2007 DARPA Urban Challenge
The Ben Franklin Racing Team
Team B156 Technical Paper*

University of Pennsylvania†
Lehigh University
Lockheed Martin Advanced Technology Laboratories

June 1, 2007

Executive summary

This paper describes the design, construction, and testing of an autonomous ground vehicle by the Ben Franklin Racing Team for the DARPA Urban Challenge. After analyzing the performance demands required of an autonomous vehicle traveling in an uncertain urban environment, we have designed our sensing, planning, navigation, and actuation systems to meet these demands. We have chosen an array of GPS/INS, LADAR's, and stereo cameras to provide timely information about the surrounding environment at the appropriate ranges. This sensor information is incorporated into a probabilistic dynamic map that can handle GPS dropouts and errors. Our planning algorithms consist of a high-level mission planner that uses information from the provided RNDF and MDF to select routes, while the lower level planner uses the latest dynamic map information to optimize a feasible trajectory to the next waypoint. The vehicle is actuated by a cost-based controller that efficiently handles steering, throttle, and braking maneuvers in both forward and reverse directions. Our software modules are integrated within a hierarchical architecture that allows rapid development and testing of the system performance. Quantitative evaluations show that our vehicle and algorithms are able to meet the demanding requirements needed to successfully complete the Urban Challenge.

*DISCLAIMER: The information contained in this paper does not represent the official policies, either express or implied, of the Defense Advanced Research Projects Agency (DARPA) or the Department of Defense. DARPA does not guarantee the accuracy or reliability of the information in this paper.

†Corresponding author: Daniel D. Lee, *ddlee@seas.upenn.edu*

1 Introduction and overview

The goal of the 2007 DARPA Urban Challenge is to build an autonomous ground vehicle which will execute a simulated military supply mission safely and effectively in a mock urban area. Compared with previous DARPA Grand Challenges, this particular challenge necessitates that robot vehicles perform autonomous maneuvers safely in traffic [1]. To address this challenge, the Ben Franklin Racing Team was formed by students and faculty at the University of Pennsylvania, Lehigh University, and engineers at Lockheed Martin Advanced Technology Laboratory. For the past year, the Ben Franklin Racing Team has been busy preparing "Little Ben," a drive-by-wire Toyota Prius with an array of onboard sensors and computers shown in Figure 1. The following sections details our team's approach in designing, constructing, and testing our hardware and software algorithms for the upcoming Urban Challenge.

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

1.1 Design considerations

The Urban Challenge presents unique challenges to autonomous sensing, navigation, and control. Some of the scenarios that our vehicle will need to be able to handle include 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 require that obstacles and lane markings are detected at a distance, and that the vehicle reacts quickly and appropriately while following the local traffic laws and conventions. An overarching requirement is that a successful system adhere to a stringent set of real-time processing constraints in its detection and reaction to its environment. This is mainly reflected 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 is predicated on achieving a reaction time that ensures 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 require that at least one vehicle length of separation is 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 anti-lock brake system (ABS). Therefore, the ABS reaction dynamics is reserved as an additional safety margin when dry pavement conditions are not present.

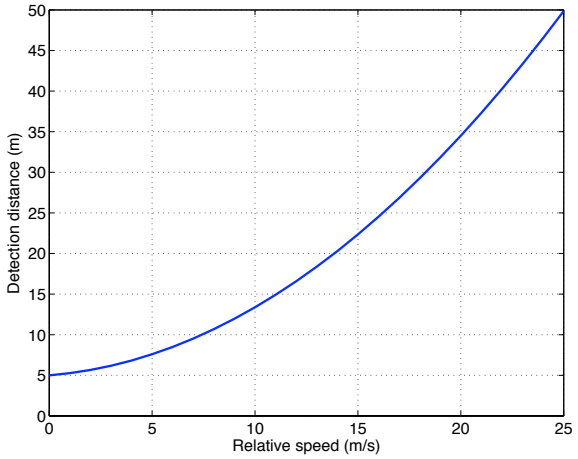

Figure 2: Required detection distance at various speeds taking into account worst-case latencies in system processing time.

We have evaluated a range of possible system sample rates, and have selected 10 Hz as the desired system processing rate. Our calculations in Figure 2 take 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 crosses the sensor detection threshold, and the second sample period is assumed to be used for the necessary computational processing.

The hardware and software systems have been selected to meet the desired detection distance and processing time objectives. Sensors and their respective mounting positions have been chosen to maximize their long range detection characteristics. Drive-by-wire actuation and computer hardware systems have been selected to minimize processing latencies. Similarly, our software modules are also optimized to maximize detection distance and minimize processing delays. This

combination of hardware sensing systems with efficient, reactive software modules will allow our vehicle to achieve the requisite safety margins for driving in urban traffic situations.

1.2 System architecture

After considering the various design requirements needed for autonomous traffic navigation and control in urban environments, the following system components were selected and installed.

1.2.1 Hardware systems

As shown in Figure 3, the drive-by-wire vehicle conversion was performed by Electronic Mobility Controls (EMC) of Baton Rouge, Louisiana. The steering system was modified with a high-speed servomotor attached to the underside of the steering column. A second motor actuates the throttle and brake pedal through a lever and cable pull. In addition, EMC provided a computer interface to control most other driving systems including transmission shift actuation, parking brake, headlights, windshield wipers, turn signals, and horn. We have also installed a CAN bus interface to the Toyota OBD-II connector to verify vehicle state information directly from vehicle's electronic control unit (ECU) computers.

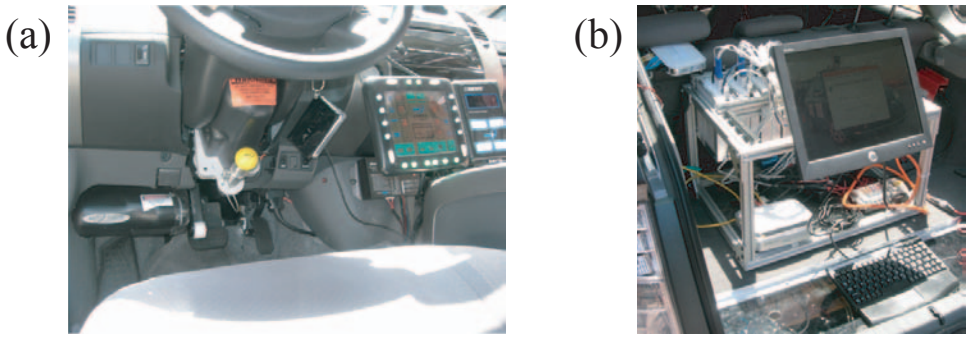

Figure 3: Hardware components located inside the vehicle: (a) EMC-provided drive-by-wire controls and (b) Mac Mini computing cluster in the trunk.

Computational processing power is provided by a cluster of Mac Mini Core Duo computers located in the trunk running Linux Ubuntu. The cluster consists of 4–8 Mac Mini's interconnected through a Gigabit ethernet switch. Serial connections to EMC's microprocessors as well as our other vehicle-interface microcontrollers are provided over the network via a Comtrol serial device server. This allows the system to automatically switch required processes over to a redundant computing node in the event of a computer failure.

Additionally, the computers are interfaced to an array of hardware LADAR and vision sensors mounted in a custom exterior roof rack. These sensors and their placement have been optimized for complete azimuthal and long range coverage and are described in more detail in a later section.

1.2.2 Software architecture

As depicted in Figure 4, the software architecture has been divided hierarchically into a series of modules, connected via interprocess communication messages. At the lowest level is the driving module which is 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 is the pose software module which integrates readings from the GPS and inertial navigation system to provide the latest pose information at 100 Hz. These two hardware interface modules can be readily replaced by a simulation module which allows us to rapidly test the software without requiring the processes to be physically connected to the vehicle systems.

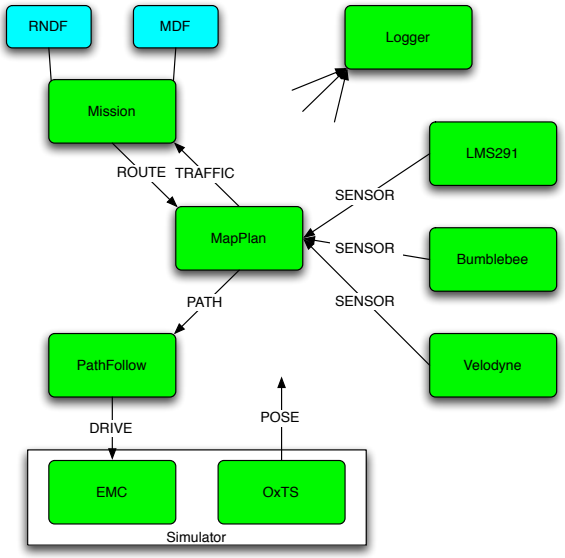

Figure 4: Software architecture showing system modules and corresponding interprocess communication messages.

At the highest level, the Mission planning module reads the appropriate RNDF and MDF files to determine the optimal sequencing of waypoints needed to complete the mission objectives. Next are the sensor modules which gather data from all the LADAR's and stereo cameras to provide probabilistic real-time estimates of the terrain, road markings, and static and dynamic obstacles. These modules consolidate 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 is then responsible for integrating all the sensor information into a probabilistic dynamic map, and computing the appropriate vehicle path to reach the next desired waypoint as determined by the high-level Mission planner. It also checks to ensure that this path avoids all known obstacles, while obeying vehicle dynamic constraints as well as local traffic rules. The PathFollow module takes the desired vehicle path from the MapPlan process and generates the optimal steering, throttle, and braking commands needed by the low-level driving module.

All the processes communicate with each other via well-defined message formats sent through the Spread messaging toolkit [2]. This open-source messaging system provides message reliability

in the presence of machine failures and process crashes, while maintaining low latencies across the network. It also enables convenient logging of these messages with appropriate timestamps. These logs allow us to rapidly identify and debug bad processes, as well as replay logged messages for diagnostic purposes.

These modules are written using a combination of C++ and Matlab. We have implemented a development environment that incorporates Subversion for source code tracking, Bugzilla for assigning tasks, and a Wiki for writing documentation. All the documentation is readably 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 are stored within the vehicle to allow us to make offline changes that are merged with our central servers after testing. With these tools, rapid prototyping and development can be accomplished by our team both in the laboratory and in the field.

2 Analysis and design

A variety of hardware systems and software modules have been implemented on our vehicle to handle the difficult sensing, navigation, and control environments of the Urban Challenge. In the following, we detail how these systems were designed, and show our analysis of their expected behavior.

2.1 Safety systems

Since safe operation is an all-inclusive requirement, we have taken major steps toward minimizing the risk of injury or damage due to undesired behavior of the vehicle. The emergency stop system has been designed to make human intervention safe, quick, and reliable. In order to achieve fail-safe operation, redundancy has been incorporated on multiple levels using watchdog timers and heartbeat monitors in Figure 5.

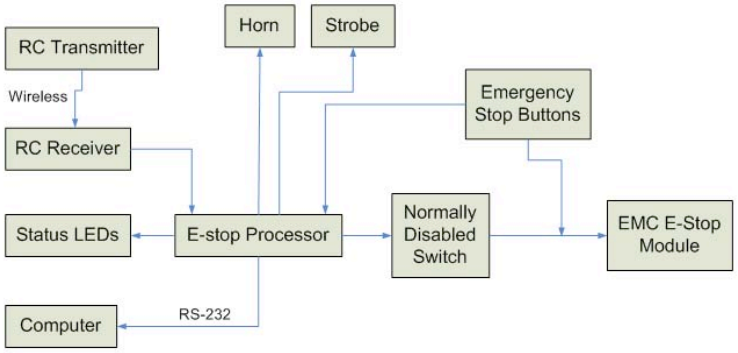

Figure 5: Block diagram depicting the emergency stop and safety systems incorporated into the vehicle.

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

Our “disable” mode is an extension of the “pause” mode. In addition to braking the vehicle and disabling computer control of the throttle, the E-stop processor verifies the vehicle speed is zero on the Toyota CAN bus, sets the transmission to *park*, and all the Toyota Prius systems are then powered down. In this state, the vehicle needs to be manually restarted in order to reactivate autonomous control. The vehicle can easily be disabled via the dedicated remote control or the manual E-stop buttons located on either side of the car.

2.2 Sensor rack

The sensors chosen to meet our sensing requirements include a variety of 2D and 3D LADAR’s as well as stereo cameras. Meeting the performance demands of the Urban Challenge requires sensors to detect obstacles from 5–50 meters around the vehicle (see Figure 2). In order to maximize the detection range of our sensors, the sensor rack shown in Figure 6 was custom designed to allow optimal viewing angles for as many of these sensors as possible. In particular, we designed the rack to accomodate the Velodyne HD LADAR to give the omnidirectional sensor a fully unoccluded 360 degree azimuthal view. By mounting the Velodyne 8” above the rest of the sensor rack, we take advantage of the full complement of elevation angles in the sensor to provide a sensing range from 4 to 60 meters around the vehicle.

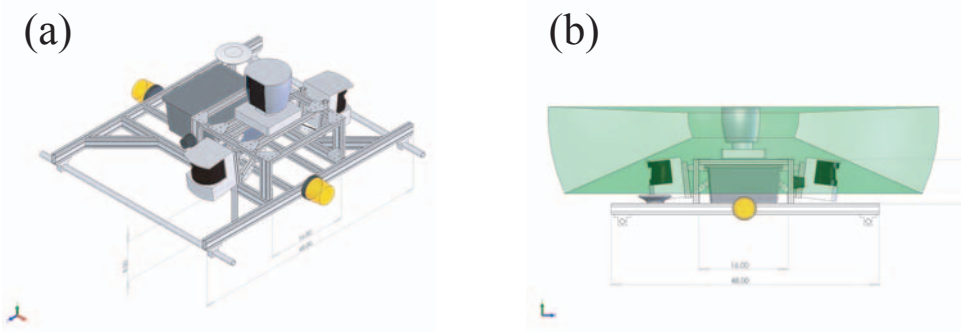

Figure 6: Rooftop sensor rack designed to provide optimal viewing angles.

The rack also incorporates a set of forward and rear facing SICK 2D LMS-291 LADAR sensors. These sensors are tilted downward in order to intersect the ground at approximately 30 meters ahead and behind the vehicle. In these positions, the SICK LADAR’s are just within their range limitations and can provide for both ground plane extraction as well as obstacle detection. These sensors are arranged so that they do not occlude the Velodyne’s field of view, and provide a complementary stream of range data.

The rack contains the warning siren, strobe lights, and mounting points for the GPS antennas. It also provides space for a weatherproof electronics enclosure. This contains and protects the power distribution block and sensor connectors which are routed from inside the vehicle through the sunroof. The construction materials for the rack were chosen so that all the sensor components are rigidly mounted, yet the frame is light enough so that it can be easily mounted and removed from the roof by two people.

2.3 Vision system

In addition to the LADAR systems, we have also incorporated a stereo vision system to detect lane markings and obstacles. This is necessary because when driving in urban canyons without GPS reception, lane markings may be the only exteroceptive signal indicating where the lane is located. In these situations, vision may be the only sensor that can accurately register the vehicle with the desired lane of travel.

Adoption of LADAR systems by nearly every competitor in the Urban Challenge raises the risk of inter-vehicle crosstalk. There is uncertainty about how LADAR's within close proximity in the Urban Challenge will affect their performance. In constrast, the passive nature of camera systems eliminates the potential for such data corruption. Unfortunately, vision system performance is highly dependent upon good ambient light levels. To maximize robustness, we incorporated both vision and LADAR based solutions for sensing to leverage the orthogonal failure modes of these sensor systems.

We have integrated a Point Grey Bumblebee 2 high-resolution, color stereo camera for lane segmentation and stop-line detection. This vision system can also serve as a secondary sensing modality for vehicle and obstacle detection as illustrated in Figure 7. Lane boundary segmentation is accomplished via adaptive thresholding on the image intensity values to account for variations in illumination across the image. The resulting feature set is further refined using stereo disparity estimates. By assuming that markings of interest are constrained to the ground plane, predicted disparity measurements at each pixel location in the near-field can be used to eliminate features inconsistent with the ground plane hypothesis [3].

(a)  (b) 

Figure 7: Vision system: (a) Bumblebee stereo vision head used to (b) track vehicles and lane markings.

2.4 LADAR reflectivity analysis

Both the Velodyne and the SICK LMS-291 LADAR sensors output simultaneous range and reflectivity measurements with each scan. The latter can be interpreted similarly to image data, and the 8-bit reflectivity resolution is equivalent to many grayscale cameras. Segmenting the lane markings from the reflectivity data can then be accomplished through traditional signal processing techniques. To demonstrate this, representative reflectivity measurements of darker and lighter asphalt road surfaces with a yellow center line marking are shown at Figure 8. This preliminary analysis shows that detecting the center line in either case is relatively straightforward, and can be used to help guide the vehicle for lane following, merging, and passing scenarios.

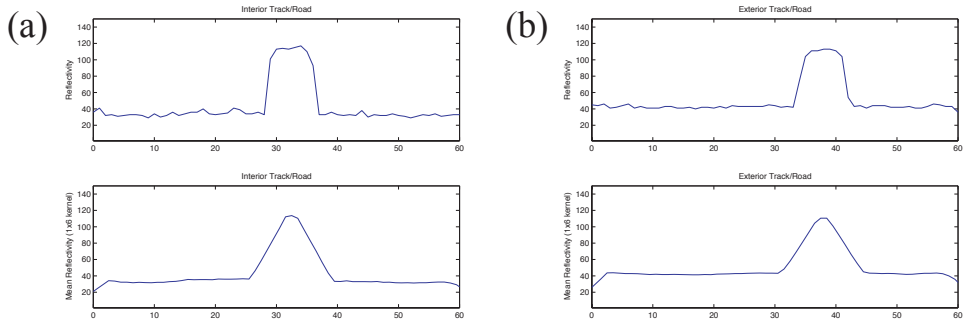

Figure 8: Representative reflectivity measurements from a SICK LMS291 on (a) dark colored and (b) lighter colored asphalt roads. The peak in both plots correspond to a yellow center line marking.

2.5 Pose estimation

Ben's primary pose estimation system is an Oxford Technical Solutions RT-3050 unit. The RT-3050 is a self-contained unit which uses a Kalman filter based algorithm to combine inertial sensors, Global Positioning System (GPS) updates with differential corrections from the OmniStar VBS service and vehicle odometry information from the native embedded vehicle systems [4]. The RT-3050 is able to provide pose estimates at a high update rate of 100 Hz with a stated accuracy of 0.5 meter. The unit is specifically designed for ground vehicle testing applications and is capable of providing estimates during periods of sustained GPS outages or poor GPS performance due to environmental effects such as multipath reflections.

The RT-3050 meets the performance requirements for operating in an urban environment; however, since pose estimation is so crucial for urban mapping and navigation, we have also constructed a redundant secondary particle-filter based pose estimation system [5]. Our secondary system consists of a Xsens MEMS IMU and a Navcom 2050 GPS unit with differential corrections provided by the Starfire service, neither of which are used by the primary RT-3050 unit. This level of redundancy is designed to provide increased fault tolerance in the unlikely event that the primary system should fail.

2.6 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. We address this issue by modeling uncertainty with factored Bayesian models and performing inference on these models using multiple complementary techniques [6].

For the Urban Challenge, prior information about the local environment is provided by the RNDF which gives the location of road features. We need to accurately reconcile this prior information with current perceptual data as measured by our LADAR and vision sensors, to accurately handle situations such as in merging and parking where tight navigation is a necessity. This information can be systematically integrated and registered by performing inference on factored Bayesian models as follows.

As the vehicle traverses its environment, perceptual data are distilled into local occupancy grid maps [7]. These maps are referenced to the local coordinate system of the vehicle and reflect the state of the world as observed at a specific instant in time. Given corresponding pose estimates, the local maps are fused over time to create a global map. If the pose estimation and mapping process are performed independently, issues with consistency often arise, as illustrated in the left two subfigures of Figure 9. This is often due to the nature of GPS error, which tends to exhibit more low-frequency drift as opposed to high-frequency noise. This drift causes obstacles to smear over long time scales, which becomes an issue when revisiting areas that have previously been mapped, or when the vehicle lingers in an area for a time on the order of the drift timescale.

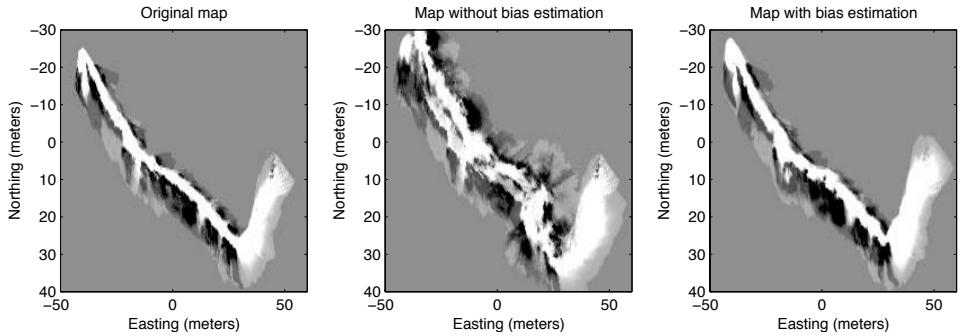

Figure 9: Comparison of original map with those generated on a second traversal of the same course, with and without bias estimation and correction.

We address this issue by modeling the pose output used for mapping as being corrupted by a hidden additive bias. Given the corrupted pose estimate and observations of maps over time, we can deduce the bias over time by assuming that multiple observations of the same area should appear consistent. Probabilistically, we model this as a discrete-state hidden Markov model in which we attempt to estimate the bias from map observations.

Given that we want to consider a discrete set of possible biases at the current time (b_t), we can apply Bayes' rule to obtain a recursive filter that updates our prior belief about the bias ($p(b_t|z^{t-1})$) given a measurement model $p(z_t|b_t)$ and a bias dynamics model $p(b_t|b_{t-1})$, where z_t denotes the measurement at time t, and z^t denotes the sequence of measurements obtained up to time t.

$$p(b_t|z^t) = \alpha p(z_t|b_t) \sum_{b_{t-1}} p(b_t|b_{t-1}) p(b_{t-1}|z^{t-1}) \tag{1}$$

Our measurement model is provided by finding the cross-correlation of the local map with the global map at certain shifts. This procedure results in a correlation score that we interpret as the likelihood that the map was observed under each possible value of the GPS bias. This is illustrated in Figure 10.

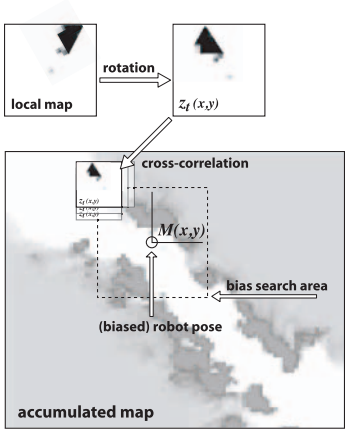

Figure 10: Illustration of map measurement model evaluation for GPS bias estimator.

Given this form of the measurement model, along with a simple diffusion dynamics model for the GPS bias, all the major operations of the estimator can be implemented using convolutions, which can be implemented very efficiently using our computer processors. Figure 9 demonstrates the effect of estimating the bias with mapping. Without bias estimation, the map is notably smeared after a second traversal of the same area. With bias estimation, no corruption is evident in the map after the second traversal of the course.

Simulation results are shown in Figure 11. A known artificial bias was added to the true pose for mapping, and this was then filtered via our method to estimate the bias. The results indicate that the filter was able to track the true bias accurately throughout the length of the trial.

This technique can be employed to ensure consistency between the lane information in the RNDF and what is perceived via the sensors. Without employing any sort of compensation, the location of the lanes in the RNDF are likely to be significantly different from the location of the lanes perceived by our vision and LADAR's. Here the RNDF serves as a prior on the lane location. We can merge this prior with an actual observation of the lane position arising from the vision system in conjunction with the output of the pose system described above. By again applying a discrete Markov filter, we can estimate the true global position of the lane, and will ensure that we can construct an consistent, dynamic map. This will allow our vehicle to handle environments where it needs to accurately sense driving lanes in the presence of traffic, such as with intersections and merging scenarios.

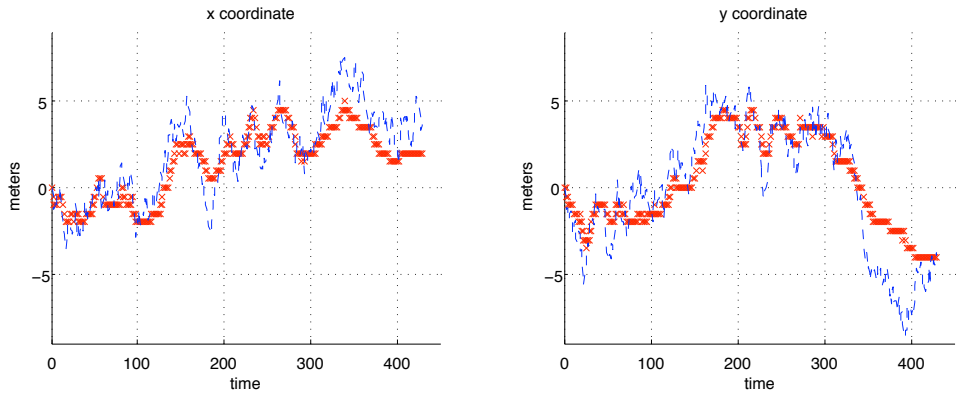

Figure 11: Plots illustrating performance of secondary bias estimator in simulation. Dashed line indicates magnitude of artificially injected bias and the crossed line indicates estimated bias. Left and right figures show x and y components of the bias, respectively.

2.7 Planning

In our hierarchical software architecture, planning is performed in two stages. At the highest level, the mission planner estimates travel times between waypoints and then computes the optimal sequence of waypoints to traverse in order to minimize the overall mission time. When a particular lane or intersection is blocked, the mission planner recomputes an alternative sequence of waypoints using Dijkstra's algorithm to adaptively respond to traffic conditions [8]. Figure 12 illustrates the display from the mission planner as it monitors the progress of the vehicle through the route network.

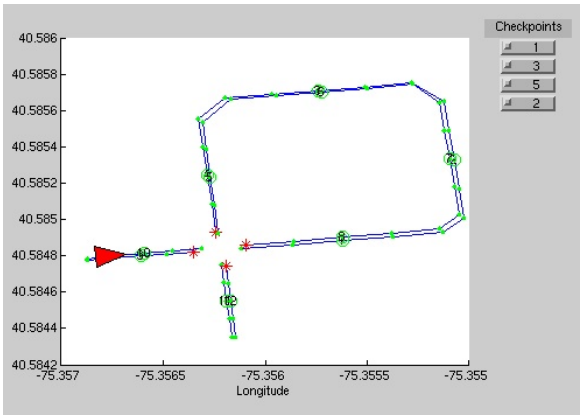

Figure 12: Mission planner uses information from the RNDF and MDF to plan optimal routes through the traffic network.

The next stage of planning incorporates information from the dynamic map by computing a detailed path to the next waypoint. This process is split across several specialized local planners that individually handle specific Urban Challenge scenarios such as lane following, lane changing,

U-turns, and intersections.

As an example of one of our local planners, our lane following planner models the desired vehicle path as a series of parametric cubic splines:

$$x(t) = a_0 + a_1 t + \frac{1}{2} a_2 t^2 + \frac{1}{6} a_3 t^3 \quad (2)$$

$$y(t) = b_0 + b_1 t + \frac{1}{2} b_2 t^2 + \frac{1}{6} b_3 t^3. \quad (3)$$

The parameters of this spline path are then optimized by minimizing the expected map cost $C(x, y)$ of the dynamic path, subject to the constraints that the path start and end at the corresponding waypoints with the proper heading directions.

$$\min_{(a_i, b_i)} \int_{t_0}^{t_1} C(x(t), y(t)) dt. \quad (4)$$

This computed path is then checked for possible collisions with static and dynamic obstacles before it is sent to be executed by the path following module.

For the more difficult scenario where the vehicle may need to backup such as in a U-turn or parking situation, we replace the parameterized spline path with a set of Dubins paths in the forward and reverse directions [9]. Each of these paths are then scored using the path cost functional in Equation 4, and the optimal path is then chosen and executed.

2.8 Path following

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

Previous approaches to steering control for autonomous car-like vehicles have used PID control based methods with error terms that combine both the lateral and heading offsets from the desired trajectory [10, 11, 12]. A weakness of these controllers in this application is that they do not explicitly consider 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.

In order to avoid these shortcomings, we have developed an alternative approach for steering control which integrates the dynamics of a vehicle model (Figure 13) to predict the resulting change in pose after a short period of time after a set of possible steering commands as shown in Figure 14 [13]. A cost function is then evaluated for each of the predicted poses and the steering command which minimizes this cost function is chosen.

As illustrated in Figure 15, the particular form of the cost function used in our controller is as follows:

$$C(\phi_i) = E_{lateral}^2 + \left[R_\theta \sin(\frac{E_{\theta_i}}{2}) \right]^2 \quad (5)$$

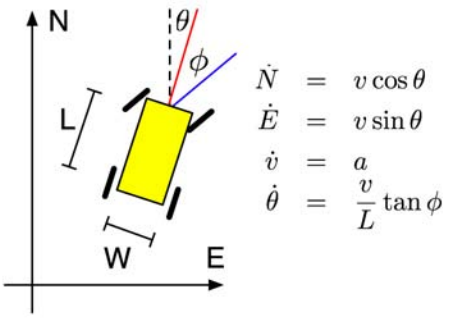

Figure 13: “Bicycle” model of the car dynamics used for control.

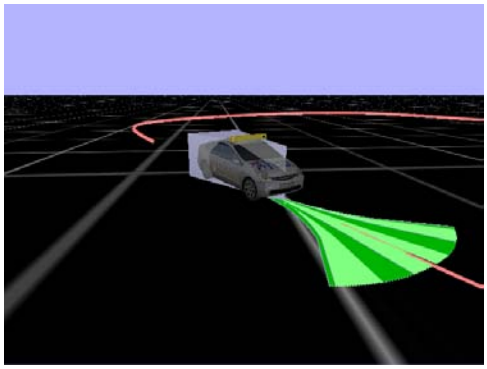

Figure 14: Estimated future vehicle poses for a set of possible steering commands in a simulated environment.

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 which is used to scale the heading error relative to the position error, which we can adaptively tune to maximize performance in the different Urban Challenge scenarios.

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

3 Results and performance

In order to test the overall performance of Ben’s hardware and software systems, we have performed an initial battery of performance measurements. These tests demonstrate some of the fundamental capabilities of the vehicle that are necessary to perform the maneuvers required in the Urban Challenge.

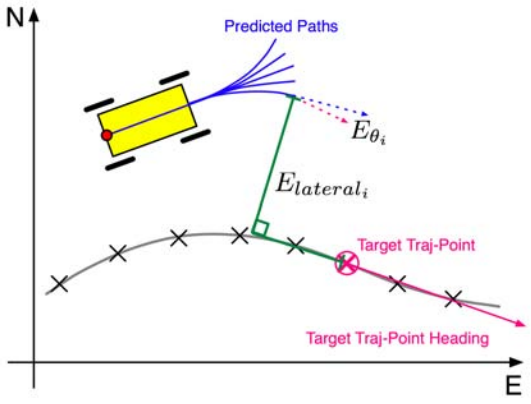

Figure 15: Graphical representation of the terms included in the controller cost function.

3.1 Stopping performance

We measured the vehicle stopping distance after "pause" mode had been activated as a function of initial velocity. Figure 16 shows the results for a range of speed, demonstrating that at 20 mph (9 m/s), the stopping distance is about 8 meters, much less than the stated requirement of 20 meters. The standard deviation of all the measurements over repeated trials was less than 1 meter in all the tests.

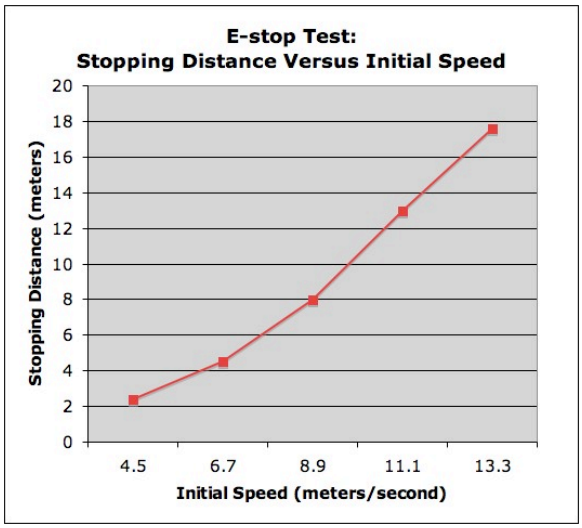

Figure 16: Stopping distance as a function of initial vehicle speed.

3.2 Trajectory tracking

To measure the accuracy of our control algorithms for urban driving, a series of tests over two laps of our test course at Lehigh University was performed and shown in Figure 17. Our test

course is approximately 900 meters long and was completed with an average speed of 10 mph, with a maximum speed of 18 mph reached on the straight sections. The tracking performance of the controller results in a mean lateral offset of -0.1 m, with a standard deviation of 0.3 m and a mean heading error of 0.2 degrees with a standard deviation of 2.6 degrees. It can be seen that the performance of the controller is highly repeatable, evident from the smooth and repetitive plot of the steering angle.

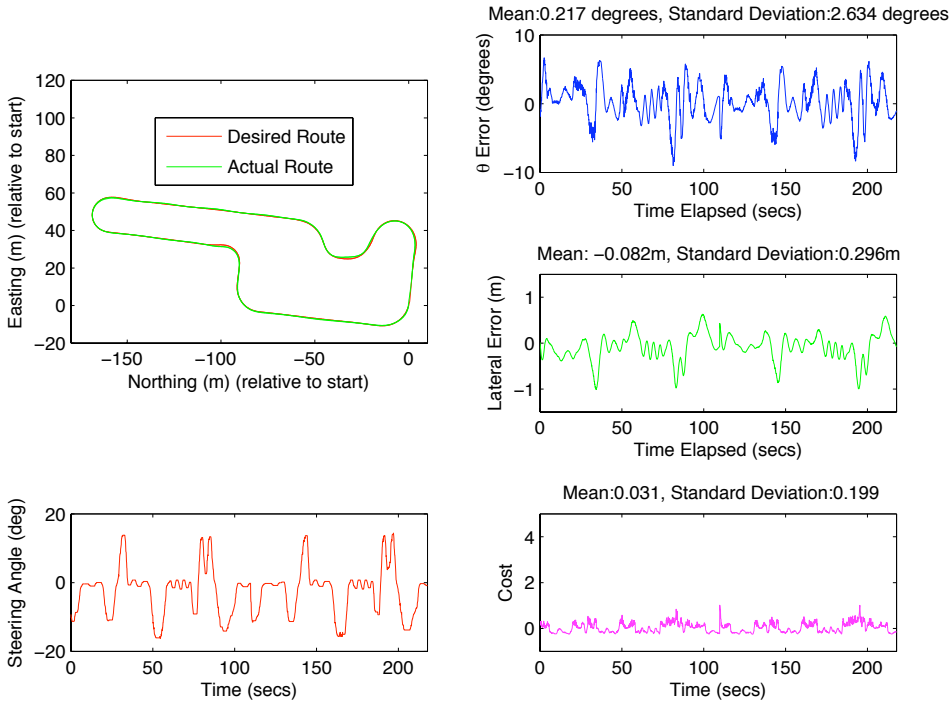

Figure 17: Results for the steering control component of the path following module during two laps on a test course.

3.3 Vision performance

To assess the accuracy and reliability of our vision system for detecting lane features in urban environments, 10 trials were conducted to assess the expected detection range of intersection stop lines. In each trial, our vehicle was driven at a velocity of 15 mph towards the intersection of our site visit course while the vision system attempted to detect the appropriate stop line. The trials were conducted against each of the four stop lines, and under a range of illumination conditions including significant shadows in the travel lanes. Data from the on-board pose system were also recorded as ground truth for determining the relative vehicle position. A plot of the detection (true positive) versus range is shown in Figure 18. These results indicate that we can expect reliable stop line segmentation at ranges of 20 meters or more. It should be noted that system parameters (*e.g.*, segmentation and voting thresholds) were chosen conservatively during these trials to avoid false positives; none were observed during our testing.

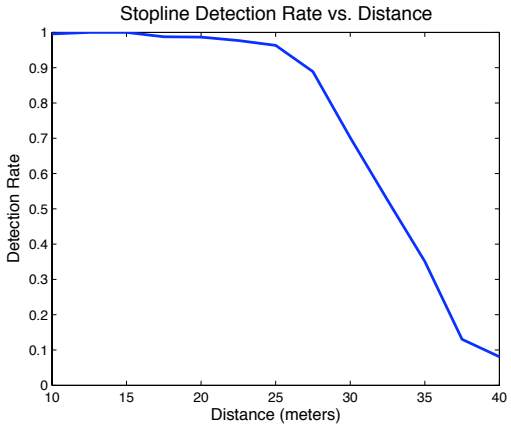

Figure 18: Stop line detection rate versus intersection distance. Results indicate reliable detection can be accomplished in excess of 20 meteres.

To characterize the performance of the stereo vision system for moving vehicle detection, we conducted a series of 15 trials where a test vehicle was driven at a velocity of 15 mph towards a stopped vehicle in the opposite lane. The test vehicle then attempted to detect the opposing vehicle solely from stereo disparity measurements. Again, trials were conducted under varying illumination conditions and an logged pose was used to estimate the detection range. A histogram of the trial results is shown in Figure 19. The "initial" detection distance represents when the opposing vehicle was first detected, while the "final" distance denotes when the vehicle was continuously tracked. The mean detection range for these was 15.4 and 14.6 meters, respectively. These distances were limited by the narrow baseline of the camera system (12 cm). However, even the minimum final detection range (8.0 meters) would be sufficient to meet the Urban Challenge stopping criterion at 15 mph.

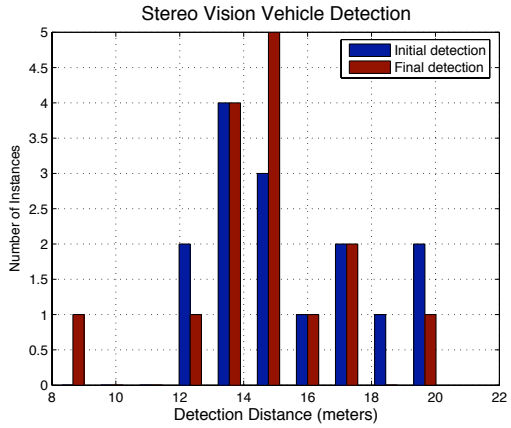

Figure 19: Stereo vision for reliably detecting vehicles was demonstrated at mean distances of 15 meters.

3.4 LADAR visibility

To assess the performance of using LADAR reflectivity measurements in lane marking segmentation, a SICK LMS-291 was mounted parallel to the ground plane on our test vehicle roof rack at a height of approximately 1.5 meters above ground level. The vehicle was then driven by a human operator at a velocity of 15 mph over our site visit course. Separate trials were conducted to collect data with the road center line in and out of the LADAR's field of view to detect both false negatives and false positives. The SICK scan rate was set at 70 Hz.

Of 11,631 scans with the road center line visible, there were 11,630 positive segmentations (1 false negative). Of the 12,253 scans of asphalt with the center line not visible, no center lines were detected (0 false positives). The success rate can be attributed to the significantly different reflectivity characteristics of the asphalt and lane markings. This is clearly illustrated in the scan histogram of the filtered reflectivity measurements in Figure 20.

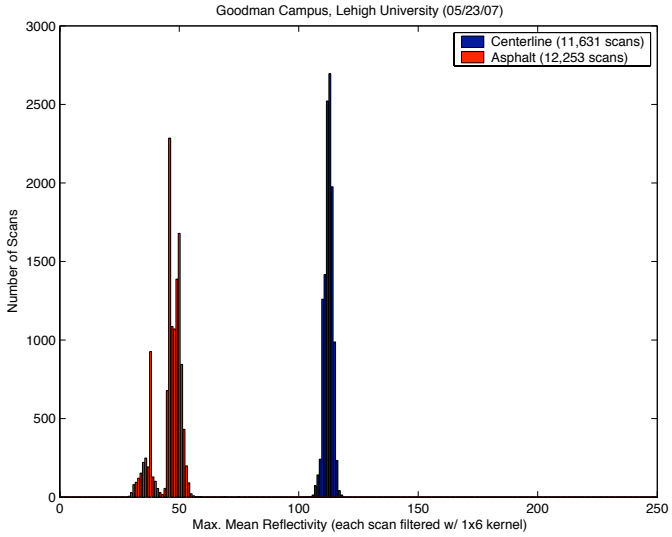

Figure 20: Histogram of reflectivity measurements. After mean filtering, the distributions for asphalt and the road center line can be readily segmented with a threshold of 85.

We expect the better range performance, greater angular resolution (0.09° vs. 0.50°), and multiple scan lines (64 vs. 1) of the Velodyne system will offer similar performance over longer distances. Characterizing the lane segmentation performance of the Velodyne system over a representative cross-section of roads is currently being performed.

3.5 Intersection test

We measured the range and accuracy of our detection systems to meet the requirements to traverse an urban intersection. Our tests consisted of two independent evaluations at 4-way stop intersections. The first was in a controlled environment where our vehicle pulled up to the stop line, saw a car present, waited for the car to clear, and then proceeded through the intersection. This evalua-

tion performed well through several trials, but was limited to testing a single additional car driven by a team member.

In our second evaluation, we took measurements with unpredictable drivers. This was performed offline by collecting log data at a busy public intersection. Our vehicle was positioned in an adjacent parking space and observed traffic passing through the intersection. In total, we collected data in 15 instances with a single vehicle, 9 instances where two vehicles pulled up at different entrances, and 2 instances with three vehicles. The vehicles included an assortment of cars, trucks, and a bus. In simulation, we then tested to see whether our planning algorithm could meet the timing requirements of the Urban Challenge for proceeding through an intersection. An example of our assessment is shown in Figure 21, where the decision to proceed was made within one second of the intersection being cleared.

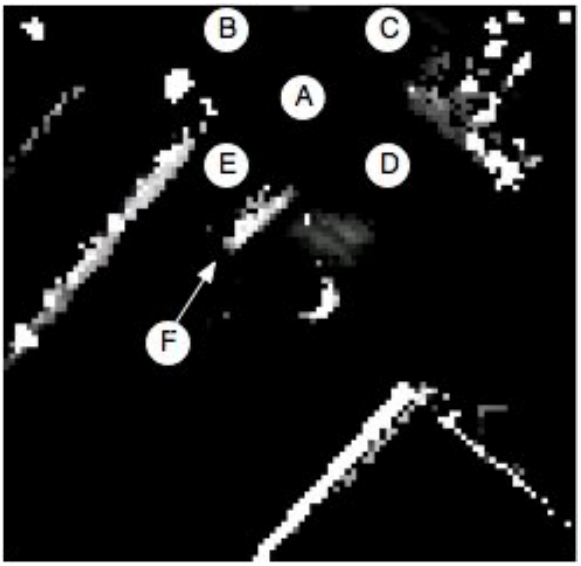

Figure 21: Occupancy grid of an intersection at 0.5 by 0.5 meter resolution. Black cells represent the ground plane while lighter cells show obstacles: Label A is the center of the intersection while labels B,C,D,E are the lanes approaching the intersection. Label F represents the detection of a passing car.

In one particular instance out of the 26 trials, our algorithm failed to correctly identify the proper time to proceed. We are currently working to improve the algorithm to more robustly handle these types of scenarios for the Urban Challenge.

4 Conclusions

This paper has presented an overview of the current state of 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, LADAR's and vision sensors were 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. In particular, we have designed and implemented the following systems in response to some of the unique performance requirements of the Urban Challenge:

- To meet the safety requirements, we have designed and implemented a fail-safe system with manual and remote emergency stop capabilities.
- To accurately traverse lanes, we have implemented algorithms to sense lane markings, and to properly integrate them into our maps.
- To sense moving obstacles, we have selected an array of long-range LADAR and vision sensors and implemented fast, reactive processing routines to robustly detect the obstacles.
- To stop and drive in traffic consistent with local rules and conventions, we have developed planning modules that can accomodate the various Urban Challenge scenarios.
- To address the challenge of parking, we have developed a controller that can accurately navigate in forward and reverse modes.

Work is still ongoing to develop the complete set of systems needed to successfully complete the Urban Challenge. However, our initial testing indicates that our current hardware and software algorithms should be capable of handling the navigation and traffic scenarios that will be encountered in the Urban Challenge. We look forward to the opportunity to demonstrate our autonomous vehicle during our site visit and at future tests.

References

[1] DARPA. (2007). Darpa Grand Challenge rules. Defense Advanced Research Projects Agency. Retrieved from http://www.darpa.mil/grandchallenge/rules.asp

[2] Y. Amir, et al. (2004). The spread toolkit: architecture and performance. (Tech. Rep. CNDS-2004-1). Baltimore, MD: Johns Hopkins University.

[3] S. Se and M. Brady. (2002). Ground plane estimation, error analysis, and applications. Robotics and Autonomous Systems, 39, 59–71.

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

[5] P. Vernaza and D. D. Lee. (2006). Rao-Blackwellized particle filtering for 6-DOF estimation of attitude and position via GPS and inertial sensors. In Proceedings of the IEEE International Conference on Robotics and Automation.

[6] P. Vernaza and D. D. Lee. (2006). Robust GPS/INS-aided localization and mapping via GPS bias estimation. In Proceedings of the 10th International Symposium on Experimental Robotics.

[7] A. Elfes. (1989). Using occupancy grids for mobile robot perception and navigation. IEEE Computer Magazine, 22(6), 46–57.

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

[9] L. E. Dubins. (1957). On curves of minimum length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of Mathematics, 79, 497–516.

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

[11] S. Thrun, et al. (2006). Stanley: The Robot that Won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 661–692.

[12] R. Murray, et al. (2006). Alice: An Information-Rich Autonomous Vehicle for High-Speed Desert Navigation. Journal of Field Robotics, 23(9), 777–810.

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