DARPA Urban Challenge
The Golem Group LLC
Technical Report

Richard Mason Jim Radford Robert Walters David Caldwell
Bill Caldwell Dmitriy Kogan

April 13, 2007

Abstract

We report progress on the various separate problems necessary for the Urban Challenge. We can distinguish vehicles from non-vehicles at long range, track their position and orientation over time, and estimate their velocity and future heading; results are presented from real traffic data. In simulation, we can conduct entire Urban Challenge missions, including construction of the elaborate trajectories necessary to perform multipoint turns or to conduct passing maneuvers. We had previously built a large vehicle capable of driving forward down a desert trail, but we have now constructed a smaller, more agile urban vehicle capable of driving in forward or reverse; this vehicle is now fully constructed with all actuation and sensors in place. The remaining work prior to the Urban Challenge is to integrate these separate threads of development and conduct full real-world system tests.

1 Introduction

The Golem Group LLC is the successor organization to the Golem Group team which competed in the first two DARPA Grand Challenge events. For a description of our past results, see Mason et al. [2006].

The Golem Group is distinctive in being by far the smallest Track A participant in the Urban Challenge. It is also the only Track A participant which did not exist prior to 2003 but was created specifically in response to the DARPA Grand Challenge. Our experiences as a "garage team" in the previous DARPA Grand Challenge have shaped our design philosophy: a minimalist approach that emphasizes simplicity and drivability.

Simplicity. While it appears somewhat typical for DARPA Grand Challenge teams to equip their vehicles with racks of networked computers, to date we have been able to perform all computational tasks using a single laptop computer.[1] Therefore we will contine to pursue a single-computer solution, unless and until it becomes clear that some vital task cannot be performed without more processing power. In the meantime we avoid the delays and overhead associated with a network system.

Our lean computational approach forces us to focus on the most essential information. It easily could require enormous processing power to monitor and predict every observable detail in the environment. However we believe that most superfluous sensor data can be rapidly discarded, leaving a manageable amount of information relevant to driving.

The same minimalist philosophy applied to sensors would suggest using as few sensors as possible. While certainly an argument can be made for "sensor fusion" exploiting the individual strengths of ladar, radar, video, et cetera, there is a cost in time and complexity associated with each new sensor type, which might have been better spent on developing a single sensor modality. In past years we have dabbled at one time or another with ladar, millimeter-wave radar, monocular video cameras, stereo video cameras, long-wave infrared cameras, and ground-feeling whisker sensors. In hindsight this seems like a monstrous dilution of effort. This year we have focused on the single sensor type, ladar, that has yielded the best results for us in the past. At this time we believe that a single ladar can perform all the necessary sensing tasks for the Urban Challenge.

Drivability. From the beginning we have considered it very important for the development and testing process that our autonomous vehicles remain human-usable. This means:

- using a full-size street-legal base vehicle;
- leaving the standard controls intact so that a human driver can practically, and legally, drive the vehicle around without towing, remote control, or other complex operations;
- enabling fast and reliable transition from human to computer control, and more importantly from computer to human control;
- leaving the interior of the vehicle unobstructed so that passengers apart from the driver can comfortably ride in the vehicle and conduct on-road development.

Our vehicles have therefore always had a relatively "stock" configuration.[2] We consider the philosophy of testing and developing from within the vehicle to be vindicated by the results of previous DARPA Grand Challenges, in which drivable, near-stock vehicles have generally outperformed vehicles which were not drivable in this sense.

In formulating our approach we are guided by concerns which are not strictly speaking Urban Challenge requirements, but are clearly important in the larger picture of autonomous vehicle development. If two vehicles perform equally well in the Urban Challenge Event,

[1]Of course, the capability of a single laptop is continually increasing with time. In today's vehicle, we use a 2.33 GHz Intel Core 2 Duo processor.

[2]This has been an evolving standard. Our first autonomous vehicle, Golem 1, appeared somewhat outlandish on the street with monster tires and a giant sensor-laden bumper, but it looked conventional compared to the moon rovers and built-from-scratch vehicles which were the norm in the 2004 DARPA Grand Challenge. Since then the field has shifted in our direction, so that conventional full-size platforms were more commonplace in the 2005 Challenge, and are the only platforms permitted in the Urban Challenge. Meanwhile our own designs have sought to move closer and closer to "factory" configuration.

there is no prize for the vehicle which uses one computer instead of many, or which uses one sensor instead of many, or which sacrifices the least interior space, or which can be more easily reproduced instead of being an expensive one-off. Nevertheless, we think these are valid design considerations in the long run, even if they are not laid down as program requirements.

2 Overview of Important Problems

Perhaps the most critical problem at the heart of the Urban Challenge is sensing and predicting the motion of other vehicles. Whereas in the previous DARPA Grand Challenge we only had to detect static obstacles in a generally forward direction, in the Urban Challenge we must detect moving vehicles in all directions, which demands expanding the effective sensor field of view while retaining a rapid revisit rate. The most stressing case from a sensor point of view is that of detecting 30-mile-per-hour cross-traffic prior to turning into it. In this case the robot must detect oncoming cars at an imprecisely known angle at a range of at least 65 meters, in order to have 3.5 seconds to complete the turn and leave a three-vehicle-length separation distance.

Furthermore, in other cases we must be able to distinguish vehicles from other objects by shape cues, and not only by motion detection, because vehicles have a special status even when not moving. A temporarily stopped car may have the right of way at an intersection, or may be waiting in a queue which the robot is expected to join.

Finally, we must be able to detect the boundaries of roads in areas were only very poor initial information is available. It would also be useful to detect line markings on the ground in order to improve the precision of the required tasks of driving in a lane and stopping within one meter of a stop line.

We describe our approach to sensing and tracking in Section 3.1, and current results in Section 4.1.

Supposing that the current and future trajectories of other vehicles have been accurately predicted, there is still the problem of computing a path which will take the robot towards its next mission waypoint while avoiding those obstacles and respecting the lane and zone boundaries and other conditions imposed by the rules of the road. Motion plans are required to be more complex than in the previous Challenge, including a requirement for multipoint maneuvers with both forward and reverse segments. We describe our optimization-based approach to planning in Section 3.2 and demonstrate results using simulated obstacles in Section 4.2.

The change from a wide-open off-road environment to a crowded urban environment also changes the ideal physical platform to be used for the autonomous vehicle. We discuss the issues behind our selection of platform in Section 3.3, and present the realized vehicle in Section 4.3.

Ladar System	**Angular Resolution**	**Elevation Field of View**	**Points Per Second**	**Revisit Rate**	**Cost**
Golem-2 System Sick LMS-221 x 2 Sick LMS-291 x 4	Δaz = 0.5° Δelev = 0.5°	90° (-60° to +30°) for three values of azimuth; ∼1° elsewhere	81,000	37.5 Hz	$26,000
Velodyne HDL-64E x 1	Elevation>-9°: Δaz = 0.15° Δelev = 0.3° Elevation<-9°: Δaz = 0.6° Δelev = 0.5°	27° (-25° to +2°)	1,024,000	13.3 Hz	$75,000
IBEO Alasca XT x 2	Δaz = 0.25° Δelev = 0.8° × cos(az)	3.2° × cos(az)	86,496	25.5 Hz	$66,800

Table 1: Performance parameters of three ladar systems considered. The values shown for azimuthal angular resolution and revisit rate are representative, but each one of these systems could be configured for a faster revisit rate, at the cost of a proportional reduction in azimuthal resolution (or vice versa).

3 Analysis and Design

3.1 Sensor System Design

The basic dilemma of a ladar system with finite data rate is that a beam return at any single elevation may be ambiguous for various reasons, so the ladar must sample at several elevation angles in a given azimuthal direction to develop an unambiguous picture of whether there is an obstacle in that direction. But for fixed data rate, the more data the ladar acquires along any particular azimuth, the less azimuthal arc it is able to cover. There is also the mechanical problem of how to direct the beams through the desired combination of elevation and azimuth. The data rates and other specifications for three different ladar systems are listed in Table 1.

In the 2005 Grand Challenge we used an arrangement of Sick LMS ladars, each one of which swept its beam through an arc in either a horizontal or vertical plane. Each of the vertical-plane Sicks acquired a ground profile in a particular azimuthal direction. This data was extrapolated to a model of the ground in other directions. Data returned from the horizontal-plane Sicks, which covered a wide arc of azimuth, was then compared to this ground model to see whether it was consistent with reflections from traversable ground, or whether it suggested a non-traversable obstacle rising above the ground. Naturally, one would expect the extrapolated ground model to be most accurate near the azimuth of the vertical-plane Sicks. For the 2005 Grand Challenge, the vertical-plane Sicks were all oriented forward so that the vehicle had maximum perception of the road ahead, somewhat more ambiguous information about the periphery of the forward 180° arc, and no information at all about the rear of the vehicle. An adaptation of this system for the Urban Challenge would require Sicks to be distributed around the vehicle.

We reported in Mason et al. [2006] that this system could detect static obstacles such as cars at ranges of 40-80 meters, and it generally did a good job of correctly classifying obstacles from a rolling and pitching vehicle moving over terrain of varying slope. It could be confused in certain cases, notably by reflective road markers, which caused off-axis ladar returns creating "phantom obstacles" at certain angles. Also, its effective field of view could be cut off by a significant abrupt change in slope.

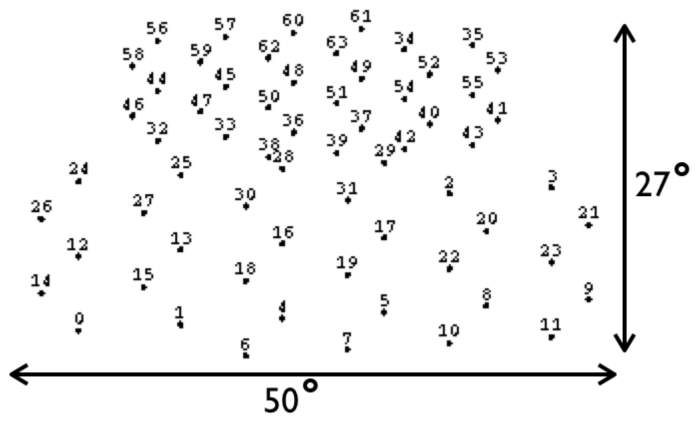
We continued to gather data with the Sick-based system while driving non-autonomously in real traffic at intersections and on roads and freeways. Using this data we were able to identify other cars by their quasi-rectangular outline and track their position over time, as discussed further in Section 4.1. We noted that the tracking algorithm could be confused by large trucks, because at the level where the side of a car would be, the horizontally sweeping Sick beam would see instead the truck's wheels, mudflaps, fifth wheel, etc. The truck presented a larger, more complicated, non-convex outline, and tended to be interpreted as a train of close-following cars.

The 80-meter Sick detection range of cars represents a best case scenario, when the laser beam encounters a fully normal facing surface or a tail reflector of the detected car. When cross-traffic was approaching at an intersection, the Sick ladar would typically get its first glimpses of a car at 65 meters, but would not reliably resolve the rectangular shape of the car until about 40 meters. This is marginally satisfactory at best for the task of merging into 30-mile-per-hour traffic.

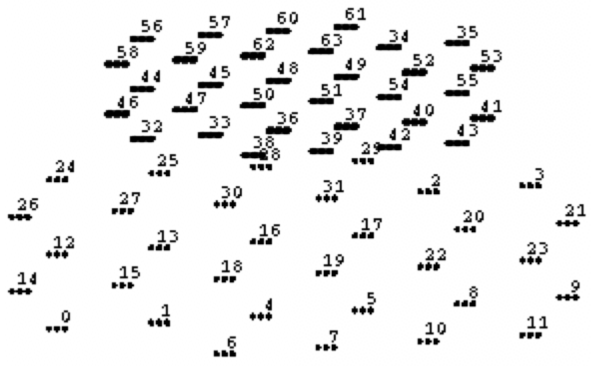
We decided to continue development with our existing system, but move to supplement or replace it with a much higher-data-rate ladar, the Velodyne HDL-64E. The HDL-64E uses 64 laser diodes to take range measurements at 64 elevation angles, while rotating through 360° of azimuth. If it performs to specification it should be able to resolve the entire surroundtrings of the vehicle with less ambiguity than our Sick-based system, correctly interpret road markings, and be robust to substantial changes of slope such as at the crest of a hill.

One limitation of the HDL-64E is that, since it must be roof-mounted for an unobstructed 360° field of view, there will be a minimum range $r_{\min} \approx (\text{sensor height})/\tan(25°)$ within which it will not be able to see low-lying ground features such as road markings. In particular, since an Urban Challenge requirement is to stop within 1 meter of a stop line, the HDL-64E will not be able to perceive the stop line all the way up to a stop. However, it should be able to provide guidance until relatively close to the stop line, with the vehicle's GPS, inertial, and odometric navigation performing the rest of the task.

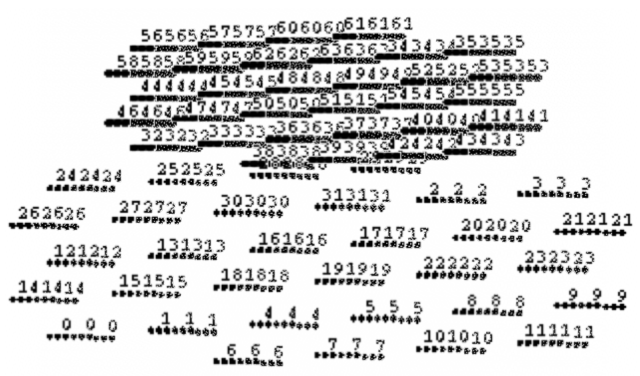
We also considered the IBEO Alasca XT ladar. The beam optics for this sensor are such that it samples at 4 elevation angles spanning 3.2° at zero azimuth directly in front of the vehicle, but the separation of the elevation angles is proportional to cos(azimuth), so at the sides of the vehicle it is effectively scanning in a horizontal plane with no range of elevation. It is vaguely similar to our 2005 Sick-based system in having some elevation data to the front but not to the sides. Since the Alasca XT was not an obvious improvement over our existing equipment, and did not have a wide vertical field of view to cope with hilly terrain, we did not pursue it further. The Alasca XT does promise long detection ranges of up to 200 meters, but such long ranges might only be effective near zero azimuth, whereas the most stressing detection range requirement in the Urban Challenge is for cross traffic near 90° azimuth.



(a)



(b)



(c)

Figure 1: (a) The angular arrangement of laser diodes in the Velodyne HDL-64E is shown. (b) Diodes 0-31 fire three times and diodes 32-63 fire nine times in a single Velodyne data packet. The pixel grouping for a single packet is shown. The rotational rate of the sensor is 4800°/second here although in principle this rate can be varied. (c) After three successive packets of data, comparisons can be made between some pairs of data points that are separated by only 0.3° of elevation and about one millisecond of time. The local angular area in the upper region will be densely filled with data after fifteen packets or five milliseconds (not shown). The lower region will be densely filled after thirty packets or ten milliseconds.

In order to resolve the shape of a detected object, we want several ladar returns from the object to be close together in space and time. We particularly want a reasonable density of data points in elevation, in order to determine whether the object is a nearly-vertical surface (thus an obstacle) or merely a drivable piece of ground. The angular arrangement of the 64 diodes of the Velodyne HDL-64E, shown in Figure 1(a), is such that each diode is roughly 3° apart from its nearest neighbor, and 4° of elevation apart from the nearest diode with the same azimuth. Thus, simple pairwise comparisons between different diodes at exactly the same laser firing time would give a low-resolution result. However, by rotating the sensor and collecting a number of measurements over several milliseconds, as illustrated in Figure 1(c), we can make comparisons between data measurements that are only 0.3° apart.

We therefore create a "bucket" for measurements made in a particular section of azimuth. The "bucket" is partly or entirely filled over the course of five or ten milliseconds as the separate diodes of the HDL-64E sensor rotate through the section of azimuth and most of the diodes take a range measurement. The data in the bucket then represents a ground profile along a ray of azimuth from the Golem vehicle, which should have moved at most fifteen centimeters and turned at most 0.3° during the time when the data was gathered. The ground profile is searched for near-vertical surfaces representing obstacles, and the range and bearing of detected obstacles is passed to the tracking algorithm.

We found that the detectors in the HDL-64E are sensitive and occasionally one will register a single return when there is no perceptible object or other cause present. The obstacle detection algorithm must ignore these occasional outliers and only register vertical surfaces which are indicated by reasonably dense data, which should not be lacking for real obstacles.

Either the Sick-based ladar system or the Velodyne ladar passes the range and bearing of detected vertical surfaces to the tracking algorithms, which segments the data, identifies other vehicles or potential vehicles from shape cues, and estimates their future trajectories in order to respect traffic rules and avoid collisions.

When the ladar image of another vehicle is reduced to the set of points closest to the sensor in the horizontal plane at which there was a detected vertical surface, the resulting two-dimensional outline generally forms either one or two sides of a quasi-rectangular shape. The corner may be somewhat rounded. Our tracking algorithm exploits these predictable outlines to infer the extent and heading of another car even if only a small corner of it is visible. We proceed by looking for breaks or gaps in the outline and divide the data into connected objects. For each segment of data, we look for a corner at the point in the segment which has maximal deviation from a straight line joining the two endpoints of the segment. If the data segment is a good fit to a rectangle passing through this corner and the two endpoints of the segment (including the case of a rectangle with one straight side passing through all three points) and the sides of the rectangle are of an appropriate length, then the data segment is considered a return from a possible vehicle. Objects which do not appear to be vehicles because they are the wrong size, or not rectilinear enough, are presumed to be static (or at least, not moving with vehicular speeds) and can be treated using the methods developed in the previous DARPA Grand Challenge.

The ladar images of potential vehicles must be associated from scan to scan, in order to build up a track history and a velocity estimate. We solve the Global Nearest Neighbor problem using Bertsekas's algorithm [Bar-Shalom and Li, 1995] to find the best matching of vehicle tracks from scan to scan. A Kalman filter is used to estimate the state of the tracked vehicle given sequential measurements. Currently simple constant-velocity filters are used, but we plan to implement more complicated models taking into account the perceived heading and lane of the vehicle to predict its future trajectory.

3.2 Navigation System Design

The navigation problem in the Urban Challenge is significantly more difficult than that of the Grand Challenges. In the Urban Challenge, the vehicles need to correctly interact with traffic lanes, stop-signs and various traffic impedances, including other vehicles. Further, the vehicle needs the ability to drive backwards in order to handle getting in and out of parking spaces, and to execute multi-point turnaround maneuvers.

Due to the diversity of the problems that need to be addressed, we designed our navigation system as a pipeline of several layers, each handling a subset of the necessary tasks. The upper layer produces a very rough spatial path that visits checkpoints and avoids regions of heavy traffic. The path produced by this layer is interpreted by a state machine that keeps track of the vehicle going into and coming out of certain maneuvers. The output of this state machine, along with the spatial path produced by the upper layer is fed into the last planning layer, an algorithm that produces a drivable trajectory that is specified both spatially and temporally.

3.2.1 Top level planning

The main task of the top-level planner is to produce turn-by-turn directions to navigate the vehicle from checkpoint to checkpoint. This should be done without a lot of concern about the details of motion such as the location of the vehicle within a lane, since those details are handled at a lower level. The A*/D* graph-search algorithm is an obvious choice for the implementation of this top-level planner [Stentz, 1994].

We chose to apply A* to a lane-centric graph, rather than a road-centric graph. This allows for lane-changing decisions to be made by the A* planner, leaving the lower level the task of determining exactly how to change lanes, not whether or not to. Thus, the nodes entered into the graph are equidistantly-spaced positions in each lane, with a spacing of about 10m. The graph connections come from forward motion in a lane, from entry/exit pairs in the RNDF, and from lane changes that conform to the restrictions indicated by the lane boundary types. The edge cost in the graph is specified as traversal time in order for the algorithm to find the fastest routes. Additionally, obstacle and traffic flow information is also integrated into the edge cost. This is designed to select obstacle-free or traffic free lanes.

3.2.2 Maneuver state machine

The top-level planner produces the desired topology of the "correct" trajectory the vehicle should drive. However, this alone is not enough to be able to generate the drivable trajectory at all times. The lower level planner needs to also know whether we are about to stop at a stop sign, or whether we are currently or are about to turn around. At this point in the development, these are the only checks that are performed for the lower level planner. It may be nesessary to add extra checks, like lane changing, with a lower level planner specifically tuned to that purpose. This extra information is passed on to the next planning level, along with the top-level result.

3.2.3 Drivable path generation

The lower layer of the navigation system is the algorithm that produces a trajectory to be followed by the feedback controller in the vehicle. This trajectory should contain both spatial and temporal information, and should be drivable, to make the trajectory following smooth and relatively simple. We are currently looking at several approaches to generating this trajectory. One of these is based on numerical nonlinear optimization [Kogan, 2005].

In general, a nonlinear optimization problem is stated as

$$\text{Find } \vec{x} \in \mathbb{R}^n \text{ that minimizes } G(\vec{x}) \in \mathbb{R} \text{ subject to } \vec{L} \leq \vec{F}(\vec{x}) \leq \vec{U} \tag{1}$$

where $\vec{L}, \vec{U} \in \mathbb{R}^m$. In other words, find $\vec{x}$ that satisfies all the constraints given by $\vec{F}(\vec{x})$, and is best according to some cost metric $G(\vec{x})$. This is useful to us because we want to find the trajectory that is fast, drivable, goes where we want, and does not impact any obstacles.

As seen in (1), in order to apply an optimization method to the planning problem, the problem has to be stated in explicit mathematical terms. We are generating trajectories with both spatial and temporal components. It is very convenient to use a representation that keeps these two components separate. We thus define our trajectory as $v(s)$ and $\theta(s)$ curves, representing the speed and yaw along the trajectory, respectively. Here s is a unitless measure of distance along the trajectory, with $s \in [0, 1]$. We define S as the total length of the trajectory. The spatial representation is illustrated in Figure 2. We want our trajectory

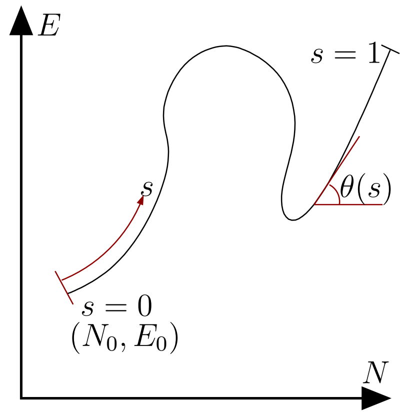


Figure 2: The spatial trajectory representation.

to be drivable, so we need to define driveability. For this purpose, we assume the vehicle satisfies the rear-axle-based bicycle model:

$$\begin{aligned} \dot{N} &= v\cos\theta \\ \dot{E} &= v\sin\theta \\ \dot{v} &= a \\ \dot{\theta} &= \tfrac{v}{L}\tan\phi \end{aligned} \tag{2}$$

where N and E are the northing and easting coordinates of the middle of the rear axle, v is the scalar speed of the center of the rear axle, θ is the yaw of the vehicle, a is the scalar longitudinal acceleration of the vehicle, L is the wheelbase of the vehicle, and ϕ is the steering angle. Given this, we can come up with expressions for various vehicle parameters:

$$\begin{array}{rcl} N(s) & = & N_0 + S\int_0^s \cos(\theta(s))ds \\ E(s) & = & E_0 + S\int_0^s \sin(\theta(s))ds \\ \dot{N}(s) & = & S\cos(\theta(s))v(s) \\ \dot{E}(s) & = & S\sin(\theta(s))v(s) \\ a & = & \frac{v}{S}\frac{dv}{ds} \\ \dot{\theta} & = & \frac{v}{S}\frac{d\theta}{ds} \\ \tan\phi & = & \frac{L}{S}\frac{d\theta}{ds} \\ \dot{\phi} & = & \frac{LS\frac{d^2\theta}{ds^2}}{S^2+\left(L\frac{d\theta}{ds}\right)^2} \\ T & = & S\int_0^s \frac{1}{v(s)}ds \end{array} \tag{3}$$

With the bicycle model in (2, we say that driveability means continuous ϕ and v. From (3), this implies continuous $\dot{\theta}$. We can thus represent our vehicle trajectories with piecewise quadratic $\theta(s)$ and piecewise linear $v(s)$. These basis functions allow the desired continuity conditions to be met while allowing enough freedom in the spline to represent any trajectory we may want.

We define a vector $\vec{x}$ to represent S and the $\theta(s)$ and $v(s)$ of desired continuity. We can then specify a vector of constraints for the trajectory $\vec{F}(\vec{x})$. These constraints can include obstacle avoidance (obstacles may be moving), lane boundaries, speed and acceleration limits, steering limits and aggressiveness limits. We can also specify a scalar-valued cost function $G(\vec{x})$ that can include traversal time and control effort. We then use an off-the shelf nonlinear optimization library, SNOPT, to perform the actual computation [Gill et al., 2006].

This method is very flexible. To stop at a stop sign, we can constrain the location and speed at the end of the trajectory. We can plan around moving obstacles. by entering time-dependent obstacles into the problem. Driving in reverse is covered under the bicycle model, so multi-point turns can be computed with this method. Multiple lanes can be entered into the cost and constraint functions, as necessary. Further, we can optimize only the spatial or only the temporal variables if a simpler computation is desired. This is useful if another spatial-only planner is used for the lower-level planner, and we only need to compute the speed profile. This can also be applied to multi-point turns, which can be computed spatially-only, since we can assume that the maneuver is going to be taken very slowly.

We are also continuing development of the “Avoider” graph-search path planning algorithm used by our team in the past two Grand Challenge events. This is a “greedy” approach that assumes that obstacles are sparse in the reachable state space of vehicle trajectories. When this assumption is correct, the code rapidly returns a drivable path; typical run times are on the order of ten microseconds. However, when the density of obstacles increases, the algorithm can be significantly slower than our optimization based planner. The “Avoider” algorithm does not guarantee convergence, and therefore also requires the use of watchdog routines that end the process according to certain failure conditions.

We have made significant progress in abstracting the graph management aspects of the

"Avoider" algorithm away from the specific mathematics of trajectory generation and evaluation. Recoding in an object oriented architecture (although still in C) resulted in a four fold reduction in code length (in terms of line count) and approximately 50% improvement in performance.

The kernel of the algorithm is an edge operation that replaces a high cost edge with a heuristically generated sub-graph. The replacements occur while maintaining a directed acyclic structure that allows linear time computation of the globally least cost traversal by means of a topological sort. The specifics of the sub-graph generation are unimportant from the point of view of the graph management, but should be designed to allow an efficient exploration of state space. In practice the heuristics we use are variations on "go around to the left" and "go around to the right".

Computation time tends to be dominated by the complexity of the cost evaluation function, which requires a search for intersections between trajectories and obstacles and the calculation of a traversal time cost. We are considering several open source and commercial computational geometry codes for this purpose, including CGAL, SISL, SOLID, and FASTGeo. In parallel we are working on code of our own based on the strip-line method.

In order to handle moving obstacles, it is necessary to introduce time as a third parameter into the state space. We are still evaluating whether the "Avoider" approach offers any computational speed advantage over the nonlinear optimization planning routines in this case.

3.3 Vehicle Platform Selection

After a thorough review of potential platforms, we selected a Toyota Prius to serve as the base vehicle for Golem 3. The reasons for using a Prius instead of the Dodge Ram 2500 pickup truck which we had used in the previous Grand Challenge, or any other platform, were the following:

Drive-By-Wire. The Toyota Prius makes extensive use of drive-by-wire technology, including accelerate-by-wire electronic throttle control, shift-by-wire electronic transmission control, and electric motor-driven power steering.[3]

By using a car with drive-by-wire actuation, designed and produced in industrial quantities by a major auto manufacturer as original equipment, we hope to obtain greater reliability and better control than is likely with custom retrofit actuators. For example, our previous efforts using chain drives to actuate the steering of our trucks suffered from mechanical backlash that interfered with precise steering control. By using Toyota's standard power steering in the Prius, consisting of an electric motor built around the steering column, we've all but eliminated the backlash.

Also, looking beyond the Urban Challenge requirement for a single vehicle, we want to develop an electronic control unit that could be reproduced and readily installed in any Prius, or perhaps any vehicle using Toyota's Hybrid Synergy Drive and associated drive-by-wire technology, to make a low-cost autonomous platform.

Turning Radius. While previous Grand Challenges required off-road capability and large

[3]In Europe and Japan, the Prius is available with Intelligent Parking Assist making use of the motorized steering.

vehicle size was arguably advantageous, the most challenging physical tasks of the Urban Challenge involve maneuvering in confined spaces, in and out of a parking space, or through a three-point turn on a blocked street. The most desirable physical characteristics for an Urban Challenge vehicle would appear to be small size, agility, tight turning radius, and short stopping distance. By these measures the Dodge Ram 2500 4x4 is quite bad (turning diameter 47.5 feet [Dodge.com, 2006]) and the Toyota Prius is quite good (turning diameter 34 feet [Toyota.com, 2007], stopping distance 27.16 feet from 25 miles per hour [Francfort et al., 2001]).

Almost no vehicles have a shorter turning radius than the Toyota Prius while meeting the minimum weight and other requirements for an Urban Challenge platform. One exception is the LTI TXII or London Taxi, with a turning diameter of 25 feet. We gave serious consideration to the TXII on maneuverability grounds; unlike any other potential platform we are aware of, it could complete the DARPA-required 180-degree turn with a simple U-turn without the necessity for a three-point turn. However, the TXII lacks the Prius's drive-by-wire technology. Also, a TXII in the United States costs approximately $50,000. The Prius is a low-cost high-availability platform compared to relatively exotic options like the TXII.

Roof Height. The relatively small size of the Prius is more compatible with a 360-degree rotating roof-mounted sensor such as the Velodyne HDL-64E ladar. The roofline of the Prius allows the Velodyne HDL-64E ladar to scan in a 360-degree circle with minimal obstruction. The 58-inch height of the Prius [Toyota.com, 2007] implies that the minimum range at which the Velodyne detects the ground is 161 inches from the sensor location.

In contrast, if the HDL-64E is placed on the 78-inch-high roof of the Dodge Ram 2500, the minimum range at which the ladar will detect the ground is 205 inches from the sensor, with the rear of the vehicle causing some beam obstruction. The Prius, at under 7 feet with the HDL-64E on top, can be driven into a standard garage, while the Dodge Ram, similarly equipped, cannot.

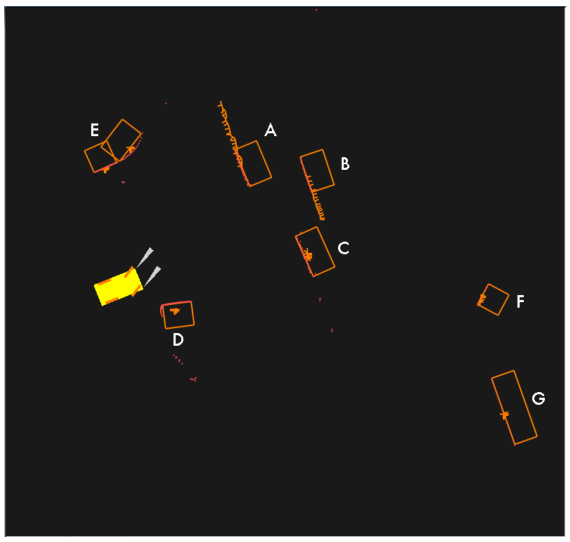
4 Results and Performance

4.1 Tracking System Performance

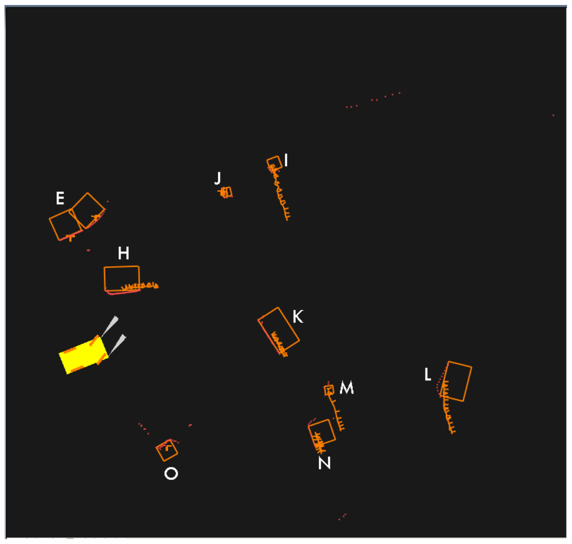
Vehicle tracking results using the Sick-based ladar system are shown in Figure 3. The algorithm does a good job of identifying the rectilinear shapes of potential cars. In real traffic situations, the number of objects to be tracked is typically about 20, a manageable number for the Global Nearest Neighbor problem even allowing for a quadratic number of pairwise comparisons between objects.

High clutter and track confusion can occur off the road (where most detected obstacles are not cars), but on the road, the data association from scan to scan works well, so that we can form coherent track histories for each vehicle. Even cars passing in close proximity to each other in different directions remain well-tracked with little observed confusion or merging of tracks.

There is a fundamental limit on the ability of the system to distinguish between a car which is temporarily paused, a car which is parked or permanently disabled, and a stationary object which is rectangular in shape. The system can perceive that a long wall is too long



(a)



(b)

Figure 3: (a) Vehicle tracking using Sick ladar data at a busy intersection. (b) More vehicle tracks at the same intersection. Rectangles are fit to the inferred locations of cars (or carlike objects), and the attached trail shows the motion of the inferred vehicle over time. The correct interpretation of the data is as follows: A and B are cars driving through the intersection. C and D are paused cars waiting to make a left and right turn respectively. E, F, G, and O are not cars but just walls or stationary objects with a somewhat rectilinear shape. H and K are cars in the process of turning left. I, J, M, and N are cars approaching or leaving the intersection for which only a small part of the outline is visible, probably due to occlusion by the other vehicles in the scene. Finally, L is a section of wall with a spurious velocity track; the ladar shadows of the passing vehicles on the corner of the intersection have created the impression of movement along the wall.

to be a car, but if only a short section of wall or K-rail is seen, it is difficult to distinguish from the side of a car. More elaborate feature detection may be possible, but rather than relying on that to work, it may be preferable for Urban Challenge purposes to assume that a parked car or K-rail in the road is an active vehicle until a certain time has elapsed with zero detected velocity.

In Figure 3(b) we see one instance of a spurious velocity track caused by the ladar shadows cast on a wall by moving vehicles. How to eliminate such tracks is an area for future work. In this case, if the vehicle had determined the boundaries of the road and knew that the wall was off the road, it could have rejected the track on that basis. Perhaps we could also do more to handle occlusions and maintain continuity of data that is temporarily occluded. Small occlusions can be disregarded and a coherent outline of the background object can be maintained, but in this case the occlusions caused by the passing vehicles may not have been sufficiently small.

We had to update some of our infrastructure in order to cope with the greatly increased data rate of the HDL-64E; for example, we now log data in binary form rather than using ASCII data logs as previously. The same tracking algorithm can be applied to data derived from the HDL-64E, with good results. Figure 4 shows that the rectilinear shape of a car is resolved immediately even when the car is at the farthest extent of the HDL-64E's current 65 meter range. This detection range should be adequate for dealing with Urban Challenge traffic. Figure 5 shows detection and tracking of moving obstacles using the high-density HDL-64E ladar data.

Because the HDL-64E measures intensity of return as well as range, it can detect if some patches of a surface are more infrared-reflective than others. It can therefore locate lane markings on the road, like those faintly visible in Figure 4. The change in elevation marking the edge of the road is even more plainly visible. We have not actually demonstrated fitting lines to the detected lane and road boundaries yet, but mere inspection of the data gives us confidence that it is possible. We expect therefore that we will be able to detect road boundaries when they are not specified in the RNDF, and the HDL-64E may be able to serve as the single sensor for an autonomous vehicle, without the need for additional cameras.

4.2 Navigation System Results

We have implemented every part of the navigation system described in Section 3.2. The top level planner is currently implemented with A*. If field testing shows this to be too slow, we will change it to a D* implementation. The whole system has been tested in simulation, and is able to drive RNDFs, change lanes, stop at stop signs, take turns and initiate 3-point turns. To further explicate the nonlinear optimization-based planner described in Section 3.2.3, we present some simulation results.

Figure 6 shows the result produced by the optimizer to drive down an empty lane. Figure 7 shows the behavior when this lane contains an obstacle. Figure 8 shows what happens when this obstacle is moving across the path of the vehicle. Figure 9 shows a lane change to avoid hitting an obstacle. Finally, Figure 10 shows an optimizer-generated 3-point turn. The vehicle is moving from the bottom to the top. The optimizer was tasked with finding the 3-point turn path that has the desired location and yaw of the vehicle at the end. We are assuming that these maneuvers are going to be taken at a very slow speed, so only the

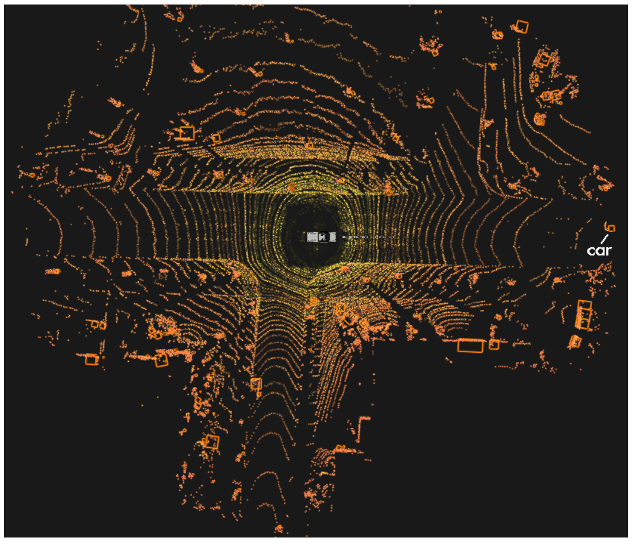


Figure 4: We can identify cars as soon as they enter the 65-meter range of the HDL-64E. Also note that curbs and to a lesser extent lane markings are clearly visible in the data.

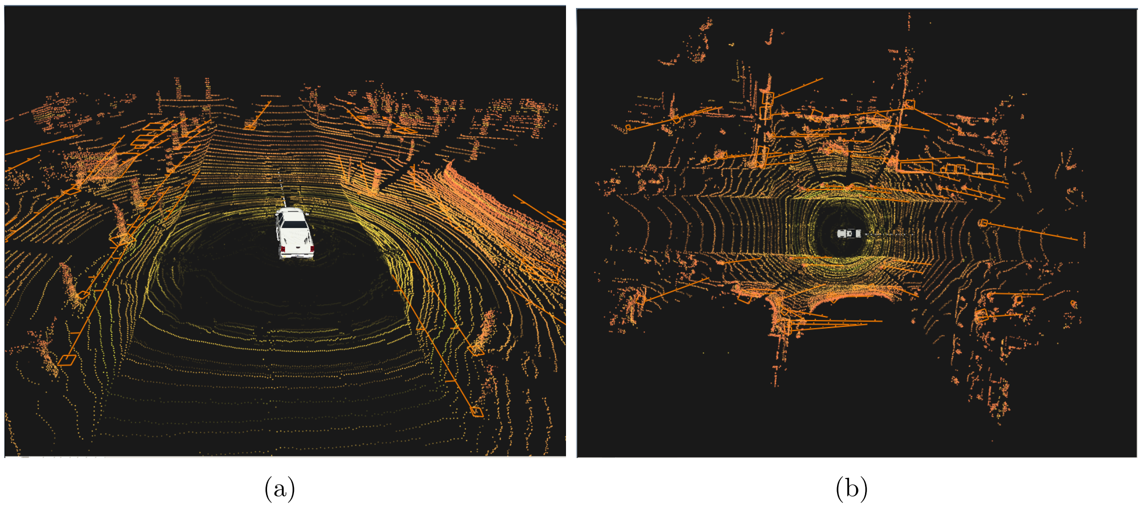
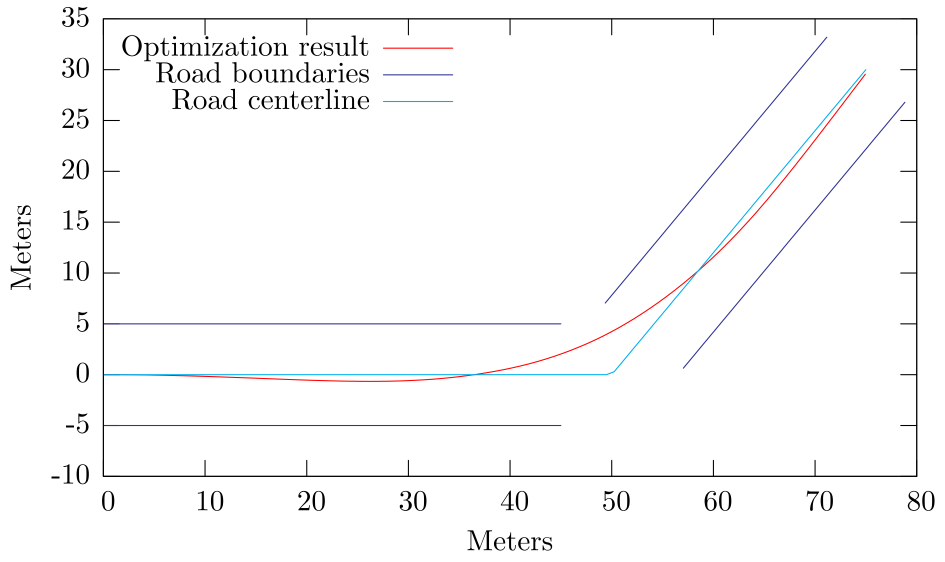


Figure 5: Obstacle detection and tracking with the Velodyne HDL-64E. In this figure all obstacles are tracked; for example the trees are tracked and not discarded simply because they are too tall and narrow to be cars. A rectangular box is drawn around each obstacle and the attached line indicates a velocity estimate, in the rest frame of the robot. Since the robot is moving even the trees have a high velocity relative to it. Note the oncoming car.



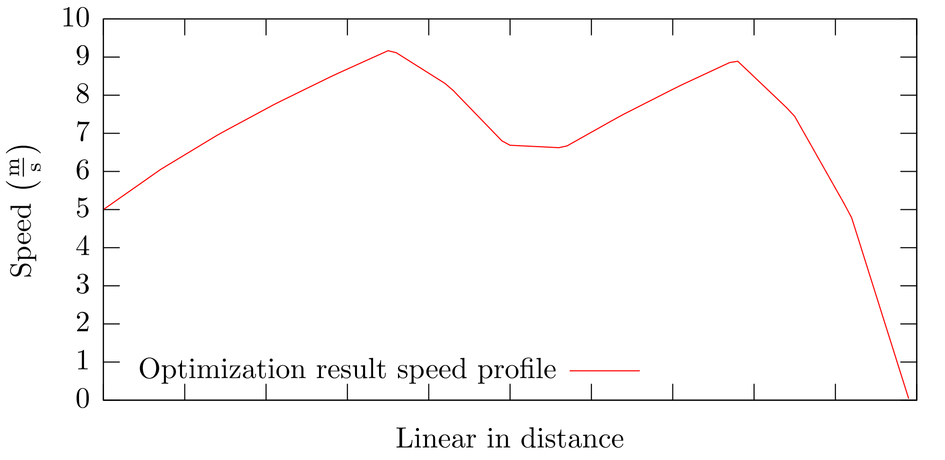


Figure 6: The optimization result of a vehicle driving down a lane. The vehicle is moving from left to right. There is a stop sign at the right end of the lane.

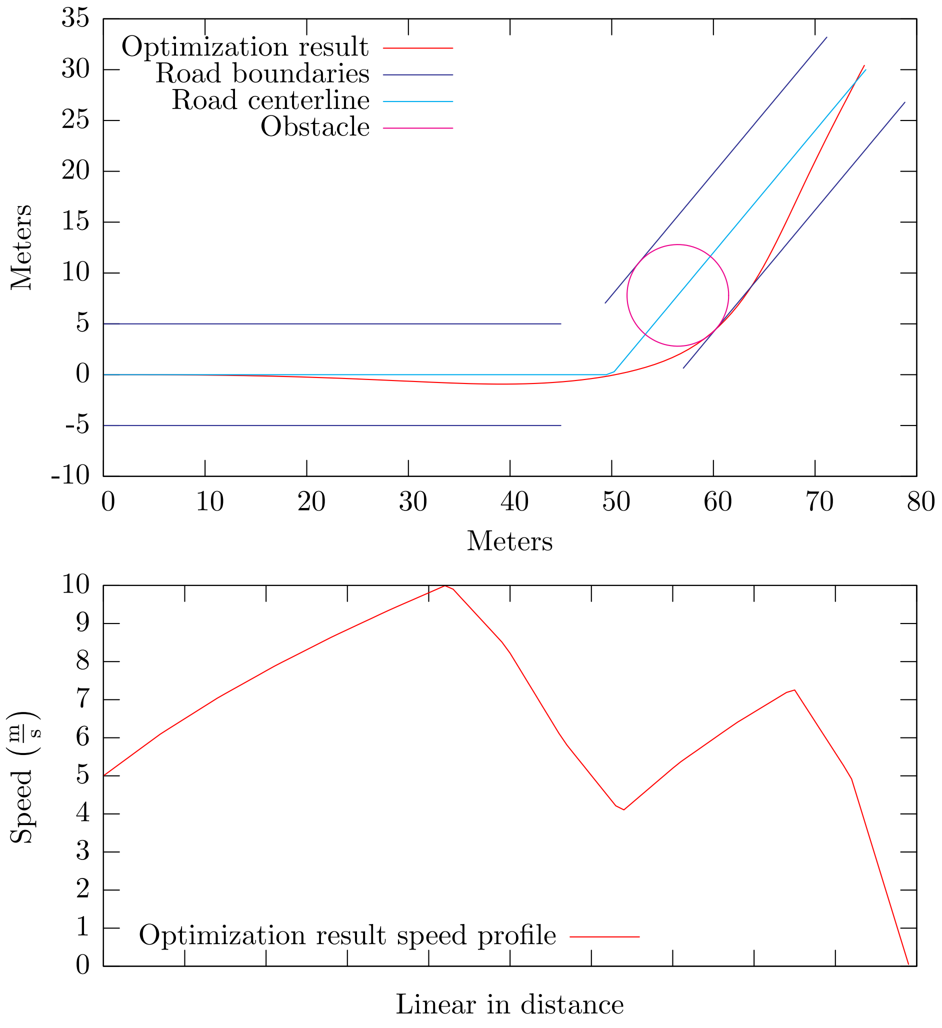


Figure 7: The optimization result of a vehicle driving avoiding an obstacle. The vehicle is moving from left to right. There is a stop sign at the right end of the lane.

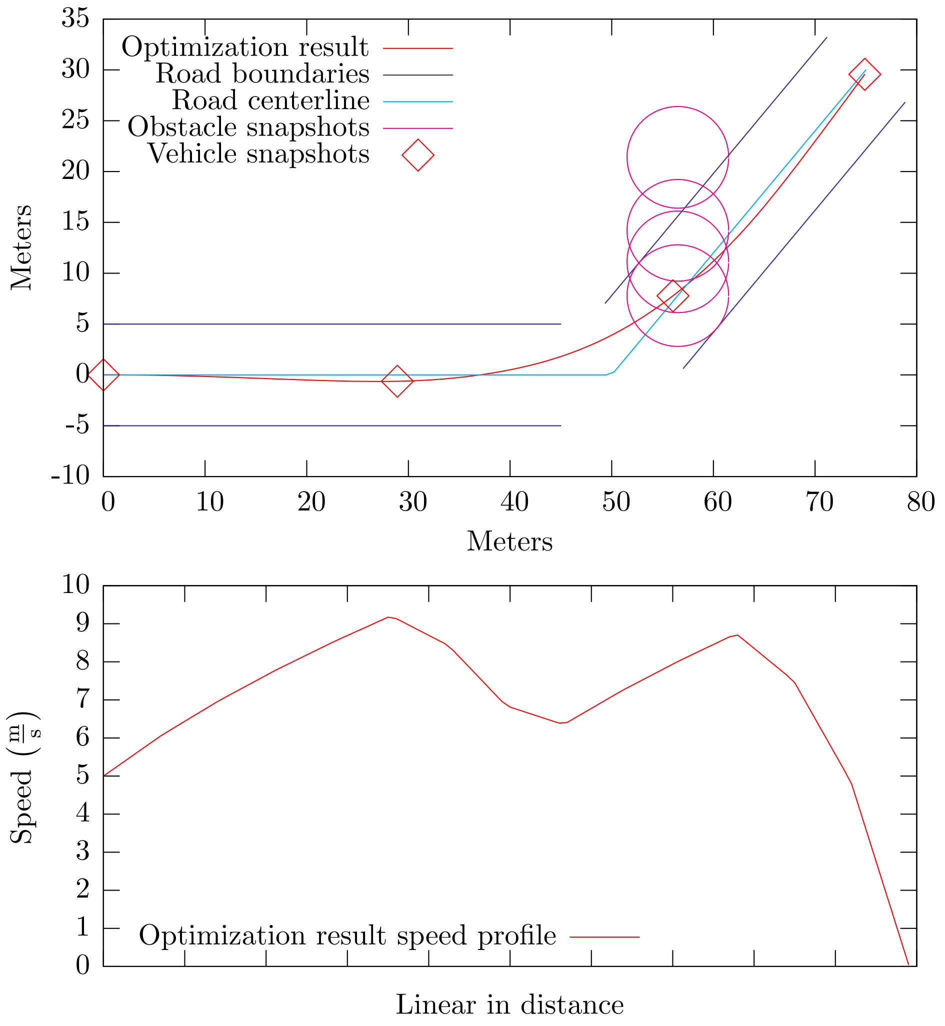


Figure 8: The optimization result of a vehicle avoiding a moving obstacle. The vehicle is moving from left to right, while the obstacle is moving up. There is a stop sign at the right end of the lane. The simultaneous positions of the vehicle and the obstacle are shown at 4 instants in time. The main point of interest is the 3rd time instant, where we see the vehicle clear the obstacle's current location, but move through its previous path.

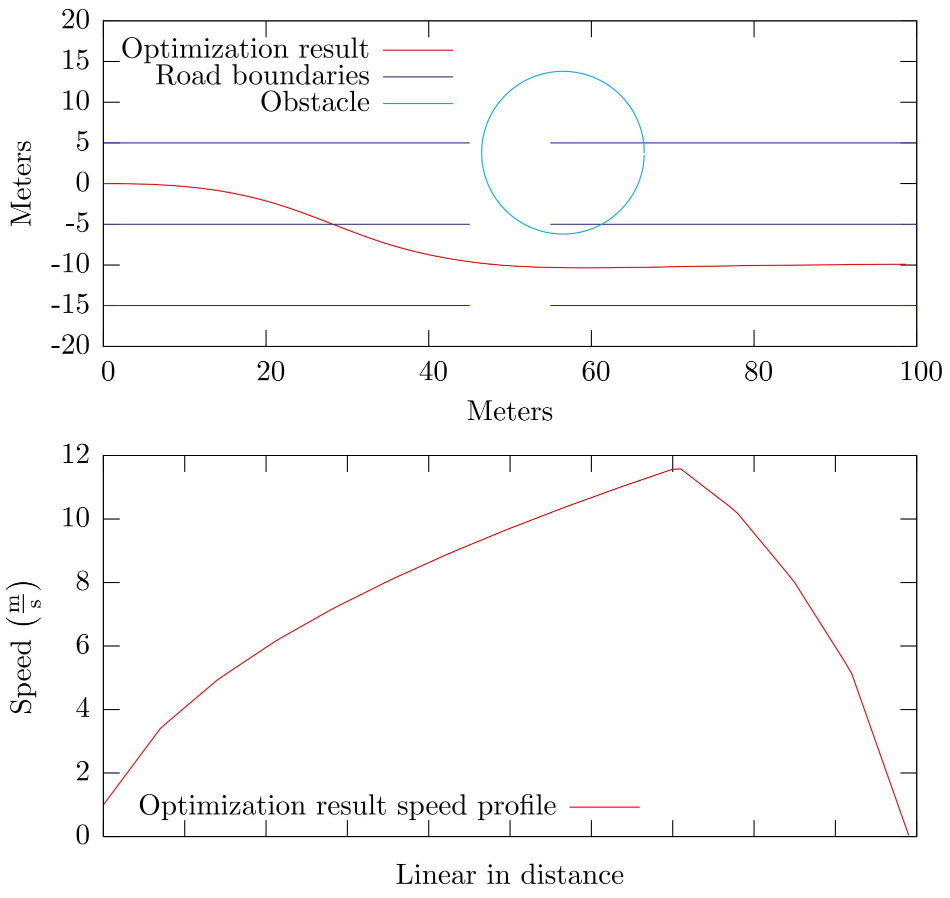


Figure 9: The optimization result of a vehicle avoiding a stationary obstacle, and changing lanes to do so. The vehicle is moving from left to right. There is a stop sign at the right end of the lanes.

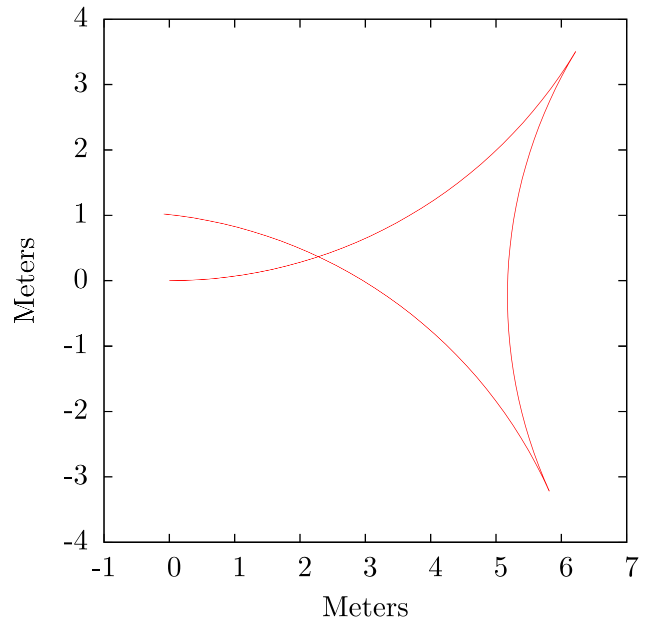


Figure 10: The optimization result of a vehicle performing a 3-point turn maneuver. The vehicle is moving from bottom-left to top-left. The segment at the right side of the figure is meant to be driven in reverse.

spatial path was computed in this run of the optimizer.

4.3 Vehicle Platform Implementation

We have purchased a 2007 Toyota Prius and turned it into a complete autonomous vehicle ready to compete in the Urban Challenge. We have fully actuated the steering, brakes, throttle and gear shift. We have access to the built-in steering angle, wheel speed, brake pressure and gear shift sensors necessary to fully control the vehicle. A diagram of the implemented control system is shown in Figure 11. A Honeywell HG-1700 inertial measurement unit, a NovAtel SPAN-enabled GPS receiver, and the Velodyne HDL-64E ladar have been mounted on the vehicle, which is shown in Figure 12. We have installed signal lights and our E-stop system. Except for the ladar, which is Ethernet-based, all the sensors are routed to our main driving computer via USB.

We had hoped to control the necessary actuators via the vehicles electronic command bus, as we know is possible. However, not being privy to the necessary commands, we have elected to interrupt the sensor inputs to the steering, throttle and gear shift sub-systems and route them through and embedded micro-controller. This allows us to selectively modify the inputs to the car when under autonomous control or pass them through when under manual control. A single switch allows the driver to take control at any time.

The Prius's brake system is both conventional and regenerative and the combination of the two is controlled by the vehicle's computer. Additionally the brake sensors, pressure and throw, are integral to the vehicle's stability control and anti-lock braking sub-systems.

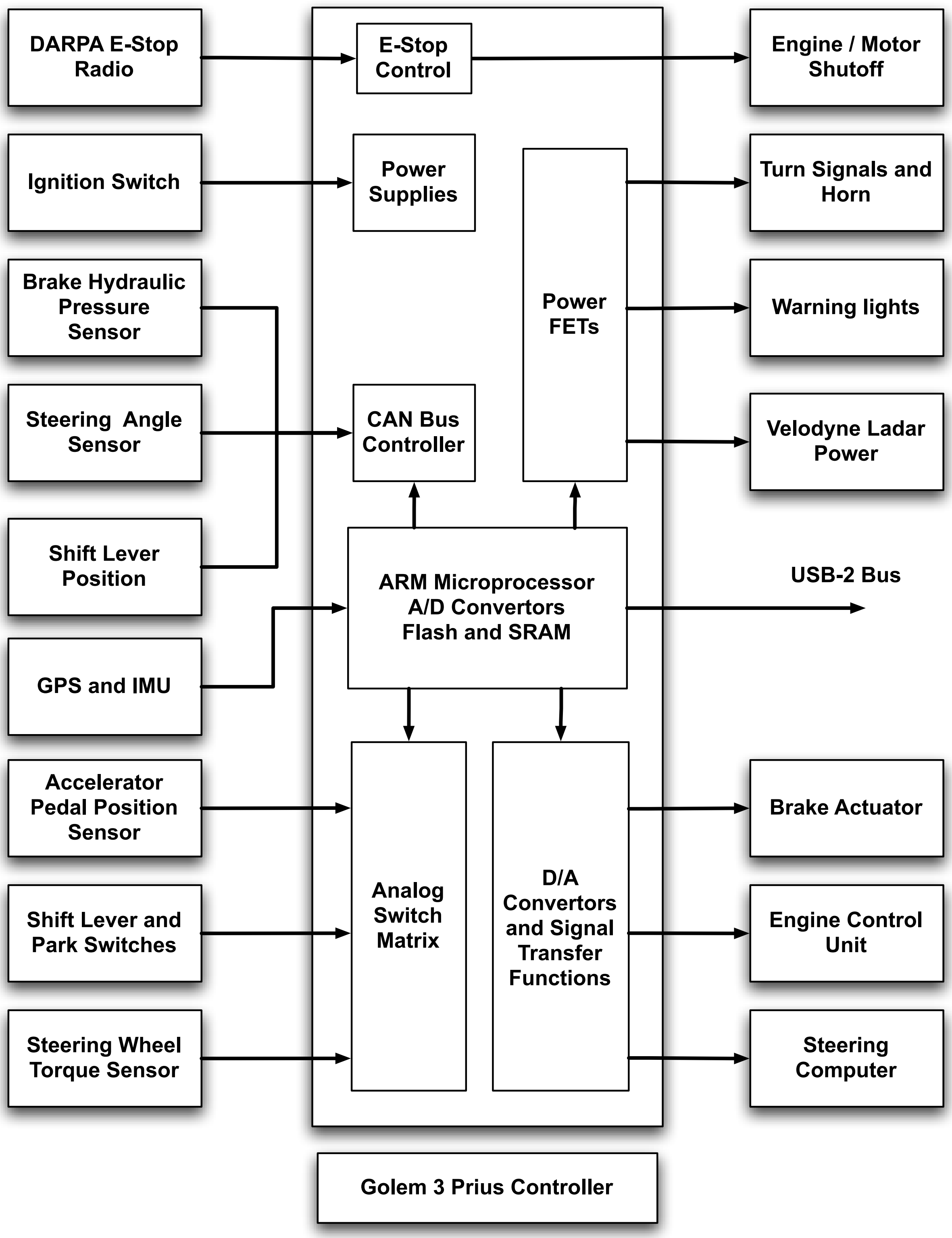


Figure 11: Control system block diagram.

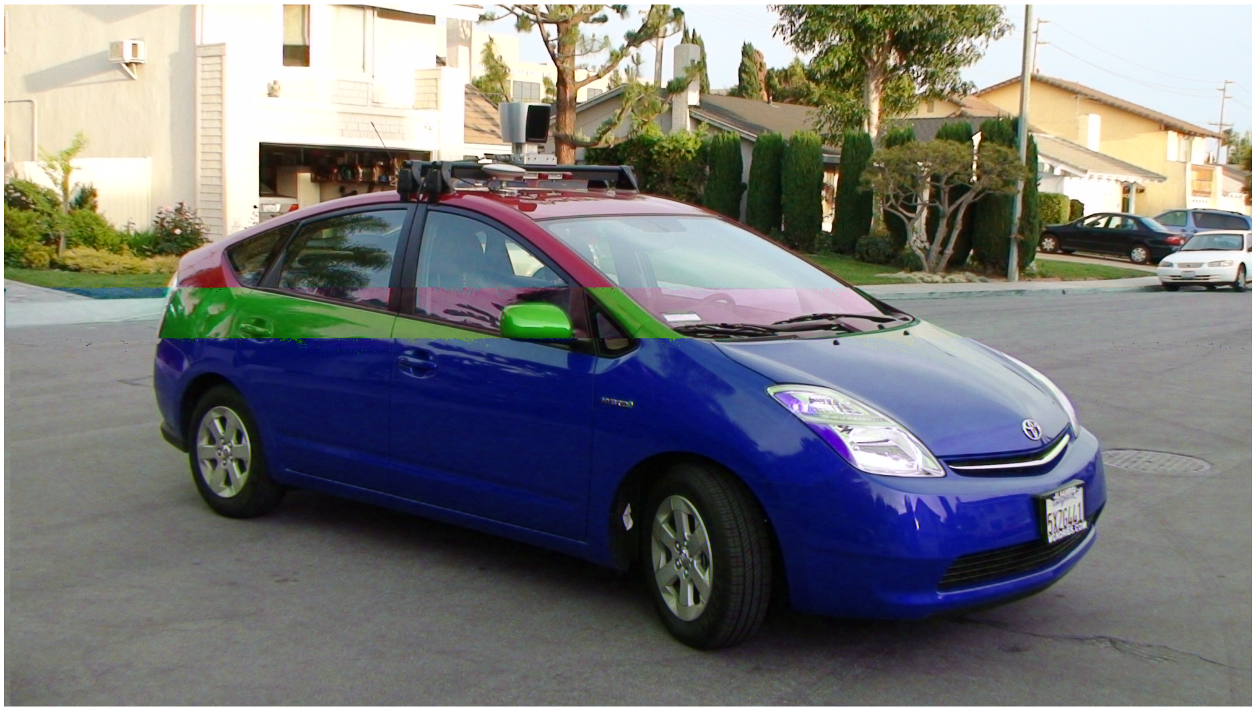


Figure 12: Golem 3.

Rather than interrupt this balance we have elected to control the brakes at the pedal via an external pneumatic system that allows an optional driver to apply additional braking at any time.

The E-stop system is also built in to the Prius controller to allow the complete disconnection of all signals necessary to stop both the motor and engine systems instantly and, at the same time, actuating the brake system in a preset controlled fashion to bring the vehicle to a quick safe stop. Provisions have been made to accommodate radios like those used in previous Grand Challenge as well as a hobby remote control to be used at the Milestone 2 site visit.

In order to keep the vehicle as fully human-usable as possible, we have minimized each subsystem to fit inside the vehicle without disturbing the interior cabin. The changes to the vehicle are not externally evident other than the addition of the Velodyne ladar system, the GPS antenna, the E-stop external switches and the warning lights, which are all mounted on a roof rack.

References

Bar-Shalom, Y. and Li, X. R. (1995). *Multitarget-Multisensor Tracking: Principles and Techniques*. YBS Publishing, Box U-157, Storrs, Connecticut.

Dodge.com (2006). *2006 Dodge Ram 2500 Specifications*. `http://www-5.dodge.com/vehsuite/VehicleCompare.jsp`.

Francfort, J., Nguyen, N., Phung, J., Smith, J., and Wehrey, M. (2001). Field operations program toyota prius hybrid electric vehicle performance characterization report. Technical Report INEEL/EXT-01-01522, Idaho National Engineering and Environmental Laboratory.

Gill, P. E., Murray, W., and Saunders, M. A. (2006). *User's Guide For SNOPT Version 7: Software For Large-Scale Nonlinear Programming*.

Kogan, D. (2005). Realtime path planning through optimization methods. Master's thesis, California Institute of Technology.

Mason, R., Radford, J., Kumar, D., Walters, R., Fulkerson, B., Jones, E., Caldwell, D., Meltzer, J., Alon, Y., Shashua, A., Hattori, H., Frazzoli, E., and Soatto, S. (2006). The Golem Group / UCLA Autonomous Ground Vehicle in the DARPA Grand Challenge. *Journal of Field Robotics*, 23:527–553.

Stentz, A. (1994). Optimal and efficient path planning for partially-known environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3310–3317.

Toyota.com (2007). *2007 Prius Specifications*. `http://www.toyota.com/prius/specs.html`.