Little Ben: The Ben Franklin Racing Team's Entry in the 2007 DARPA Urban Challenge

Jonathan Bohren, Tully Foote, Jim Keller, Alex Kushleyev, Daniel Lee, Alex Stewart, and Paul Vernaza
University of Pennsylvania
Philadelphia, Pennsylvania 19104
e-mail: ddlee@seas.upenn.edu

Jason Derenick and John Spletzer
Computer Science and Engineering
Lehigh University
Bethlehem, Pennsylvania 18015

Brian Satterfield
Lockheed Martin Advanced Technology Laboratories
3 Executive Campus, Suite 600
Cherry Hill, New Jersey 08002

Received 13 February 2008; accepted 28 July 2008

This paper describes "Little Ben," an autonomous ground vehicle constructed by the Ben Franklin Racing Team for the 2007 DARPA Urban Challenge in under a year and for less than $250,000. The sensing, planning, navigation, and actuation systems for Little Ben were carefully designed to meet the performance demands required of an autonomous vehicle traveling in an uncertain urban environment. We incorporated an array of a global positioning system (GPS)/inertial navigation system, LIDARs, and stereo cameras to provide timely information about the surrounding environment at the appropriate ranges. This sensor information was integrated into a dynamic map that could robustly handle GPS dropouts and errors. Our planning algorithms consisted of a high-level mission planner that used information from the provided route network definition and mission data files to select routes, whereas the lower level planner used the latest dynamic map information to optimize a feasible trajectory to the next waypoint. The vehicle was actuated by a cost-based controller that efficiently handled steering, throttle, and braking maneuvers in both forward and reverse directions. Our software modules were integrated within a hierarchical architecture that allowed rapid development and testing of the system performance. The resulting vehicle was one of six to successfully finish the Urban Challenge.

1. INTRODUCTION

The goal of the 2007 DARPA Urban Challenge was to build an autonomous ground vehicle that could execute a simulated military supply mission safely and effectively in a mock urban area. Compared with previous DARPA Grand Challenges, this particular challenge necessitated that robot vehicles perform autonomous maneuvers safely in traffic (DARPA, 2007). To address this challenge, the Ben Franklin Racing Team was formed by students and faculty at the University of Pennsylvania and Lehigh University and engineers at Lockheed Martin Advanced Technology Laboratory. In under a year and with a limited budget, the Ben Franklin Racing Team was able to construct "Little Ben," a drive-by-wire Toyota Prius with an array of onboard sensors and computers shown in Figure 1.

The Urban Challenge presented unique challenges to autonomous sensing, navigation, and control. Some of the scenarios that our vehicle needed to be able to handle included the following:

- Maintain appropriate safety margins at all times
- Accurately follow a lane within prescribed lane boundaries
- Detect and avoid moving traffic
- Stop and drive into a new lane in the presence of other vehicles
- Park in constrained spaces in dynamic environments

These situations required that obstacles and lane markings be detected at a distance and that the vehicle react quickly and appropriately while following the local traffic laws and conventions. An overarching requirement was that a successful system adhere to a stringent set of real-time processing constraints in its detection and reaction to its environment. This was reflected mainly in the system reaction time, as governed by the processing sample rate. Low sample rates increase the distance at which obstacles and other traffic vehicles must be detected for safe operation. Conversely, high sample rates are attainable only by using overly simplified sensing and control algorithms.

The design of our vehicle's hardware and software systems was predicated on achieving a reaction

Figure 1. Little Ben is a Toyota Prius hybrid vehicle modified for drive-by-wire operation with an onboard array of sensors and computers.

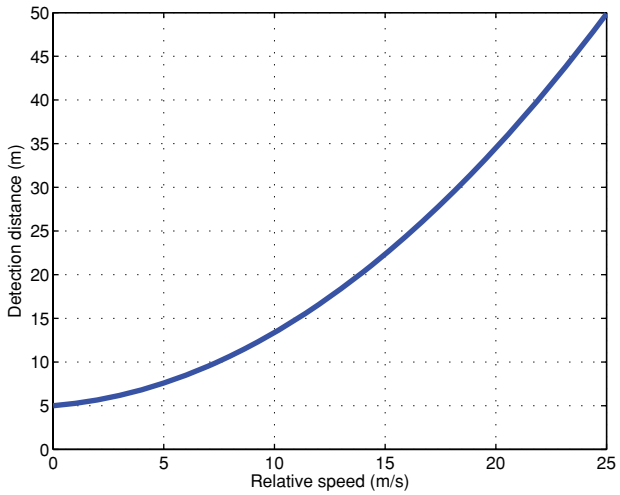

Figure 2. Required detection distance at various speeds taking into account worst-case latencies in system processing time.

time that ensured safe operation of driving maneuvers at the mandated upper speed limit of 30 mph (13.4 m/s). As an example of our design methodology, Figure 2 shows our calculation of the required detection distance of another vehicle in order for our vehicle to properly react and stop at various relative speeds up to 60 mph (26.8 m/s). In our calculations, we required that at least one vehicle length of separation be maintained at the end of the maneuver. We have also identified the maximum braking acceleration that can be introduced in this speed range without triggering the antilock brake system (ABS). Therefore, the ABS reaction dynamics were reserved as an additional safety margin when dry pavement conditions were not present.

We evaluated a range of possible system sample rates and selected 10 Hz as the desired system processing rate. Our calculations in Figure 2 took into account a two-sample-period delay (200 ms) as the worst-case scenario for detection and reaction; the first sample period could elapse just before an obstacle crossed the sensor detection threshold, and the second sample period was assumed to be used for the necessary computational processing.

The hardware and software systems were selected to meet the desired detection distance and processing time objectives. Sensors and their respective mounting positions were chosen to maximize their long-range detection characteristics. Drive-by-wire actuation and computer hardware systems were selected to minimize processing latencies. Similarly, our software modules were also optimized to maximize detection distance and minimize processing delays. This combination of hardware sensing systems with efficient, reactive software modules allowed Little Ben to achieve the requisite safety margins for driving in urban traffic situations.

2. VEHICLE PLATFORM

Little Ben was built from a 2006 Toyota Prius hybrid vehicle with modified controls to allow drive-by-wire as well as manual operation. Because the Urban Challenge took place in a *mostly* urban setting, there was no need for a large off-road vehicle. The Prius's compact size made many driving maneuvers easier to accomplish than by other larger vehicles and also proved to be very stable, reliable, and easy to work with.

Unlike most standard automobiles, Little Ben did not have an alternator. Instead it used power provided by the built-in 200-V hybrid battery via a dc–dc converter to power all standard 12-V vehicle components as well as the additional hardware that we installed. As shown in Table I, the total peak power consumption of the additional hardware systems was less than 700 W peak, well below the maximum 1-kW power output of the stock dc–dc converter. Thus, Little Ben did not require any specialized alternators or additional generators or cooling hardware. This overall power and fuel efficiency enabled Little Ben to finish the 57 miles (92 km) of the Urban Challenge using only about 1 gallon (3.8 liters) of gasoline.

Table I. Power consumption of sensors, computers, and vehicle actuation on Little Ben.

Device	Peak power consumption (W)
EMC (drive-by-wire)	150
3 SICK LMS-291	60
2 SICK LDLRS	60
1 Velodyne	60
7 Mac Minis	250
3 Hokuyo URG-04LX	15
1 OxTS Pose System	10
2 serial device servers	5
Ethernet switches, router	5
Total peak power	~650
Actual steady state	~450

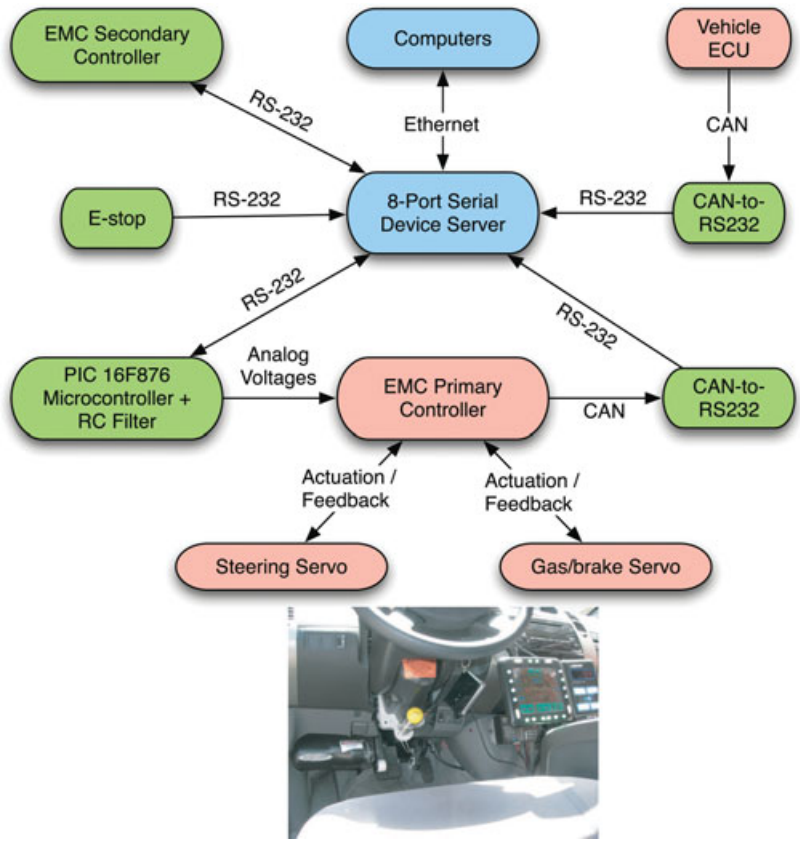

Figure 3. Components that interface to the drive-by-wire system.

2.1. Drive-by-Wire Actuation

As depicted in Figure 3, the drive-by-wire vehicle actuation was performed by Electronic Mobility Controls (EMC) of Baton Rouge, Louisiana. This conversion included dc servomotors to actuate the steering wheel and gas/brake pedals, along with triple-redundant motor controllers to ensure safe operation. Two analog dc voltage inputs were provided by EMC to control the steering wheel and gas/brake pedal position. To transmit the digital control signal from the computers to the drive-by-wire system, we implemented a very simple digital-to-analog converter using a cheap digital programmable integrated circuit (PIC) microcontroller with a resistor-capacitor (RC)-filtered pulse-width modulation (PWM) output. Given that the drive-by-wire analog signals were sampled at 100 Hz, the RC time constant of the filter was chosen to be approximately 10 ms. This ensured that the full actuation bandwidth was preserved while smoothing any electrical noise interference in the vehicle. The PWM frequency of the microcontroller was set to 20 kHz with 8-bit resolution, which was sufficient for smooth and accurate control of the actuators.

Other vehicle controls such as transmission shifting, turn signals, and parking brake were interfaced via a single RS-232 connection to EMC's secondary controller unit. Additionally, we installed a CAN bus interface to the Toyota onboard diagnostic (OBD) connector in order to verify vehicle state information directly from the car's electronic control unit (ECU). The CAN interface provided accurate brake pedal position at 100 Hz, steering encoder feedback at 70 Hz, and other vehicle state information such as transmission shift setting at slightly lower rates.

2.2. Emergency Stop

Because safety is a top priority with autonomous vehicles, we took major steps toward minimizing the

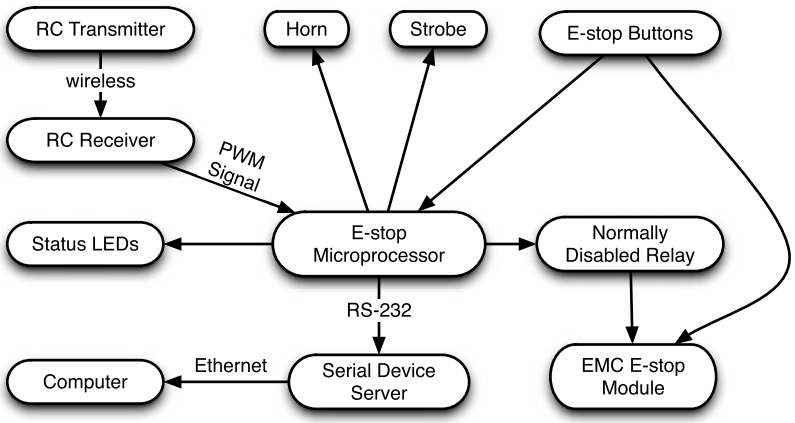

Figure 4. Block diagram depicting the emergency stop and safety systems incorporated into the vehicle.

risk of injury or damage due to undesired behavior of the vehicle. The emergency stop system was designed to make human intervention safe, quick, and reliable. To achieve fail-safe operation, redundancy was incorporated on multiple levels using watchdog timers and heartbeat monitors as shown in Figure 4.

When the "pause" mode was activated, either by a human operator or when the radio-controlled transmitter was out of range, the throttle commands from the computers were automatically overridden and the brake was applied at near-maximum braking acceleration to ensure smooth stopping within the allowed distance. In this mode, the computers were unable to drive the vehicle unless the "run" command was explicitly given. After the run command was given, the audible and visual strobe warning devices were activated, and control was returned to the computer systems after a 5-s delay.

Our "disable" mode was an extension of the pause mode. In addition to braking the vehicle and disabling computer control of the throttle, the E-stop processor verified that the vehicle speed was zero on the Toyota CAN bus and set the transmission to *park*. All the Toyota Prius systems were then powered down. In this state, the vehicle would need to be manually restarted in order to reactivate autonomous control. The vehicle could easily be disabled via the dedicated remote control or the manual E-stop buttons located on either side of the car.

To achieve high reliability, the most crucial components of the safety system were implemented using simple PIC microcontrollers and fail-safe mechanical relays. These were powered using the backup battery system of the EMC drive-by-wire system, so even without vehicle power or computers, the car was guaranteed to respond properly to pause and disable commands.

2.3. Roof Rack

Because of constraints on our local storage facilities, our primary sensor rack was designed such that it could be quickly mounted and unmounted as a single structure without having to recalibrate the sensors. The stock beams from a Yakima roof rack were replaced with aluminum pipes onto which an 80/20 aluminum structure was rigidly fixed. Once locked into place, the sensor rack was connected to the vehicle power and computing systems through a single umbilical connector.

To maximize the detection range of our sensors, the rack shown in Figure 5 was custom designed to allow optimal viewing angles for as many of these sensors as possible. In particular, we designed the rack to accommodate a Velodyne HD LIDAR to give the omnidirectional sensor a fully unoccluded 360-deg azimuthal view. By mounting the Velodyne 8 in. (20 cm) above the rest of the sensor rack, we took advantage of the full complement of elevation angles in the sensor to provide a sensing range from 4 to 60 m around the vehicle. Its lateral left-of-center position was also optimized for navigating around obstacles on American roads.

The rack also integrated a set of forward- and rear-facing SICK LMS-291 S14 LIDAR sensors. These 90-deg-field-of-view sensors were tilted downward

Figure 5. Roof sensor rack designed to provide optimal viewing angles.

in order to intersect the ground at approximately 6–7 m ahead of and behind the vehicle. In these positions, the SICK LIDARs were well within their range limitations and could provide for both ground plane, obstacle and lane marking detection. These sensors were also arranged so that they did not occlude the Velodyne's field of view and provided a complementary stream of range data.

The rack also contained the warning siren, strobe lights, and mounting points for the global positioning system (GPS) antennas. Additionally, it provided space for a weatherproof electronics enclosure. This contained and protected the power distribution block and connectors for sensors mounted on other parts of the vehicle.

2.4. Hood Sensors

Little Ben also integrated several sensors that were mounted on its hood. At the front center of the hood was a vertical scanning SICK LMS-291 LIDAR, as well as a high-resolution Point Grey Bumblebee stereo camera as shown in Figure 6. The $1{,}024 \times 768$ resolution color camera had a horizontal 50-deg field of view, with a frame rate of 15 Hz. To minimize the potential for image blooming caused by sunlight, the camera was pitched down 15 deg to minimize the field of view over the horizon. A visor was also integrated as a further level of protection.

In addition, two SICK LD-LRS LIDAR scanners were mounted parallel to the road surface at the front left and front right corners of the hood. These scanners provided overlapping 270-deg fields of coverage and were used to detect obstacles and track moving vehicles in front and at the sides of Little Ben. The LD-LRS sensors employed a scanning frequency of 10 Hz, reporting laser returns at 0.5-deg increments. Owing to their vulnerable position and possible misalignment in the event of a crash, a simple cardboard fiducial marker was attached to the hood and used to automatically verify correct operation of these sensors.

2.5. Other Sensors

Three compact Hokuyo URG-04LX LIDAR scanners were also used to cover blind spots in sensor coverage at short range around the vehicle as shown in Figure 7. Although these sensors were rated only for indoor use, through experimentation we found that they could be used in outdoor conditions as long as they were properly shielded from water and from light in the back. Two Hokuyo scanners were mounted underneath the side mirrors for detecting obstacles such as curbs at the sides of the front wheels, as well as nearby lane markings on the

Figure 6. Sensors mounted on the hood of Little Ben.

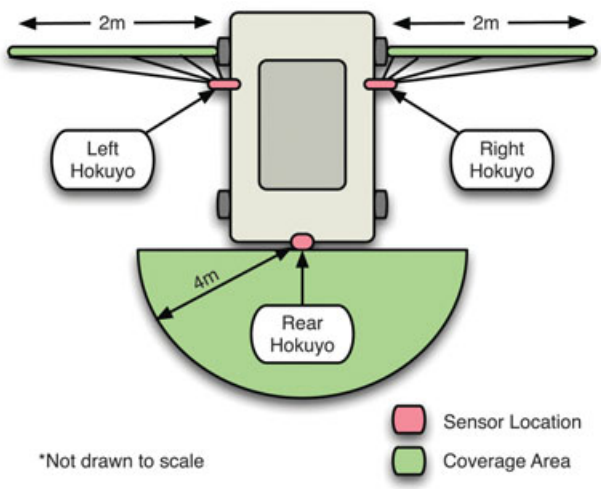

Figure 7. Additional Hokuyo scanners used to eliminate blind spots at short range.

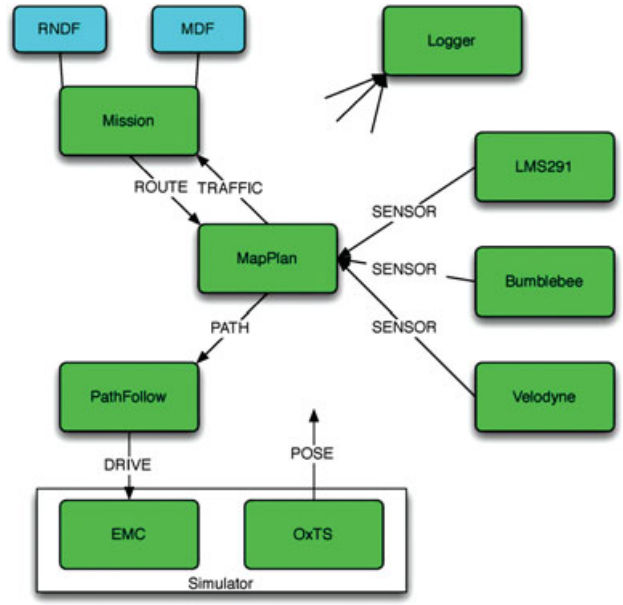

Figure 8. Software architecture showing system modules and corresponding interprocess communication messages.

ground. The third sensor was mounted slightly above the rear bumper and allowed for accurate maneuvering between tightly spaced obstacles while in reverse.

3. SOFTWARE ARCHITECTURE

As depicted in Figure 8, the software architecture was divided hierarchically into a series of modules, connected via interprocess communication messages. At the lowest level was the driving module, which was responsible for interfacing to the vehicle controller hardware and verifying correct operation of the steering, throttle, braking, and transmission. Also present at this low level was the pose software module, which integrated readings from the GPS and inertial navigation system (INS) to provide the latest pose information at 100 Hz. These two hardware interface modules could be readily replaced by a simulation module, which allowed us to rapidly test the software without requiring the processes to be physically connected to the vehicle systems.

At the highest level, the mission planning module read the appropriate route network definition file (RNDF) and mission data file (MDF) to determine the optimal sequencing of waypoints needed to complete the mission objectives. Next were the sensor modules, which gathered data from all the LIDARs and the stereo camera to provide probabilistic real-time estimates of the terrain, road markings, and static and dynamic obstacles. These modules consolidated the large amount of sensor data into a compact representation in the vehicle's local reference frame before sending this information onto the MapPlan process.

The MapPlan process was then responsible for integrating all the sensor information into a probabilistic dynamic map and then computing the appropriate vehicle path to reach the next desired waypoint as determined by the high-level mission planner. It also checked to ensure that this path avoided all known obstacles while obeying vehicle dynamic constraints as well as local traffic rules. The PathFollow module took the desired vehicle path from the MapPlan process and generated the optimal steering, throttle, and braking commands needed by the low-level driving module.

All the processes communicated with each other via well-defined message formats sent through the Spread messaging toolkit (Amir, Danilov, Miskin-Amir, Schultz, & Stanton, 2004). This open-source messaging system provided message reliability in the presence of machine failures and process crashes, while maintaining low latencies across the network. It also enabled convenient logging of these messages with appropriate time stamps. These logs allowed us to rapidly identify and debug bad processes, as well as replay logged messages for diagnostic purposes.

These modules were written mainly in Matlab with some ancillary C++ routines. The use of high-level Matlab enabled the software system to be written in fewer than 5,000 lines of code. We also implemented a development environment that incorporated Subversion for source code tracking, Bugzilla for assigning tasks, and a Wiki for writing documentation. All the documentation was readily accessible to the whole team with convenient search functionality to allow easy collaboration. During field testing, local copies of the Bugzilla and source code repository were stored within the vehicle to allow us to make offline changes that were merged with our central servers after testing. With these tools, rapid prototyping and development was accomplished by the team both in the laboratory and in the field.

All the software processes were run on a small computer cluster, consisting of seven Mac Minis with Core 2 Duo processors running Ubuntu Linux. The computers were interconnected through a gigabit Ethernet network in the vehicle trunk as shown in Figure 9. Serial connections to the vehicle's low-level hardware and sensors were also provided over the Ethernet network via Comtrol serial device servers. In the event of a computer failure, our system automatically switched the affected processes over to a redundant computing node without having to manually reconfigure any connections. Special monitoring software (monit) was used to constantly check the status of all the computers and processes to detect software crashes and other possible failures.

To prevent the various data streams from interfering with one another, Little Ben contained three separate subnetworks isolated using hardware routers and switches. The first subnet was used for normal interprocess communications between the Mac Minis. The second subnet was used to isolate the various LIDAR sensors: it was found that some of the SICK sensors contained buggy network protocol implementations and would lock up in the presence of extraneous Ethernet traffic. Finally, the third subnetwork was used to isolate the large amounts of data broadcast by the Velodyne LIDAR sensor (approximately 3 MB/s); only those computers processing this data stream would subscribe to this subnet.

4. PERCEPTION

Little Ben's perception system was responsible for providing information about the locations of static

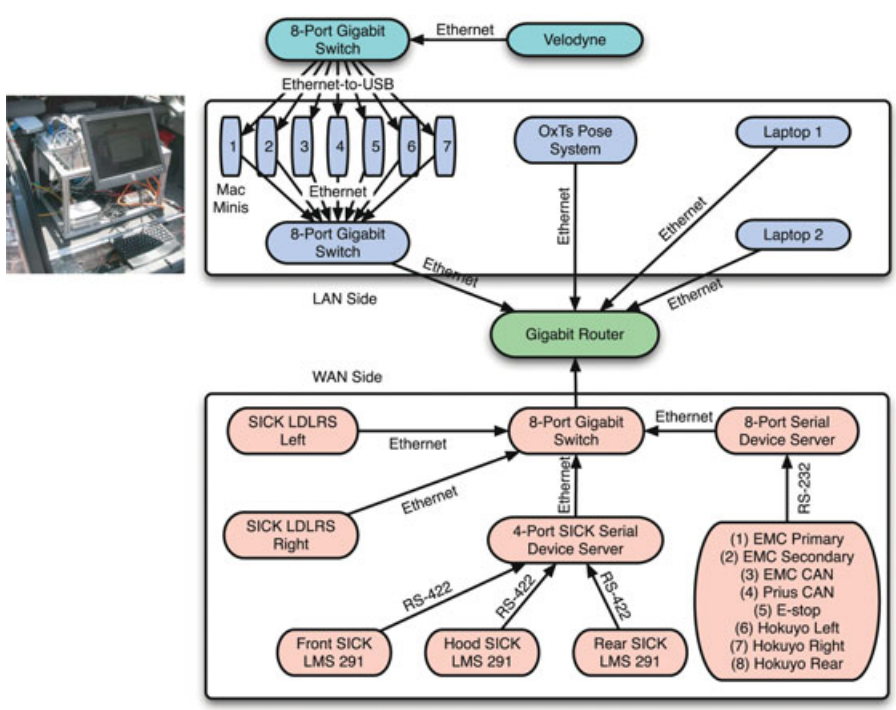

Figure 9. Computing and networking systems onboard Little Ben.

obstacles, traversable ground, moving vehicles, and lane markings on the road. This processing was performed in a highly redundant manner by the various sensors on the vehicle: Velodyne LIDAR, SICK LIDARs, Hokuyo LIDARs, and Bumblebee stereo camera.

4.1. Velodyne Processing

The Velodyne HDL-64E LIDAR was Little Ben's primary medium-to-long-range sensor. It was used for geometric obstacle/ground classification and road marking extraction, as well as dynamic obstacle velocity tracking. The Velodyne houses 64 905-nm lasers and can spin between 5 and 15 Hz, yielding a field of view of 360-deg azimuth and −24- to +2-deg zenith. We configured the sensor to spin at 10 Hz in order to acquire a high point density and capture frames at the same rate as our control system. A sample scan is shown in Figure 10.

Although we were supplied with the factory horizontal and vertical correction factors, we found that the individual lasers required an additional distance offset that we calibrated using comparisons to the readings from the SICK LMS-291 sensors. Even with this extra calibration, we found that our particular Velodyne sensor would sometimes report laser ranges with uncertainties on the order of 30 cm, much larger than the stated 5-cm accuracy. Because of this large uncertainty, it was necessary to process the Velodyne data as 64 independent scans, rather than aggregating returns between different lasers.

Some of the individual lasers would also sporadically return large noisy outliers. Because of this, the range and reflectivity values from each laser were carefully monitored online and rejected if any inconsistent outliers were detected. Classifications of ground versus obstacle were also never based on a single return but on the statistics of a consecutive set of four to five points.

In this manner, we were able to classify reflective obstacles such as other vehicles out to 60 m and detect ground points out to 30 m under good conditions, depending on the reflectivity of the ground. The reflectivity data were also used to detect lane markings during the Urban Challenge within a range of 15 m.

4.2. LIDAR Ground/Obstacle Detection

The range scans from the downward-angled LIDARs, SICK LMS-291s and side-mounted Hokuyos, were processed in the following manner. In the first phase

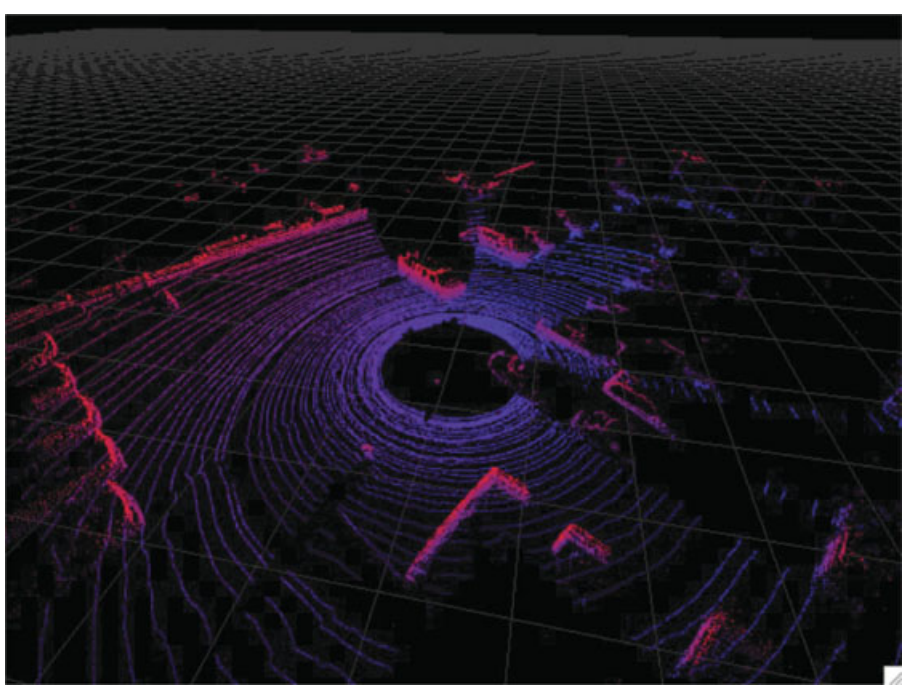

Figure 10. Three-dimensional point cloud from a single Velodyne scan classified as ground and obstacle.

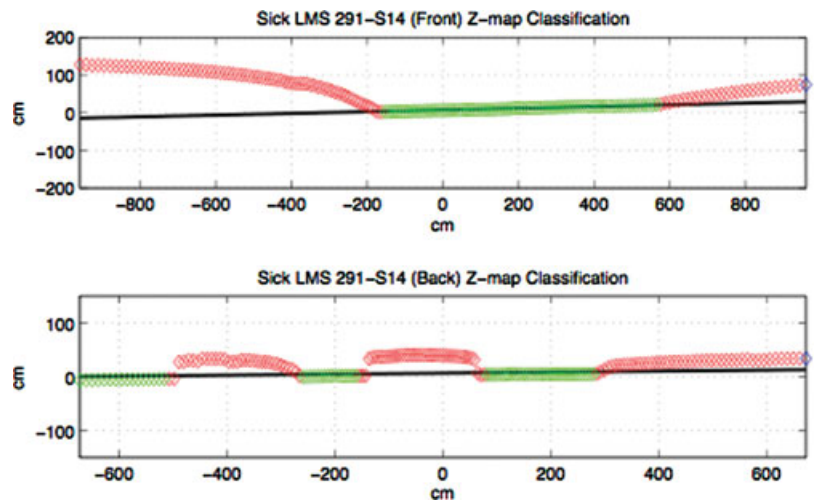

Figure 11. Robust ground plane extraction and ground/obstacle classification from the front- and rear-facing SICK LMS-291 sensors.

of the algorithm, a ground plane was fitted in a robust fashion to the observed points. The range values from a scan were first passed through a median filter to remove spurious returns due to airborne dust particles, rain, etc. Then given the observed Cartesian points from the filtered LIDAR scan, (x_i, z_i), we minimized the following objective:

$$\min_{m,b} \sum_i f(z_i - mx_i - b), \tag{1}$$

where f was an error measure that was quadratic near zero but decreased much more slowly for larger values. This minimization was performed in an incremental fashion using interative least squares (Guitton, 2000).

To improve the robustness of the ground plane fit, we also employed regularization of the ground plane parameters based on the relative geometry of the vehicle and sensor calibration. This enabled our algorithm to accurately track the ground as shown in Figure 11, even in the presence of significant pitch changes as well as highly linear obstacle features.

Once an accurate ground plane had been determined, it was relatively easy to classify the various observed scan points as obstacle or ground based on their deviation to the ground plane. Another example of this classification is shown in detecting a nearby curb from a side-mounted Hokuyo scanner as shown in Figure 12.

4.3. LIDAR Lane Marking Detection

In addition to streaming range information, the Velodyne, SICK LMS-291-S14, and Hokuyo LIDARs also returned corresponding reflectivity values. This information was used for detecting and identifying lane markers in order to compensate for any positional shifts in our pose system.

To detect lane markings from the LIDAR returns, the reflectivity readings of identified ground points were first analyzed. Sections of ground whose reflectivity values were significantly higher than the surrounding road surface were then identified as

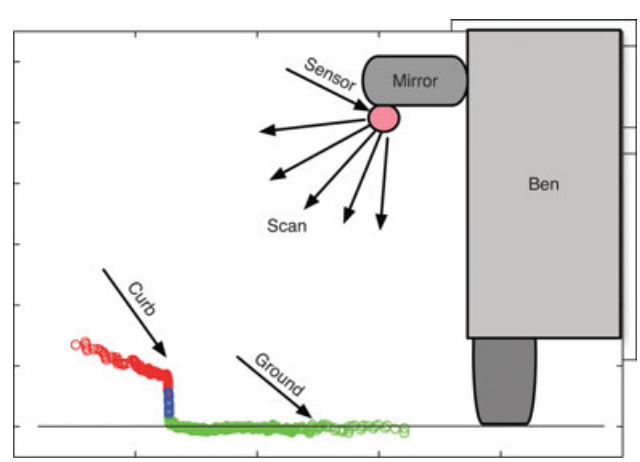

Figure 12. Curb/ground detection from side-facing Hokuyo LIDAR.

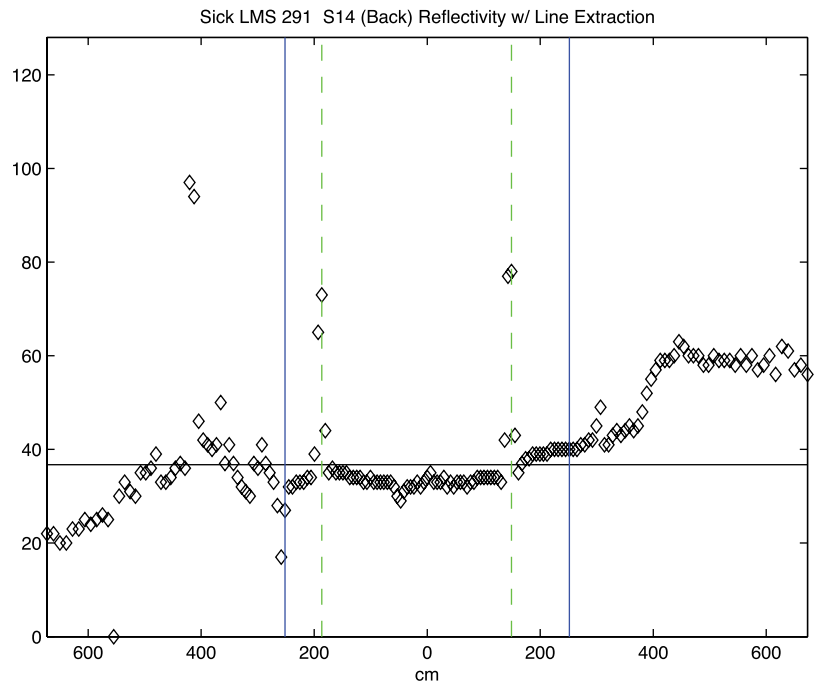

Figure 13. Lane marking detection and identification from front SICK LMS-291.

potential markings. These sections were then checked to see whether they corresponded to line widths between 10 and 15 cm wide. Figure 13 shows an example of lane marking detection and identification from the front-facing SICK LMS-291 sensor. The two peaks correspond to the left and right lane markings present in the lane.

4.4. Dynamic Obstacle Tracking

To successfully navigate through dynamic obstacles, obstacle velocities as well as positions needed to be accurately estimated. Budgetary and time constraints precluded us from incorporating any type of RADAR sensors, so Little Ben tracked dynamic obstacles using consecutive LIDAR returns from the Velodyne sensor as well as consecutive returns from the hood-mounted SICK LD-LRS scanners.

First, classified obstacle range returns were grouped into local line features. The line features were then tracked across consecutive scans using a multiple-hypothesis Kalman filter. This filter rejected spurious detections based on prior constraints on the size and velocities of known obstacles.

Figure 14 shows the output of the algorithm tracking two vehicles based on consecutive range readings from one of the hood-mounted SICK LD-LRS scanners. In this manner, moving obstacles within a range of approximately 60 m could be tracked with an accuracy of about 1 m/s.

4.5. Vision

The Bumblebee stereo vision system was also used to recover road markings at ranges from 4 to 15 m ahead of the vehicle. Constraining our interest to this region yielded more robust feature segmentation and more reliable stereo disparity estimates and allowed a linear model to be used in reconstructing the lane markings.

Images were processed at 512×384 resolution at a rate of 15 frames per second, using approximately 50% of a single core of one of the Mac Mini processors. Figure 15 shows an example of the output of our vision system in detecting and locating lane markings relative to the vehicle.

Images from the stereo camera were first enhanced and then subtracted from corresponding pixel-shifted locations in the left and right images. The appropriate pixel shifts were determined by calibrating the camera relative to the ground plane. This could also be done adaptively using the observed

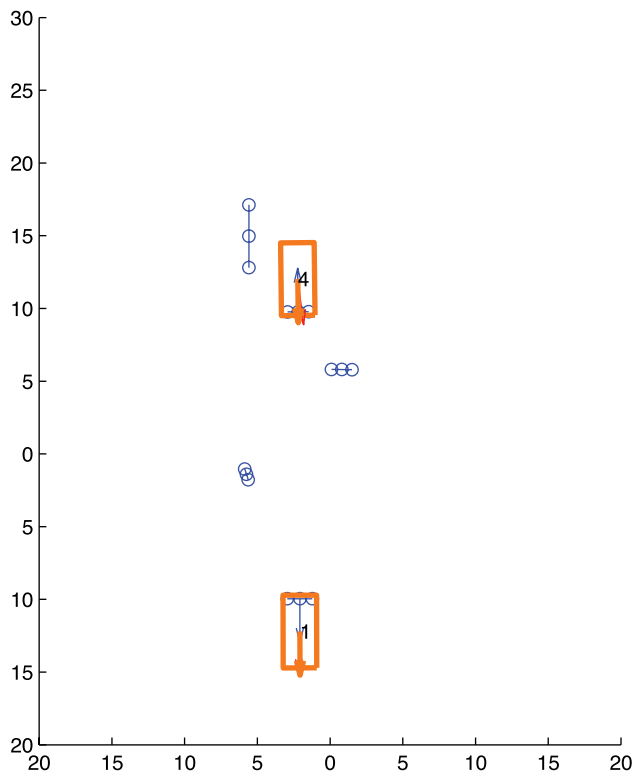

Figure 14. Vehicle heading and velocity tracking from SICK LD-LRS sensor; Little Ben is located at the origin of the figure.

stereo disparity values. Valid lane markings were then detected using a variety of filters that test image region candidates based on width, length, and area constraints. Candidate lines were then projected onto the road surface and placed into the map relative to the current vehicle location.

5. MAPPING

Our previous experience with autonomous outdoor navigation has underscored the need for robust mapping that is consistent with perceptual data as well as prior information about the environment (Vernaza & Lee, 2006). As the vehicle traversed its environment, perceptual data were distilled into local occupancy grid maps (Elfes, 1989). These maps were referenced to the local coordinate system of the vehicle and reflected the state of the world as observed at a specific instant in time.

As information about static obstacles, dynamic obstacles, and lane markings was sent by the perceptual modules, the MapPlan module updated the various ground/obstacle and lane marking likelihoods in a 300 × 300 m map, roughly centered at the current vehicle location. The current vehicle pose was obtained from an Oxford Technical Solutions RT-3050 unit. The RT-3050 is a self-contained unit that uses a Kalman filter–based algorithm to combine inertial sensors, GPS updates with differential corrections from the OmniStar VBS service, and vehicle odometry information from the native embedded vehicle systems (Kalman & Bucy, 1961). The RT-3050 was able to provide pose estimates at a high update rate of 100 Hz with a stated accuracy of 0.5 m. The unit was specifically designed for ground vehicle testing applications and was capable of providing pose estimates during periods of sustained GPS outages or poor GPS performance due to environmental effects such as multipath reflections.

Given the vehicle pose estimates, the various perceptual measurements were fused into the current map. Figure 16 shows a snapshot of the map shortly after the beginning of the Urban Challenge, when

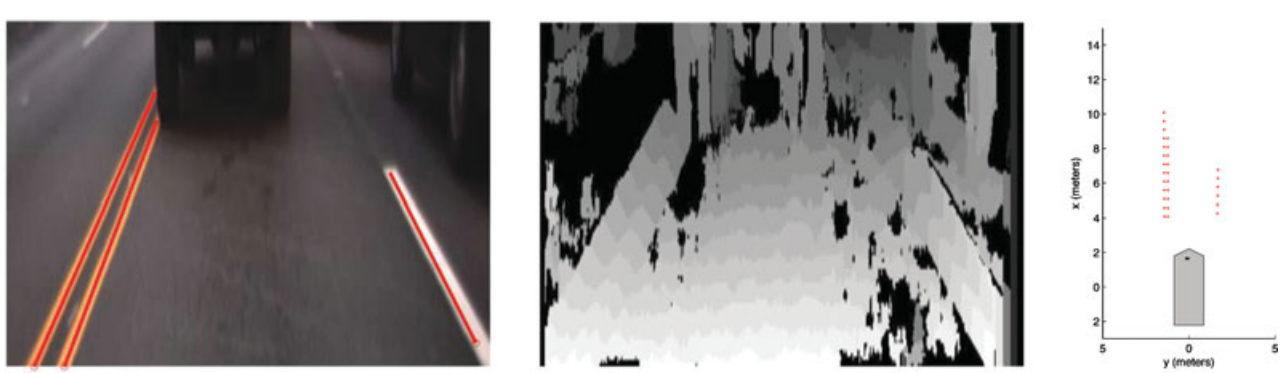

Figure 15. (Left) Camera image with segmented lanes; (center) corresponding disparity image; (right) reconstructed lane relative to vehicle.

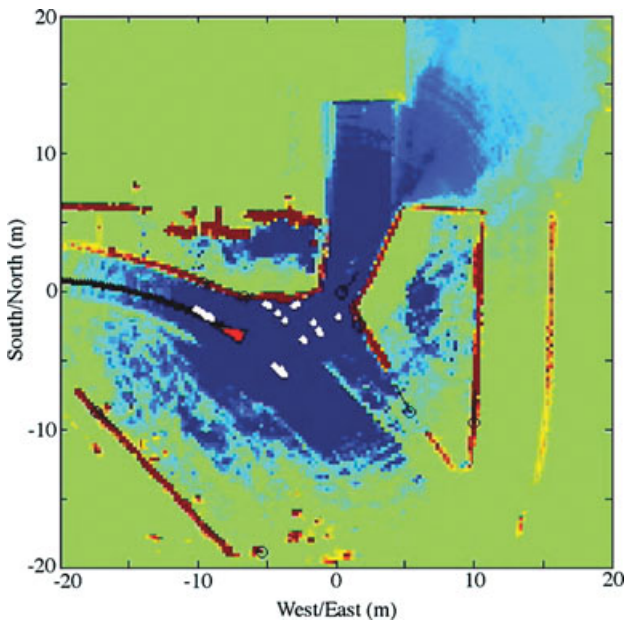

Figure 16. Probabilistic map with obstacle/road (red/blue) likelihoods along with lane markings (white) generated near the beginning of the Urban Challenge course.

Little Ben entered the two-lane loop. The walls and road surface, as well as road markings, can be clearly seen in this map.

6. PLANNING AND CONTROL

6.1. Mission and Path Planning

In our hierarchical software architecture, planning was performed in two stages. At the highest level, the mission planner estimated travel times between waypoints and then computed the optimal sequence of waypoints to traverse in order to minimize the overall mission time. When a particular lane or intersection was blocked, the mission planner recomputed an alternative sequence of waypoints using Dijkstra's algorithm to adaptively respond to traffic conditions (Dijkstra, 1959). Figure 17 illustrates the display from the mission planner as it monitors the progress of the vehicle through a route network.

The next stage of planning incorporated information from the dynamic map by computing a detailed path to the next waypoint. Depending on the current sequence and next waypoint type in the RNDF, a specialized local planner that dealt with lane following, U-turns, intersections, and zones separately was selected. The lane following planner optimized

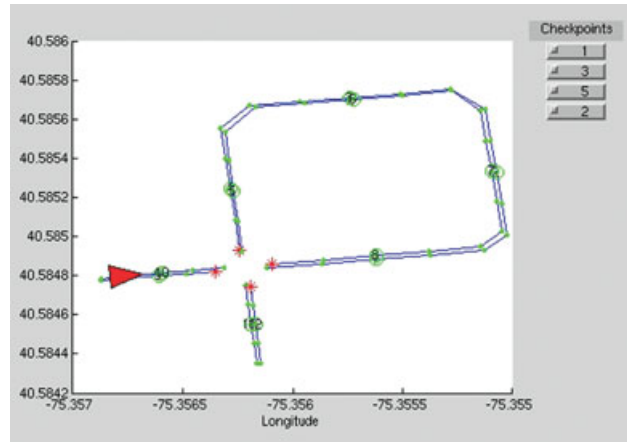

Figure 17. Mission planner uses information from the RNDF and MDF to plan optimal routes through the traffic network.

a continuous set of lateral offsets from the path given by the RNDF. The U-turn planner monitored the road edges and obstacles while transitioning between forward and reverse driving modes. On the other hand, the intersection planner first monitored the waiting time and other vehicle positions while computing the optimal path through the intersection box. Finally, the zone parking planner used a fast nonholonomic path planner to find an optimal path to the next waypoint in the zone. Each of these planners computed the desired geometric path using the current map costs and a maximum safe driving speed by computed time to possible collisions from the tracked dynamic obstacles.

6.2. Path Following

The path follower module was responsible for calculating the vehicle steering and throttle–brake actuation commands required to follow the desired trajectory as accurately as possible. The trajectory specified the desired route as a set of points for which the spatial position and the first and second derivatives were defined.

Previous approaches to steering control for autonomous car-like vehicles have used proportional–integral–derivative (PID) control–based methods with error terms that combine both the lateral and heading offsets from the desired trajectory (Coulter, 1992; Rasmussen, Stewart, Burdick, & Murray, 2006; Thrun et al., 2006). A weakness of these controllers in this application is that they do not explicitly consider

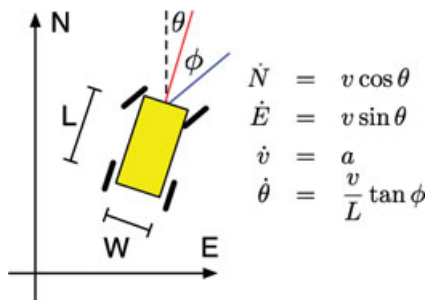

Figure 18. "Bicycle" model of the car dynamics used for control.

the kinematic constraints of the vehicle when calculating the steering command. These controllers also typically require significant reparameterization in order to operate the vehicle in reverse.

To avoid these shortcomings, we developed an alternative approach for steering control that integrated the dynamics of a vehicle model (Figure 18) to predict the resulting change in pose after a short period of time under a set of possible steering commands as shown in Figure 19 (Gillespie, 1992). A cost function is then evaluated for each of the predicted poses, and the steering command that minimized this cost function was chosen.

As illustrated in Figure 20, the particular form of the cost function used in our controller was as follows:

$$C(\phi_i) = E_{\text{lateral}}^2 + \left[R_\theta \sin\left(\frac{E_{\theta_i}}{2}\right)\right]^2, \qquad (2)$$

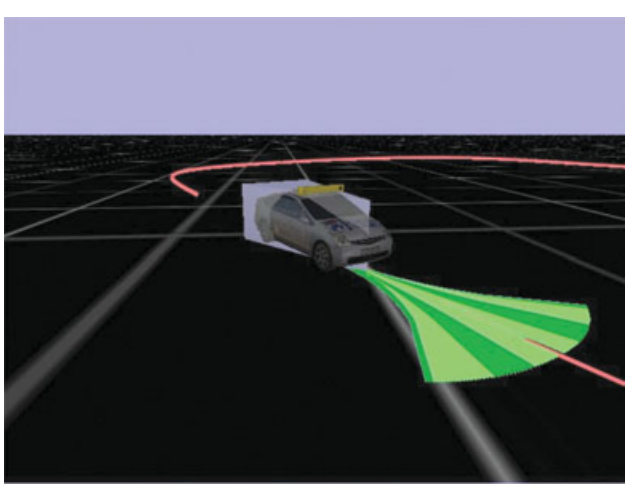

Figure 19. Estimated future vehicle poses for a set of possible steering commands in a simulated environment.

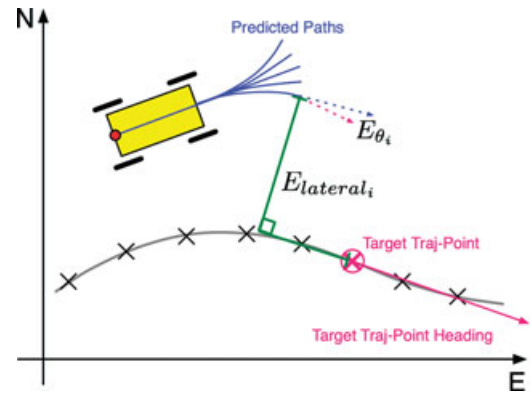

Figure 20. Graphical representation of the terms included in the controller cost function.

where E_{lateral} and E_{θ_i} are the lateral and heading offsets of the vehicle relative to the target point on the trajectory. Note that there is a length parameter R_θ in the cost function that was used to scale the heading error relative to the position error. This parameter was adaptively tuned to maximize performance in the different Urban Challenge scenarios.

The advantage of this value-based controller was that it was quite robust to vehicle dynamics and could be used just as effectively when the vehicle was operated in either reverse or forward. This allowed us to accurately control the vehicle in situations requiring tight navigation such as in lane changing or parking.

The speed of the vehicle was controlled by a proportional-integral (PI) controller after linearization of the throttle and brake dynamics. The controller's error term was the difference between the desired speed set by the path planner and the current speed as measured by the pose system.

6.3. Overhead Imagery Registration

We used the DARPA-provided overhead imagery to aid in preprocessing the RNDF to yield a more precise description of the roadways. In particular, our primary goal in processing the RNDF was to add a heading to each lane point corresponding to the tangent vector to the road at that point. These tangents were then used to better plan smooth trajectories connecting pairs of waypoints. We did not artifically add extra waypoints to "densify" the given RNDF.

Before this could be accomplished, we needed a good estimate of the mapping from universal transverse mercator (UTM) coordinates to image coordinates in the overhead imagery. We found this

transformation by fitting an affine model mapping the UTM coordinates of the known corner points to their known pixel locations. To refine this fit, we found additional fiducial points to include in the regression, in the form of the four surveyed points in the team pits whose GPS coordinates were given by DARPA. We deduced the image coordinates of these points by measuring their real-world distances from visible fiducials in the image.

7. NQE PERFORMANCE

Prior to the National Qualifying Event (NQE), a "Red Team" was formed by engineers from Lockheed Martin Advanced Technology Laboratory who were not involved in the design and development of Little Ben. This team came up with a series of eight weekly tests at three different sites in the 2 months before the NQE. These tests stressed various aspects of autonomous driving as specified in the DARPA guidelines. This independent validation gave the team confidence in Little Ben's abilities in unknown environments during the NQE.

On the basis of its performance during the qualifying rounds, Little Ben was chosen as a finalist for the Urban Challenge Final Event (UFE). However, several significant refinements were made to both the vehicle hardware and software prior to the UFE in order to account for deficiencies identified during the NQE test phase.

For example, prior to the NQE the dynamic vehicle tracking described in Section 4.4 was performed entirely by the Velodyne system. However, the merge operation required in NQE Course A exposed significant blind spots of the Velodyne due to occlusions from road signage, as well as the large electronic speed monitor used by DARPA test vehicles for speed management. This motivated the additional integration of the SICK LD-LRS hood-mounted units for vehicle tracking. As the Velodyne was already processed as 64 independent LIDAR instances, integrating data from the two additional LD-LRS units was relatively straightforward.

A second potential blind spot was identified during the parking operation required in NQE Course B. As part of this test, the vehicle was required to pull straight into an open parking spot with parked cars on both the left and right sides. Because the front of the parking spot was also blocked by a third car, exiting would require Little Ben to reverse out. However, reversing was aggravated by the placement of a large obstacle parallel to the row of cars. Whereas during the NQE there was sufficient clearance for Little Ben to exit without incident, a more severe test during the UFE might result in losing sight of a low-height obstacle. It was this requirement that motivated the integration of the rear-bumper-mounted Hokuyo LIDAR described in Section 2.5. In fact, this sensor was integrated the evening before the UFE! Although the parking test during the UFE was in fact far simpler than the NQE requirement, the ability to seamlessly integrate another sensor only hours before the final event—and with very limited testing—is itself a testament to the flexibility and modularity of our software architecture.

One final observation from the NQE was regarding our approach to processing the LIDAR data. Because Little Ben relied almost entirely on LIDAR for exteroceptive sensing (no RADARs), we placed significant emphasis on robust estimation and outlier rejection. We observed other vehicles misclassify dust clouds thrown up by their tires as phantom obstacles and stop until these clouds dissipated. However, the temporal filtering and spatial smoothness constraints used with the SICK and Velodyne systems, respectively, made Little Ben robust to such false positives.

8. UFE PERFORMANCE

On the basis of its performance during the NQE, Little Ben was seeded fourth entering the UFE competition. Overall, Ben's performance during the final event was quite good. The most significant shortcoming during the 57-mile (92 km) race occurred during the first mission of the UFE. This involved the off-road portion of the course known as "the Outback" that connected Montana Street with Phantom Road East. During development, we had operated exclusively on paved roads and approximately planar off-road surfaces. In contrast, the Outback was a steep grade with dramatic pitch changes over short distances. As a result, Little Ben stalled at the bottom of the Outback as it transitioned to Phantom Road East. Owing to the extreme pitch of the dirt road at the requisite stop line and low suspension and bumper clearance of Little Ben, the paved road surface was nearly touching the front bumper at this point in the course. This is shown in Figure 21. From this pose, the perceptual system interpreted the road surface as an obstacle immediately in front of the vehicle and refused to proceed through the intersection. By repositioning

Figure 21. Little Ben temporarily stalled on the transition from the Outback to Phantom Road East. The steep pitch of the transition resulted in the road surface being falsely classified as an obstacle.

Little Ben a few meters ahead, he was able to continue through the remainder of the course.

Mission 1 also saw Little Ben execute what was arguably the most intelligent maneuver of the UFE. This occurred at a four-way intersection and required Little Ben to interact with three other robot vehicles and an even larger number of human-operated traffic vehicles, as shown in Figure 22. Little Ben (in dashed box) arrived at the intersection with the UCF entry temporarily stalled to the right and the MIT vehicle stopped to the left (upper left). Little Ben obeyed intersection precedence and waited for MIT and UCF to proceed. The MIT vehicle made a right turn but then became stopped temporarily against a curb. Upon determining that the UCF vehicle was stalled, Little Ben began his planned right turn (upper right). Immediately, this brought Little Ben face to face with the Cornell vehicle, which had stopped temporarily in the wrong lane while attempting to pass other traffic (lower right). After several seconds, Little Ben executed a pass to maneuver around the Cornell vehicle, and then returned to his own lane (lower left). Whereas the correctness of this behavior may seem obvious, what is significant is that of the four robots that appeared at this intersection, only Little Ben appeared to proceed as a human operator would have done.

Little Ben finished the 57-mile (92 km) course in approximately 305 min, not including any penalty time that may have been assessed by DARPA. Remarkably, he was the only Track B entry that was able to complete the challenge.

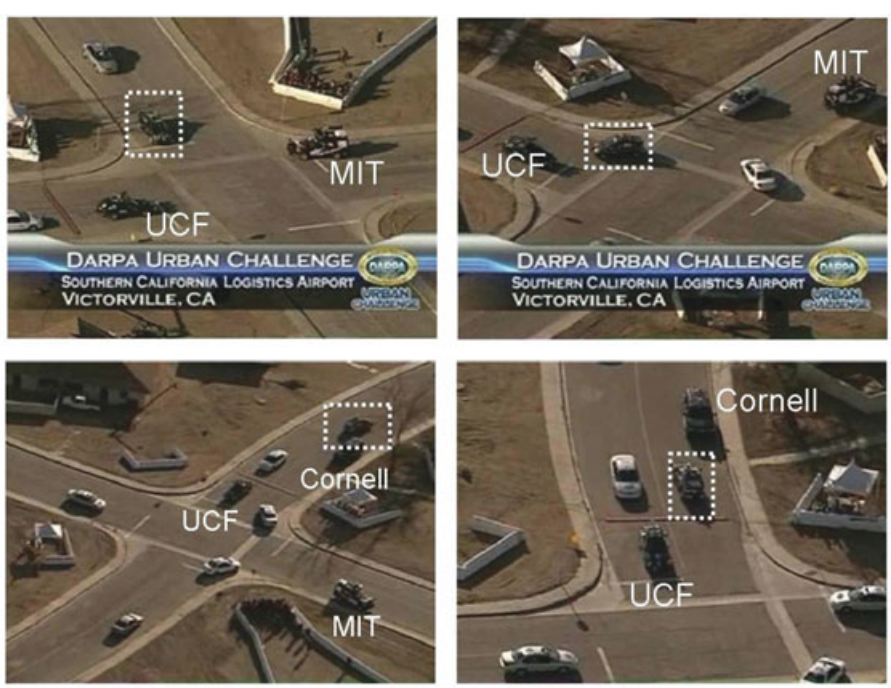

Figure 22. (Clockwise from upper left) Little Ben performs a right turn while passing a robot vehicle stopped in the wrong lane.

9. SUMMARY

This paper has presented some of the technical details of Little Ben, the autonomous ground vehicle built by the Ben Franklin Racing Team for the 2007 DARPA Urban Challenge. After quantifying the sensing and reaction time performance requirements needed for the upcoming challenge, the hardware and software systems for Ben were designed to meet these stringent criteria. An array of GPS/INS, LIDARs, and vision sensors was chosen to provide both omnidirectional and long-range sensing information. The software modules were written to robustly integrate information from the sensors, build an accurate map of the surrounding environment, and plan an optimal trajectory through the traffic network. This allowed the vehicle to successfully complete the 2007 DARPA Urban Challenge, even though we were severely constrained by time and budget constraints.

ACKNOWLEDGMENTS

The team would like to express our gratitude to the following individuals who contributed to the success of this project: Allen Biddinger, Brett Breslow, Gilad Buchman, Heeten Choxi, Kostas Daniilidis, the Footes, Rich Fritz, Daniel Garofalo, Chao Gao, Erika Gross, Drew Houston, Ani Hsieh, Steve Jamison, Michael Kozak, Vijay Kumar, Samantha Kupersmith, Bob Lightner, Gerry Mayer, Tom Miller, George Pappas, Ray Quinn, Ellen Solvibile, CJ Taylor, Chris Wojciechowsk, and many others. We would also like to thank Mitch Herbets and Thales Communications for sponsoring our entry. Finally, thanks to Global360 for allowing the use of video images in this report.

REFERENCES

Amir, Y., Danilov, C., Miskin-Amir, M., Schultz, J., & Stanton, J. (2004). The Spread toolkit: Architecture and performance (Tech. Rep. CNDS-2004-1), Baltimore, MD: Johns Hopkins University.

Coulter, R. (1992). Implementation of the pure pursuit path tracking algorithm (Tech. Rep. CMU-RI-TR-92-01). Pittsburgh, PA: Carnegie Mellon University.

DARPA (2007). Retrieved July 24, 2008, from http://www.darpa.mil/grandchallenge/rules.asp.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. Numerische Mathematik, 1, 269–271.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. IEEE Computer Magazine, 22, 46–57.

Gillespie, T. D. (1992). Fundamentals of vehicle dynamics. Warrendale, PA: Society of Automotive Engineers.

Guitton, A. (2000). The iteratively reweighted least squares method. Stanford University Lecture Notes. Stanford, CA: Stanford University.

Kalman, R. E., & Bucy, R. S. (1961). New results in linear filtering and prediction theory. Transactions of the ASME, 83, 95–107.

Rasmussen, C., Stewart, A., Burdick, J., & Murray, R. M. (2006). Alice: An information-rich autonomous vehicle for high-speed desert navigation. Journal of Field Robotics, 23(9), 777–810.

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., & Mahoney, P. (2006). Stanley: The robot that won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692.

Vernaza, P., & Lee, D. D. (2006). Robust GPS/INS-aided localization and mapping via GPS bias estimation. In Proceedings of the 10th International Symposium on Experimental Robotics, Rio de Janeiro, Brazil (pp. 101–110). Berlin: Springer.