maxim Likhachev, and John Dolan

***Abstract*— Urban driving is a demanding task for autonomous vehicles as it requires the development and integration of several challenging capabilities, including high-level route planning, interaction with other vehicles, complex maneuvers, and ultra-reliability. In this paper, we present a reasoning framework for an autonomous vehicle navigating through urban environments. Our approach combines route-level planning, context-sensitive local decision making, and sophisticated motion planning to produce safe, intelligent actions for the vehicle. We provide examples from an implementation on an autonomous passenger vehicle that has driven over** 3000 **autonomous kilometers and competed in, and won, the Urban Challenge.**

I. INTRODUCTION

Autonomous passenger vehicles present an extremely promising solution to traffic accidents caused by driver error. However, developing systems that are sophisticated enough and reliable enough to operate in everyday driving scenarios is a huge challenge. As a result, up until very recently, autonomous vehicle technology has been limited to either off-road, unstructured environments where complex interaction with other vehicles is non-existent [1], [2], [3], [4], [5], [6], or very simple on-road maneuvers such as highway-based lane following [7]. However, to live up to their enormous potential, such systems have to make the transition to unrestricted on-road driving.

In November 2007 the United States Defense Advanced Research Projects Agency (DARPA) held a competition for autonomous vehicles intended to accelerate this transition. Dubbed 'The Urban Challenge', the competition consisted of a series of navigation missions through an urban environment. Each vehicle had to navigate through single and multi-lane roads, traffic circles and intersections, open areas and unpaved sections, and cope with road blockages and complex parking tasks. They had to do this for roughly 60 miles, all in the presence of other human-driven and autonomous vehicles, and all while abiding by speed limits and California driving rules.

This challenge required significant advances over the state of the art in autonomous vehicle technology. In this paper, we describe the reasoning framework developed for Carnegie Mellon University's winning entry into the Urban Challenge, "Boss". This framework enabled Boss to plan fast routes through the urban road network to complete its missions; interact safely and intelligently with obstacles and other vehicles on roads, at intersections, and in parking lots; and perform sophisticated maneuvers to complete complex parking tasks.

Fig. 1. "Boss": Tartan Racing's winning entry in the Urban Challenge.

We first describe in more detail the Urban Challenge and the required autonomous vehicle capabilities. We then present Boss' reasoning architecture and describe each of the major components in this architecture. We conclude with discussion and future extensions.

II. THE URBAN CHALLENGE

The DARPA Urban Challenge was an autonomous vehicle race through roughly 60 miles of urban roads, intersections, and parking lots. 11 vehicles were selected for the final event on November 3, 2007, and 6 completed the course.

Twenty four hours before the race, DARPA provided each vehicle with a rough road map of the environment, known as a Route Network Definition File (RNDF), that provided the location of intersections, parking lots (known as zones), and the connectivity of the roads. The RNDF also provided the positions of key locations known as checkpoints that were used for specifying the set of ordered goal locations in each navigation mission, and waypoints along each road that provided some road shape information, all given in lat/long coordinates. However, the density of these waypoints was not enough to provide accurate road shape information for blind waypoint following. DARPA also provided publicly-available overhead imagery of the area that could be used for improving the road map; however, they made no guarantees

---

This work would not have been possible without the dedicated efforts of the Tartan Racing team and the generous support of our sponsors including General Motors, Caterpillar, and Continental. This work was further supported by DARPA under contract HR0011-06-C-0142.

D. Ferguson is with Intel Research Pittsburgh and Carnegie Mellon University, Pittsburgh, PA, USA. Email: `dave.ferguson@intel.com`

C. Baker and J. Dolan are with Carnegie Mellon University, Pittsburgh, PA, USA. Email: `{cbaker,jdolan}@andrew.cmu.edu`

M. Likhachev is with The University of Pennsylvania, Philadelphia, PA, USA. Email: `maximl@seas.upenn.edu`

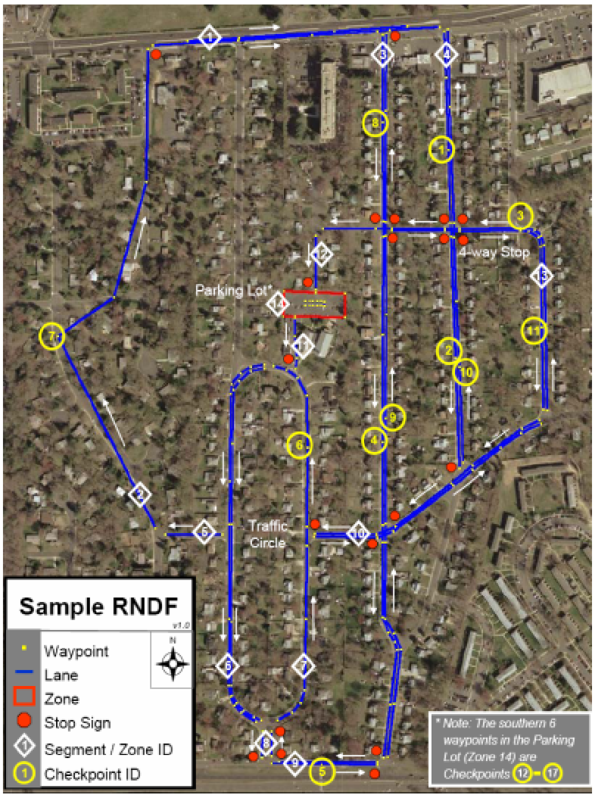

Fig. 2. A sample Route Network Definition File (RNDF) provided by DARPA. The yellow points represent the waypoints provided and the blue line segments connect adjacent waypoints. The yellow circled numbers represent checkpoints.

as to the accuracy of this imagery. Fig. 2 provides a sample RNDF file and imagery provided by DARPA.

On race day, DARPA gave each vehicle a mission file consisting of a set of ordered checkpoints from the RNDF that had to be visited. The vehicle had five minutes to process this file and begin its mission. After completing its mission, the vehicle would be given a new mission file, and so on until all the missions were complete.

III. SYSTEM ARCHITECTURE

The completion of the Urban Challenge required extremely reliable urban driving capability. In Boss, this capability was achieved through a software system architecture that was decomposed into four major blocks (see Fig. 3).

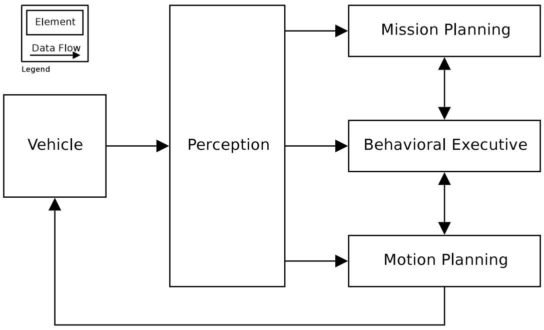

Fig. 3. Boss' Software System Architecture

A. *Perception*

The Perception component provides a composite picture of the world to the rest of the system by interfacing to sensors, processing the raw sensor data, and fusing the multiple streams together into a collection of semantically-rich data elements. The most important of these elements are:

- **Vehicle State**, globally-referenced position, attitude and speed for Boss;
- **Road World Model**, globally-referenced geometric information about the roads, parking zones, and intersections in the world;
- **Moving Obstacle Set**, an estimation of other vehicles in the vicinity of Boss;
- **Static Obstacle Map**, a grid representation of free, dangerous, and lethal space in the world; and
- **Road Blockages**, an estimation of clearly impassable road sections.

B. *Mission Planning*

The Mission Planning component computes the fastest route through the road network to reach the next checkpoint in the mission. The mission planner reasons about the optimal path to a particular checkpoint much like a human would plan a route from their current position to a desired destination such as a grocery store or gas station. Routes are evaluated based on knowledge of road blockages, speed limits, and the nominal time required to make special maneuvers such as lane changes or u-turns.

C. *Behavioral Executive*

The Behavioral Executive combines the strategic information provided by Mission Planning with local traffic and obstacle information provided by Perception and generates a sequence of local pose goals for the Motion Planning component. These local goals are grouped into three abstract behavioral contexts, each of which requires the system to behave according to a specific group of rules:

- **Lane Driving**, in which the system traverses a road of one or more lanes while maintaining safe vehicle separation and adhering to rules governing passing maneuvers and stop-and-go traffic;
- **Intersection Handling**, requiring the determination of precedence among stopped vehicles and safe merging into or across moving traffic at an intersection; and
- **Zone Maneuvering**, in which the system maneuvers through an unstructured obstacle or parking zone.

These contexts are primarily determined by the location of the system and the current objectives set forth by the Mission Planner and can be triggered by status from the Motion Planner and by the overall health of the various vehicle and software subsystems.

D. *Motion Planning*

The Motion Planning component takes the local pose goal from the Behavioral Executive and generates a trajectory that will safely drive Boss towards this goal. Two broad contexts for motion planning exist: on-road driving and unstructured driving. In each context, the motion planner generates a set of candidate trajectories based on constraints from the Behavioral Executive and selects the best collision-free trajectory from this set to execute.

Together, the mission, behavioral, and motion planning components perform the reasoning behind Boss' every move. Each of these components and their relationships to one another are described further in the following sections.

IV. MISSION PLANNING

The mission planner is responsible for generating a cost-to-goal value for every waypoint in the world. In our setting, this value can be thought of as the minimum time required to reach the goal. A path to the goal from any point in the world can then easily be extracted by selecting, from any given point, the waypoint in the vicinity that minimizes the sum of this cost-to-goal value plus the time taken to reach the waypoint, then repeating this process to step through waypoints until the goal is reached.

To generate mission plans, the data provided in the RNDF is used to create a graph that encodes the connectivity of the environment. Each waypoint in the RNDF becomes a node in this graph, and directional edges (representing lanes) are inserted between a given waypoint and all other waypoints that it can reach. These edges are also assigned time costs based on a combination of several factors, including the distance of the edge, the speed limit, and the complexity of the corresponding area of the environment. This graph is searched to compute cost-to-goal values for each position in the graph given a desired goal position, such as the first checkpoint in the mission. In addition to providing the Behavioral Executive more information to reason about, computing cost-to-goal values for every position is useful because it allows the navigation system to behave correctly should the vehicle be unable to perfectly execute the original path (e.g. if a particular intersection is passed through by mistake, we can immediately extract the best path from the vehicle's current position).

As the vehicle navigates through the environment, the mission planner updates its graph to incorporate newly-observed information, such as road blockages or previously-unknown intersections or roads. Each time a change is observed, the mission planner re-generates new cost-to-goal values. Because the size of the graph is relatively small, this replanning can be performed extremely quickly, allowing for immediate response to environmental changes.

V. BEHAVIORAL EXECUTIVE

The Behavioral Executive is responsible for using the cost-to-goal value function from the Mission Planner and local perceptual information to generate local tasks for the Motion Planner. It is also responsible for the system's adherence to various rules of the road, especially those concerning structured interactions with other traffic and road blockages, and for detection of and recovery from anomalous situations. The local tasks take the form of simple, discrete motion goals to be executed by the Motion Planner, such as driving along a road to a specific point or maneuvering to a specific parking spot. The issuance of these goals is predicated on safety and traffic concerns such as precedence among vehicles stopped at an intersection and windows-of-opportunity in yield situations such as at T-intersections. In the case of driving along a road, periodic lane tracking and speed government commands are used to implement behaviors such as safety gap maintenance, passing maneuvers and queueing in stop-and-go traffic.

The design of the Behavioral Executive is built upon the identification of a set of driving contexts, each of which requires the vehicle to focus on a specific collection of environmental features. At the macroscopic level, the three relevant contexts are on-road, at-intersection, and in-zone. Their corresponding behaviors are respectively Lane Driving, Intersection Handling, and Zone Maneuvering. The first two are highly structured, both in geometry and road-rules, and are thus strongly reflected in the Behavioral Executive's architecture. The third, Zone Maneuvering, occurs in unstructured and largely unconstrained environments, including parking lots, jammed intersections and recovery situations where the only guiding rules are to avoid obstacles and achieve a specified pose. Due to the reduced structure in these areas, this driving context does not require complex reasoning at the Behavioral Executive level. Fig. 4 shows the primary functional elements within the Behavioral Executive, grouped by functional context and highlighting the data flow between them.

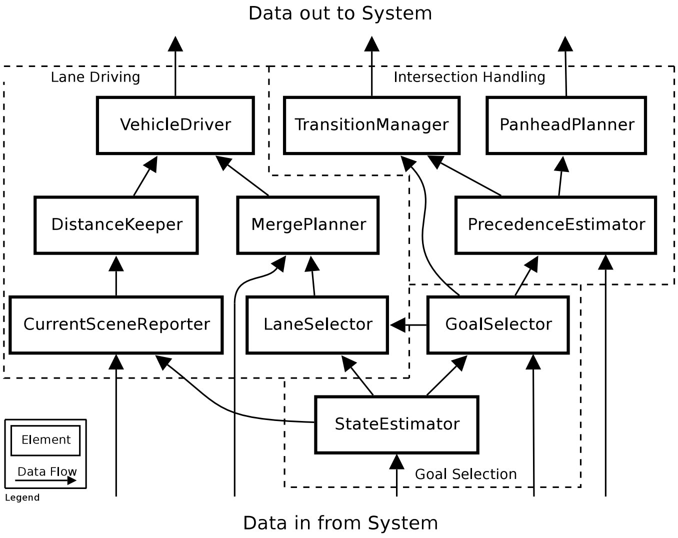

Fig. 4. Behavioral Executive software architecture, showing grouped dominant elements and data paths.

The system elements shown in Figure 4 perform the following functions:

Goal Selection

- `StateEstimator` combines the vehicle's current position with the world model to produce a discrete and semantically rich representation of the vehicle's logical position within the model.
- `GoalSelector` uses the current logical location as

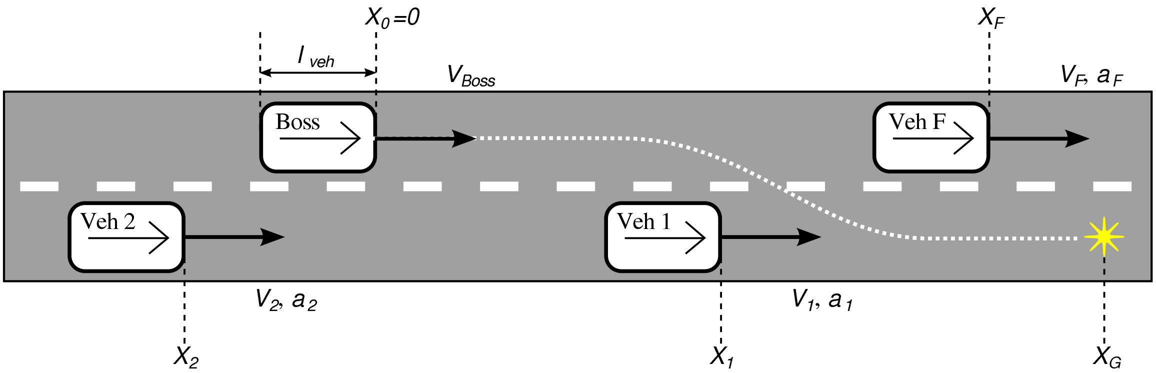

Fig. 5. An example lane driving scenario.

reported by StateEstimator to generate the next series of local goals for execution by the Motion Planner.

Lane Driving

- `CurrentSceneReporter` distills the list of known vehicles and lane blockages into a few discrete data elements, most notably the distance to and velocity of the nearest vehicle in front of Boss in the current lane.
- `LaneSelector` uses the surrounding traffic conditions to determine the optimal lane to be in at any instant and requests a merge into that lane if necessary.
- `MergePlanner` determines the feasibility of a merge into a lane proposed by LaneSelector and commands merge speeds and the lane-change maneuver when appropriate.
- `DistanceKeeper` uses the system's current speed and the lead vehicle information from `CurrentSceneReporter` to determine the necessary in-lane vehicle safety gaps and govern the vehicle's speed accordingly.
- `VehicleDriver` combines the outputs of `DistanceKeeper` and `MergePlanner` with its own internal rules to generate a periodic message to the Motion Planner that governs road driving parameters such as speed, acceleration and a desired tracking lane.

Intersection Handling

- `PrecedenceEstimator` uses the list of known other vehicles and their state information to determine precedence at an intersection.
- `PanheadPlanner` aims panning long-range radar and laser sensors to gain the most relevant information for intersection precedence decisions.
- `TransitionManager` manages the discrete-goal interface between the Behavioral Executive and the Motion Planner, using the goals from `GoalSelector` and the results from `PrecedenceEstimator` to determine when to transmit the next sequence of goals.

It is important to note that the final implementation includes many more functional elements and more convoluted data paths than listed above, but that these generally belong to auxiliary functionality such as diagnostic state reporting or else are an artifact of the system's rapid development cycle. These elements and connections were omitted to allow for a more clear and concise description of the system's functionality.

To illustrate the system's operation, we highlight two example scenarios in Sections V-A and V-B: lane driving and intersection handling.

A. Lane Driving

Fig. 5 shows a standard driving scenario along a road with two lanes in the same direction, separated by a dashed white line. In this scenario, Boss is driving in the left lane along with a collection of traffic vehicles and toward a goal in the right lane some distance x_G down the road. To satisfy this scenario, the system must:

- Maintain the maximum speed possible along the segment (to minimize travel time),
- Maintain a safe forward separation to the lead vehicle, "Veh F",
- Reach the upcoming goal in the correct lane and at the correct speed or else abort and select an alternate route (if possible).
- Merge into the correct lane with sufficient spacing so as to not violate spacing rules relative to the vehicles in that lane, "Veh 1" and "Veh 2".

The ultimate output of the Lane Driving System, produced by the `VehicleDriver` element, is an instantaneously desired speed and an instantaneously desired tracking lane. The tracking lane is produced directly by the Merge Planning element and is discussed below. The speed output is the minimum of the road's speed limit, an externally-imposed speed limit and a subsumptive [8] selection between the speeds necessary for Distance Keeping and Merge Planning, where the Distance Keeping output speed is *suppressed* by the Merge Planning output speed when the Merge Planner is active to give the Merge Planner unfettered control of the vehicle's speed and room to bend the Distance Keeping rules if necessary to continue forward. In this driving context, the aforementioned system elements are all active simultaneously, performing the following functions:

- The `CurrentSceneReporter` identifies "Veh F" as the lead vehicle in the current lane and provides the distance $x_L = (x_F - l_{veh})$ and the velocity $v_L = v_F$ to

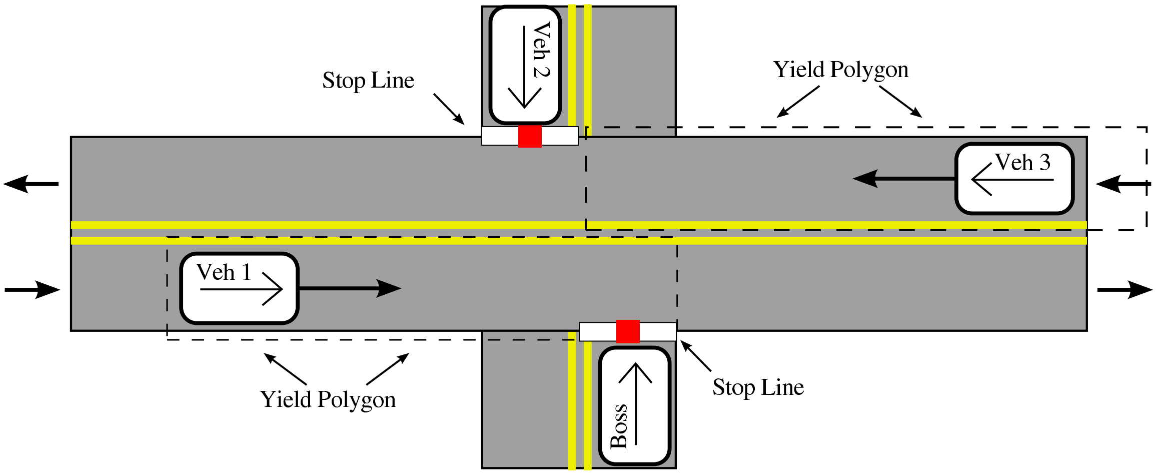

Fig. 6. An Example Intersection Handling Scenario.

the rest of the system as the distance to the lead vehicle and the lead vehicle's speed respectively.

- The `DistanceKeeper` computes a velocity command for tracking "Veh F" at a safe distance as $v_{DK} = K_{gap} * (x_L - gap_{desired})$, where K_{gap} is a configurable proportional gain and $gap_{desired}$ is computed as a function of Boss' current speed.
- The `LaneSelector` determines both that the current goal is a distance x_G away along the right-hand lane and that progress in the current lane is being inhibited by "Veh F" (if v_{DK} is less than the speed limit). If the goal is sufficiently far away and the Lane Selector is attempting to pass "Veh 2", it may continue to hold the current tracking lane for a short time. Otherwise, it will request an immediate merge into the right-hand lane.
- The `MergePlanner`, assuming that `LaneSelector` has requested a merge, identifies three potential merge slots in the right hand lane: before "Veh 2", after "Veh 1", and in-between the two. Each slot is evaluated both for instantaneous and predicted feasibility, and the `MergePlanner` may command a slow-down in the current lane to let "Veh 2" pass , an immediate merge between, or else to continue tracking "Veh F" in the current lane to pass "Veh 1", all depending on the specifics of the scenario.

B. Intersection Handling

Fig. 6 shows Boss approaching a stop line in a four-way, two-stop intersection with the intent to cross and continue forward. Another vehicle, "Veh 2", approaches the other stop-line, and there is traffic flowing on the horizontal road in both directions ("Veh 1" and "Veh 3").

The intersection handling subsystem is active both on approach to the intersection and when sitting at the stop-line waiting for precedence. On approach, the `PrecedenceEstimator` computes the set of relevant stop-lines to monitor for *precedence* among static vehicles and a set of yield polygons to monitor for *clearance* through moving traffic. These two boolean states, *precedence* and *clearance*, are forwarded to the `TransitionManager` to control when to issue the command to proceed through the intersection. The yield polygons are also used by the `PanheadPlanner` to optimize sensor coverage of the areas of the road where moving traffic is likely to be found.

Precedence is determined among stop-lines via a notion of *occupancy*, and the stop-line Boss is approaching is treated no differently than the others. When any vehicle (i.e. Boss or "Veh 2") is inside a pre-computed polygon around the stop-line, that stop-line is considered to be *occupied* and the stop-line with the earliest occupied time is considered to have precedence. Precedence is signaled to the `TransitionManager` when Boss is stopped at its stop-line and that stop-line has precedence.

Clearance is computed by estimating a time-of-arrival (ETA) for each vehicle in a Yield Polygon (i.e. "Veh 1" and "Veh 3") at the crash point for that polygon, which is where the vehicle would first intersect Boss' projected path through the intersection. These ETAs are compared to a conservative estimate of the time Boss would require to traverse the intersection, and instantaneous clearance is granted when all the ETAs in question exceed this estimate. To compensate for errors in the estimation of other vehicles, instantaneous clearance must be believed for at least one continous second before *clearance* is granted to the `TransitionManager` for goal propagation.

Once the system reaches the stop-line, the `TransitionManager` receives a set of local goals to proceed through the intersection from the `GoalSelector`. It then waits for the `PrecedenceEstimator` to signal that Boss has both precedence among stop-lines and clearance through traffic before actually issuing those goals to the motion planner. This has the benefit of isolating the motion planner from rules regarding discrete traffic interaction and allowing it to focus on lane following and collision avoidance.

VI. MOTION PLANNING

The motion planning layer is responsible for executing the current motion goal issued from the behaviors layer. This goal may be a location within a road lane when

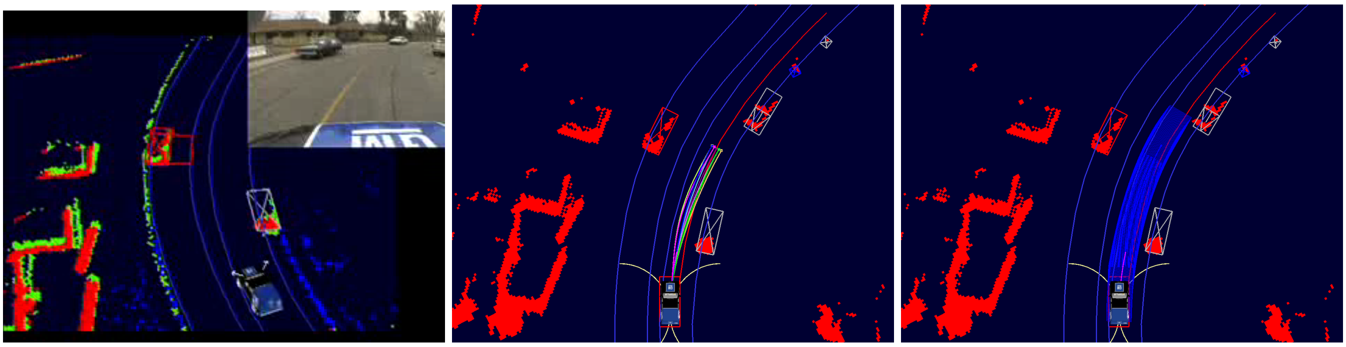

Fig. 7. Following a road lane. These images show a single timeframe from the Urban Challenge.

performing nominal on-road driving, a location within a zone when traversing through a zone, or any location in the environment when performing error recovery. The motion planner constrains itself based on the context of the goal to abide by the rules of the road.

In all cases, the motion planner creates a path towards the desired goal, then tracks this path by generating a set of candidate trajectories that follow the path to varying degrees and selecting from this set the best trajectory according to an evaluation function. This evaluation function differs depending on the context but includes consideration of static and dynamic obstacles, curbs, speed, curvature, and deviation from the path.

A. *Lane Driving*

During on-road navigation, the motion goal from the Behavioral Executive is a location within a road lane. The motion planner then attempts to generate a trajectory that moves the vehicle towards this goal location in the desired lane. To do this, it first constructs a curve along the centerline of the desired lane, representing the nominal path for the vehicle. To robustly follow the desired lane and to avoid static and dynamic obstacles, the motion planner generates trajectories to a set of local goals derived from this centerline path.

The goals are placed at a fixed longitudinal distance down the centerline path, but vary in lateral offset to provide several options for the planner. A model-based trajectory generation algorithm is used to compute dynamically feasible trajectories to these local goals [9]. The velocity profile used for each of these trajectories is computed based on several factors, including: the maximum velocity bound given from the Behavioral Executive based on safe following distance to the lead vehicle, the speed limit of the current road segment, the maximum velocity feasible given the curvature of the centerline path, and the desired velocity at the goal (e.g. if it is a stop-line).

The resulting trajectories are then evaluated against their proximity to static and dynamic obstacles in the environment, as well as their distance from the centerline path, their smoothness, and various other metrics. The best trajectory according to these metrics is selected and executed by the vehicle.

Fig. 7 provides an example of the local planner following a road lane. The left-most image shows the view from the vehicle overlaid on an overhead road and traversability map (lane extents are shown as blue curves, obstacles shown in red). The center image shows a set of trajectories generated to follow the right lane (the centerline of the lane is shown as a red curve), and the right image shows the trajectory selected for execution (the convolution of the vehicle along this trajectory is shown as a sequence of blue polygons).

B. *Unstructured Driving*

When driving in unstructured areas, the motion goal from the Behavioral Executive is a desired pose for the vehicle such as a parking spot. The motion planner attempts to generate a trajectory that moves the vehicle towards this goal pose. However, driving in unstructured environments, such as zones, significantly differs from driving on roads. As mentioned in the previous section, when traveling on roads the desired lane implicitly provides a preferred path for the vehicle (the centerline of the lane). In zones there are no driving lanes and thus the movement of the vehicle is far less constrained.

To efficiently plan a smooth path to a distant goal pose in a zone, we use a lattice planner that searches over vehicle position (x, y), orientation (θ), and velocity (v) to generate a sequence of feasible maneuvers that are collision-free with respect to the static and dynamic obstacles observed in the environment. This path is also biased away from undesirable areas within the environment such as curbs. To efficiently generate complex plans over large, obstacle-laden environments, the planner relies on the anytime, replanning search algorithm Anytime D* [10].

The resulting plan is then tracked by the local planner in a similar manner to the paths extracted from road lanes. However, in contrast to when following lane paths, the trajectories generated to follow the zone path all attempt to terminate on the path, reducing the risk that the vehicle might move away from the path and not easily be able to return to it. Fig. 8 shows Boss tracking a lattice plan into a parking spot.

This lattice planner is flexible enough to be used in a large variety of cases that can occur during on-road and zone navigation. In particular, it is used during error recovery

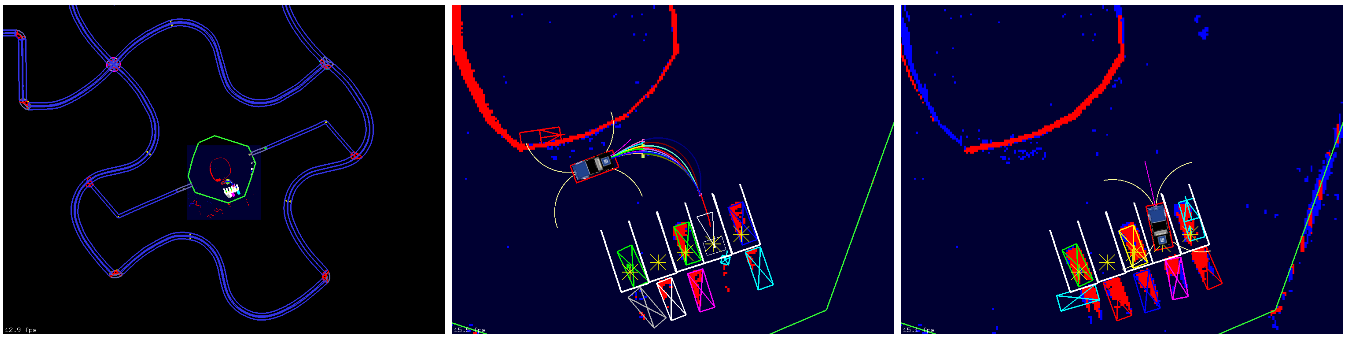

Fig. 8. Following a lattice plan to a parking spot. These images are from a qualification run at the Urban Challenge

when navigating congested intersections, to perform difficult u-turns, and to get the vehicle back on track after emergency defensive driving maneuvers. In these error recovery scenarios the lattice planner is biased to avoid areas that could result in unsafe behavior (such as oncoming lanes when on roads).

VII. CONCLUSIONS

We have presented a reasoning framework for autonomous urban driving. Performing this task safely and reliably requires intelligent consideration of other vehicles, context-aware decision making, and sophisticated motion planning. Our framework provides these capabilities through a three-tiered architecture that facilitates incremental addition of competencies and has been proven in over 3000 kilometers of autonomous driving, including winning the Urban Challenge. The approach applies to general urban driving and can be used in either fully autonomous systems or intelligent driver assistance systems.

REFERENCES

[1] A. Stentz and M. Hebert, “A complete navigation system for goal acquisition in unknown environments,” *Autonomous Robots*, vol. 2, no. 2, pp. 127–145, 1995.

[2] A. Kelly, “An intelligent predictive control approach to the high speed cross country autonomous navigation problem,” Ph.D. dissertation, Carnegie Mellon University, 1995.

[3] S. Singh, R. Simmons, T. Smith, A. Stentz, V. Verma, A. Yahja, and K. Schwehr, “Recent progress in local and global traversability for planetary rovers,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[4] “Special Issue on the DARPA Grand Challenge, Part 1,” *Journal of Field Robotics*, vol. 23, no. 8, 2006.

[5] “Special Issue on the DARPA Grand Challenge, Part 2,” *Journal of Field Robotics*, vol. 23, no. 9, 2006.

[6] J. Carsten, A. Rankin, D. Ferguson, and A. Stentz, “Global path planning on-board the mars exploration rovers,” in *Proceedings of the IEEE Aerospace Conference*, 2007.

[7] C. Thorpe, T. Jochem, and D. Pomerleau, “The 1997 automated highway demonstration,” in *Proceedings of the International Symposium on Robotics Research (ISRR)*, 1997.

[8] R. A. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[9] T. Howard and A. Kelly, “Optimal rough terrain trajectory generation for wheeled mobile robots,” *International Journal of Robotics Research*, vol. 26, no. 2, pp. 141–166, 2007.

[10] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, “Anytime Dynamic A*: An Anytime, Replanning Algorithm,” in *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2005.