Motion Planning in Urban Environments: Part II

Dave Ferguson
Intel Research Pittsburgh
Pittsburgh, PA
dave.ferguson@intel.com

Thomas M. Howard
Carnegie Mellon University
Pittsburgh, PA
thoward@ri.cmu.edu

Maxim Likhachev
University of Pennsylvania
Philadelphia, PA
maximl@seas.upenn.edu

***Abstract*— We present the motion planning framework for an autonomous vehicle navigating through urban environments. Such environments present a number of motion planning challenges, including ultra-reliability, high-speed operation, complex inter-vehicle interaction, parking in large unstructured lots, and constrained maneuvers. Our approach combines a model-predictive trajectory generation algorithm for computing dynamically-feasible actions with two higher-level planners for generating long range plans in both on-road and unstructured areas of the environment. In this Part II of a two-part paper, we describe the unstructured planning component of this system used for navigating through parking lots and recovering from anomalous on-road scenarios. We provide examples and results from "Boss", an autonomous SUV that has driven itself over 3000 kilometers and competed in, and won, the Urban Challenge.**

I. Introduction

Motion planning for autonomous vehicles operating in urban environments is extremely challenging, requiring both high speed navigation on roads and complex maneuvering in unstructured parking lots, all while interacting with other vehicles and obstacles. In this Part II of a two-part paper, we describe the unstructured driving component of a motion planning system for autonomous urban driving. This system was developed for Carnegie Mellon University's winning entry into the Urban Challenge, "Boss". This system enabled Boss to travel quickly and smoothly through parking lots and off-road areas, as well as perform complex error recovery maneuvers in anomalous on-road scenarios.

Part I of this two-part paper described the on-road driving component of the motion planner, as well as the underlying trajectory generation algorithm used to track paths in both on-road and unstructured planning scenarios. Part I also provided some background of the architecture of the overall software system used by Boss. However, in the following section for comprehensiveness we re-iterate the high-level responsibilities of the motion planner within our architecture. We then describe how Boss generates complex maneuvers in large, unstructured environments, along with how it exploits the context of the scenario to efficiently bias its behavior appropriately. Throughout, we present example results and illustrations drawn from over 3000 kilometers of autonomous driving and the Urban Challenge competition itself. We conclude with a discussion of lessons learned and related work.

II. Motion Planning

In Boss' software architecture, the motion planning layer is responsible for executing the current motion goal issued from the Behavioral Executive. This goal may be a location within a road lane when performing nominal on-road driving, a location within a parking lot or obstacle field when traversing through one of these areas, or any location in the environment when performing error recovery. The motion planner constrains itself based on the context of the goal and the environment to abide by the rules of the road.

During nominal on-road driving, the goal entails a desired lane and a desired position within that lane (typically a stop-line at the end of the lane). In such cases, the motion planner invokes a high-speed lane-based planner to generate a path that tracks the desired lane. During unstructured driving, such as when navigating through parking lots, the goal consists of a desired pose of the vehicle in the world. In these cases, the motion planner invokes a 4D lattice planner that generates a global path to the desired pose. These unstructured motion goals are also used when the vehicle encounters an anomalous situation during on-road driving and needs to perform a complex maneuver (such as when an intersection is partially blocked and cannot be traversed through in the desired lane).

Given a motion goal, 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. As mentioned above, the nature of the path generated differs based on the context of the motion goal and the environment. In addition, the evaluation function differs depending on the context but always includes consideration of static and dynamic obstacles, curbs, speed, curvature, and deviation from the path. The selected trajectory is then directly executed by the vehicle.

III. Unstructured Planning

During unstructured area navigation, the motion goal from the Behavioral Executive is a pose (or set of poses) in the environment. The motion planner attempts to generate a trajectory that moves the vehicle towards this goal pose. However, driving in unstructured environments significantly differs from driving on roads. As mentioned in the Part I, when traveling on roads the desired lane implicitly provides a preferred path for the vehicle (the centerline of the lane). In unstructured environments 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, we use a lattice planner that searches over vehicle position

Fig. 1. Replanning when new information is received

(x, y), orientation (θ), and velocity (v). The set of possible local maneuvers considered for each (x, y, θ, v) state in the planner's search space are constructed offline using the same vehicle model as used in trajectory generation, so that they can be accurately executed by the vehicle. This planner searches in a backwards direction out from the goal pose(s) and generates a path consisting of a sequence of feasible high-fidelity maneuvers that are collision-free with respect to the static obstacles observed in the environment. This path is also biased away from undesirable areas within the environment such as curbs and locations in the vicinity of dynamic obstacles.

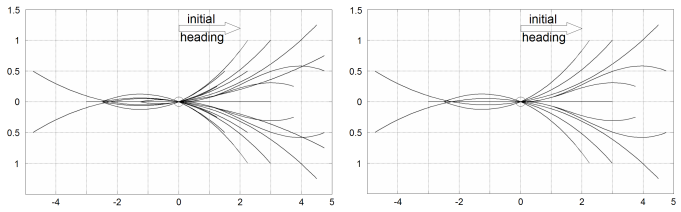
This global high-fidelity path is then tracked by a local planner that operates similarly to the on-road lane tracker, by generating a set of candidate trajectories that follow the path while allowing for some flexibility in local maneuvering. The local planner runs at a fixed 10Hz during operation, with the lattice planner also nominally run at 10Hz. However, in very difficult planning scenarios the lattice planner may take longer (up to a couple seconds) to generate its initial solution and it is for this reason that pre-planning is performed whenever possible (as will be discussed later). In the following sections, we describe in more details the elements of our approach and how it exploits the context of its instantiation to adapt its behavior based on the situation (e.g. parking lot driving vs off-road error recovery).

IV. Planning Complex Maneuvers

To efficiently generate complex plans over large, obstacle-laden environments, the planner relies on an anytime, re-planning search algorithm known as Anytime Dynamic A* (Anytime D*), developed by Likhachev et al. [1]. Anytime D* quickly generates an initial, suboptimal plan for the vehicle and then improves the quality of this solution while deliberation time allows. The algorithm is also able to provide control over the suboptimality bound of the solution at all times during planning. Figure 8 shows an initial, suboptimal path converging over time to the optimal solution.

When new information concerning the environment is received (for instance, a new static or dynamic obstacle is observed), Anytime D* is able to efficiently repair its existing solution to account for the new information. This repair process is expedited by performing the search in a backwards direction, as in such a scenario updated information in the vicinity of the vehicle affects a smaller portion of the search space and so Anytime D* is able to reuse a large portion of its previously-constructed search tree in re-computing a new path. Figure 1 illustrates this replanning capability. These images were taken from a parking task performed during the National Qualification Event (the bottom-left image shows the parking lot in green and the neighboring roads in blue). The top-left image shows the initial path planned for the vehicle to enter the parking spot indicated by the white triangle. Several of the other spots were occupied by other vehicles (shown as rectangles of varying colors), with detected obstacles shown as red areas. The trajectories generated to follow the path are shown emanating from our vehicle (discussed later). As the vehicle gets closer to its intended spot, it observes more of the vehicle parked in the right-most parking spot (top, second from left image). At this point, it realizes its current path is infeasible and replans a new path that has the vehicle perform a loop and pull in smoothly. This path was favored in terms of time over stopping and backing up to re-position.

To further improve efficiency, the lattice planner uses a multi-resolution state and action space. In the vicinity of the goal and vehicle, where very complex maneuvering may be required, a dense set of actions and a fine-grained discretization of orientation are used during the search. In other areas, a coarser set of actions and discretization of orientation are employed. However, these coarse and dense resolution areas both share the same dimensionality (specifically, x, y, θ, v) and seamlessly interface with each other, so that resulting solution paths overlapping both coarse and dense areas of the space are smooth and feasible. Figure 2 illustrates how the dense and



(a) Dense action space (b) Coarse action space

Fig. 2. Dense and coarse resolution action spaces. The coarse action space contains many fewer actions (24 versus 36 in the dense action space) with transitions only to states with a coarse-resolution heading discretization (in our case, 16 headings versus 32 in the dense-resolution discretization). In both cases the discretization in position is 0.25m.

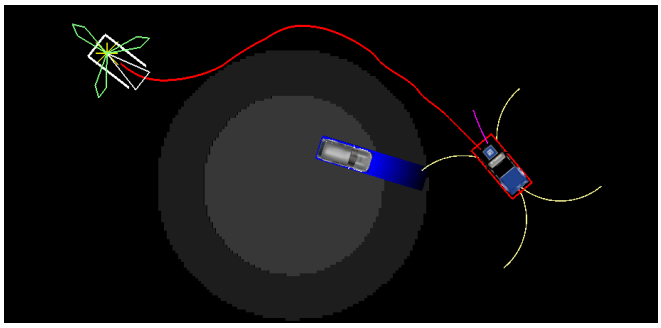


Fig. 3. Biasing the cost map for the lattice planner so that the vehicle keeps away from dynamic obstacles. Notice that the high-cost region around the dynamic obstacle is offset to the left so that Boss will prefer moving to the right of the vehicle.

coarse action and state spaces differ.

The effectiveness of the Anytime D* algorithm is highly dependent on its use of an informed heuristic to focus its search. An accurate heuristic can reduce the time and memory required to generate a solution by orders of magnitude, while a poor heuristic can diminish the benefits of the algorithm. It is thus important to devote careful consideration to the heuristic used for a given search space.

Since in our setup Anytime D* searches backwards, the heuristic value of a state estimates the cost of a path from the robot pose to that state. Anytime D* requires these values to be admissible (not to overestimate the actual path cost) and consistent [2]. For any state (x, y, θ, v), the heuristic we use is the maximum of two values. The first value is the cost of an optimal path from the robot pose to (x, y, θ, v) assuming a completely empty environment. These values are pre-computed offline and stored in a heuristic lookup table [3]. This is a very well informed heuristic function when operating in sparse environments and is guaranteed to be admissible. The second value is the cost of a 2D path from the robot (x_r, y_r) coordinates to (x, y) given the actual environment. These values are computed online by a 2D grid-based Dijkstra's search. This second heuristic function is very useful when operating in obstacle-laden environments. By taking the maximum of these two heuristic values we are able to incorporate both the constraints of the vehicle and the constraints imposed by the obstacles in the environment. The result is a very well-informed heuristic function that can speed up the search by an order of magnitude relative to either of the component heuristics alone.

For more details concerning the benefit of this combined heuristic function and other optimizations implemented in our lattice planner, including its multi-resolution search space and how it efficiently plans and replans, see [4].

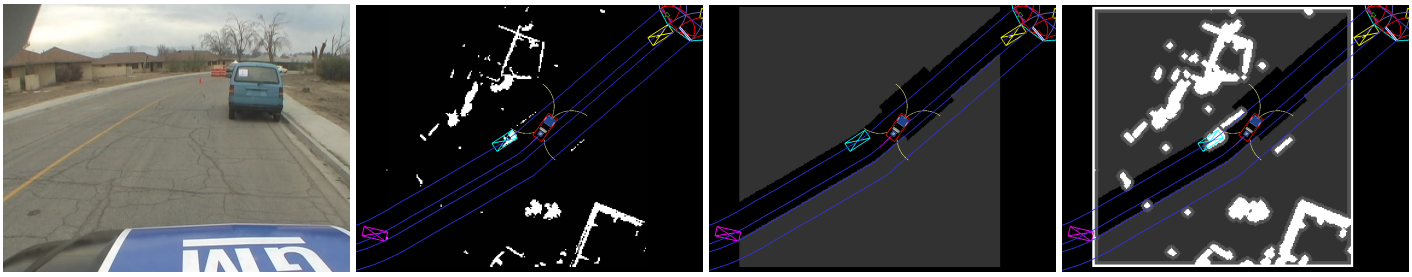
A. Incorporating Environmental Constraints

In addition to the geometric information provided in the static obstacle map from perception, we incorporate context-specific constraints on the movement of the vehicle by creating an additional cost map known as a constrained map. This 2D grid-based cost map encodes the relative desirability of different areas of the environment based on the road structure in the vicinity and, if available, prior terrain information. This constrained cost map is then combined with the static map from perception to create the final combined cost map to be used by the lattice planner. Specifically, for each cell (i, j) in the combined cost map $\mathcal{C}$, the value of $\mathcal{C}(i, j)$ is computed as the maximum of $\mathcal{EPC}(i, j)$ and $\mathcal{CO}(i, j)$, where $\mathcal{EPC}(i, j)$ is the static map value at (i, j) and $\mathcal{CO}(i, j)$ is the constrained cost map value at (i, j).

For instance, when invoking the lattice planner to plan a maneuver around a parked car or jammed intersection, the constrained cost map is used to specify that staying within the desired road lane is preferable to traveling in an oncoming lane, and similarly that driving off-road to navigate through a cluttered intersection is dangerous. To do this, undesirable areas of the environment based on the road structure are assigned high costs in the constrained cost map. These can be both soft constraints (undesirable but allowed areas), which correspond to high costs, and hard constraints (forbidden areas), which correspond to infinite costs. Figure 4 shows the constrained cost map generated for an on-road maneuver, along with the expanded perception cost map and the resulting combined cost map used by the planner.

B. Incorporating Dynamic Obstacles

The combined cost map of the planner is also used to represent dynamic obstacles in the environment so that these can be avoided by the planner. The perception system of Boss represents static and dynamic obstacles independently, which allows the motion planner to treat each type of obstacle differently. The lattice planner adapts the dynamic obstacle avoidance behavior of the vehicle based on its current proximity to each dynamic obstacle. If the vehicle is close to a particular dynamic obstacle, that obstacle and a short-term prediction of its future trajectory is encoded into the combined cost map as a hard constraint so that it is strictly avoided. For every dynamic obstacle, both near and far, the planner encodes a varying high-cost region around the obstacle reflecting the uncertainty regarding its future behavior to provide a safe clearance. Although these high-cost regions are not hard constraints, they result in the vehicle avoiding the vicinity of the dynamic obstacles if at all possible. Further,



(a) (b) (c) (d)

Fig. 4. A snapshot from a qualification run during the Urban Challenge, showing (b) the obstacle map from perception (obstacles in white), (c) the constrained cost map based on the road structure (lighter areas are more costly), and (c) the resulting combined cost map used by the planner.

the generality of this approach allows us to influence the behavior of our vehicle based on the specific behavior of the dynamic obstacles. For instance, we offset the high-cost region based on the relative position of the dynamic obstacle and our vehicle so that we will favor moving to the right, resulting in yielding behavior in unstructured environments quite similar to how humans react in these scenarios. Figure 3 provides an example scenario involving a dynamic obstacle along with the corresponding cost map generated.

V. Tracking Complex Paths

The resulting lattice plan is then tracked by the local planner in a similar manner to the paths extracted from road lanes: the motion planner generates a set of trajectories that attempt to follow the plan while also allowing for local maneuverability. However, in contrast to when following lane paths, the trajectories generated to follow the lattice path all attempt to terminate on the path. Each trajectory is in fact a concatenation of two short trajectories, with the first of the two short trajectories ending at an offset position from the path and the second ending back on the path. By having all concatenated trajectories return to the path we significantly reduce the risk of having the vehicle move itself into a state that is difficult to leave.

Figure 7 illustrates the local planner following a lattice plan to a specified parking spot. Figure 7(a) shows the lattice plan generated for the vehicle (in red) towards the desired parking spot (desired pose of the vehicle shown as the white triangle). Figure 7(b) shows the set of trajectories generated by the vehicle to track this plan, and Figure 7(c) shows the best trajectory selected by the vehicle to follow the path.

Both forwards and reverse trajectories are generated as appropriate based on the velocity of the lattice path being tracked. When the path contains an upcoming velocity switching point, or cusp point, the local planner generates trajectories that bring the vehicle to a stop at the cusp point. Figure 5 shows reverse trajectories generated to a cusp point in the lattice path.

As mentioned above, one of the desired capabilities of our vehicle was to be able to exhibit human-like yielding behavior in parking lots, to allow for safe, natural interaction with other vehicles. Through our biased cost function, the lattice planner typically generates paths through parking lots that keep to the right of other vehicles. However, it is possible that another vehicle may be quickly heading directly towards Boss, requiring evasive action similar to the on-road defensive driving maneuvers discussed in Part I. In such a case, Boss' local planner detects that it is unable to continue along its current course without colliding with the other vehicle and it then generates a set of trajectories that are offset to the right of the path. The intended behavior here is for each vehicle to move to the right to avoid a collision. Figure 6 provides an example of this behavior in a large parking lot.

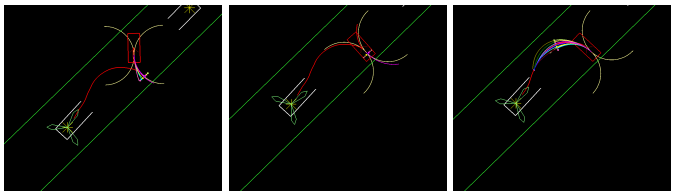
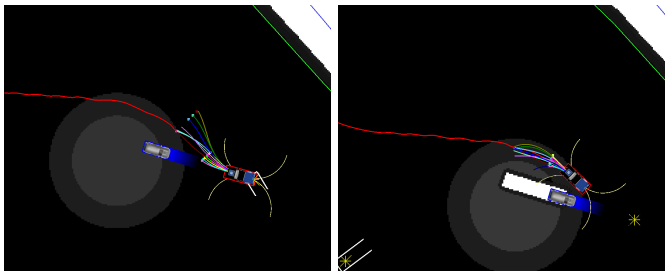


Fig. 5. Reversing during path tracking.



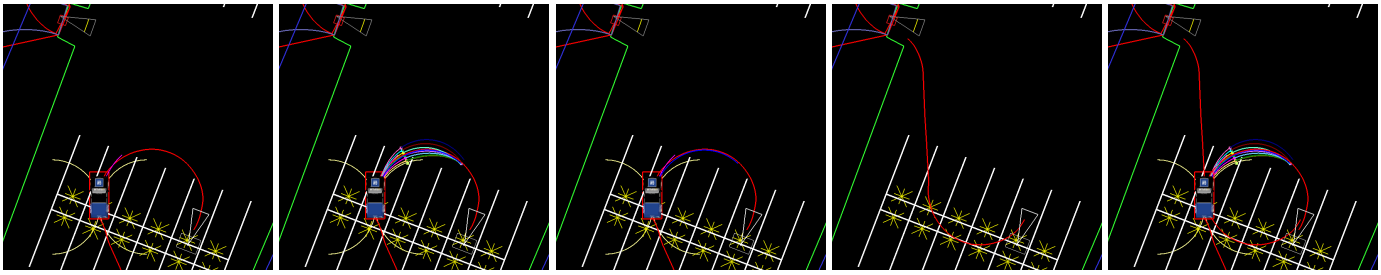
(a) (b)

Fig. 6. Defensive driving when in unstructured environments. (a) Trajectories thrown to right of path (other vehicle should likewise go to right). (b) New path planned from new position.

In addition to the general optimizations employed by the lattice planner, there are several context-specific steps performed in different urban driving scenarios for which the lattice planner is invoked. In the following two sections we describe different methods used to provide optimized, intelligent behavior in parking lots and on-road error recovery scenarios.

VI. Planning in Parking Lots

Because the location of parking lots is known a priori, this information can be exploited by the motion planning system to improve the efficiency and behavior of the lattice planner



(a) (b) (c) (d) (e)

Fig. 7. Following a lattice plan to a parking spot. Here, one planner is updating the path to the spot while another is pre-planning a path out of the spot.

within these areas. First, we can use the extents of the parking lot to constrain the vehicle through the constrained cost map. To do this, we use the a priori specified extents of the parking lot to set all cells outside the lot in the constrained cost map to be hard constraints. This constrains the vehicle to operate only inside the lot. We also include a high cost buffer around the perimeter of the parking lot to bias the vehicle away from the boundaries of the lot.

When prior terrain information exists such as overhead imagery, this information can also be incorporated into the constrained cost map to help provide global guidance for the vehicle. For instance, this information can be used to detect features such as curbs or trees in parking lots that should be avoided, so that these features can be used in planning before they are detected by onboard perception.

Further, because the parking lot goals are also known in advance of entering the parking lot (e.g. the vehicle knows which parking spot it is intending on reaching), the lattice planner can pre-plan to the first goal pose within the parking lot while the vehicle is still approaching the lot. By planning a path from the entry point of the parking lot in advance, the vehicle can seamlessly transition into the lot without needing to stop, even for very large and complex lots. Figure 8 illustrates the pre-planning used by the lattice planner.

In a similar vein, when the vehicle is in a lot traveling towards a parking spot, we use a second lattice planner to simultaneously plan a path from that spot to the next desired location (e.g. the next parking spot to reach or an exit of the lot). When the vehicle reaches its intended parking spot, it then immediately follows the path from this second planner, again eliminating any time spent waiting for a plan to be generated. Figure 7 provides an example of the use of multiple concurrent lattice planners. Figure 7(a) shows the lattice plan generated towards the desired parking spot. Figure 7(d) shows the path simultaneously being planned out of this spot to the exit of the parking lot, and Figure 7(e) shows the paths from both planners at the same time.

VII. Planning in Error Recovery Scenarios

The lattice planner is flexible enough to be used in a large variety of cases that can occur during on-road and unstructured navigation. In particular, it is used during error recovery when navigating congested lanes or intersections and to perform difficult U-turns. In such cases, the nominal on-road motion planner determines that it is unable to generate any feasible trajectory and reports its failure to the Behavioral Executive, which in turn issues an unstructured goal pose (or set of poses) to the motion planner and indicates that it is in an error recovery mode. The motion planner then uses the lattice planner to generate a path to the set of goals, with the lattice planner determining during its planning which goal is easiest to reach. 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) through increasing the cost of undesirable areas in the constrained cost map (see Figure 4).

The ability to cope with anomalous situations was a key focus of Boss' software system and the lattice planner was used as a powerful tool to maneuver the vehicle to arbitrary locations in the environment.

The lattice planner is also invoked when the road shape is not known a priori and cannot be reliably detected online due to the absence of road lane markers or clear geometric features such as curbs. In such cases the lattice planner is used to generate obstacle-free paths for the vehicle and it is biased to stay within any detected geometric extents of the road.

VIII. Results and Discussion

Our motion planning system was developed over the course of more than a year and tested over thousands of kilometers of autonomous operation in three different testing sites, as well as the Urban Challenge event itself. Through this extensive testing all components were rigorously debugged and the planners were incrementally improved upon.

A key factor in our system-level design was that Boss should never give up. As such, the lattice planner was designed to be general enough and powerful enough to plan in extremely difficult scenarios. In addition, the Behavioral Executive was designed to issue an infinite sequence of pose goals to the motion planner should it continue to fail to generate plans (see [5] for details of this error recovery framework). And in the final Urban Challenge event, as anticipated, this error recovery played a large part in Boss' successful completion of the course. Over the course of the three final event missions, there were 17 different instances where the lattice planner was invoked due to an encountered anomalous situation (some of these included getting cut off at intersections, coming across

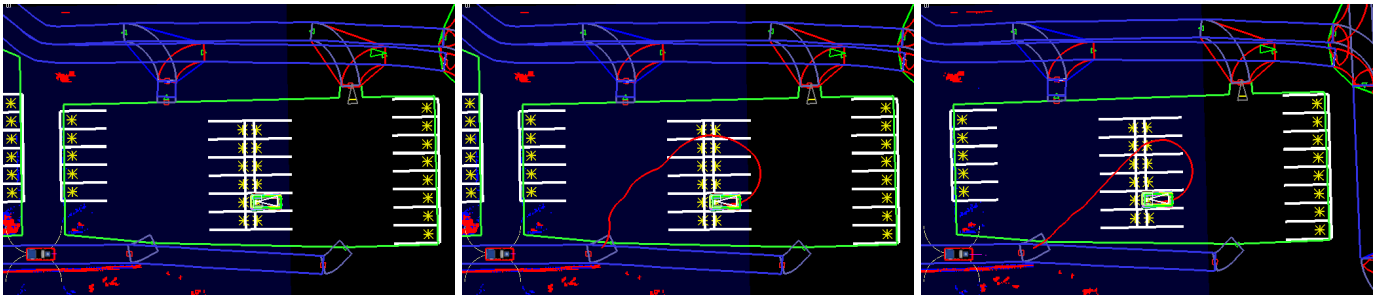


Fig. 8. Pre-planning a path into a parking spot (parking lot boundary shown in green, parking spot indicated by white triangle and multicolored goals) and improving this path in an anytime fashion. A set of goal poses are generated that satisfy the parking spot and an initial path is planned while the vehicle is still outside the parking lot. This path is improved as the vehicle approaches, converging to the optimal solution shown in the right image.

other vehicles blocking the lane, and perception occasionally observing heavy dust clouds as static obstacles).

Incorporation of an accurate vehicle model was also an important design choice for Boss' motion planning system. This allowed Boss to push the limits of acceleration and speed and travel as fast as possible along the road network, confident in its execution. Combining this model with an efficient on-road planner that could safely handle the high-speeds involved was also central to Boss' on-road performance.

In addition, the efficiency and path quality of the lattice planner enabled Boss to also travel smoothly and quickly through parking lot areas, without ever needing to pause to generate a plan. As well as generating smooth paths in (x, y, θ), by also considering the velocity dimension v the lattice planner was able to explicitly reason about the time required to change direction of travel and was thus able to generate very fast paths even when complex maneuvers were required. Overall, the focus on execution speed and smoothness strongly contributed to Boss finishing the four-hour race 19 minutes and 8 seconds faster than its nearest competitor [6].

One of the important lessons learned during the development of this system was that it is often extremely beneficial to exploit prior, offline processing to provide efficient online planning performance. We used this idea in several places, from the generation of lookup tables for the trajectory generator and lattice planner heuristic function to the pre-computing of constrained cost maps for parking lots. This prior processing saved us considerably at run-time. Further, even when faced with calculations that cannot be pre-computed offline, such as planning paths through novel environments, it can often pay to begin planning before a plan is required. This concept was the basis for our pre-planning on approach to parking lots and our concurrent planning to both current and future goals, and it enabled us to produce high quality solutions without needing to wait for these solutions to be generated.

Finally, although simplicity was central to our high-level system development and significant effort was put into making the interfacing between processes as lightweight as possible, we found that in regards to motion planning, although simple, approximate planning algorithms can work well in most cases, generality and completeness when needed are priceless. Using a high-fidelity, high-dimensional lattice planner for unstructured planning problems proved time and time again to be the right choice for our system.

IX. Prior Work

Existing research on motion planning for autonomous outdoor vehicles can be roughly broken into two classes: motion planning for autonomous vehicles following roads and motion planning for autonomous vehicles navigating unstructured environments including off-road areas and parking lots. A key difference between road following and navigating unstructured environments is that in the former case a global plan is already encoded by the road (lane) itself, and therefore the planner only needs to generate short-term motion trajectories that follow the road, while in the latter case no such global plan is provided.

A. On-road Planning

A vast amount of research has been conducted in the area of road following. Some of the best known and fully-operational systems include the CMU NavLab project [7], the INRIA autonomous car project [8] in France, the VaMoRs [9] and VITA projects [10] in Germany, and the Personal Vehicle System (PVS) project [11] in Japan. Approaches to road following in these and other systems vary drastically. For example, one of the road following algorithms used by NavLab vehicles is ALVINN [12] which learns the mapping from road images onto control commands using Neural Network by observing how the car is driven manually for several minutes. In the VITA project, on the other hand, the planner was used to track the lane while regulating the velocity of the vehicle in response to the curvature of the road and the distance to nearby vehicles and obstacles. Stanford University's entry in the second DARPA Grand Challenge also exhibited lane following behavior through evaluating a set of candidate trajectories that tracked the desired path [13]. Our lane planning approach is closely related to theirs, however to generate their candidate trajectories they sample the control space around a base trajectory (e.g. the trajectory leading down the center of the lane), while we sample the state space along the road lane. Some significant advantages of using a state space approach include the ability to finely control position and heading at the

terminal state of each trajectory (which we can align with the road shape), the ability to impose the requirement that each trajectory terminates at exactly the same distance along the path, allowing for fairer evaluation of candidate actions, and the simplification of generating complex maneuvers such as U-turns and lane changes.

However, several of these existing approaches have been shown to be very effective in road following in normal conditions. A major strength of our approach, on the other hand, is that it can handle difficult scenarios, such as when a road is partially blocked (e.g., by an obstacle, a stalled car, a slow-moving car or a car driving in an opposite direction but moving out of the bounds of its own lane). Our system can handle these scenarios robustly and at high speeds.

B. Unstructured Planning

Roboticists have concentrated on the problem of mobile robot navigation in unstructured environments for several decades. Early approaches concentrated on performing local planning, where very short term reasoning is performed to generate the next action for the vehicle [14], [15]. A major limitation of these purely local approaches was their capacity to get the vehicle stuck in local minima en route to the goal (for instance, cul-de-sacs). To improve upon this limitation, algorithms were developed that incorporated global as well as local information [16], [17]. Subsequent approaches have focused on improving the local planning component of these approaches by using more sophisticated local action sets that better follow the global value function [13], [18], and by generating sequences of actions to perform more complex local maneuvers [19]. In parallel, researchers have concentrated on improving the quality of global planning, so that a global path can be easily tracked by the vehicle [20], [1], [21], [3]. However, the computational expense of generating complex global plans over large distances has remained very challenging, and the approaches to date have been restricted to either small distances, fairly simple environments, or highly suboptimal solutions. Our lattice-based global planner is able to efficiently generate feasible global paths over much larger distances than previously possible, while providing suboptimality bounds on the quality of the solutions and anytime improvement of the solutions generated.

X. Conclusions

We have presented the motion planning framework for an autonomous vehicle navigating through urban environments. Our approach combines a model-predictive trajectory generation algorithm for computing dynamically-feasible actions with an efficient lane-based planner, for on-road planning, and a 4D lattice planner, for planning in unstructured environments. It has been implemented on an autonomous vehicle that has traveled over 3000 autonomous kilometers and we have presented sample illustrations and results from the Urban Challenge, which it won in November 2007.

XI. Acknowledgements

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.

References

[1] 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.

[2] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

[3] R. Knepper and A. Kelly, "High performance state lattice planning using heuristic look-up tables," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[4] M. Likhachev and D. Ferguson, "Planning Dynamically Feasible Long Range Maneuvers for Autonomous Vehicles," in *Proceedings of Robotics: Science and Systems (RSS)*, 2008.

[5] C. Baker, D. Ferguson, and J. Dolan, "Robust Mission Execution for Autonomous Urban Driving," 2008, submitted to International Conference on Intelligent Autonomous Systems (IAS).

[6] DARPA, "DARPA Urban Challenge Official Results," 2008, posted at http://www.darpa.mil/GRANDCHALLENGE/mediafaq.asp.

[7] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.

[8] J. Baber, J. Kolodko, T. Noel, M. Parent, and L. Vlacic, "Cooperative autonomous driving: intelligent vehicles sharing city roads," *IEEE Robotics and Automation Magazine*, vol. 12, no. 1, pp. 44–49, 2005.

[9] E. Dickmanns, R. Behringer, C. Brudigam, D. Dickmanns, F. Thomanek, and V. Holt, "All-transputer visual autobahn-autopilot/copilot," in *Proceedings of the 4th Int. Conference on Computer Vision ICCV*, 1993, pp. 608–615.

[10] B. Ulmer, "VITA - an autonomous road vehicle (arv) for collision avoidance in traffic," in *Proceedings of Intelligent Vehicle Symposium*, 1992, pp. 36–41.

[11] A. Hattori, A. Hosaka, M. Taniguchi, and E. Nakano, "Driving control system for an autonomous vehicle using multiple observed point information," in *Proceedings of Intelligent Vehicle Symposium*, 1992.

[12] D. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[13] S. Thrun *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, August 2006.

[14] R. Simmons, "The curvature velocity method for local obstacle avoidance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1996.

[15] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance." *IEEE Robotics and Automation*, vol. 4, no. 1, 1997.

[16] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

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

[18] 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.

[19] C. Stachniss and W. Burgard, "An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[20] G. Song and N. Amato, "Randomized motion planning for car-like robots with C-PRM," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2001.

[21] M. Pivtoraiko and A. Kelly, "Constrained motion planning in discrete state spaces," in *Proceedings of the International Conference on Advanced Robotics (FSR)*, 2005.