

2008

Robust Mission Execution for Autonomous Urban Driving

Christopher R. Baker
Carnegie Mellon University

David I. Ferguson
Intel Research

John M. Dolan
Carnegie Mellon University

Follow this and additional works at: <http://repository.cmu.edu/robotics>

 Part of the [Robotics Commons](#)

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Robotics Institute by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

Robust Mission Execution for Autonomous Urban Driving

Christopher R. BAKER^{a,1}, David I. FERGUSON^b, and John M. DOLAN^a

^a*Carnegie Mellon University, Pittsburgh, PA, USA*

^b*Intel Research Pittsburgh, Pittsburgh, PA, USA*

Abstract. We describe a multi-modal software system for executing navigation missions in an urban environment, focusing on the robust treatment of anomalous situations such as blocked roads, stalled vehicles and tight maneuvering. Various recovery mechanisms are described relative to the nominal mode of operation, and results are discussed from the system's deployment in the DARPA Urban Challenge.

Keywords. Error Recovery, Autonomous Vehicles, Urban Challenge, Tartan Racing, Boss

Introduction

The ability to cope reliably with unforeseen circumstances is a critical feature of any fully autonomous robotic system. This is especially true in the case of complex, highly structured environments such as urban road networks in the context of the Urban Challenge [6], an autonomous vehicle competition sponsored by the US Defense Advanced Research Projects Agency (DARPA). Contestant robots were required to autonomously execute a series of navigation missions in a simplified urban environment consisting of roads, intersections, and parking lots, while obeying road rules and interacting safely and correctly with other vehicles. Unlike the previous challenge [1,2], which focused on rough-terrain navigation, this competition combined the possibilities of local obstructions with the uncertainties of interacting with other traffic, forming a complex problem space that defied rigorous treatment *a priori*. As with other mission-oriented, fully autonomous robotic systems [3], robust mission execution required a system for selecting and executing various recovery maneuvers aimed at overcoming or circumventing anomalous situations. This paper describes the mission recovery mechanisms employed by Boss, Tartan Racing's winning entry to the Urban Challenge, shown in Figure 1(a). Section 1 provides a brief overview of the total software system to provide a context for discussion of the robot's motion planning capabilities and error recovery mechanisms in Sections 2 and 3. The performance of the system during the Urban Challenge Final Event is reviewed in Section 4, and we conclude with key lessons learned in Section 5.

¹Corresponding Author: Christopher R. Baker, Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA; E-mail cbaker@andrew.cmu.edu

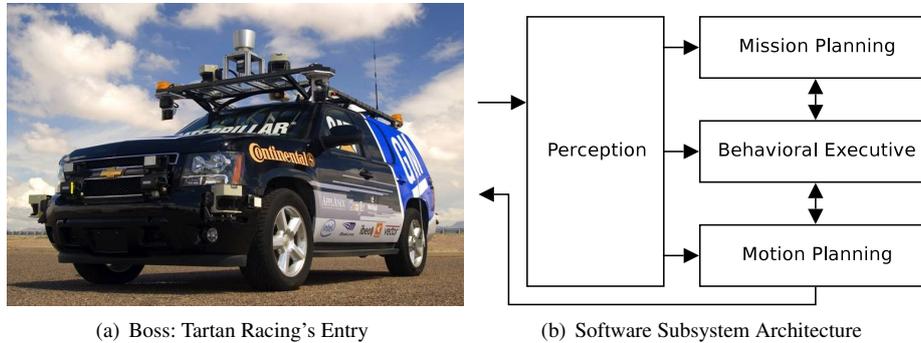


Figure 1. Total system structure for the Urban Challenge

1. System Architecture

The software system that controls Boss is divided into four primary subsystems, shown in Figure 1(b). They communicate via message-passing according to the anonymous publish-subscribe [4] pattern and work in concert to ensure safe, reliable and timely mission execution. Their responsibilities and interactions, discussed briefly below, are presented more thoroughly in [9].

The *Perception* subsystem processes sensor data from the vehicle and produces a collection of semantically-rich data elements such as the current pose of the robot, the geometry of the road network, and the location and nature of various obstacles such as road blockages and other vehicles.

The *Mission Planning* subsystem computes the fastest route to reach the next checkpoint from all possible locations within the road network, encoded as an estimated time-to-goal from each waypoint in the network. This estimate incorporates knowledge of road blockages, speed limits, intersection complexity, and the nominal time required to make special maneuvers such as lane changes or U-turns.

The *Behavioral Executive* combines the global route information provided by the Mission Planner with local traffic and obstacle information provided by Perception to select a sequence of incremental goals for the Motion Planning subsystem to execute. Typical goals include driving to the end of the current lane or maneuvering to a particular parking spot, and their issuance is predicated on conditions such as precedence at an intersection or the detection of certain anomalous situations. These anomalous situations trigger the selection of recovery goals as described in Section 3.

The *Motion Planning* subsystem is responsible for the safe, timely execution of the incremental goals issued by the Behavioral Executive. The isolation of goal selection from goal execution promotes the development of powerful, highly general planning capabilities, which fall into two broad contexts: on-road driving and unstructured driving. A separate path-planning algorithm is used for each context, and the nature and capabilities of each planner have a strong influence on the overall capabilities of the system, including the nature of common failure scenarios and the options for attempting recovery maneuvers. The following section describes these planning elements in greater detail to provide a foundation for discussion of the rest of the recovery system.

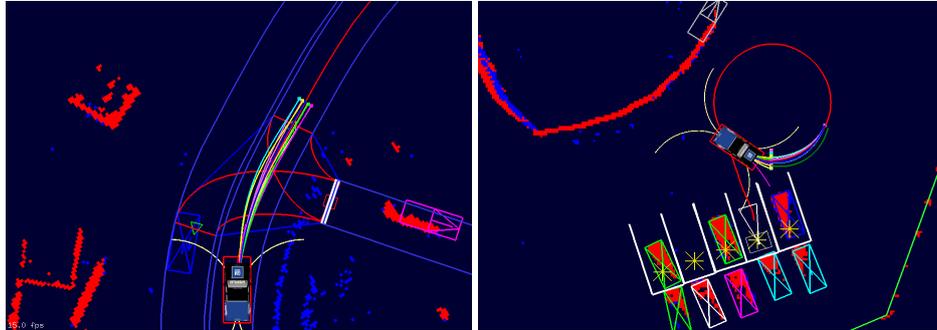


Figure 2. **Left:** Motion planning on roads. The centerline of the desired lane is shown (red) with the generated trajectories (various colors) extending along the lane. The obstacle map is overlaid (red cells are lethal; blue are curb estimates), and two detected vehicles are shown as rectangles. **Right:** Motion planning in unstructured areas. The vehicle is executing a path (red) to a parking spot between several other vehicles.

2. Motion Planning

Given a goal from the Behavioral Executive, the Motion Planner first creates a path towards the goal, then tracks it by generating a set of candidate trajectories following the path to varying degrees and selecting the best collision-free trajectory from this set. The algorithms used to generate this path are different for on-road and unstructured driving situations, as illustrated in Figure 2.

During on-road navigation, the goal from the Behavioral Executive describes a desired position along a lane. In this case, the Motion Planner constructs a curve along the centerline of the desired lane, representing the nominal path of the vehicle. A set of local goals are selected at a fixed longitudinal distance down this centerline path, varying in lateral offset to provide several options for the planner. Then, a model-based trajectory generation algorithm [7] is used to compute dynamically feasible trajectories to these local goals. The resulting trajectories are evaluated against their proximity to static and dynamic obstacles, their distance from the centerline path, their smoothness, and various other metrics, and the best trajectory is selected for execution by the vehicle. This technique is very fast and produces smooth, high-speed trajectories for the vehicle, but it can fail to produce feasible trajectories when confronted with aberrant scenarios, such as blocked lanes or extremely tight turns. In these cases, the recovery algorithms discussed in Section 3 call on the unstructured planner described below to extricate the system.

When driving in unstructured areas, the goal from the Behavioral Executive simply describes a desired pose for the vehicle, typically a parking spot or lot exit. To achieve this pose, a lattice planner 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 static and dynamic obstacles [8]. This planner is much more powerful and flexible than the on-road planner, but it also requires more computational resources and is limited to speeds below 15 mph. The recovery system leverages this flexibility to navigate congested intersections, perform difficult U-turns, and circumvent or overcome obstacles that block the progress of the vehicle. In these situations, the lattice planner is typically biased to avoid unsafe areas, such as oncoming traffic lanes, but this bias can be removed as the situation requires.

3. Behavioral Executive

During normal operation, the Behavioral Executive selects and issues goals to the Motion Planner according to three situational contexts:

- **Lane Driving**, in which the system traverses a road of one or more lanes while requiring maintenance of safe inter-vehicle separation and adherence 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.

The active context is determined by the location of the system within the road network and by the best incremental action to take from that location to reach the next checkpoint. That action is issued as a goal to the Motion Planning subsystem, and its execution is monitored until it is either completed successfully or it is determined to have failed. Goal failures can either be directly reported by the Motion Planner or inferred by monitoring forward progress over some span of time. In both cases, the failure is treated equally and triggers the selection and issuance of a recovery goal.

A good recovery goal selection algorithm should be able to generate a non-repeating sequence of novel recovery goals in the face of repeated failures *ad infinitum*. It should be sensitive to the original driving context, as each calls out a specific underlying planner which has certain properties that must be considered when selecting recovery goals. In addition, each context is governed by a specific set of road rules that must be adhered to if possible, but disregarded if necessary in a recovery situation. The recovery system should also be minimally complex so as to be implementable in the available time.

Keeping these requirements in mind, recovery goals are selected as a direct function of the original failed goal, which encodes the driving context, and a notion of the current *recovery level*. When in normal operation, the recovery level is set to zero, and it is incremented whenever any goal, normal or recovery, fails. The intent is that smaller values for the recovery level yield low-risk, easy-to-execute maneuvers that are tuned to handle common and benign situations. If these initial recovery goals fail, increasing values for the recovery level yield higher-risk maneuvers, enabling more drastic measures in more convoluted situations. It is important to note that the recovery goal selection process does not explicitly incorporate any surrounding obstacle data, making it intrinsically robust to transient environmental effects or perception artifacts that may cause spurious failures in other subsystems.

All recovery goals are specified as a span of poses to achieve, leveraging the power of the underlying lattice planner to perform whatever actions are necessary to get the system back on-course. The successful completion of any recovery goal resets the system back to normal operation, requiring that all recovery goals terminate at valid locations in the world. This eliminates the possibility of complex multi-maneuver recovery schemes, but it was deemed that situations requiring a more complex recovery system were simply out of scope. Should the same, or a very similar, normal goal fail immediately after a seemingly successful recovery sequence, the previous recovery level is reinstated instead of simply incrementing from zero to one. This bypasses the recovery levels that presumably failed to get the system out of trouble and immediately selects goals at a higher level.

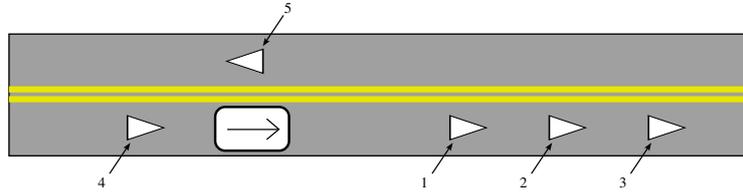


Figure 3. Example Lane Recovery Sequence

Each nominal driving context is backed by a unique recovery goal selection algorithm designed to complement the functionality of the underlying planner and observe as many situational rules as possible while still moving forward with the mission. The first and most commonly encountered recovery scenario is an error reported by the Motion Planner while driving down a lane. Typical causes for such a failure include:

- Small or transient obstacles, such as traffic cones, that do not block, but constrain the road such that the assumptions made by the lane planner are violated. In this case, a recovery goal specified a short distance ahead in the center of the lane will allow the lattice planner to quickly generate a path around the partial obstruction;
- Larger obstacles such as road barrels, or other cars that are detected too late for the distance-keeping behavior to come to a graceful stop, with the resultant pose requiring a back-up or other non-trivial maneuver to extricate the system;
- Low-hanging canopy that requires cautious and careful planning, perhaps including departure from the lane of travel, to guarantee the safety of the system;
- Lanes, especially virtual paths through an intersection, whose local shapes are kinematically infeasible, requiring a series of three-point turn maneuvers to realign the system with the road.

Keeping these and other similar causes in mind, the algorithm for lane recovery goal selection, illustrated in Figure 3, selects an initial set of goals forward along the lane from the original failure position with the distance forward described by:

$$D_{\text{recovery}} = D_{\text{offset}} + \text{RecoveryLevel} * D_{\text{incremental}} \quad (1)$$

The values for D_{offset} and $D_{\text{incremental}}$ were tuned through testing to $20m$ and $10m$ respectively, providing good results in most situations. These forward goals (Goals 1,2,3 in Figure 3) are constrained to remain in the original lane of travel and are selected out to some maximum distance (roughly $50m$, corresponding to the system's high-fidelity sensor-range), after which a goal is selected a short distance behind the vehicle (Goal 4) with the intent of backing up and getting a different perspective on the situation.

After backing up, the sequence of forward goals is allowed to repeat once more with a $5m$ offset, after which continued failure triggers higher-risk maneuvers depending on the nature of the road. If there is a lane available in the opposing direction, the road is eventually marked as fully blocked, and the system issues a U-Turn goal (Goal 5) and attempts to follow an alternate path to goal. Otherwise, the system is trapped on a one-way road, so the forward goal selection process from Equation 1 is allowed to continue forward beyond the $50m$ limit. These farther-reaching goals remove the constraint of staying in the original lane, giving the lattice planner complete freedom to achieve the goals by any means possible. In either case, these maneuvers disregard all normal rules

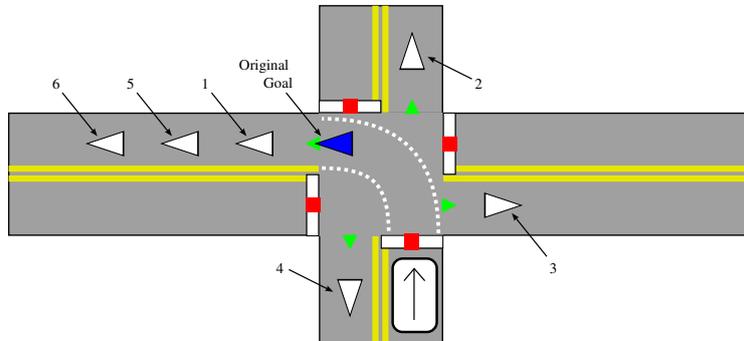


Figure 4. Example Intersection Recovery Sequence

governing interaction with other traffic on the road and thus are only issued at high recovery levels, typically greater than 20.

The second and perhaps most intricate scenario is recovering from failed intersection goals. The desired path through an intersection is a smooth interpolation between the two connected roads, as shown in Figure 4. It can often be narrow with high curvature and can intersect various obstacles such as guard rails and road signs, so the first and simplest recovery goal (Goal 1) is to try to achieve a pose slightly beyond the original goal using the lattice planner, giving the whole intersection as its workspace. This quickly recovers the system from small or spurious failures in intersections and compensates for intersections with turns that are beyond the one-pass kinematic capabilities of the robot.

If that first recovery goal fails, alternate routes out of the intersection (Goals 2,3,4) are attempted until all alternate routes to goal are exhausted. Thereafter, the system removes the constraint of staying in the intersection and selects pose goals incrementally deeper (Goals 5,6) into the lane or parking lot associated with the original failed goal. For lanes, this is almost identical to the selection of lane recovery goals, except that there is no initial offset. For parking lots, the goals are specified as incrementally deeper and larger spaces within the lot, as the specific terminal pose of the vehicle does not matter so long as it finds its way into the parking lot.

The case where specifics do matter, however, is the third recovery scenario, failures in parking lots. Because the lattice planner is always invoked when planning in parking lots, failure implies one of the following:

1. The path to the goal has been transiently blocked by another vehicle. While the rules guaranteed that parking spots will always be free, they made no such guarantees about traffic accumulations at lot exits, or about the number of vehicles that may be operating in the parking lot at any one time. In these cases, the system should try to move close to the original goal and allow the blockage some time to pass.
2. Due to some sensor artifact, the system believes there is no viable path to goal. In this case, selecting nearby goals that offer different perspectives on the original goal area may relieve the situation.
3. The path to the goal is genuinely fully blocked and the parking lot is effectively subdivided. This was considered to be beyond the scope and spirit of the competition and is very challenging to handle in the general case.

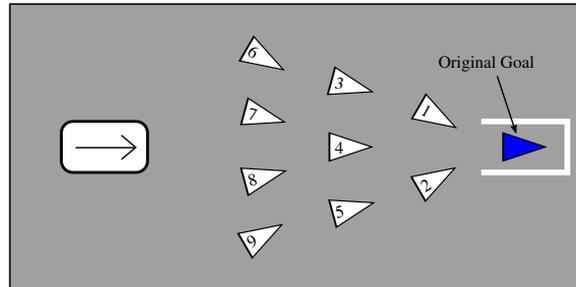


Figure 5. Example Parking Lot Recovery Sequence

The goal selection algorithm for failed parking lot goals selects a sequence of goals in a regular, triangular pattern, centered on the original goal as shown in Figure 5. If failures continue through the entire pattern, the next set of actions is determined by the exact nature of the original goal. For parking spot goals, the pattern is repeated with a small incremental angular offset *ad infinitum*, guided by the assumption that the parking spot is empty. Over time, all possible approaches to the parking spot may be attempted, but the system will not proceed on the mission until the parking spot has been attained². Lot exits, on the other hand, will be marked as blocked and the system will attempt to re-route through alternate exits in a way similar to the alternate path selection in the intersection recovery algorithm above.

If all other exits are exhausted, the system issues goals extending outward along the road network from the exit according to a semi-random breadth-first search. In this fourth and last-ditch recovery strategy, increasing values of the recovery level call out farther paths, and the lattice planner is not constrained to stay on roads or in parking lots. These high-risk, loosely specified goals are meant to be invoked when each of the other recovery goal selection schemes has been exhausted. In addition to parking lot exits, these are selected for lane recovery goals that run off the end of the lane and similarly for intersection recovery goals when all other attempts to get out of an intersection have failed.

Through these four recovery algorithms, all possible paths forward can be explored from any single location, leaving only the possibility that the system is stuck due to a sensor artifact or software bug that requires a small local adjustment to escape, as opposed to the selection of a farther, riskier goal. To compensate for this possibility, a completely separate recovery mechanism monitors the vehicle's absolute motion, and if the vehicle does not move at least one meter every five minutes, it overrides the current goal with a randomized local maneuver. When that maneuver is completed, the entire goal selection system is re-initialized from the new location, clearing error recovery and execution state for another attempt. The goals selected by this recovery mechanism are meant to be kinematically feasible and can be either in front of or behind the vehicle's current pose. They are biased somewhat forward of the robot's position so there is statistically net-forward motion if the robot is forced to choose these goals repeatedly over time. This functionality *suppresses* the functionality of the other goal selection systems, and is analogous to the "Wander" behavior from [5].

²The parking spot could easily be bypassed after some number of attempts, but the penalty for skipping a checkpoint was unspecified and may have been interpreted as a failed mission.

	Mission Run Time	Recovery Events	Recovery Time		Single-Level Recoveries	Multi-Level Recoveries
			Total	Average		
Mission 1	89m, 41s	10	8m, 18s	49.8s	8	2(9, 22)
Mission 2	56m, 21s	3	39s	13s	3	0
Mission 3	85m, 53s	4	2m, 37s	39.2s	3	1(23)
Cumulative	231m, 55s	17	11m, 34s	40.8s	14	3

Table 1. Recovery System Performance in the Urban Challenge Final Event

4. Event Analysis

The Urban Challenge Final Event (UCFE) consisted of a series of three missions, covering roughly 60 miles of urban roads and meant to be completed in less than 6 hours. Analysis of over 140 gigabytes of logs from the event shows that the recovery system played an important role in the vehicle’s success, being called on to recover the system from over a dozen anomalous situations as summarized in Table 1.

Of the 17 recovery events during the UCFE, only three exceeded the first recovery level. These more complicated situations consumed 5m, 41s, representing nearly half of the total time spent in recovery, and can be traced to small collections of software bugs or sensor malfunctions. The level-9 recovery in the first mission was incited by an *all-pause*³ event that lasted roughly ten minutes. When the *all-pause* was lifted, recovery goals were issued along the road, but the planner was unable to find a collision-free action away from a roadside barrier due to GPS drift while the vehicle was paused. Eventually, the positioning system corrected itself and the robot was able to proceed. The primary contribution of the recovery system in this instance was the continual issuance of novel goals in the face of repeated failures, though no one particular goal resolved the situation.

The two higher-order recovery events, level 22 in mission one and level 23 in mission three, were due to a combination of perception and planning bugs that caused the system to believe that the path forward along the current lane was completely and persistently blocked by other traffic where there was, in fact, no traffic at all. After repeated attempts to move forward, the recovery system declared the lane fully blocked, commanded a U-Turn, and followed an alternate route to goal. This demonstrated the recovery system’s ability to compensate for subtle but potentially lethal bugs in other components.

Discarding these more complex events, the remaining 14 were resolved in an average of 25 seconds and had a span of causes ranging from dust and transient sensor artifacts to vehicles proceeding out of turn at intersections. In all cases, the system recovered quickly and gracefully, demonstrating an appropriate emphasis on the selection of nearby, low-risk recovery maneuvers early in the recovery sequence.

5. Conclusions

The results from deployment in the UCFE show that the recovery system was well suited to the challenges posed by this competition. The decision against complex environmental reasoning in favor of comparatively simple, incremental goal selection algorithms led to

³Urban Challenge officials had the ability to temporarily disable all vehicles on the course to, for example, remove a disabled vehicle from the course or to resolve some dangerous situation.

a highly robust mechanism for overcoming a wide range of anomalous circumstances. The ability to quickly recover from simple failures also provided a safety net for all other elements of the system, as transient false-positives from obstacle detection systems or tracking failures from the Motion Planning subsystem were easily tolerated, often transparently so to an external observer. More complex scenarios often took longer than desirable to resolve, but the relative infrequency of these scenarios, when coupled with the fact that the system eventually recovered itself in every occurrence, demonstrate again the system's effectiveness in the scope of the Urban Challenge. That the Final Event did not exercise the deepest reaches of the various recovery mechanisms is a testament to the system's preparedness and to the importance of rigorous, clever and sometimes adversarial testing by a highly dedicated test team.

While many of the details of the recovery heuristics were tailored to the nature of the competition, several core concepts are generally applicable to autonomous mobile robots. Splitting the navigation problem into separate entities for goal selection and execution allows for the development of powerful, highly general motion planning capabilities which can then be leveraged by any goal selection mechanism to explore many possible resolutions to any one failure. Given a powerful underlying planner, the blind selection of incrementally higher-risk recovery goals is an effective means of quickly and safely resolving benign situations while retaining the ability to incrementally discard environmental rules regarding structure and interaction in favor of continuing onward. Most importantly, an effective recovery mechanism is a critical element of any fully autonomous robotic system, providing alternate paths forward from both internal faults and external obstacles and guaranteeing the timely completion of the system's mission.

Acknowledgments

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] Special Issue on the DARPA Grand Challenge, Part 1. *Journal of Field Robotics*, 23(8), 2006.
- [2] Special Issue on the DARPA Grand Challenge, Part 2. *Journal of Field Robotics*, 23(9), 2006.
- [3] Christopher Baker et al. A campaign in autonomous mine mapping. In *Proceedings of the IEEE Conference on Robotics and Automation (ICRA)*, volume 2, pages 2004 – 2009, April 2004.
- [4] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 2003.
- [5] Rodney A Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.
- [6] Defense Advanced Research Projects Agency (DARPA). Urban challenge website, July 2007. <http://www.darpa.mil/grandchallenge>.
- [7] T. Howard and A. Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. *International Journal of Robotics Research*, 26(2):141–166, 2007.
- [8] M. Likhachev and D. Ferguson. Planning Dynamically Feasible Long Range Maneuvers for Autonomous Vehicles. In *Proceedings of Robotics: Science and Systems (RSS)*, 2008.
- [9] Chris Urmson et al. Autonomous Driving in Urban Environments: Boss and the DARPA Urban Challenge. *Accepted to Journal of Field Robotics*, 2008.