Autonomous Driving in Traffic: Boss and the Urban Challenge

Chris Urmson, Chris Baker, John Dolan, Paul Rybski, Bryan Salesky, William "Red" Whittaker, Dave Ferguson, and Michael Darms

■ *The DARPA Urban Challenge was a competition to develop autonomous vehicles capable of safely, reliably, and robustly driving in traffic. In this article we introduce Boss, the autonomous vehicle that won the challenge. Boss is a complex artificially intelligent software system embodied in a 2007 Chevy Tahoe. To navigate safely, the vehicle builds a model of the world around it in real time. This model is used to generate safe routes and motion plans both on roads and in unstructured zones. An essential part of Boss's success stems from its ability to safely handle both abnormal situations and system glitches.*

In 2003 the Defense Advanced Research Projects Agency (DARPA) announced the first Grand Challenge with the goal of developing vehicles capable of autonomously navigating desert trails and roads at high speeds. The competition was generated as a response to a mandate from the United States Congress that a third of U.S. military ground vehicles be unmanned by 2015. To achieve this goal DARPA drew from inspirational successes such as Charles Lindbergh's May 1927 solo transatlantic flight aboard the *Spirit of St. Louis* and the May 1997 IBM Deep Blue chess victory over reigning world chess champion Garry Kasparov. Instead of conventional funded research, these successes were organized as challenges with monetary prizes for achieving specific goals.

DARPA's intent was not only to develop technology but also to rally the field of robotics and ignite interest in science and engineering on a scale comparable to the Apollo program. Thus, the Grand Challenges were born. Organized as competitions with cash prizes awarded to the winners ($1 million in 2004; $2 million in 2005; and $2 million for first, $1 million for second, and $500,000 for third in 2007). For the first two years the government provided no research funding to teams, forcing them to be resourceful in funding their endeavours. For the 2007 Urban Challenge, some research funding was provided to teams through research contracts. By providing limited funding, the competition encouraged teams to find corporate sponsorship, planting the seeds for strong future relationships between academia (the source of many of the teams) and industry.

The Grand Challenge[1] was framed as a cross-country race where vehicles had to drive a prescribed route in the minimum time. There would be no moving obstacles on the course, and no one was allowed to predrive the course. To complete the

Copyright © 2009, Association for the Advancement of Artificial Intelligence. All rights reserved. ISSN 0738-4602

Figure 1. The Urban Challenge Final Event.

Eleven teams qualified for the Urban Challenge final event. Eighty-nine teams from the United States and countries as far away as China registered for the competition.

challenge, the autonomous vehicles would need to drive the route in less than 10 hours. The competition was scoped to require driving both on and off road, through underpasses, and even through standing water. These diverse challenges lead to a field with a wide variety of vehicles, ranging from military high-mobility multipurpose wheeled vehicles (HMMWVs) and sport utility vehicles (SUVs) to motorcycles and custom-built dune buggies.

The first running of the Grand Challenge occurred on March 3, 2004. The 142-mile course was too difficult for the field, with the best vehicle traveling 7.4 miles. Though described as a failure in the popular press, the competition was a technical success; though the vehicles drove a small fraction of the course, the top competitors did so at high, sustained speeds that had not been previously demonstrated. Given the excitement and promise generated by this first challenge, the competition was rerun in 2005.

The 2005 Grand Challenge began at 6:30 AM on October 8. Vehicles launched at intervals from three start chutes in the Mojave desert. Approximately 7 hours later, the first of five finishers completed the challenge. Stanley (Thrun et al. 2006), from Stanford University, finished first, in 6 hours and 53 minutes, followed soon after by the two Carnegie Mellon robots, Sandstorm and H1ghlander (Urmson et al. 2006). Kat-5, funded by the Gray Insurance Company, and TerraMax, a 30,000-pound behemoth developed by the OshKosh Truck Corporation, rounded out the finishers. With five vehicles completing the course, the 2005 Grand Challenge redefined the perception of the capability of autonomous mobile robots.

The DARPA 2007 Urban Challenge[2] continued the excitement generated by the previous challenges, engaging researchers from around the world to advance intelligent, autonomous vehicle technology (figure 1). Competitors were required to develop full-size autonomous vehicles cable of driving with moving traffic in an urban environment at speeds up to 30 miles per hour (mph). The vehicles were required to demonstrate capabilities that were undeniably intelligent, including obeying traffic rules while driving along roads, negotiating intersections, merging with moving traffic, parking, and independently handling abnormal situations.

The Urban Challenge was fundamentally about intelligent software. The most visible artifacts of the research and development are the numerous autonomous vehicles that were entered in the competition, but the magic is really in the algorithms that make these vehicles drive. The competition was a 60-mile race through urban traffic; a drive that many people face daily. Thus the challenge was not in developing a mechanism that could survive these rigors (most automotive manufacturers do this already), but in developing the algorithms and software that would robustly handle driving, even during unexpected situations. To complete the challenge, vehicles had to consistently display safe and correct driving behavior; any unsafe or illegal maneuver could result in elimination from the competition.

In this article we present an overview of Boss, the vehicle that won the Urban Challenge (figure 2). Boss was developed by a team of researchers and students from Carnegie Mellon University, General Motors, Caterpillar, Continental, and Intel. Boss drives at speeds up to 30 mph by monitoring its location and environment and constantly updating its motion plan and the model of the world around it. Throughout this article we provide pointers to other publications that delve into each area in greater depth.

Figure 2. Boss.

Boss uses a combination of 17 sensors to navigate safely in traffic at speeds up to 30 mph.

The Vehicle

Boss is a 2007 Chevrolet Tahoe modified for autonomous driving. While it was built from the chassis up for the Urban Challenge, it incorporated many lessons learned from the previous challenges. Like most of the vehicles that competed in the Urban, we built Boss from a standard chassis, to reduce mechanical complexity and benefit from the reliability of a highly engineered platform. The Tahoe provides ample room to mount sensors, computers, and other components while maintaining seating for four, which is important for safe and efficient testing and development. Computer control of the vehicle is made possible by using electric motors to actuate the brake pedal, gear selector, and steering column and through controller area network (CAN) bus communication with the engine control module. Despite these changes, the Tahoe maintains its standard human driving controls, and pressing a button reverts it to a human-driven vehicle.

Boss integrates a combination of 17 different sensors (a mixture of light detection and ranging [lidar], radio detection and ranging [radar], and global positioning system [GPS]). These sensors are mounted to provide a full 360 degree field of view around the vehicle. In many cases, sensor field of views overlap, providing redundancy and in some cases complementary information. The primary sensor is a Velodyne HDL-64 lidar, mounted on the top of the vehicle. This lidar provides dense range data around the vehicle. Three Sick laser range finders are used to provide a virtual bumper around the vehicle while two other long-range lidars are used to track vehicles at ranges of up to 100 m. Three Continental ACC radars are mounted in the front (two) and rear (one) bumpers to detect and track moving vehicles at long ranges. In addition, a pair of Continental sensors (lidar and

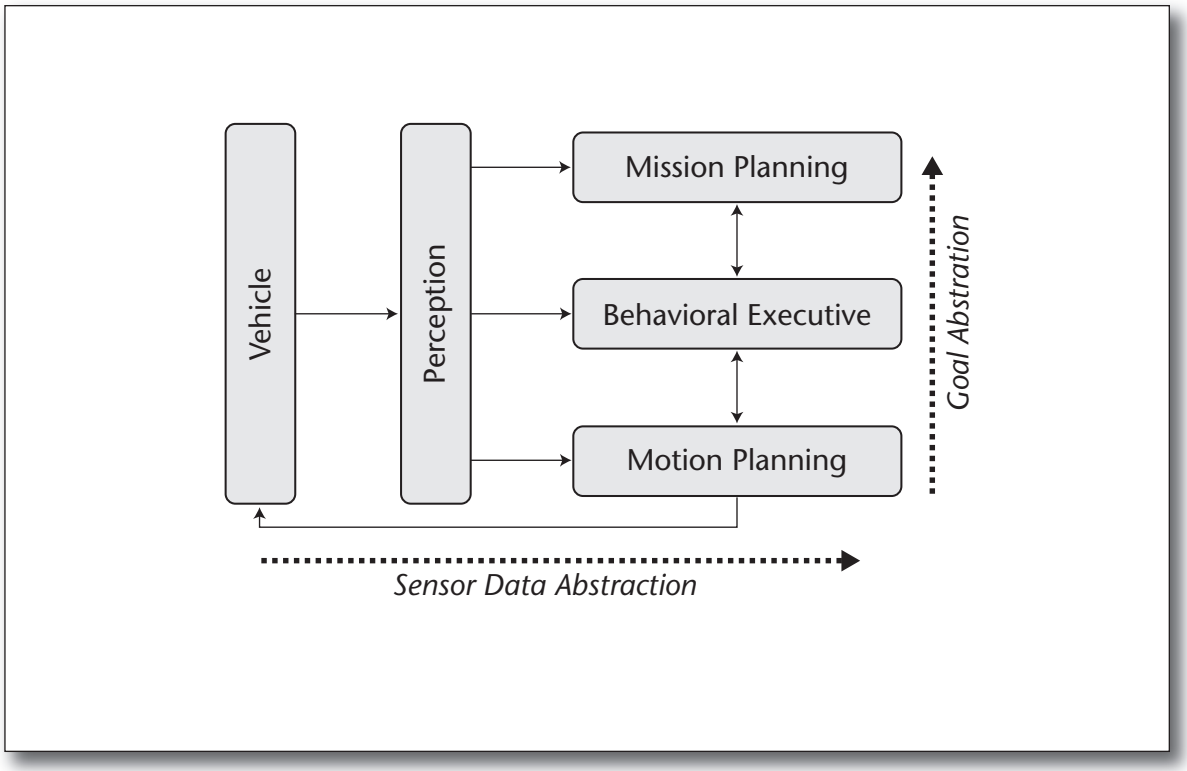

Figure 3. The Boss Software Architecture.

The software architecture consists of a perception layer that provides a model of the world to a classical three-tier control architecture.

radar) are mounted on pointable bases, enabling Boss to "turn its head" to look long distances down cross streets and merging roads. To determine its location, Boss uses a combination of GPS and inertial data fused by an Applanix POS-LV system. In addition to this system, Boss combines data from a pair of down looking SICK laser range finders to localize itself relative to lane markings described in its on-board map. The number of sensors incorporated into Boss provides a great deal of information to the perception system.

For computation, Boss uses a CompactPCI chassis with ten 2.16 gigahertz Core2Duo processors, each with 2 gigabytes of memory and a pair of gigabit Ethernet ports. Each computer boots off of a 4 gigabyte flash drive, reducing the likelihood of a disk failure. Two of the machines also mount 500 gigabyte hard drives for data logging, and all are time synchronized through a custom pulse-per-second adapter board. The computer cluster provides ample processing power, so we were able to focus our efforts on algorithm development rather than software optimizations.

Software Architecture

To drive safely in traffic, the software system must be reliable and robust and able to run in real time. Boss's software architecture is fundamentally a distributed system, enabling decoupled development and testing. In all, 60 processes combine to implement Boss's driving behavior. Architecturally these are clustered into four major components: perception, mission planning, behavioral execution, and motion planning (figure 3).

The *perception* component interfaces to Boss's numerous sensors and fuses the data into a coherent world model. The world model includes the current state of the vehicle, the current and future positions of other vehicles, static obstacles, and the location of the road. The perception system uses a variety of probabilistic techniques in generating these estimates.

The *mission planning* component computes the fastest route through the road network to reach the next checkpoint in the mission, based on knowledge of road blockages, speed limits, and the nominal time required to make special maneuvers such as lane changes or U-turns. The route is actually a policy generated by performing gradient descent on a value function calculated over the road network. Thus the current best route can be extracted at any time.

The *behavioral executive* executes the mission

plan, accounting for traffic and other features in the world (Ferguson et al. 2008). It encodes the rules of the roads and how and when these rules should be violated. It also implements an error recovery system that is responsible for handling off-nominal occurrences, such as crowded intersections or disabled vehicles. The executive is implemented as a family of state machines that translate the mission plan into a sequence of motion planning goals appropriate to the current driving context and conditions.

The *motion planning* component takes the motion goal from the behavioral executive and generates and executes a trajectory that will safely drive Boss toward this goal, as described in the following section. Two broad contexts for motion planning exist: on-road driving and unstructured driving.

Software Components

In this section we describe several of the key software modules (world model, moving obstacle detection and tracking, motion planning, intersection handling, and error recovery) that enable Boss to drive intelligently in traffic.

World Model

Boss's world model is capable of representing environments containing both static and moving obstacles. While this representation was developed for the Urban Challenge, it can be also used more broadly for general autonomous driving. The world model includes the road network, static obstacles, tracked vehicles, and the position and velocity of Boss. The road network defines where and how vehicles are allowed to drive, traffic rules (such as speed limits), and the intersections between roads. The road representation captures sufficient information to describe the shape (the number of lanes, lane widths, and geometry) and the rules governing driving on that road (direction of travel, lane markings, stop lines, and so on).

Static obstacles are those that do not move during some observation period. This includes obstacles off and on road. Examples of static obstacles include buildings, traffic cones, and parked cars. In contrast, dynamic obstacles are defined as objects that may move during an observation period. With this definition all vehicles participating actively in traffic are dynamic obstacles even if they are temporarily stopped at an intersection. By these definitions an object is either static or dynamic; it should not be represented in both classes. An object can, however, change from static to dynamic and vice versa: a person may, for example, get into a parked car and actively begin participating in traffic. The information about whether an object is static or dynamic is not tied exclusively to its motion; it is an interpretation of the current state of the world around the vehicle.

Each dynamic obstacle is represented by a series of oriented rectangles and velocities. Each rectangle represents the state, or predicted state, of the obstacle at an instant in time over the next few seconds. Additionally, each obstacle can be flagged as either *moving* or *not moving* and as either *observed moving* or *not observed moving*. An obstacle is flagged as moving if its instantaneous velocity estimate is above some minimum noise threshold. The obstacle is not flagged as observed moving until the instantaneous velocity remains above a noise floor for some threshold time. Similarly, a moving obstacle does not transition from observed moving to not observed moving until its instantaneous velocity remains below threshold for some large amount of time. These flags provide a hysteretic semantic state that is used in the behavioral executive to improve behavior while maintaining the availability of full fidelity state information for motion planning and other modules.

Static obstacles are represented in a grid of regularly spaced cells. The advantages of this representation are that it provides sufficient information for motion planning and is relatively inexpensive to compute and query. The disadvantages are that this simple representation makes no attempt to classify the type of obstacle and uses a relatively large amount of memory. To generate the map, range data is first segmented into ground and nonground returns and then accumulated in a map using an occupancy gridlike algorithm. To maintain a stable world representation, dynamic obstacles that are flagged as not observed moving and not moving are painted into the map as obstacles. Once a dynamic obstacle achieves the state of observed moving, the cells in the map that it overlaps are cleared. In the process of building the static obstacle map, an instantaneous obstacle map is generated. This map contains all static and dynamic obstacles (that is, data associated to dynamic obstacle hypotheses with the observed moving flag set are not removed). This map is much noisier than the static obstacle map and is used only for validating potential moving obstacle tracks within the dynamic obstacle tracking algorithms.

Moving Obstacle Detection and Tracking

The moving obstacle detection and tracking algorithms fuse sensor data to generate a list of any dynamic obstacles in a scene (Darms, Rybski, and Urmson 2008). The tracking system is divided into two layers: a sensor layer and a fusion layer (see figure 4). The sensor layer includes modules that are specialized for each class of sensor and runs a module for each physical sensor on the vehicle. The sensor layer modules perform sensor-specific tasks such as feature extraction, validation, association,

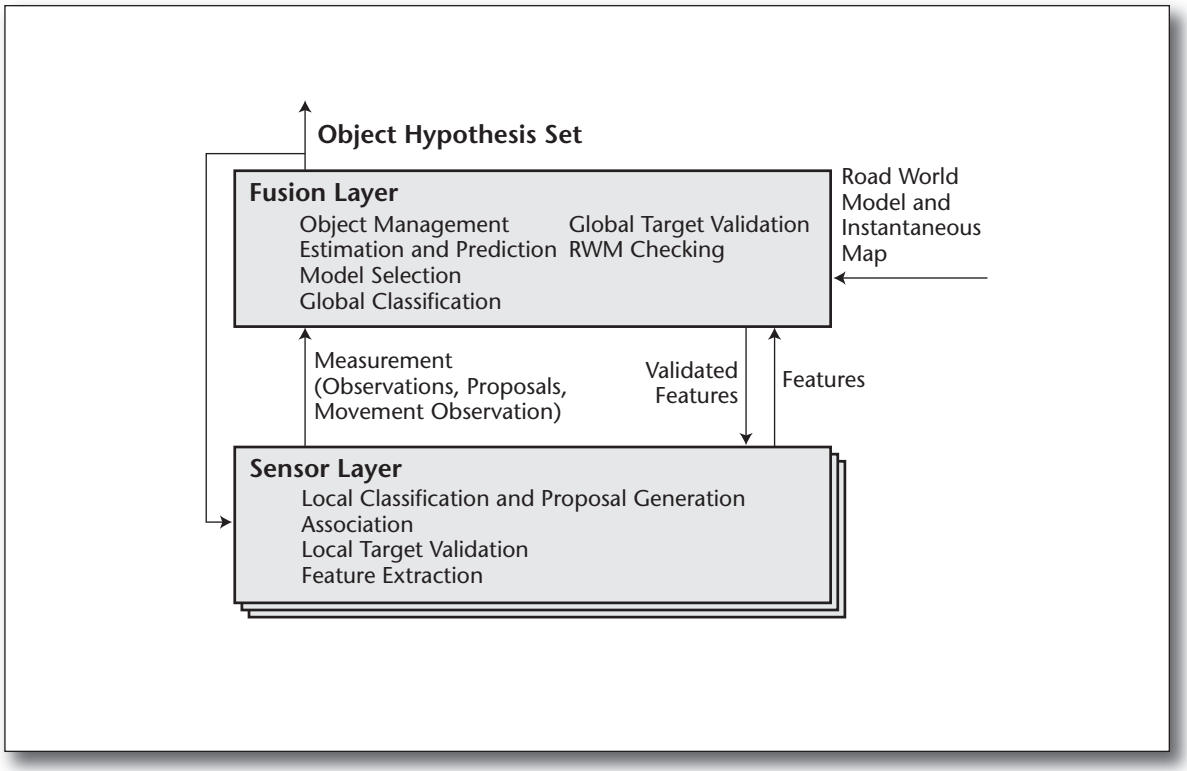

Figure 4. The Moving Obstacle Tracking System Is Decomposed into Fusion and Sensor Layers.

and proposal and observation generation. Each of these tasks relies on sensor-specific knowledge, generally encoded as heuristics; these are the important details that enable the machinery of the Kalman filter to work correctly. For example, radar sensors may generate a specific class of false echoes that can be heuristically filtered, but the filtering approach would be irrelevant for a lidar or even a different class of radar. By encapsulating these sensor-specific details within sensor modules we were able to modularize the algorithms and maintain a clean, general implementation in the fusion layer.

The fusion layer combines data from each of the sensor modules into the dynamic obstacle list, which is communicated to other components. The fusion layer performs a global validation of sensor observations, selects an appropriate tracking model, incorporates measurements into the model and predicts future states of each obstacle. The global validation step uses the road network and instantaneous obstacle map to validate measurements generated by the sensor layer. Any observation either too far from a road, or without supporting data in the instantaneous obstacle map, is discarded. For each remaining observation, an appropriate tracking model is selected, and the state estimate for each obstacle is updated. The future state of each moving obstacle is generated by either extrapolating its track (over short time intervals) or by using the road network and driving context (for long time intervals).

Motion Planning

Motion planning is responsible for moving the vehicle safely through the environment. To do this, it first creates a path toward 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. During on-road navigation, the motion planner generates its nominal path along the center line of the desired lane (Ferguson, Howard, and Likhachev 2008a). A set of local goals are selected at a fixed longitudinal distance down this center line path, varying in lateral offset to provide several options for the planner. Then, a model-based trajectory generation algorithm (Howard et al. 2008) 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 center line path, their smoothness, and various other metrics to determine the best trajectory for execution by the vehicle. This technique is sufficiently 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 error recovery system is called on to rescue the system.

When driving in unstructured areas the motion planner's goal is to achieve a specific pose (Ferguson, Howard, and Likhachev 2008b). To achieve a pose, a lattice planner searches over vehicle position, orientation, and velocity to generate a sequence of feasible maneuvers that are collision free with respect to static and dynamic obstacles. 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.

Intersection Handling

Correctly handling precedence at intersections is one function of the behavioral executive (Baker and Dolan 2008). Within the executive, the *precedence estimator* has primary responsibility for ensuring that Boss is able to safely and robustly handle intersections. The precedence estimator encapsulates all of the algorithms for determining when it is safe to proceed through an intersection. Before entering the intersection, the intersection must be clear (both instantaneously and throughout the time it predicts Boss will take to travel through the intersection) and Boss must have precedence. The precedence order is determined using typical driving rules (that is, the first to arrive has precedence through the intersection and in cases of simultaneous arrival, yield to the right).

The precedence estimator uses knowledge of the vehicle's route, to determine which intersection should be monitored, and the world model, which includes dynamic obstacles around the robot, to calculate the precedence ordering at the upcoming intersection. To ensure robustness to temporary noise in the dynamic obstacle list, the algorithm for computing precedence does not make use of any information beyond the geometric properties of the intersection and the instantaneous position of each dynamic obstacle. Instead of assigning precedence to specific vehicles, precedence is assigned to individual stop lines at an intersection, using a geometric notion of "occupancy" to determine arrival times. Polygons are computed around each stop line, extending backward along the incoming lane by a configurable distance and exceeding the lane geometry by a configurable margin. If the front bumper of a moving obstacle is inside this polygon, it is considered an occupant of the associated stop line. Boss's front bumper is added to the pool of front bumpers and is treated in the same manner to ensure equitable estimation of arrival times. Figure 5 shows a typical occupancy polygon extending three meters back along the lane from the stop line. Once the polygon becomes occupied the time is recorded and used to estimate precedence. To improve robustness of the dynamic obstacle list, the occupancy state is not cleared instantly when there is no obstacle in the polygon. Instead, the polygon remains occupied for a small amount of time. Should an obstacle reappear in the polygon in this time, the occupancy state of the polygon is maintained. This allows moving obstacles to flicker out of existence for short intervals without disturbing the precedence ordering.

In cases where Boss stops at an intersection and none of the other vehicles move, Boss will wait 10 seconds before breaking the deadlock. In this case the precedence estimator assumes that the other traffic at the intersection is not taking precedence and asserts that Boss has precedence. In these situations other elements of the behavioral executive limit the robot's maximum speed to 5 mph, causing it to proceed cautiously through the intersection. Once the robot has begun moving through the intersection, there is no facility for aborting the maneuver, and the system instead relies on the motion planner's collision-avoidance algorithms and, ultimately, the intersection recovery algorithms to guarantee safe, forward progress should another robot enter the intersection at the same time.

Error Recovery

Despite best efforts to implement a correct and reliable system, there will always be bugs in the implementation and unforeseen and possibly illegal behavior by other vehicles in the environment. The error recovery system was designed to ensure that Boss handles these situations and "never gives up." To this end, a good recovery algorithm should generate a sequence of nonrepeating motion goals that gradually accept greater risk; it should adapt behavior to the driving context, since each context includes road rules that should be adhered to if possible and disregarded only if necessary in a recovery situation; and finally, the recovery system should also be kept as simple as possible to reduce the likelihood of introducing further software bugs.

Our approach is to use a small family of error recovery modes and maintain a recovery level. Each recovery mode corresponds to a nominal driving context (driving down a lane, handling an intersection or maneuvering in an unstructured area). When in normal operation, the recovery level is zero, but as failures occur, it is incremented. The recovery level is used to encode the level of risk that is acceptable. It is important to note that

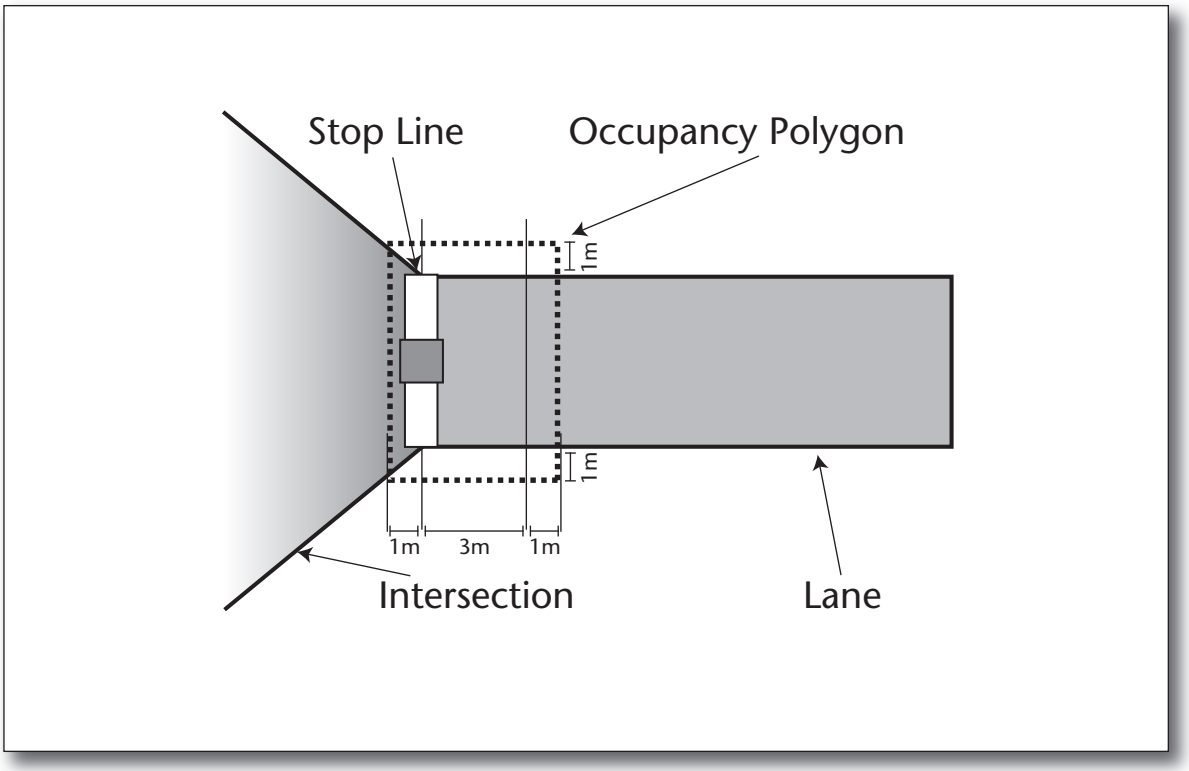

Figure 5. A Typical Occupancy Polygon for a Stopline.

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 maneuvers are specified as a set of poses to be achieved and utilize the movement zone lattice planner to generate paths through potentially complicated situations. The successful completion of any recovery goal returns the system to normal operation, resetting the recovery level. 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 a goal at a higher level.

The on-road recovery algorithm involves generating goals at incrementally greater distances down the current road lane to some maximum distance. These forward goals (goals 1, 2, 3 in figure 6a) are constrained to remain in the original lane of travel. If none of the forward goals succeed, a goal is selected a short distance behind the vehicle (goal 4) with the intent of backing up to provide a new perspective. After backing up, the sequence of forward goals is allowed to repeat once more with an offset, after which continued failure triggers higher-risk maneuvers. For example, if there is a lane available in the opposing direction, the road is eventually marked as blocked, and the system issues a U-turn goal (goal 5) and attempts to follow an alternate path to the goal. The intersection recovery algorithm involves attempting to navigate alternate paths through an intersection and then alternative routes through the road network. The desired path through an intersection is a smooth interpolation between the incoming and outgoing lanes that may be blocked by some obstacle in the intersection or may be infeasible to drive. Thus, the first recovery level (goal 1 in figure 6b) is to try to achieve a pose slightly beyond the original goal using the lattice planner, with 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 tighter than the vehicle can easily make. If the first recovery goal fails, alternate routes out of the intersection (goals 2, 3, 4) are attempted until all alternate routes to the goal are exhausted. Thereafter, the system removes the constraint of staying in the intersection and selects goals incrementally further out from the intersection (goals 5, 6).

The unstructured zone recovery procedure varies depending on the current objective. Initially, the algorithm selects a sequence of goals in a regular,

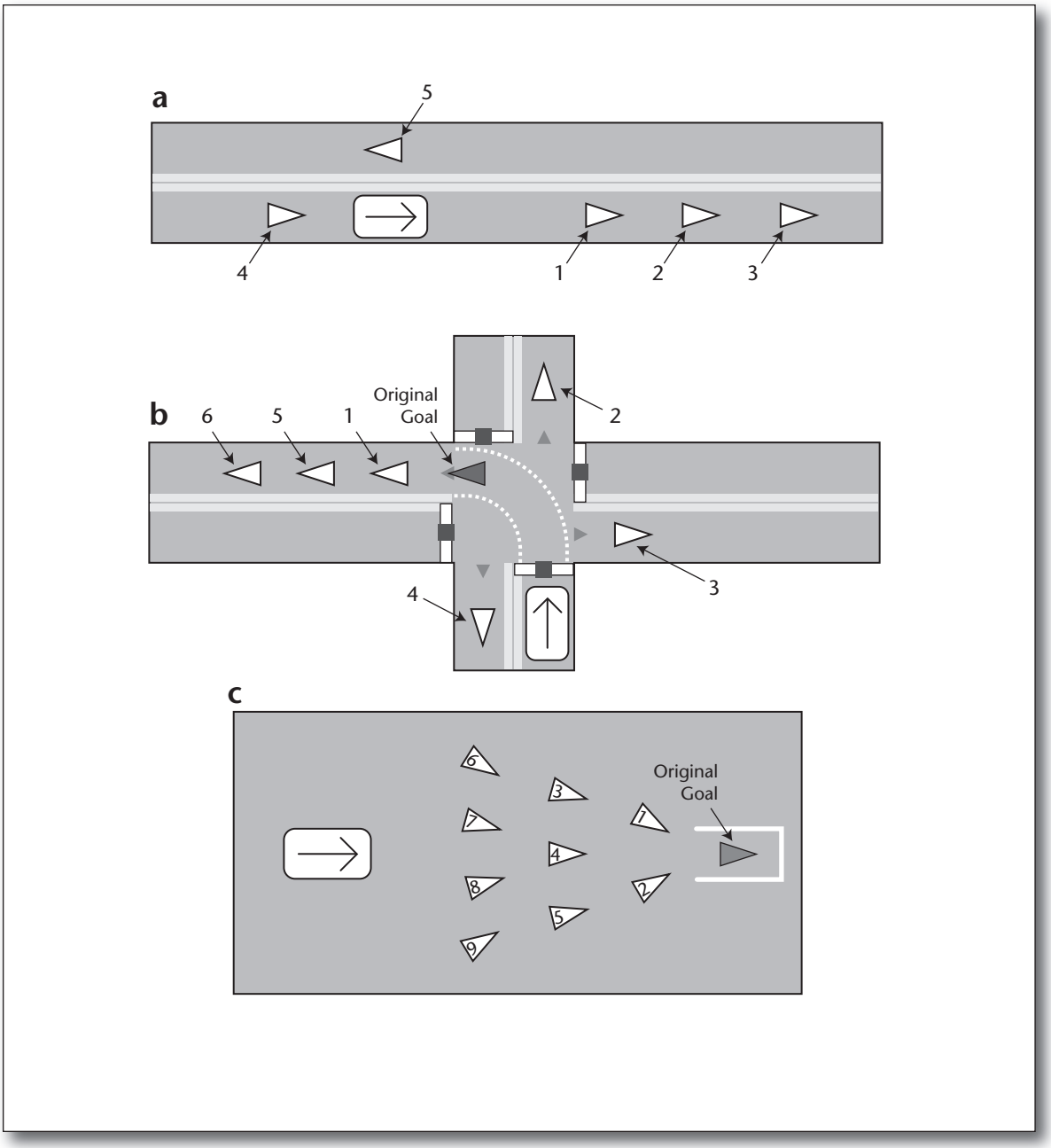

Figure 6. Boss Uses Context-Specific Recovery Goals to Provide Robustness to Software Bugs and Abnormal Behavior of Other Vehicles.

triangular pattern, centered on the original goal as shown in figure 6(c). If, after attempting this complete pattern, the vehicle continues to fail to make progress, a more specialized recovery process is invoked. For parking spot goals, the pattern is continuously repeated with small incremental angular offsets, based on the assumption that the parking spot will eventually be correctly perceived as empty, or the obstacle (temporarily) in the spot will move. If the current goal is to exit the parking lot, the algorithm tries first to use alternative exits, if those exits fail, then the algorithm issues goals to locations on the road network outside of the parking lot, giving the lattice planner free reign in generating a solution out of the parking lot.

An independent last-ditch recovery system monitors the vehicle's motion. If the vehicle does not move at least one meter every five minutes, it overrides any other goals with a randomly selected local motion. The hope is that some small local

Figure 7. The Urban Challenge Final Event Was Held on a Road Network in and around the Former George Air Force Base.

motion will either physically dislodge the vehicle or, by changing the point of view, clear any artifacts due to noisy perceptions that are causing the failure. Upon successful completion of this motion, the recovery level is reset, enabling the vehicle to continue on its way. This functionality is analogous to the "wander" behavior (Brooks 1986).

The components described in this section are just a few examples of the autonomy components that caused Boss to drive safely. For a more in depth overview, we point the reader to Urmson et al. (2008)

Performance

During testing and development, Boss completed more than 3000 kilometers of autonomous driving. The most visible portion of this testing occurred during qualification and competition at the DARPA Urban Challenge. The event was held at the former George Air Force Base in Victorville, California, USA (figure 7). During qualification, vehicles were required to demonstrate their abilities in three test areas. Area A required the autonomous vehicles to merge into and turn across dense moving traffic. Vehicles had to judge the size of gaps between moving vehicles, assess safety, and then maneuver without excessive delay. For many of the vehicles, this was the most difficult challenge, as it involved significant reasoning about moving obstacles. Area B was a relatively long road course that wound through a neighborhood. Vehicles were challenged to find their way through the road network while avoiding parked cars, construction areas, and other road obstacles but did not encounter moving traffic. Area C was a relatively short course but required

autonomous vehicles to demonstrate correct behavior with traffic at four-way intersections and to demonstrate rerouting around an unexpectedly blocked road. Boss demonstrated safe and reliable driving across all of these areas and was the first vehicle to qualify for the Urban Challenge competition.

During the final event, the 11 finalists were required to drive 52 miles in and around the base while interacting with each other and fifty human-driven vehicles. Six of the 11 teams that entered the final competition completed the course (Buehler, Lagnemma, and Singh 2008). Boss finished approximately 19 minutes faster than the second-place vehicle, Junior (from Stanford University), and 26 minutes ahead of the third-place vehicle, Odin (from Virginia Polytechnic Institue and State University). The vehicles from Cornell University, the Massachusetts Institute of Technology, and the University of Pennsylvania rounded out the finishers, but each of these vehicles required some human intervention to complete the course. All of the vehicles that completed the challenge performed impressively with only relatively minor driving errors: Boss twice incorrectly determined that it needed to make a U-turn, resulting in driving an unnecessary 2 miles; Junior had a minor bug that caused it to repeatedly loop twice through one section of the course; and Odin incurred a significant GPS error that caused it to drive partially off the road for small periods of time. Despite these glitches, these vehicles represent a new state of the art for autonomous urban driving.

During the final competition, the value of Boss's error recovery system was demonstrated on 17 occasions. All but 3 of these errors were resolved by the first error recovery level, which, in general, meant that an outside observer would not detect that anything untoward had occurred. The remaining three recoveries consumed more than half of the 11 minutes the vehicle spent in error recovery modes and were generally due to software bugs or sensor artifacts. For example, the U-turn mistakes described above 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 when 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 its goal. While suboptimal, this behavior demonstrated the recovery system's ability to compensate for subtle but potentially lethal bugs in the software system. The end result is that Boss demonstrated robust autonomy in the face of the unexpected and its own shortcomings.

Discussion

The Urban Challenge was a compelling demonstration of the state of the art in autonomous robotic vehicles. It has helped to reinvigorate belief in the promise of autonomous vehicles. While the Urban Challenge is not wholly representative of daily driving, it does represent a significant step toward achieving general driving. Some of the remaining challenges include diagnostics, validation, improved modeling, and autonomous vehicle and driver interaction.

Diagnostics: It is essential that autonomous vehicles understand when they are in a safe working condition and when they should not be operated. Reliable detection and diagnosis of certain classes of sensor and operational faults are very difficult.

Validation: Prior to deploying something as safety critical as an autonomous vehicle, we must have confidence that the vehicle will behave as desired. While there has been significant progress in both systems engineering and automated software validation, neither field is yet up to the task of validating these complex autonomous systems.

Improved modeling: The Urban Challenge offered only a subset of the interactions a driver faces daily; there were no pedestrians, no traffic lights, no adversarial drivers, and no bumper-to-bumper traffic. A sufficient and correct model of the world for complex urban driving tasks is yet to be developed.

Autonomous vehicle and driver interaction: By not allowing drivers in the Urban Challenge robots, this problem was irrelevant, but when autonomous vehicles are deployed, the sharing of control between driver and vehicle will be critically important. We are only just beginning to understand how to approach this problem.

Many of these items will best be addressed by applying artificial intelligence and machine-learning techniques since they require complicated models that are difficult to explicitly encode. Despite these challenges, autonomous vehicles are here to stay. We have already begun work with Caterpillar to use techniques and algorithms developed during the Urban Challenge to automate large off-highway trucks for mining applications. The automotive industry is interested in improving existing driver assistance systems by using autonomous vehicle techniques to further reduce traffic accidents and fatalities (global traffic accident fatalities exceed 1.2 million people each year), reduce congestion on crowded roads, and improve fuel economy. The question is no longer *if* vehicles will become autonomous but *when and where* can I get one?

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

AAAI Executive Council Nominations Needed!

To submit a candidate's name for consideration, please send the individual's name, address, phone number, and e-mail address to Carol Hamilton, Executive Director, AAAI, 445 Burgess Drive, Menlo Park, CA 94025; by fax to 1-650-321-4457; or by e-mail to hamilton@aaai.org. Nominators should contact candidates prior to submitting their names to verify that they are willing to serve, should they be elected. The deadline for nominations is November 1, 2009.

General Motors, Caterpillar, and Continental. This work was further supported by DARPA under contract HR0011- 06-C-0142.

Notes

1. See the DARPA 2005 Grand Challenge website (www.darpa.mil/grandchallenge05).

2. See the DARPA 2007 Urban Challenge website (www.darpa.mil/grandchallenge).

References

Baker, C., and Dolan, J. 2008. Traffic Interaction in the Urban Challenge: Putting Boss on Its Best Behavior. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*, 1752–1758. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Brooks, R. A. 1986. A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation* 2(1): 14–23.

Buehler, M.; Lagnemma, K.; and Singh, S.; eds. 2008. Special issues on the 2007 DARPA Urban Challenge, Parts 1–3. *Journal of Field Robotics* 25(8, 9, 10).

Darms, M.; Rybski, P.; and Urmson, C. 2008. Classification and Tracking of Dynamic Objects with Multiple Sensors for Autonomous Driving in Urban Environments. In *Proceedings of the 2008 IEEE Intelligent Vehicles Symposium,* 1197–1202. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Ferguson, D.; Baker, C.; Likhachev, M.; and Dolan, J. 2008. A Reasoning Framework for Autonomous Urban Driving. *Proceedings of the 2008 IEEE Intelligent Vehicles Symposium*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Ferguson, D.; Howard, T.; and Likhachev, M. 2008a. Motion Planning in Urban Environments: Part I. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Ferguson, D.; Howard, T.; and Likhachev, M. 2008b. Motion Planning in Urban Environments: Part II. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '08)*. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Howard, T.; Green, C.; Ferguson, D.; and Kelly, A. 2008. State Space Sampling of Feasible Motions for High Performance Mobile Robot Navigation in Complex Environments. *Journal of Field Robotics* 25(1): 325–345.

Thrun, S.; Montemerlo, M.; Dahlkamp, H.; Stavens, D.; Aron, A.; Diebel, J.; Fong, P.; Gale, J.; Halpenny, M.; Hoffmann, G.; Lau, K.; Oakley, C.; Palatucci, M.; Pratt, V.; Stang, P.; Strohband, S.; Dupont, C.; Jendrossek, L.-E.; Koelen, C.; Markey, C.; Rummel, C.; van Niekerk, J.; Jensen, E.; Alessandrini, P.; Bradski, G.; Davies, B.; Ettinger, S.; Kaehler, A.; Nefian, A.; and Mahoney, P. 2006. Stanley: The Robot that Won the DARPA Grand Challenge. *Journal of Field Robotics* 23(9): 661–692.

Urmson, C.; Anhalt, J.; Bartz, D.; Clark, M.; Galatali, T.; Gutierrez, A.; Harbaugh, S.; Johnston, J.; Kato, H.; Koon, P. L.; Messner, W.; Miller, N.; Mosher, A.; Peterson, K.; Ragusa, C.; Ray, D.; Smith, B. K.; Snider, J. M.; Spiker, S.; Struble, J. C.; Ziglar, J.; ; and Whittaker, W. R. L. 2006. A Robust Approach to High-Speed Navigation for Unrehearsed Desert Terrain. *Journal of Field Robotics* 23(8): 467–508.

Urmson, C.; Anhalt, J.; Bae, H.; Bagnell, J. D.; Baker, C.; Bittner, R. E.; Brown, T.; Clark, M. N.; Darms, M.; Demitrish, D.; Dolan, J.; Duggins, D.; Ferguson, D.; Galatali, T.; Geyer, C. M.; Gittleman, M.; Harbaugh, S.; Hebert, M.; Howard, T.; Kolski, S.; Likhachev, M.; Litkouhi, B.; Kelly, A.; McNaughton, M.; Miller, N.; Nickolaou, J.; Peterson, K.; Pilnick, B.; Rajkumar, R.; Rybski, P.; Sadekar, V.; Salesky, B.; Seo, Y.-W.; Singh, S.; Snider, J. M.; Struble, J. C.; Stentz, A. T.; Taylor, M.; Whittaker, W. R. L.; Wolkowicki, Z.; Zhang, W.; and Ziglar, J. 2008. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *Journal of Field Robotics* 25(8): 425–466

Chris Urmson is an assistant research professor in the Robotics Institute at Carnegie Mellon University. He served as the director of technology for Tartan Racing, leading the development of Boss.

Chris Baker is a graduate student at the Robotics Institute at Carnegie Mellon University. He worked on Boss's behavioral reasoning and fault recovery algorithms.

John Dolan is a senior systems scientist in the Robotics Institute at Carnegie Mellon University. He served as the behaviors lead for Tartan Racing.

Paul E. Rybski is a systems scientist in the Robotics Institute at Carnegie Mellon University. He served as the perception lead for Tartan Racing.

Bryan Salesky is a commercialization specialist at the National Robotics Engineering Center of the Robotics Institute at Carnegie Mellon University. He served as the software lead for Tartan Racing.

William "Red" Whittaker is the Fredkin University Research Professor of Robotics at Carnegie Mellon University. Whittaker was Tartan Racing's team leader.

Dave Ferguson is a researcher at Two Sigma Investments. He was the planning lead for Tartan Racing's Urban Challenge team.

Michael Darms is team leader in the Advanced Engineering Department of Continental's Chassis and Safety Division. He worked on the perception system of Boss and was responsible for object tracking algorithms.