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.

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. 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

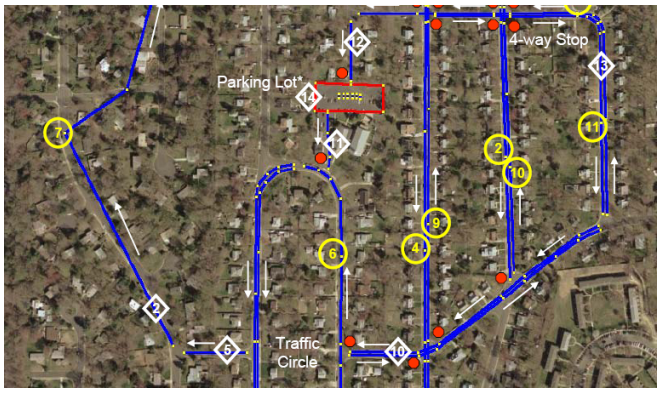

Fig. 2. Sample Route Network Definition File (RNDF) data. Small yellow dots represent waypoints, with blue line segments connecting them to form roads. Circled numbers represent the checkpoints used to compose missions.

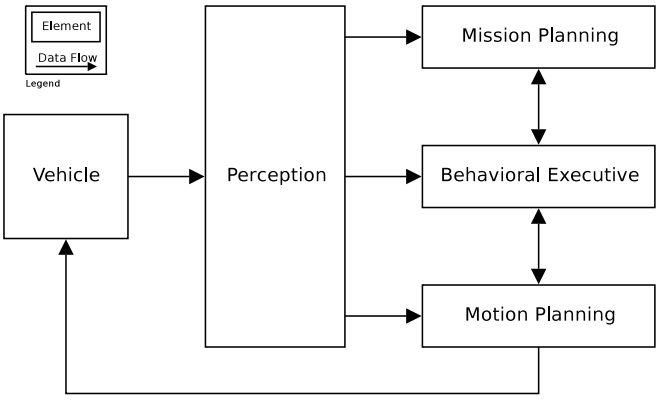

Fig. 3. Boss' software system architecture, showing primary subsystems and data paths. Communication is via message-passing according to the anonymous publish-subscribe pattern.

as to the accuracy of this imagery. Fig. 2 illustrates the sample RNDF data and imagery provided by DARPA.

On race day, DARPA gave each vehicle a Mission Data File (MDF), which described a mission as an ordered set of checkpoints to be visited. The vehicle had five minutes to process this file and begin its mission. Upon completion, the vehicle was given a new MDF, and so on until all 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 that ran asynchronously (see Fig. 3).

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 geometry and connectivity of the roads, parking zones, and intersections that comprise the road network;
- **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 road sections that are impassable due to static obstacles.

The *Mission Planning* component reasons about the optimal route to the current 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. This globally strategic information is provided to the Behavioral Executive as an estimated cost-to-checkpoint value for each waypoint in the road network.

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 incremental pose goals for execution by the Motion Planning component. These local goals are propagated and modulated to implement traffic-interactive behaviors. There are three abstract behavioral contexts, each governed by a specific set of rules. In the *Lane Driving* context, 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* requires the determination of precedence among stopped vehicles and safe merging into or across moving traffic at an intersection. Lastly, in the *Zone Maneuvering* context, the system maneuvers through an unstructured obstacle or parking zone. The active context is primarily determined by the location of the system and the global route set forth by the Mission Planner, which determines the next action the robot should take toward the current checkpoint.

The *Motion Planning* component takes the next 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 further described in the following sections.

IV. MISSION PLANNING

The mission planner is responsible for generating a cost-to-checkpoint value for every waypoint in the world. In our setting, this value can be thought of as the minimum time required to reach the checkpoint. A path to the checkpoint

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-checkpoint value plus the time taken to reach the waypoint, then repeating this process to step through waypoints until the checkpoint is reached.

To generate cost-to-checkpoint values, 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. Dijkstra's search is then run to compute cost-to-checkpoint values for each position in the graph given a desired checkpoint position, such as the first checkpoint in the mission. In addition to providing the Behavioral Executive more information to reason about, computing cost-to-checkpoint 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-checkpoint 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 following the value function from the Mission Planner to generate local tasks for the Motion Planner. Along the way, it uses local perceptual information to guarantee the system's adherence to various rules of the road, especially those concerning structured interactions with other traffic and road blockages. The local tasks take the form of simple, discrete motion goals, such as driving along a road to a specific point, navigating to a specific parking spot, or maneuvering to recover from any of a number of anomalous situations. 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. In the case of driving along a road, a periodic lane tracking and speed government message modifies the current task to implement behaviors such as safety gap maintenance, passing maneuvers and queueing in stop-and-go traffic.

The Behavioral Executive is decomposed according to its responsibilities among the system's three abstract operational contexts: Lane Driving, Intersection Handling, and Zone Maneuvering. The first two obey highly structured road and traffic rules, and are thus strongly reflected in the Behavioral Executive's architecture, represented by distinct functional groups in Fig. 4. Zone Maneuvering, on the other hand, 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. Given its reduced structure, this context is not directly represented at the behavioral level. It is important to note that the system includes many more functional elements and more convoluted data paths than indicated, but that these generally belong to auxiliary functionality. These are omitted to allow for a more clear and concise discussion of the core elements.

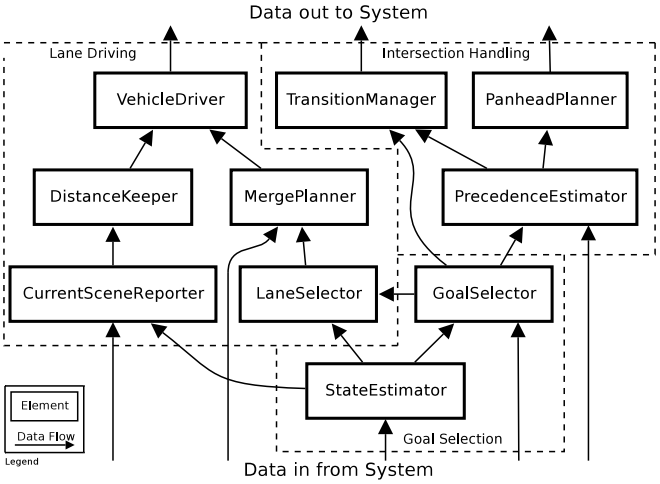

Fig. 4. Behavioral Executive software architecture, showing grouped dominant elements and data paths.

The core functionality of the Behavioral Executive begins with the selection of the current task to be executed by the Motion Planning subsystem. This responsibility is fulfilled by the *Goal Selection* group, which consists of two active elements. First the `StateEstimator` compares the vehicle's global position with the geometry of the road model to determine the vehicle's logical position within the road network. The `GoalSelector` then uses this logical location to generate the next local goal to be executed by the Motion Planner. This goal directly reflects the current driving context, activating other functional groups as described by examples in the following sections

To better illustrate the system's operation, we discuss two example scenarios, lane driving and intersection handling, in the following sections.

A. Lane Driving

The *Lane Driving* group is active while the current goal is to drive along a road consisting of one or more lanes. Its functionality is broken into five primary modules whose ultimate output is a message that governs the robot's speed and tracking lane to simultaneously exhibit two distinct behaviors, *Distance Keeping* and *Lane Selection*.

Fig. 5 shows a standard driving scenario along a road with two lanes in the same direction, separated by a dashed white

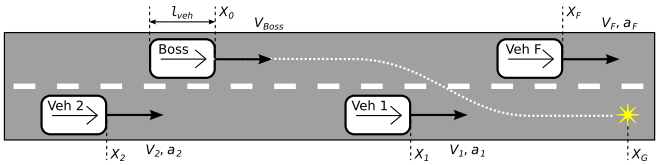

Fig. 5. An example lane driving scenario.

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); and
- 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 Distance Keeping behavior starts with the `CurrentSceneReporter`, which distills the list of known vehicles and lane blockages into a few discrete data elements, used by the `DistanceKeeper` to govern the robot's speed according to the Urban Challenge rules regarding inter-vehicle safety gaps.

The Lane Selection behavior begins with the `LaneSelector`, which makes a tactical decision about the robot's currently-desired lane according to the surrounding traffic conditions and local goal information, requesting a merge into that lane if necessary. The `MergePlanner` then determines the feasibility of a merge into that lane, governing the vehicle's speed to track to a valid merge window and commanding lane-change maneuver when appropriate.

The ultimate outputs of the Lane Driving group, produced by the `VehicleDriver` element, are the robot's instantaneously desired speed and tracking lane. 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 is *suppressed* by the Merge Planning output to give the `MergePlanner` unfettered control of the vehicle's speed, possibly bending the Distance Keeping rules if necessary. In the example shown in Fig. 5, 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 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 (possibly) that progress in the current lane is being inhibited by "Veh F". If the goal is sufficiently far away, or 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 in order to pass "Veh 1", all depending on the specifics of the scenario.

It is important to note that these behaviors are meant to operate well within the kinematic and dynamic capabilities of the robot, such that the Motion Planner is always able to find and track a collision-free path without considering the macroscopic behavior of either Boss or the other vehicles in the world.

B. Intersection Handling

The *Intersection Handling* group consists of two primary elements, which are active on approach to and while waiting at an intersection. The `TransitionManager` manages the discrete-goal interface between the Behavioral Executive and the Motion Planner, withholding the goals from `GoalSelector` until the it is time for the robot to proceed. The time to proceed is determined by the `PrecedenceEstimator`, using the list of other vehicles and their state information to determine precedence among stop lines and clearance through traffic.

Fig. 7 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").

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

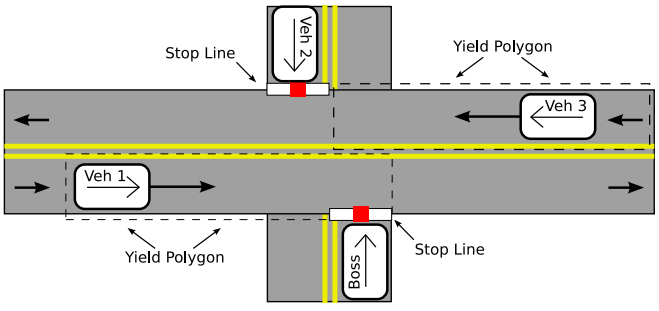

Fig. 7. An Example Intersection Handling Scenario.

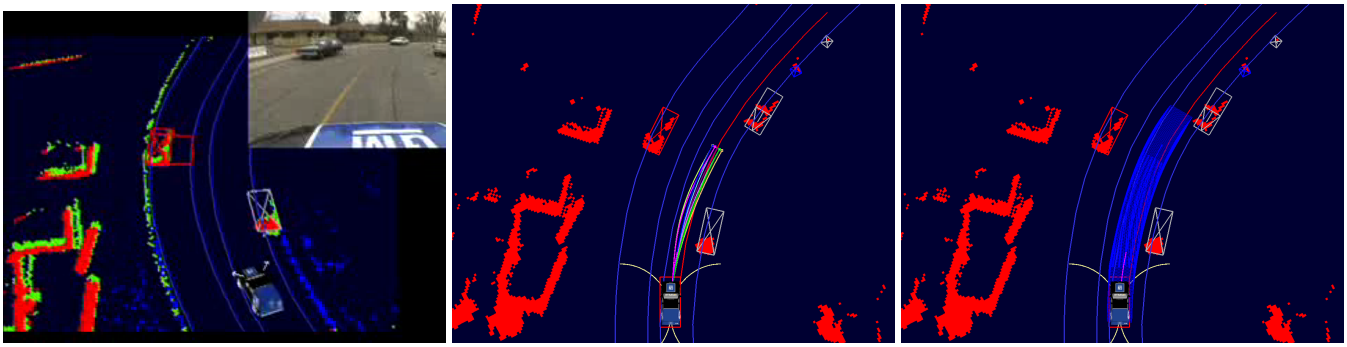

Fig. 6. Following a road lane. These images are from a qualification run at the Urban Challenge.

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*, where 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 for each Yield Polygon by estimating a time-of-arrival (ETA) for each vehicle (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 ETA's are compared to a conservative estimate of the time Boss would require to traverse the intersection, and instantaneous clearance is granted when each ETA exceeds this estimate. To compensate for transient errors in the detection of other vehicles, instantaneous clearance must be believed for at least one continuous 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. Similar to the Lane Selection behavior, this has the benefit of isolating the motion planner from rules regarding discrete traffic interaction, allowing it to focus entirely 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 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. 6 provides an example of the local planner following

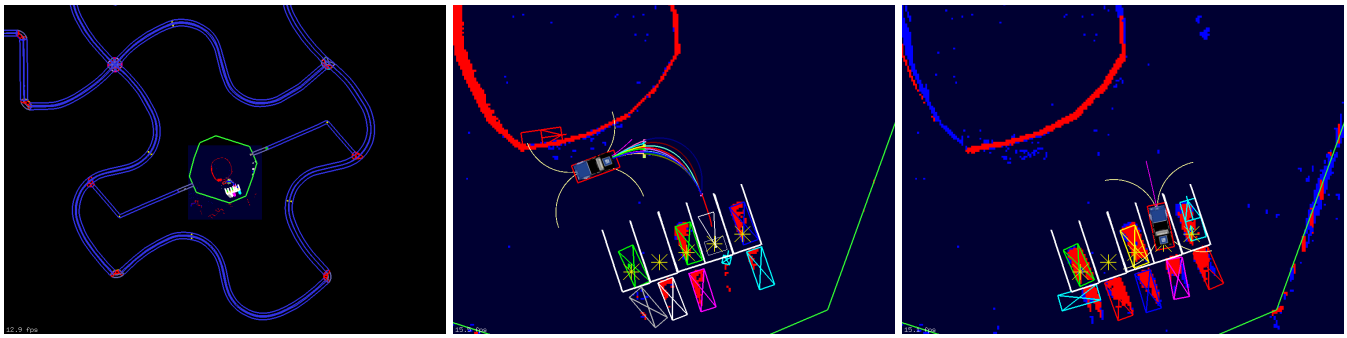

Fig. 8. Following a lattice plan to a parking spot. These images are from a qualification run at the Urban Challenge.

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 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-traffic lanes).

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.