Odin: Team VictorTango's Entry in the DARPA Urban Challenge

Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, and Chris Terwelp
TORC Technologies, LLC
Blacksburg, Virginia 24060
e-mail: bacha@torctech.com

Charles Reinholtz
Department of Mechanical Engineering
Embry-Riddle Aeronautical University
Daytona Beach, Florida 32114
e-mail: charles.reinholtz@erau.edu

Dennis Hong and Al Wicks
Department of Mechanical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24060

Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster
Unmanned Systems Group
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24060

Received 29 January 2008; accepted 14 May 2008

The DARPA Urban Challenge required robotic vehicles to travel more than 90 km through an urban environment without human intervention and included situations such as stop intersections, traffic merges, parking, and roadblocks. Team VictorTango separated the problem into three parts: base vehicle, perception, and planning. A Ford Escape outfitted with a custom drive-by-wire system and computers formed the basis for Odin. Perception used laser scanners, global positioning system, and a priori knowledge to identify obstacles, cars, and roads. Planning relied on a hybrid deliberative/reactive architecture to

Journal of Field Robotics 25(8), 467–492 (2008)
Published online in Wiley InterScience (www.interscience.wiley.com). • DOI: 10.1002/rob.20248

analyze the situation, select the appropriate behavior, and plan a safe path. All vehicle modules communicated using the JAUS (Joint Architecture for Unmanned Systems) standard. The performance of these components in the Urban Challenge is discussed and successes noted. The result of VictorTango's work was successful completion of the Urban Challenge and a third-place finish.

1. INTRODUCTION

On November 3, 2007, DARPA hosted the Urban Challenge, an autonomous ground vehicle competition in an urban environment. To meet this challenge, Virginia Tech and TORC Technologies formed team VictorTango, a collaborative effort between academia and industry. The team includes 46 undergraduate students, 8 graduate students, 4 faculty members, and 5 full-time TORC employees and industry partners, including Ford Motor Co. and Caterpillar, Inc. Together team VictorTango and its partners developed Odin, a 2005 Ford Hybrid Escape modified for autonomous operation.

In the weeks prior to competition, 35 teams prepared for the National Qualifying Event (NQE). Vehicles had to navigate various courses, merge with traffic, navigate cluttered roads and zones, park in full parking lots, and detect roadblocks. After a rigorous qualifying event, only 11 teams were deemed ready by DARPA to line up in the start chutes of the final Urban Challenge Event (UCE). The vehicles had to navigate situations similar to those they encountered during the NQE. However, each vehicle also had to share the road with the other 10 autonomous vehicles, 10 chase vehicles, and 50 human-driven traffic vehicles. Six of the 11 vehicles finished the race. This paper provides a summary of the approach, final configurations, successes, and incidents of the third-place team, VictorTango.

1.1. VictorTango Overview

Team VictorTango divided the problem posed by the Urban Challenge into three major parts: base vehicle platform, perception, and planning. Each of these sections was then subdivided into distinct components for parallel development. Team members were able to split up the required tasks, execute and debug them individually, and provide finished components for full system testing. This modular approach provided the rapid development time needed to complete a project of such magnitude in only 14 months. This section provides a description of the components that constitute the team's approach.

1.2. Base Vehicle Platform

Team VictorTango's entry in the Urban Challenge is a modified 2005 Hybrid Ford Escape named Odin, shown in Figure 1. This base vehicle platform meets the DARPA requirement of a midsize commercial automobile with a proven safety record. The use of the hybrid-electric Ford Escape provides numerous advantages in the areas of on-board power generation, reliability, safety, and autonomous operation. As required by DARPA, the drive-by-wire conversion does not bypass any of the OEM safety systems. Because the stock steering, shifting, and throttle systems on the Hybrid Escape are already drive-by-wire, these systems can be controlled electronically by emulating the command signals, eliminating the complexity and failure potential associated with the addition of external actuators. The stock hybrid power system is able to provide sufficient power for sensors and computers without the need for a separate generator.

Odin's main computing is supplied by a pair of Hewlett–Packard servers, each of which is equipped with two quad-core processors. One of the servers runs Microsoft Windows XP and is dedicated to sensor processing. Windows was selected because some of the sensor processing software uses National Instruments' LabVIEW Vision development module, requiring Windows. The other server runs Linux

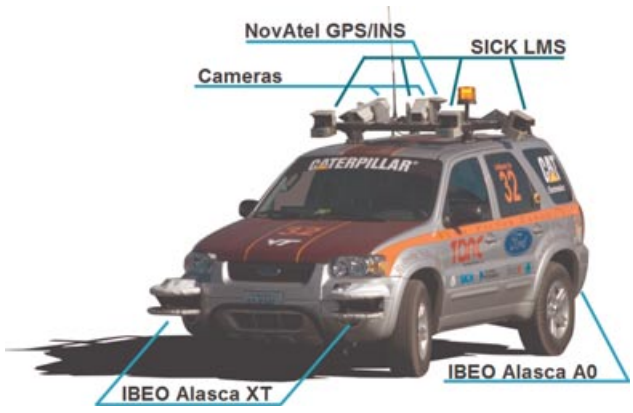

Figure 1. External view of Odin with sensors labeled.

and is further subdivided into four virtual machines for process load balancing and isolation. The Linux system, selected for its configurability and stability, runs all of the decision-making and planning modules. The vehicle hardware is controlled by a National Instruments CompactRIO unit, which contains a real-time capable operating system and a field programmable gate array (FPGA). The primary communications backbone is provided by a gigabit Ethernet network.

1.3. Perception

To fulfill the behavioral requirements of the Urban Challenge, Odin must first be able to adequately localize its position and perceive the surrounding environment. Because there may be sparse way points in a route network definition file (RNDF) and areas of poor global positioning system (GPS) coverage, the surrounding road coverage and legal lanes of travel must also be sensed. Finally, Odin must be able to perceive all obstacles in its path and appropriately classify obstacles as vehicles.

For each perception requirement, multiple sensors are desirable to achieve the highest levels of fidelity and reliability. To allow for maximum flexibility in sensor fusion, the planning software does not use any raw sensor data; rather it uses a set of sensor-independent perception messages. The perception components and the resulting messages are shown in Figure 2. The localization component determines the vehicle position and orientation in the world. The road detection component determines a road coverage map as well as the position of each lane in nearby segments. The object classification component detects obstacles and classifies them as either static or dynamic. A dynamic obstacle is any obstacle that is capable of movement, so a stopped vehicle would be classified as a dynamic obstacle with zero forward velocity.

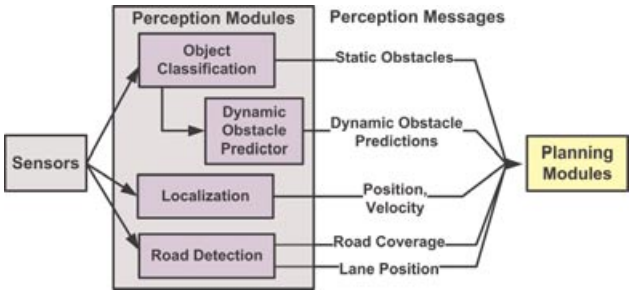

Figure 2. Perception structure overview.

1.4. Planning

The planning software on Odin uses a hybrid deliberative-reactive model dividing upper-level decisions and lower-level reactions into separate components. These components run concurrently at independent rates, allowing the vehicle to react to emergency situations without needing to replan an entire route. Splitting the decision making into separate components also allows each system to be tested independently and fosters parallel development, which is especially attractive given the short development timeline of the DARPA Urban Challenge.

The route planner component is the coarsest level of planning and is responsible for determining which road segments and zones the vehicle should use to travel to all checkpoints. The driving behaviors component is responsible for obeying the rules of the road and guiding the vehicle along the planned route. The lowest level of the planning process is the motion planning component, which determines the path and speed of Odin. Motion commands are then passed to the vehicle interface to be translated into actuator control signals. An overview of the planning process is shown in Figure 3.

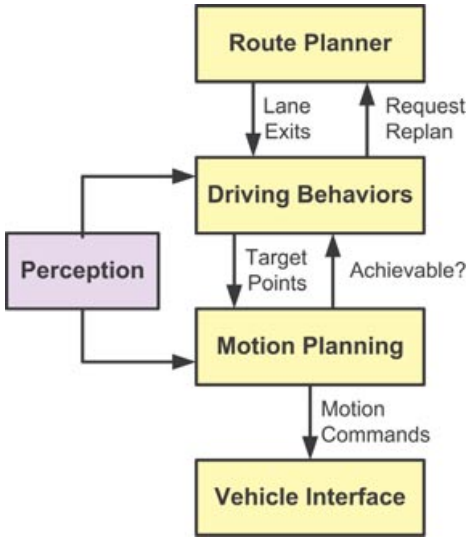

Figure 3. Planning structure overview.

2. TECHNICAL APPROACH

This section presents an overview of the major design choices made in the development of Odin, focusing on perception and planning systems. In each of these sections, an overview of the system function as well as of the design of key elements is given.

2.1. System Architecture and Communications

Whereas previous Grand Challenges could be solved using a purely reactive software architecture, the complex nature of the Urban Challenge necessitates a hybrid solution. In addition to the simpler goal-seeking behavior required in the previous challenges, Urban Challenge vehicles must maintain knowledge of intent, precedence, and timing. With many concurrent perception and planning tasks of differing complexity, priority, and computation time, parallelism is preferred to a single monolithic sense–plan–act structure (Murphy, 2000). In addition, the complexity of the Urban Challenge problem necessitates a well-defined software architecture that is modular, clearly segmented, robust, safe, and simple.

VictorTango's software structure employs a novel hybrid deliberative-reactive paradigm. Odin's perception, planning, and acting occur at several levels and in parallel tasks, acting on the most recent information received from other modules. With traditional hybrid architectures, deliberative components are usually kept at a high level, whereas the more reactive, behavior-based, components are used at a low level for direct actuator control (Konolige & Myers, 1998; Rosenblatt, 1995). With the rapid growth of computing technology, however, there has been a reemergence of deliberative methods for low-level motion planning (Thrun et al., 2006; Urmson et al., 2006). Search-based approaches provide the important traits of predictability and optimality, which are useful from an engineering point of view (Russel & Norvig, 2003). VictorTango's system architecture therefore exhibits a *deliberative–reactive–deliberative* progression. As a result, the scope of a behavioral control component can be moved from low-level reflexes to higher-level decision making for solving complex, temporal problems. An overview of the hybrid mixture of deliberative planning, reactive navigation, and concurrent sensor processing is shown in Figure 4. Each of the modules is further detailed in the following sections.

SAE AS-4 JAUS (Joint Architecture for Unmanned Systems) was implemented for communications, enabling automated dynamic configuration and enhancing the future reusability and commercialization potential of Urban Challenge software. Each software module is implemented as a JAUS component with all interactions to and from other modules occurring via JAUS messages. As such, each software module operates as a stand-alone component that can be run on any one of the computing nodes. Because dynamic configuration and data subscription are handled via JAUS, the system is highly reconfigurable, modular, expandable, and reusable beyond the Urban Challenge. An additional benefit of employing a communications standard toolkit was the easy integration of logging and simulation (both discussed further in Section 6).

2.2. Perception

Perception is defined to include all aspects of the design necessary to sense the environment and the vehicle's position in it. Each perception module transforms raw data collected from multiple sensors into information useful to the decision-making software.

2.2.1. Sensor Layout

The sensor coverage for Odin is shown in Figure 5. The largest portion of Odin's detection coverage is provided by a coordinated pair of IBEO Alasca XT Fusion laser range finders. This system comprises two four-plane, multireturn range finders and a single external control unit (ECU) that covers a 260-deg field of view, as shown in Figure 5. The system has an advertized range of almost 200 m, although the effective range to reliably identify most objects has been shown in testing to be closer to 70 m. A single IBEO Alasca A0 unit with a field of view of 150 deg is used to detect approaching vehicles behind Odin and navigate in reverse. The Alasca A0 is an earlier-generation Alasca sensor than the XT, and testing has shown a lower range of approximately 50 m for reliable object classification.

For short-range road and obstacle detection, two additional SICK LMS 291 laser range finders are angled downward on the front corners of the roof rack. These sensors are able to detect negative obstacles and smaller obstacles that may be underneath the IBEO XT vertical field of view. Two side-mounted

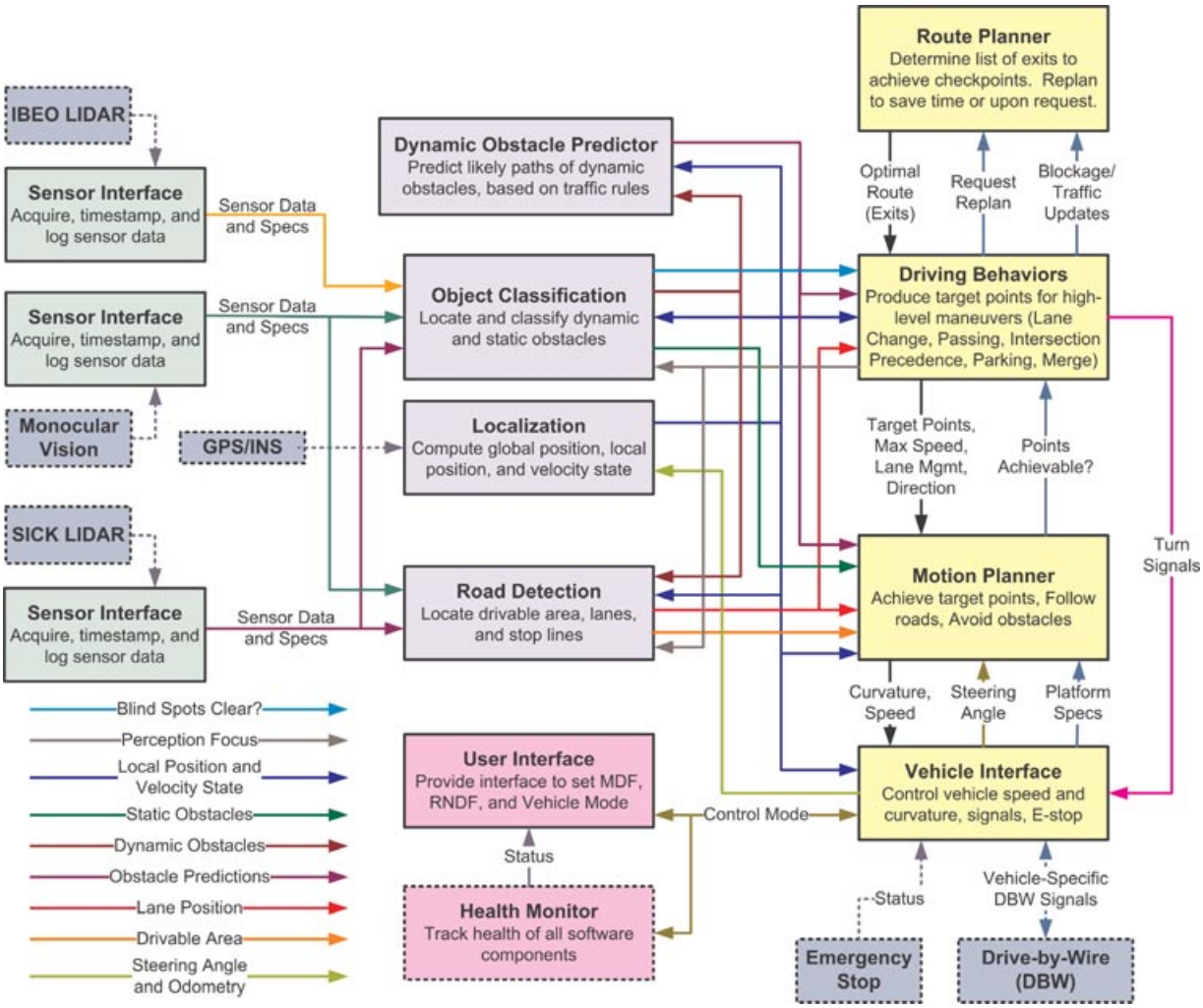

Figure 4. System architecture for Odin, omitting health monitor connections for clarity.

SICK LMS 291 single-plane range finders are used to cover the side blind spots of the vehicle and ensure 360-deg coverage. The side-mounted SICK LMS sensors are primarily utilized during passing maneuvers.

Two IEEE 1394 color cameras were intended to supplement the IBEO obstacle classification software and perform road detection but were not used in the final competition configuration. In combination, the cameras cover a 90-deg horizontal field of view in front of Odin, and each transmits raw 1,024 × 768 images at 15 frames per second.

2.2.2. Road Detection

The road detection software component provides information about nearby roads and zones in the form of lanes (report lane position) and overall drivable area (drivable area coverage). Report lane position describes the available lanes of travel and is used for decision making, vehicle navigation, and dynamic obstacle predictions. Drivable area coverage defines all areas available for Odin to drive, which is applied as a road filter for detected objects, and is used to assist with zone navigation. These two outputs are generated from three different sources: the RNDF, vision

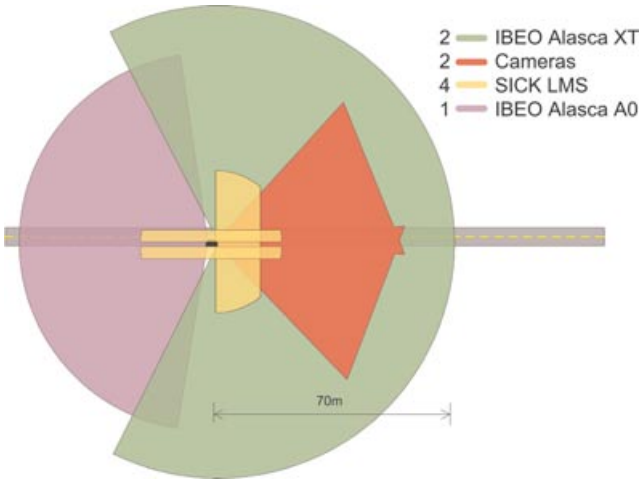

Figure 5. Odin's sensor coverage. The shaded areas indicate the maximum range of the sensor or the point at which the sensors scanning plane intersects the ground. Odin is facing to the right in this figure.

data, and SICK LIDAR data. The RNDF is used to define all lanes and drivable areas within a certain range of the vehicle. The sensor data are then used to better define roads when the way points are sparse or GPS coverage is poor. Both SICK LIDAR and vision processing can be manually enabled or disabled if not needed due to the configuration of the current course.

2.2.3. RNDF Processing

The basis for the road detection module is the RNDF supplied by DARPA. The specified lanes and exit–entrance pairs in the file are preprocessed automatically to create continuous paths for Odin. Cubic spline interpolations produce a piecewise continuous curve that passes through all way points in each lane. This interpolation uses a cubic function, the way point positions, and a desired heading to ensure a smooth transition between adjoining pieces of the lane (Eren, Fung, & Evans, 1999). The cubic function used to define the spline interpolation is

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x,$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y,$$

where $x(u)$ and $y(u)$ are the point position at u, which is incremented from zero to one to generate the spline interpolation between two points. The eight unknowns of these two equations (a_x, b_x, c_x, d_x, a_y, b_y, c_y, d_y) can be determined using the eight boundary conditions set by the way point positions and desired headings:

$$p(0) = p_k,$$
$$p(1) = p_{k+1},$$
$$p'(0) = \tfrac{1}{2}c(p_{k+1} - p_{k-1}),$$
$$p'(1) = \tfrac{1}{2}c(p_{k+2} - p_k),$$

where p_{k-1}, p_k, p_{k+1}, and p_{k+2} represent the way point positions (x and y) and c is the desired curvature control. In matrix form, this set of equations for a spline interpolation between two points is as follows:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \\ a_y \\ b_y \\ c_y \\ d_y \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ x_{k+1} \\ y_{k+1} \\ \frac{1}{2}c(x_{k+1} - x_{k-1}) \\ \frac{1}{2}c(y_{k+1} - y_{k-1}) \\ \frac{1}{2}c(x_{k+2} - x_k) \\ \frac{1}{2}c(y_{k+2} - y_k) \end{bmatrix}.$$

After solving for the unknowns, u is incremented at the desired resolution, or number of points, to create the interpolation between the two way points. The curvature control variable can be adjusted to increase or decrease the curvature between the points. This splining is also used to generate the desired path (termed a "lane branch") through intersections for every exit–entrance pair. Branches extend a lane from an exit point to the entrance point of a connecting lane, allowing lane maintenance to guide a vehicle through an intersection. Figure 6 shows an example spline interpolation for a 90-deg right turn (e.g., a typical right turn at a four-way intersection). The plot on the left shows an ideal spline for this intersection, and the plot on the right shows the effect of a lower curvature control value. Four way points, shown as round points, are required for a cubic spline interpolation. The linear connection between these points is shown as the dashed line for visual reference. The cubic spline interpolation with a resolution of five points is the solid line with square points.

All splining for the RNDF is preprocessed when an RNDF is loaded. As Odin drives the RNDF, nearby lanes and intersections are extracted and assembled

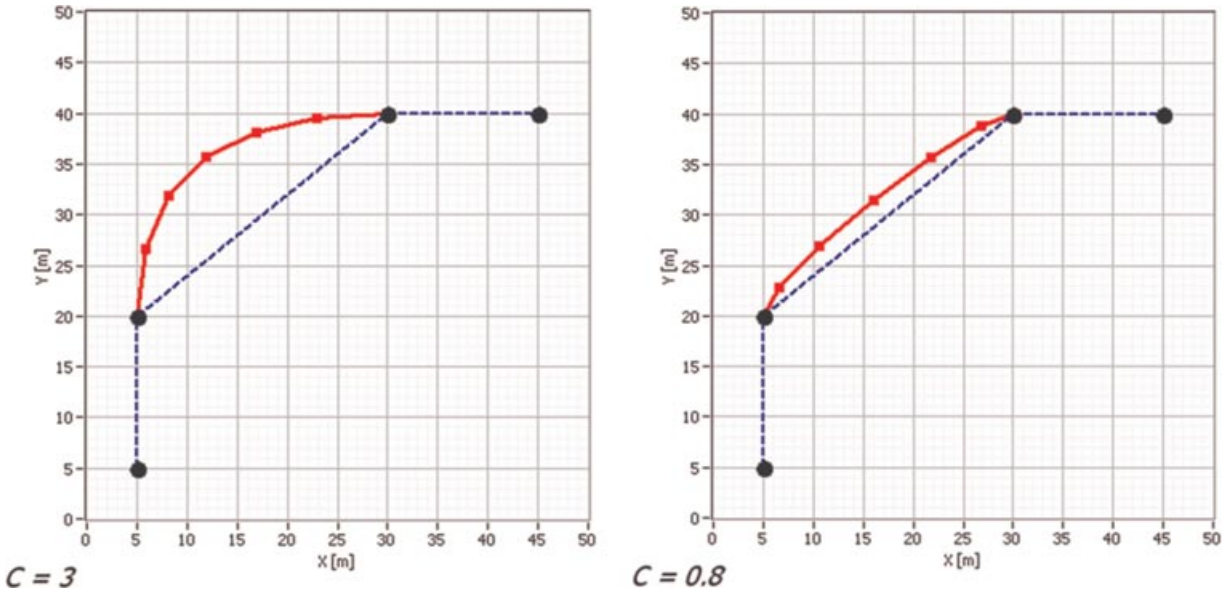

Figure 6. Example cubic spline interpolation for a 90-deg right turn with an ideal spline (left). The effect of a lower curvature control value is also plotted (right). Way points are the round points. Spline interpolation is the solid line with square points. For visual reference, linear connections are shown by the dashed line.

into the report lane position output. This output creates the possible paths for Odin and other sensed vehicles. The report lane position software was developed to automatically generate these cubic spline interpolations for an entire RNDF. This is achieved in two steps using the geometry between way points. First, it is determined whether spline interpolation is necessary. In other words, a series of straight way points or a straight through intersection does not require cubic splining, but traveling around a traffic circle would. Next, the curvature control is selected for the locations that required splining using the distance between way points, the previous and future desired headings, and knowledge gained from cubic spline data analysis. The automatic spline generator was originally designed to guide Odin through intersections as a human driver would and therefore performed close to flawlessly in these more open navigation situations. Lane splines, on the other hand, required much more precision to follow the course of the road. The splining process always provided smooth paths for the vehicle but lacked the realization of the actual geometry of the roads and intersections.

A solution to this problem was to compare the cubic spline output to geo-referenced aerial imagery, which was guaranteed to be supplied by DARPA for the competition. Using this information, the spline curvatures could be manually adjusted to help ensure that the splined lane positions match the physical lanes in the road network. Therefore, the RNDF processing software was modified to accept manual overrides to the generated splines and save a configuration file if changes were required. Figure 7 shows an example of comparing the splined report lane position output (bright overlays) to the actual road (outlined by the beige curbs) in the geo-referenced aerial imagery of Victorville.

Figure 7. Example of the splined lane position (bright overlays) output matching the actual roads in the aerial imagery. The road displayed is segment 11 of the UCE RNDF.

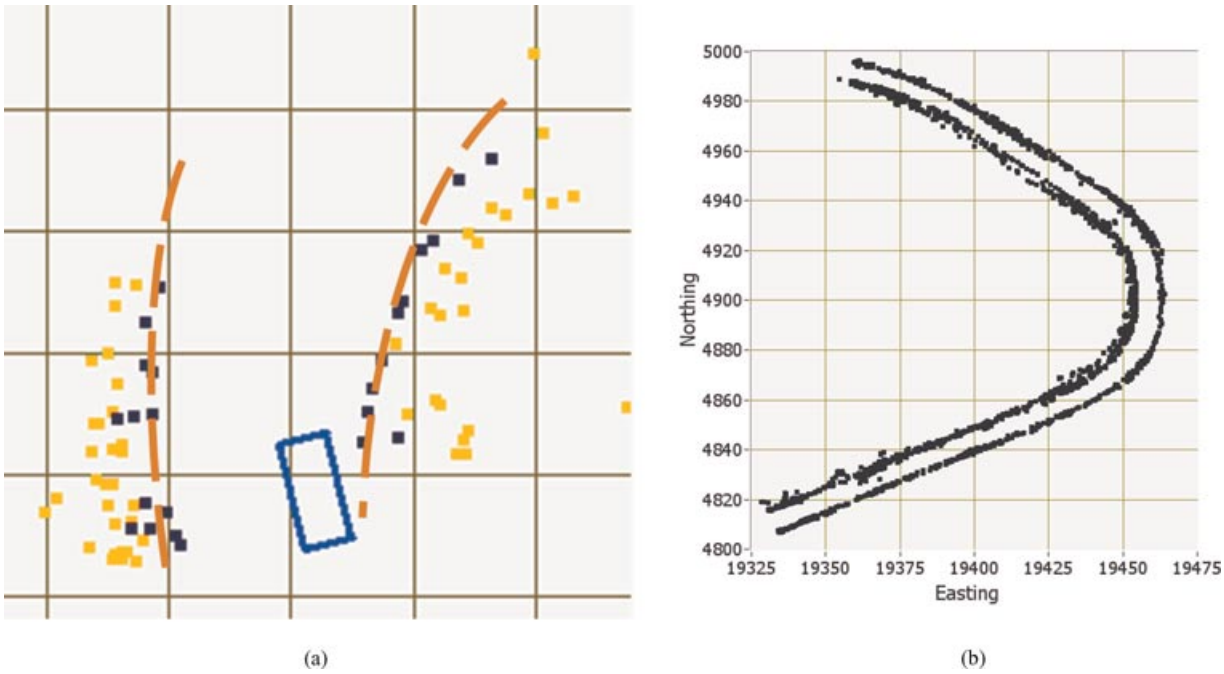

Figure 8. LIDAR-based road detection results: (a) A single frame of potential curb points in world frame with points used in the regression darkened. The regression output curve is the dashed line. Grid spacing is 5 m, and Odin is represented by the rectangle. (b) An aggregation of all curb points previously used in the lane boundary regression along a single road, plotted in universal transverse mercator (UTM) coordinates.

RNDF-based drivable area coverage uses a combination of the splined report lane position output and the zones from the RNDF. This is also preprocessed to create the drivable area coverage for the entire course. During operation, the drivable area within range of Odin is extracted and output as a drivable area map. This drivable area map is a binary image that marks all nearby areas in which Odin or another vehicle can be expected to operate.

2.2.4. Sensor Data

Odin also uses LIDAR data to supplement the RNDF-generated lane and road positions. The two forward-looking SICK LIDARs identify the road by looking for rapid changes in range that result from discontinuities in a flat road surface, such as those caused by curbs, potholes, and obstacles. The SICK LIDARs are positioned at different vertical angles to allow the algorithm to analyze multiple returns on the same features.

Lane positions can be predicted by fitting probable road boundary locations through a second-order, least-squares regression analysis. Given the locations classified as potential curb sites by the LIDAR, the curb points are defined as the points that follow the previous estimate inside an allowable scatter. This scatter is determined by the standard deviation of the previous estimate and is weighted to allow more points closer to Odin and to select the closest curb boundary detected. Figure 8(a) shows logged data indicating all potential curb points, the subset used in the regression, and resulting boundary curves. Lane position can then be determined by referencing the expected number of lanes in the RNDF. The RNDF-based report lane position output can be augmented with these sensed lanes. Figure 8(b) shows an aggregation of the curb points previously used in the regression traveling down a two-lane road. The curb points follow the shape of the of the road, but as shown in Figure 8(a), detection reaches only 10–15 m in front of the vehicle, requiring software to slow the vehicle to maintain safe operation.

In addition to LIDAR, computer vision approaches were also attempted for detecting lanes as well as improving the drivable area coverage map.

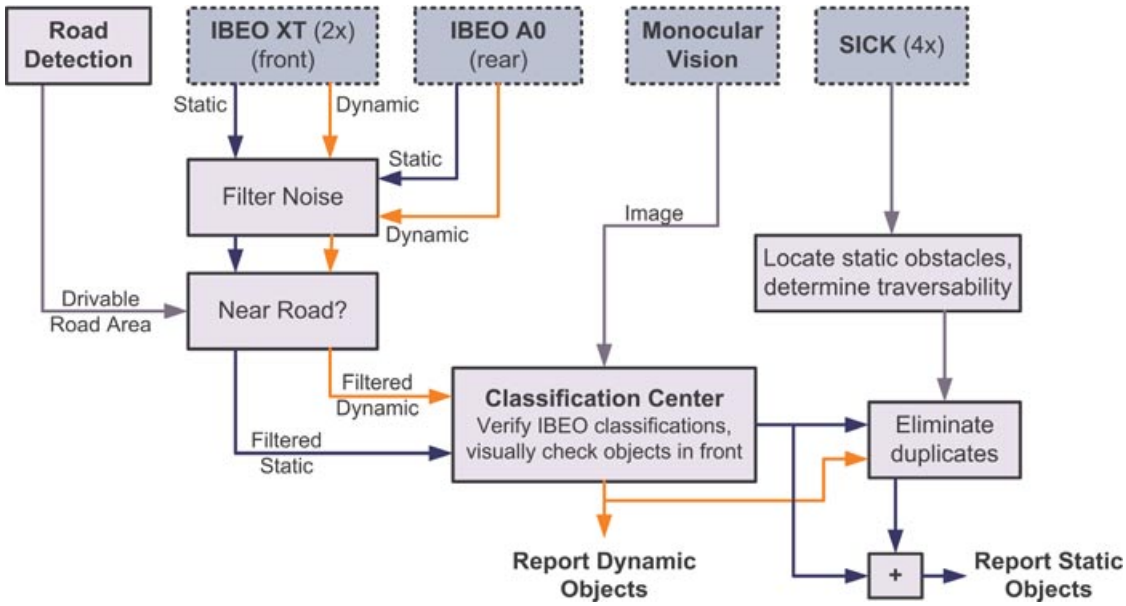

Figure 9. The object classification module localizes and classifies all perceived objects.

The visual lane detection software uses image intensity to distinguish bright lane boundaries from the darker surrounding areas of the road surface. Edge detection is applied to the results of the intensity operation, separating out the lines and edges of the road. The position of each lane is found by fitting the strongest lines on the road to a curve through a Hough transform (Duda & Hart, 1972). Vision was also used to improve the drivable area coverage map in zones by finding low-profile islands usually found in parking lots. These islands are often a color different from that of the surrounding area, and therefore vision processing is ideal for detecting them. The algorithm thresholds the entire image according to the absolute color of an area directly in front of Odin, which is assumed to be drivable. Significant color changes in this control area are ignored for a short period to improve this assumption. The detected features are then subtracted from the drivable area coverage map generated from the RNDF. Neither of these vision algorithms was used in the final competition due to a lack of sufficient testing and development, as further discussed in Section 3.

2.2.5. Object Classification

The accurate identification and classification of objects is one of the most fundamental and difficult requirements of the Urban Challenge. The vision system and the laser range finders all have advantages and disadvantages for classification. The IBEO range finders can determine the location of an object to submeter accuracy, but they have poor classification capabilities. Vision-based methods can result in accurate classification, but they are computationally intensive and have a limited horizontal field of view and range.

The classification module splits all objects into one of two categories: static objects that will not be in motion and dynamic objects that are in motion or could be in motion. Dynamic objects on the course are expected to be either manned or unmanned vehicles. The core of the classification module, shown in Figure 9, is the IBEO laser range finders. Although visual methods of detection were examined, they were determined to be too computationally intensive to return accurate information about nearby objects, especially at higher vehicle speeds. This problem is intensified because multiple cameras are needed to cover a full 360-deg sweep around Odin. The A0 and XT Fusion range finders cover almost the entire area around Odin, and objects can be detected and segmented using software available on the IBEO ECUs (Fuerstenberg, Dietmayer, & Lages, 2003; Fuerstenberg, Linzmeier, & Dietmayer, 2003). These ECU objects serve as the basis for the module. However,

small deviations in the laser pulse's reflection point and time of flight often cause variations in the results of the built-in segmentation routines. To account for these variations, a filter is applied that requires each object to have been detected and tracked for a short but continuous period before the object is considered valid. The positions of these valid objects are then checked against the drivable area coverage map; anything not on or close to drivable area is considered inconsequential and is not examined.

Once these detected objects are sufficiently filtered through their position and time of detection, their characteristics are passed to a classification center for verification. Through testing, the IBEOs have proven to be accurate in determining a moving object's velocity, and it is assumed that all large moving objects are vehicles. It is also important for the system to detect stationary vehicles in front of Odin for situations such as intersection queuing and precedence. The initial software design included verification of object classification using monocular image processing. The real-world locations of objects obtained from the IBEOs are transformed into regions of interest within the image, which are searched for features common to cars such as taillights and tires (Cacciola, 2007). By restricting processing to these regions, high-resolution imagery could be used without the usual processing penalty. This feature was effective at correcting groups of static obstacles being incorrectly classified as a dynamic obstacle. However, an inadequate amount of test time was logged to verify that certain critical failure modes, such as the vision system incorrectly identifying a dynamic object as static or the correct handling of complete vision outages due to poor lighting conditions, would not occur. Therefore, the vision portion of the classification module was not used in the final competition.

2.2.6. SICK LIDAR-Based Detection

The four SICK LMS-291 units on Odin were used for close-range object detection. The two side-mounted SICK LIDAR are devoted to blind spot checking. Figure 10 shows a history of LIDAR objects after being transformed into vehicle frame. If the areas adjacent to the vehicle sides have a return that is above a height threshold, then the blind spot is reported as not clear.

The two front-mounted SICK LIDARs are used to detect objects that are within close range of the vehicle but outside the IBEO's vertical field of view. The returns of these downward-pointed laser range finders are segmented based on range discontinuities and classified as road or an obstacle based on a height threshold. This information is compared over multiple scans to capture the overall behavior of an object through time and is used to reclassify the object if necessary. Figure 10 shows an example of classifications derived from a SICK scan cycle. This illustration also shows the potential curbs marked as circles. These points are determined after the drivable area is distinguished and defined as points where the drivable area ends or has a sharp step in the profile.

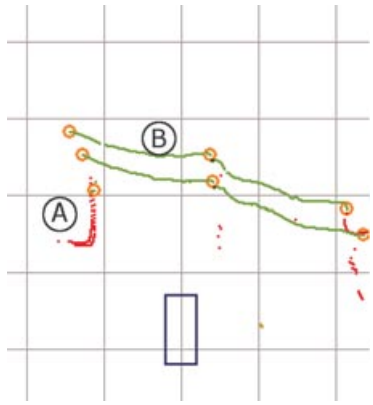

Figure 10. SICK LMS scan data transformed into vehicle frame. A (the outline of a car) has been classified as an obstacle, B has been classified as drivable road, and the circles are potential curb sites. Grid spacing is 5 m, and Odin is represented by the rectangle.

2.2.7. Dynamic Obstacle Predictor

Once an object has been classified as a vehicle, it is monitored by the dynamic obstacle predictor, which predicts likely paths for each vehicle based on road data and the motion history of the object. If there are no available lane data, such as in zones or if a dynamic obstacle does not appear to be following a lane, the dynamic obstacle predictor simply continues the current motion into the future. These predictions are used by driving behaviors for traffic interaction at intersections (such as merges) and motion planning for obstacle avoidance.

2.2.8. Localization

Odin has been equipped with a Novatel Propak LB+ system that provides a filtered GPS/INS solution. In

addition, wheel speed and steering-angle measurements are available from the vehicle interface. An extended Kalman filter (EKF) has been developed using standard methodology (Kelly, 1994) that combines these measurements with the goal of improving the Novatel solution during GPS outages. In addition, this filter provides a means for incorporating other absolute position measurements. The best-case accuracy of the position provided by this localization solution is 10 cm circular error probable (CEP).

A separate local position solution is also tracked in Odin's localization software. This solution provides a smooth and continuous position estimate that places Odin in an arbitrary local frame. The goal of this position solution is to provide perception modules with a position frame to place obstacles that is free of discontinuities caused by GPS position jumps. This is accomplished by calculating the local position using only odometry. This position typically drifts by 2.6% of the total distance traveled by the vehicle; however, this error accumulation is small enough that it does not cause significant position error within the range of the vehicle's perception sensors.

2.3. Planning

Decision making for Odin is handled by a suite of modules cooperating in a hybrid deliberative–reactive–deliberative fashion. Each of the major components is presented in sequence, from the top down.

2.3.1. Route Planning

The route planner component is the coarsest level of decision planning on Odin as it determines only which road segments should be traveled to complete a mission. It uses a priori information such as the road network and speed limits specified by the RNDF and mission data file (MDF), respectively, as well as blockage information gathered during mission runs. After processing, the route planner outputs a series of way point exits to achieve each checkpoint in the mission.

By considering only exit way points, it is easy to formulate the route planner as a graph search problem. The route planner on Odin implements the A* graph search method (Hart, Nilsson, & Raphael, 1968) using a time-based heuristic to plan the roads traveled. Whereas the A* search algorithm guarantees an optimal solution, it depends on the validity of the data used in the search. The time estimate used during the search assumes that the vehicle is able to travel at the specified segment speed limits, and it uses predefined estimates of the time for typical events, such as the time taken when traversing a stop line intersection, performing a U-turn, or entering a zone.

2.3.2. Driving Behaviors

Driving behaviors is responsible for producing three outputs. First, driving behaviors must produce a behavior profile command that defines short-term goals for motion planning in both roads and zones. Second, in the event of a roadblock, a new set of directions must be requested from the route planner. Finally, the turn signals and horn must be controlled to appropriately signal the intent of the vehicle.

The behavior profile sent to motion planning comprises six target points, a desired maximum speed, travel lane, and direction (forward, reverse, and don't care). Each target point contains a way point location in UTM coordinates, the lane, and lane branch (for intersections). Target points can also contain optional fields such as a stop flag and a desired heading. Finally, the behavior profile also contains zone and safety area flags to enable different behaviors in motion planning.

2.3.2.1. Action-Selection Mechanism

Driving behaviors must coordinate the completion of sophisticated tasks in a dynamic, partially observable, and unpredictable environment. The higher-level decision making being performed in driving behaviors must be able to handle multiple goals of continually changing importance, noisy and incomplete perception data, and noninstantaneous control. To do this, a behavior-based paradigm was implemented. The unpredictable nature of an urban environment calls for a robust, vertical decomposition of behaviors. Other advantages of using a behavior-based paradigm include modularity, the ability to test incrementally, and graceful degradation (Murphy, 2000). For instance, if a behavior responsible for handling complex traffic situations malfunctions, simpler behaviors should still be operable, allowing Odin to continue on missions with slightly less functionality.

As with any behavior-based architecture, implementation details are extremely important and can lead to drastically different emergent behaviors. Coordination becomes paramount because no centralized planning modules are used and control is shared among a variety of perception–action units,

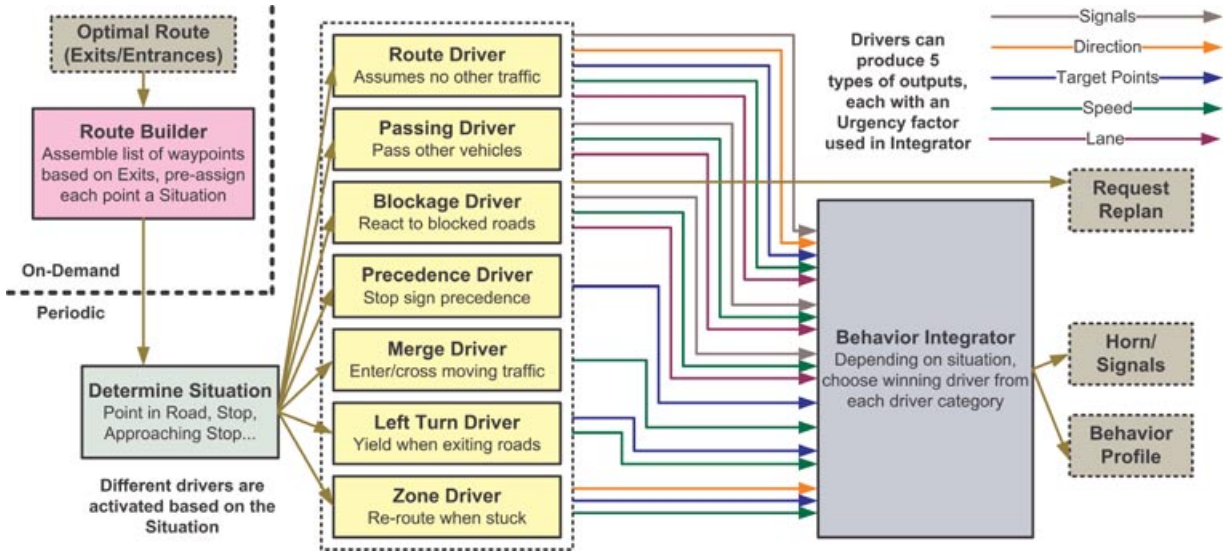

Figure 11. Flow diagram of the behavior-based, winner-takes-all driving behaviors implementation. Behavior integrator ensures that there is one winner from each driver category.

or behaviors. In the Urban Challenge environment, the problem of action selection in the case for conflicting desires is of particular interest. For example, the desire to drive in the right lane due to an upcoming right turn must supersede the desire to drive in the left lane due to a slow-moving vehicle. Furthermore, due to the strict rules of urban driving, certain maneuvers must be explicitly guaranteed by the programmer. To address this problem, a method of behavior selection is needed such that driving behaviors will actively determine and run the most appropriate behaviors given the current situation.

An *arbitration* method of action selection (Pirjanian, 1999) is used for the driving behaviors module. In the above example of choosing the appropriate lane to drive in, driving with two wheels in each lane is not an acceptable solution. Therefore, a modified winner-takes-all (Maes, 1989) mechanism was chosen. To address the situational awareness problem, a system of hierarchical finite state machines is used. Such a system allows driving behaviors to distinguish between intersection, parking lot, and normal road scenarios (Hurdus, 2008). The implementation of finite state machines also provides resilience to perception noise, and by using a hierarchical system, concurrency is easily produced. The overall architecture of driving behaviors is shown in Figure 11. A finite state machine is used to classify the situation, and each individual behavior can be viewed as a lower-level, nested-state machine. The chosen action selection mechanism operates within the behavior integrator. This approach is considered a *modified* winner-takes-all approach because all behavior outputs are broken down into one of several categories, including, but not limited to, target point drivers, speed drivers, and lane drivers.

2.3.2.2. *Passing and Blocked Roads*

When driving down a normal section of road (i.e., not in a safety zone approaching an intersection, not in an intersection polygon, and not in a zone), Odin runs three behaviors, the route driver, the passing driver, and the blockage driver. The route driver is responsible for driving the route as closely as possible to the route originally provided by the route planner. If no obstacles or traffic are ever encountered, then the route driver will maintain control of the vehicle throughout all segments of the RNDF. When entering a new segment, for example, the route driver will immediately attempt to move Odin to the correct lane for the next exit.

The passing driver is concerned with getting around slow-moving or disabled vehicles. It is therefore responsible for monitoring other vehicles in the near vicinity, deciding whether a pass is necessary,

and executing this pass in a safe and legal manner. Awareness of the roads is necessary as the passing driver must distinguish between passing in an oncoming lane and passing in a forward lane and subsequently check the appropriate areas for traffic. The passing driver does not maintain knowledge of the overall route, so it is the responsibility of the route driver to overrule the passing driver if a pass is initiated too close to an exit or intersection.

Finally, the blockage driver maintains a current list of available lanes. If static obstacles in the road or a disabled vehicle cause a lane to be blocked, the blockage driver removes this lane from the available list. If all RNDF-defined lanes are removed from the list and at least one of these lanes is an oncoming lane, then the blockage driver commands a dynamic replan. When this is necessary, the route planner is first updated with the appropriate blockage information and all behaviors are reset while a new route is generated.

The interaction of these three drivers was sufficient for giving Odin the ability to pass disabled and slow-moving vehicles in the presence of oncoming and forward traffic, pass static obstacles blocking a lane, pass over all checkpoints, be in the correct lane for any exit, and initiate dynamic replans when necessary.

2.3.2.3. *Intersections*

To handle intersections, Odin uses three drivers (precedence, merge, and left turn) in the approaching stop, stop, approaching exit, and exit situations. Of special note is that all three drivers operate by monitoring areas where vehicles (or their predictions) may be, rather than tracking vehicles by ID. Although this decision was initially made to deal with object-tracking issues in early iterations of the perception software, it turned out to greatly enhance the overall robustness of the intersection behavior.

The precedence driver activates when Odin stops at junctions with more than one stop sign. This driver overrides the route driver by maintaining the current stop target point with a high urgency until it is Odin's turn or a traffic jam has been detected and handled. The merge driver activates at intersections where Odin must enter or cross lanes of moving traffic, controlling the speed by monitoring areas of the merge lane or cross lanes for vehicles or vehicle predictions. To handle intersections with both moving traffic and other stop signs, the merge driver cannot adjust the speed until the precedence driver has indicated that it is Odin's turn or a traffic jam has been detected.

For turns off a main road (a case in which the RNDF does not explicitly state right-of-way via stop points), the left turn driver activates when Odin's desired lane branch at an upcoming exit way point crosses over oncoming-traffic lanes. In this case, the left turn driver overrides the route driver by setting the stop flag for the exit target point. Once the exit way point has been achieved, the driver controls the desired speed in the behavior profile while monitoring the cross lanes for a sufficient gap to safely achieve the left turn.

2.3.2.4. *Parking Lot Navigation*

In zones, the role of driving behaviors is to provide general zone traversal guidance by controlling the behavior profile target points. Initially, VictorTango planned to fully automate the first stage of this process, determining the accepted travel patterns through a parking lot (along the parking rows). However, zones containing only a subset of the actual parking spots (i.e., one spot per row) make it difficult to automatically identify parking rows based on the RNDF alone. Because this could very well be the case in the final event, a tool was designed to manually place "control points" in the zone, in the form of a directionally connected graph.

In the route-building stage of driving behaviors, Odin performs a guided Dijkstra search to select control points for navigating toward the parking spot and reversing out of the spot. The route driver then guides Odin along this preplanned path. If the path is blocked, the zone driver can disconnect a segment of the graph and choose a different set of control points. The parking maneuver is signaled to motion planning by enabling the stop flag and providing a desired heading on the parking checkpoint. To reverse out of the spot, the direction is constrained to be only in reverse, and a target point is placed in order to position Odin for the next parking spot or zone exit.

Driving behaviors is not responsible for any object avoidance or traffic behavior in zones. Motion planning handles tasks such as steering to the right for oncoming dynamic objects, performing the parking maneuver, and checking for safe space to reverse out of the spot. This was primarily due to the largely unknown environment in a zone and the desire to keep driving behaviors independent of the exact size and mobility constraints of the vehicle.

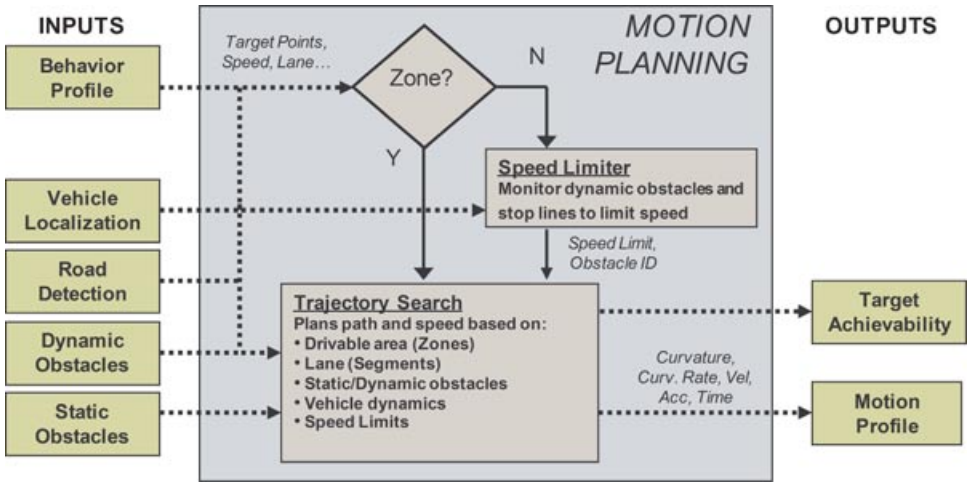

Figure 12. Software flow diagram of the motion planning component.

2.3.3. Motion Planning

Motion planning is responsible for planning the speed and path of the vehicle. Motion planning receives behavior profiles, defined in Section 2.3.2, from driving behaviors and plans a series of motion commands called a *motion profile*. The motion profile is a platform-independent series of commands that include a desired curvature, curvature rate of change, desired velocity, maximum acceleration, and time duration of each command. The motion profile consists of the entire path planned by motion planning and typically contains 2–3 s of commands. A platform-independent motion profile message allows reuse of communication messages across base platforms and allows vehicle-specific control loops to take place in the vehicle interface. However, motion planning still requires a basic model of the specific operating platform. In addition to commanding the vehicle interface, motion planning can also provide feedback to driving behaviors about whether the currently commanded behavior profile is achievable. This feedback is critical when detecting a blocked lane or even an entirely blocked roadway.

Motion planning is structured into two main components, consisting of a speed limiter and a trajectory search, as shown in Figure 12. The speed limiter commands a maximum speed based on traffic in the future path of Odin and upcoming stop commands. Dynamic obstacle predictions are analyzed to follow slower-moving traffic or to stop behind a disabled vehicle leaving enough room to pass. The speed limiter sends trajectory search this maximum speed and an obstacle ID if the speed is limited by a dynamic obstacle. The speed limiter is disabled when traveling through zones, leaving trajectory search to handle all dynamic obstacles.

The core of motion planning is the trajectory search module that simulates future motion to determine a safe and fast path through the sensed environment. Three main steps happen in trajectory search: (1) a cost map is created using lane and obstacle data, (2) a series of actions is chosen to reach a goal state, and (3) the series of actions is processed into a feasible motion profile. The trajectory search plans with the assumption of an 80-ms processing time. If run time exceeds 450 ms or all possible actions are exhausted, the goal criteria are declared unachievable. Driving behaviors is notified about the unachievable goal, and a stop is commanded, with urgency depending on the proximity of obstacles.

Trajectory search uses a fixed-grid-size cost map to represent the environment when solving for a motion path. The range and resolution of the cost map are configurable, and the final configuration used a resolution of 20 cm^2 per cell, extending 30 m in front of the vehicle and 15 m behind and to the sides of the vehicle. It is important to note that driving behaviors and the speed limiter software did not use this cost map, allowing these modules to incorporate dynamic obstacles beyond this range. The cost map stores costs as 8-bit values, allowing obstacles to be stored as highest cost at the obstacle and reduced cost around

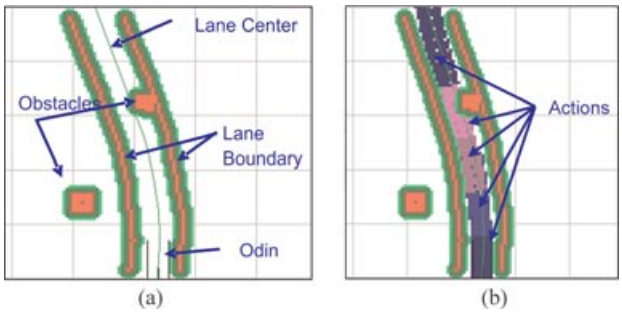

Figure 13. (a) Trajectory search cost map with labeled features. (b) Trajectory search cost map with planned solution representing each action as a differently shaded path segment.

the obstacle. Figure 13(a) shows an example cost map with costs added from obstacles and lane boundaries. Dynamic obstacles are expanded to account for future motion. However, this treatment of dynamic obstacles is very limiting, and responding to dynamic obstacles is mainly the job of the speed limiter. If the speed limiter is reacting to a dynamic obstacle, the ID is passed to trajectory search and the obstacle is omitted from the cost map. Omitting these dynamic obstacles prevents trajectory search from deciding that a slower-moving vehicle is blocking a lane.

Once a cost map is created, trajectory search produces a set of goal criteria using data from the behavior profile, such as desired lane, zone, desired gear, and heading criteria. Goal criteria may be as simple as driving a set distance down a lane or more specific such as achieving a desired position with a desired heading. The search process starts with the current vehicle state and uses an A* search to evaluate sequences of possible future actions. Each action consists of a forward velocity and steering rate over a period of time. These actions are evaluated by checking the predicted path of the vehicle due to the action against the cost map as well as other costs such as distance from the lane center and time. The search speed is improved by using only a finite set of possible actions that have precomputed motion simulation results (Lacaze, Moscovitz, DeClaris, & Murphy, 1998). Figure 13(b) shows a planned path, with each action as a differently shaded path segment.

Odin uses a precomputed results set with an initial steering angle varied at 0.25-deg increments, commanded steering rates varied from 0 to 18 deg/s at 6-deg/s increments, and commanded velocities ranging from 2 to 12.5 m/s at 3.5-m/s increments. The precomputed results contained information including which grid cells will be occupied as well as final state information. A separate set of coarser results is used in situations in which the vehicle is allowed to travel in reverse. When creating a sequence of actions, the precomputed occupancy data are translated and rotated based on the previous ending-state conditions. When the search algorithm runs, actions that are dynamically unsafe or have a commanded velocity too far from the previous velocity are filtered out. After the search is complete, the list of actions is converted to a drivable series of motion commands. This last step selects accelerations and accounts for decelerating in advance for future slower speeds. Steering rates are also scaled to match the planned path if the vehicle is traveling at a speed different from that originally planned. The trajectory search solves the goal using the fastest possible paths (usually resulting in smoother paths), and these speeds are often reduced due to MDF speed limits or due to the speed limiter.

While traveling in segments, trajectory search chooses goals that travel down a lane. In contrast, zone traversal is guided by target points along with goal criteria and search heuristics to produce different behaviors. The behaviors include forming artificial lanes to connect the points, a recovery behavior allowing the vehicle to reverse and seek the center of the fake lane, and a wandering behavior that uses no lane structure at all. These behaviors are activated by a simple state machine that chooses the next behavior if the current behavior was resulting in unachievable goals. Development time was reduced and reliability was improved by using the same trajectory search algorithm in zones as well as segments. Although the overall behavior of the vehicle could be adjusted by changing the goals and heuristics, the core functionality of planning fast collision-free motion remained constant.

2.3.4. Vehicle Interface

The main role of the vehicle interface component is to interpret the generic motion profile messages from motion planning, defined in Section 2.3.3, and output vehicle-specific throttle, brake, steering, and shifting signals. By accepting platform-independent motion commands and handling speed control and steering control at a low level, any updates to the vehicle-specific hardware or software can be made transparent to the higher-level decision-making components.

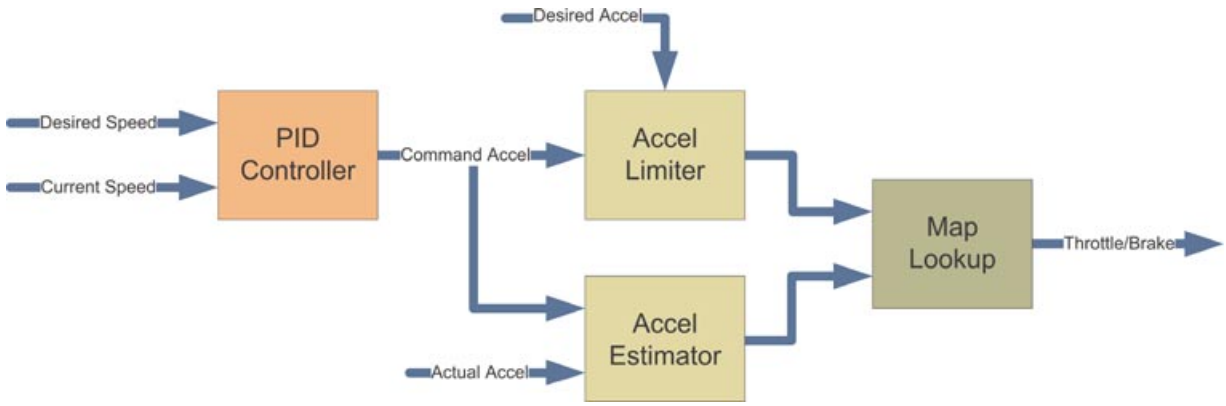

Figure 14. Block diagram of speed controller showing PID controller, acceleration controller, and map lookup linearization function.

Additionally, the vehicle interface can actuate other vehicle systems such as lights, turn signals, and the horn.

Closed-loop speed control is provided by a map-linearized proportional–integral–derivative (PID) controller. The controller, as shown in Figure 14, takes the output of the PID, band limits it to control the maximum acceleration, and inputs it to a map lookup function to produce a throttle or brake command. Terrain compensation is provided by a proportional controller that estimates the longitudinal acceleration on the vehicle and additively enhances the map input (Currier, 2008).

Steering control relies on a standard bicycle model to estimate the curvature response of the vehicle (Milliken & Milliken, 1995). This model can be shown to produce estimates accurate enough for autonomous driving in the operational conditions found in the Urban Challenge (Currier, 2008). The steering angle and rate calculated by the bicycle model is tracked by a rate-controlled PID loop to produce the desired vehicle path.

3. FINAL SOFTWARE CONFIGURATION

As the Urban Challenge approached, decisions had to be made for the final software configuration to be used in the event to ensure adequate testing time. These choices ranged from setting maximum/minimum parameters to disabling software components. This section explains test procedures and rationale for the final adjustments made to Odin's software configuration.

3.1. Motion Planning Parameters

For all NQE runs as well as the final UCE, the maximum speed that Odin would drive was limited to 10 m/s (22 mph). This allowed Odin to drive at the maximum speed specified for all segments during NQE and the maximum speed for all but two segments on the UCE. The limiting factor in determining a safe maximum speed was the distance at which obstacles could be reliably detected. During testing, the vehicle could smoothly drive roads at speeds of 13 m/s (29 mph), but would occasionally stop very close to obstacles such as a disabled vehicle or roadblock. This was due to hills or changing terrain causing obstacles to be out of the vertical sensor field of view until Odin was closer to the obstacle.

3.2. Sparse Road Detection

Prior to competition, members of the team tested the road detection suite in a wide variety of sparse way point scenarios. Odin was tested on various road compositions including dirt, gravel, and asphalt. Lane definitions were also varied. These tests were on well-lined and poorly lined roads, as well roads with curbs, drop-offs, and ditches. Laser range finder–based methods of lane detection yielded reasonably robust results, but due to the possible variety of road surfaces and markings, vision-based methods were much less robust. The primary challenge for lane detection was defining assumptions. The algorithm did not have enough engineering time to handle all possible roads. Further, the sparse way point example in

Figure 15. A merge in NQE course A with an occluded vehicle (vehicle 4).

the sample RNDF indicated that Odin would be required to negotiate intersections in sparse scenarios, which through analysis became very demanding on the algorithms.

As a result, the team felt comfortable that if sparse way points would be required, this software could be turned on; but it was deemed that splining way points would be the method of choice if the final-event RNDF permitted. Hence, before arriving in California, all code associated with vision-based lane detection was disabled, and an implementation of the software that allowed the team to selectively disable/enable the laser range finder–based lane detection was put in place. This allowed the team to turn on laser range finder–based lane detection when certain conditions are met. Upon receipt of each RNDF, the team would examine all points in the simulator visualization to determine whether the segments required the sparse way point algorithm to be enabled for that RNDF. Owing to the relative density of the way points in all events, there was never a requirement for this algorithm to be enabled.

3.3. Vision Drivable Area Coverage

As defined in Section 2.2.2, the RNDF defines the drivable areas and sensor data are used to verify the RNDF information and subtract areas that contained obstacles, such as landscaped islands in parking lots. After extensive testing in various environments and lighting conditions, the vision drivable area coverage was not reliable enough to be trusted. It would occasionally produce false positives that eliminated drivable areas directly in front of the vehicle when no obstacles were present. Also, there were no undrivable areas in zones present during the NQE or UCE courses. The team decided that the laser range finders and object classification were able to detect most static obstacles that would get in the way of Odin. Therefore, vision drivable area coverage was disabled to prevent the vehicle from stopping and getting stuck in the event of a false positive produced by the vision processing.

4. NATIONAL QUALIFYING EVENT

The team spent many long days and nights testing and preparing for the Urban Challenge NQE and UCE. This section presents and analyzes Odin's performance in the NQE.

4.1. NQE A: Traffic

This NQE course challenged a vehicle's ability to drive in heavy traffic while balancing safety and aggressiveness. The first vehicle to perform at NQE course A, Odin performed well, executing merges and left turns with only a few minor errors. Odin also had no trouble with the k-rails surrounding the course that caused many other teams problems when merging into traffic. The mistakes Odin made were subsequently fixed and could be attributed to the following: needing to adjust intersection commitment threshold, an IBEO region of interest bug, and false object classifications.

To prevent Odin from slamming on the brakes and blocking an intersection, Odin will commit to traversing an intersection once it has passed a calculated threshold. This threshold is based on being able to stop without protruding into traffic lanes. However, several times in Odin's first run on NQE area A, Odin commanded a merge in the situation shown in Figure 15. About 350 ms later, the previously occluded vehicle (no. 4) suddenly appeared to

sensors at a range that would normally prevent Odin from commanding a merge. However, because the stop line was very close to the cross lane, Odin had already passed the commit point and disabled the cancellation mechanism. For the second NQE run, the team adjusted the commit point, allowing merges to be canceled closer to the road. On the second course A run, Odin performed satisfactorily with the occluded cars by properly canceling the merge when the occluded vehicle was in view.

A bug in the IBEO factory software also resulted in Odin cutting off traffic vehicles. The IBEO sensors allow a region of interest to be defined to reduce the number of output objects, which is important given that the IBEO software has an internal limit of 64 objects. This region is in the shape of a trapezoid defined by its upper-left and lower-right corners. Objects filtered out by the region of interest can still count toward the limit of 64, which could cause objects far away from the vehicle to prevent closer objects from being returned. This behavior can be prevented by enabling an option to process only scan returns into objects within the region. However, if this option is enabled, the region is incorrectly defined as the thin rectangle within the trapezoid, causing all of the precalculated regions to be smaller than intended. This option was a configuration change enabled shortly before the challenge, and the issue was not discovered until after the second NQE course run. Afterward, the precalculated regions were redefined to ensure that vehicles were always seen in time for the behavior software to correctly interact with them.

The most serious incident on NQE course A occurred during the first attempt when Odin completed a couple of laps and then came to a stop when making the left turn off the main loop. Odin unfortunately remained stopped and did not proceed through large gaps in the traffic. The judges let the team reposition Odin, and Odin was able to continue completing laps without getting stuck again. After examining logs, it was found that the retroreflective lane markings were appearing as obstacles to the IBEO LIDAR. These sections were about the same length as a typical car and were classified as a stopped dynamic obstacle. The false objects would intermittently appear, causing Odin to be stuck only once. The software already had safeguards against waiting for stationary vehicles, but these were ineffective due to the object flickering in and out, resetting timers used to track stationary vehicles. To ensure that this would not occur in the final race, the planning software was made more robust against flickering objects. These changes prevented Odin from being stuck on multiple occasions on the second NQE course A run but did cause unnecessary delay when making a left turn. This problem was mainly a result of depending on a single sensor modality, LIDAR, for obstacle detection and classification. Additional sensing methods, such as vision or radar, could help reduce the number of false positives by providing unique information about surrounding objects detected through LIDAR.

4.2. NQE B: Navigation and Parking

NQE course B was used to test road and zone navigation, parking, and obstacle avoidance. The vehicles were given missions that took them around the large course, through parking areas, and down streets with parked cars and other blockages. Odin accepted the challenge at top speed and set himself up to complete with a competitive time. However, during the first run, Odin had minor parking confusion and got stuck in the gauntlet area, resulting in insufficient time to complete the mission. Odin experienced a significant GPS pop on the second run of NQE course B that caused the vehicle to jump a curb. After restarting back on the road, Odin finished the course without any problems.

During the first run of NQE B, Odin approached the desired space in the row of parking spots, shown in Figure 16, and stopped for a moment. Instead of pulling into the spot, Odin reversed, drove forward in a loop around the entire row of cars, and then pulled right into the spot at a crooked angle. This surprising maneuver was caused by an incorrect assumption regarding the amount of space that would be available in front of the vehicle after a parking maneuver. Motion planning could not find a clear path due to the car parked in front of the spot. After Odin tried to reposition itself without any progress, another behavior took control that tries to navigate around obstructions, causing Odin to circle the lot. When Odin returned to the parking spot, the software was able to find a parking solution with a clear path by parking at an angle. The parking software was improved between runs to more accurately check for a clear path that did not extend past the parking space. It is interesting to note that with a lower-level software failure, Odin was still able to park, but with reduced performance.

During the first run of NQE course B, Odin became stuck in the area known as the "gauntlet,"

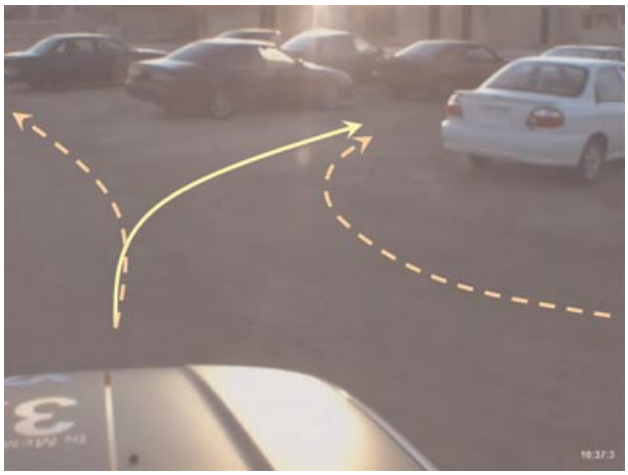

Figure 16. During NQE B, Odin did not pull into the parking spot by the direct path (solid line). Instead, Odin pulled to the left (dashed line), circling the parked cars, and eventually entered the space.

characterized by vehicles parked along both sides of the road. This length of road was further complicated with cones and traffic barrels marking construction hazards along the centerline of the road. A simulator screenshot of what was seen in the gauntlet is shown in Figure 17(a). Odin tried to change lanes to pass the disabled vehicle but was unable to execute this command because the blind spot was reported as not clear. Reviewing the logged data also revealed that Odin had difficulty traveling where both lanes of travel were significantly blocked. As seen in Figure 17(a), both lanes are mostly blocked, but there is a large space between the lanes. To solve the first problem, the lane change driver was changed to be less cautious about obstacles in a blind spot when lanes had opposite traffic directions. To allow Odin to use the entire road more effectively, motion planning would still have a strong desire to stay in the commanded lane during a lane change but was changed to not be constrained to remain entirely in the commanded lane.

Odin experienced localization failures on the second attempt at NQE course B. As Odin exited the traffic circle onto Sabre Boulevard, the position solution provided by the Novatel system suddenly jumped almost 10 m to the southeast. Because the road estimate was derived completely from GPS, this new vehicle position caused Odin to think it had left the lane. Motion planning attempted to move Odin back into the desired lane of travel, but could not do so due to a virtual boundary placed around the lane. As a result, Odin drove to the end of Sabre Boulevard, thinking that it was driving off of the road. At the end of Sabre Boulevard, Odin reached a virtual boundary caused by the intersection of two segments. Having nowhere else to go, Odin turned right and drove onto the dirt on the side of the road and was paused. Within milliseconds of the pause, the localization solution reported by the Novatel system corrected itself. Such a large jump in position was never encountered in testing and was not repeated in any NQE course or the final event.

4.3. NQE C: Intersections and Roadblocks

The final NQE course was used to test intersection precedence and dynamic replanning due to

(a) (b)

Figure 17. Gauntlet scenario from NQE area B seen in (a) simulation replay and (b) logged video.

(a) (b)

Figure 18. NQE course C contained (a) stop sign intersections and (b) road blockages.

roadblocks. Odin performed perfectly at all intersections, yielding to those who had right-of-way and taking the appropriate turn, shown in Figure 18(a). Replanning due to a road blockage was also a success. An interesting challenge presented in NQE C was a road blockage that was repeatedly approached by the vehicles during a mission. When Odin detected a blockage requiring a route replan, the route planner would add the blockage and a pair of U-turn connections to the way point on both sides of the blockage to its internal road map. Allowing a U-turn on the opposite side of a blockage provided an adequate solution for NQE C; however, it can introduce a problem in a dynamic environment where a blockage may not be permanent, allowing a vehicle to plan a U-turn in the middle of a clear road.

Odin did have trouble detecting the stop sign blockage shown in Figure 18(b). The perception module had never been presented with an obstacle that was not attached to the ground within the road area. The laser range finder sensor suite on Odin had a narrow vertical field of view, which caused difficulty perceiving this blockage. The IBEO sensors were barely able to detect the bottom of the stop signs attached to the gate while the vehicle was in motion. Once the signs were detected and the vehicle brakes were applied, the downward pitch of the vehicle caused the IBEOs to lose sight of the gate. The vehicle was then commanded to accelerate back up to speed, at which point the gate was seen again and the cycle repeated. The resulting behavior was that Odin crept forward toward the gate until the signs were detected by the close-range downward-looking SICK range finders. At times the SICKs were barely detecting the stop signs, resulting in the processed stop sign object flickering in and out of existence. Because of this, it took the behavior module a significant amount of time to initiate a replan around the blockage. To resolve this issue, the behavior software was modified to replan not only after constantly seeing a blockage for a constant period of time, but also after a blockage was seen in an area for a cumulative period of time.

4.4. Practice and Preparation

This section explains the VictorTango practice and preparation routine during the Urban Challenge events that helped make Odin successful.

4.4.1. Practice Areas

Team VictorTango always tried to maximize use of each practice time slot. The team developed a number of RNDFs that replicated sections of the NQE RNDF for each of the practice areas. A detailed test plan was created for each practice. If problems arose, the goal was not to debug software, but to gather as much test data as possible for further analysis. This proved to be an extremely effective way to test given the short amount of time allowed for each practice block.

During the practice sessions a fair amount of dust collected on the lenses of the IBEO laser range finders. Although the team was allowed to clean sensors

between missions, Odin's performance could have suffered until the vehicle returned due to this layer of dust. The IBEO sensors are capable of reading up to four returns per laser pulse in order to handle cases in which small particles such as dust or precipitation cause the light reflection. After looking at the scan profiles for the sensors while they were covered in dust, it was confirmed that a majority of the primary scan returns were contacting the dust resting on the lens. Even with the primary returns blocked, the sensors were still able to perceive all objects due to the multireturn feature. Although running with dust on the lens is not ideal, Odin is able to continue without any reduction in perception capability until the lens can be cleaned.

4.4.2. Simulation

Simulation was a tool heavily used by team VictorTango during NQE and UCE preparation. (The role of simulation in the entire software development process is discussed in Section 6.2.2.) In preparation for NQE and UCE, team VictorTango used an interactive simulator to load NQE and UCE RNDFs and dynamically add traffic vehicles or static obstacles. The team validated that Odin would be able to drive all areas of the RNDF before ever running Odin on the course and possibly wasting NQE run. For example, the planning software had trouble with the short road segments (less than 2 m long) connecting the parking zones to the surrounding road segment. Software modifications to handle this previously untested RNDF style were validated in simulation.

If a failure occurred during an NQE run, a simulation scene could be created to match the environment in which the failure occurred. As software developers fixed problems, the test team had the manpower to run simulations in five parallel groups on laptop computers. This gave the software developers the luxury of concentrating on remaining software bugs while the test team exhaustively checked new software in a full range of tests, ensuring that there were no unintended side effects.

5. URBAN CHALLENGE EVENT

The most obvious accomplishment of the VictorTango team is that Odin finished the Urban Challenge competitively. This section highlights the successes and analyzes the incidents of Odin's performance in the Urban Challenge race.

5.1. Performance Overview

First out of the gate, Odin left the start zone and drove away at top speed to complete his first mission. Not knowing what to expect, the team eagerly looked on, with people stationed at each of the viewing locations of the UCE course. Odin performed superbly, finishing in third place with no accidents or major errors. Odin's chase vehicle driver informed the team that Odin was a predictable and safe driver throughout the challenge. As the team watched the in-car video, they saw that Odin navigated the roads and zones smoothly, made smart choices at intersections, and obeyed the rules of the road. The UCE did not demand as much obstacle avoidance and intersection navigation as the NQE. However, the endurance element and unpredictable nature of robot-on-robot interactions made it just as challenging.

5.2. Perception

This section presents the perception issues Odin faced during the Urban Challenge. Perception is defined to include all aspects of the design necessary to sense the environment and transform the raw data into information useful to the decision-making software.

5.2.1. Localization Pops

During the UCE, Odin experienced a localization failure much like the one encountered during the NQE area B second attempt. Unlike the localization error during NQE, the position this time jumped to the north, causing Odin to compensate by driving onto the curb to the south. Fortunately, the position jump was not as severe. After a brief pause, the localization solution returned to the correct position and Odin returned to the road and continued through the course.

Just as in the NQE, the data needed to diagnose the true cause of this error were not being logged. Although code changes could have been made to add these data to the localization logs, the team decided that this was an unnecessary code change and that there was insufficient time to test for unexpected errors before the final competition.

5.2.2. IBEO Reset

A known problem with the IBEO sensors was that occasionally the internal ECU factory software would freeze, resulting in no scan or object data being

transmitted to the classification software module. No specific cause could be identified for this problem; however, it often occurred after the ECU software had been running for more than 4 h. To remedy this situation, a health-monitoring routine had been built into the sensor interface code. When the interface fails to receive sensor data for a full second, it can stop Odin by reporting an error to the health monitoring module. If the connection is not restored within 5 s, the power is cycled to the IBEO ECUs to reset the software. This reset takes approximately 90 s, and the vehicle will remain in a paused software state until sensor data are restored. During the third mission of the UCE, this ECU software freeze occurred as Odin approached one of the four-way intersections. The software correctly identified this failure and cycled power to the sensors. After the reset time elapsed, Odin was able to successfully proceed through the course without any human intervention.

5.2.3. Phantom Object

Early in the race, Odin traveled down the dirt road on the southeast section of the RNDF. Without incident, Odin traversed the road until the road began bending left before entering Phantom East. Odin slowed to a stop prior to making this last turn and waited for close to a minute. The forward spacing enforcer was keeping Odin from going farther because object detection was reporting a dynamic obstacle ahead in the lane. Owing to the method by which dynamic obstacles are calculated, the berm to Odin's left had a shape and height that appeared to be a vehicle. Had sparse lane detection been enabled for the race, Odin would have corrected the actual location of the road and object classification's road filter would have surely eliminated the berm as a possible vehicle. However, because this was not in place, Odin was doomed to wait indefinitely. The vehicle was never classified as disabled, because the RNDF prohibited passing in a one-lane road. Otherwise, motion planning would have easily navigated Odin beyond the phantom object. Fortunately, due to a small amount of GPS drift to the south the forward-spacing enforcer allowed Odin to pass due to an acceptable clearance between Odin, the berm to the right, and the phantom object still in view. The conditions that allowed Odin to pass were similar to conditions experienced in the gauntlet with cars parked on the side of the road. In the end, had localization data been more accurate, Odin may have waited forever.

Figure 19. Report lane position overlay for lane 8.1 of the UCE RNDF where Odin clipped a curb.

5.2.4. Road Detection

The cubic spline interpolation of the RNDF provided Odin with smooth paths to drive that followed the curvature of the roads exceptionally well. Driving directly to way points was not sufficient for navigating the Victorville RNDFs. There were only two instances during the Urban Challenge events when Odin drove over the curbs due to splining issues rather than localization errors. One example is in lane 8.1, which was the entry lane to the parking zones in the UCE RNDF. Figure 19 displays the report lane position output (overlaid on the upper lane). On the basis of the image, the spline follows the roads. However, the curb in the right turn of this s-curve was clipped every time Odin drove down this lane. This issue was due to tight geometry of the lane, the curvature control limitations of the chosen implementation of cubic splines, and human error in checking the RNDF preprocessing.

5.3. Driving Behaviors

During the UCE, driving behaviors performed well, with no major issues. Below are detailed some of the interesting situations Odin encountered in the UCE.

5.3.1. Intersections

Odin handled intersections well in the UCE, according to logs, team observation, and the driver of Odin's chase vehicle. During several merge scenarios, Odin encountered traffic cars or other robots; however, in fewer than 5% of four-way-stop scenarios did Odin have to respect precedence for cars arriving before him. It is interesting to note that on several occasions, Odin observed traffic vehicles and chase cars roll through stop signs without ever stopping. On one occasion, after properly yielding precedence to Little Ben from U-Penn, Odin safely yielded to Little Ben's

chase car, which proceeded out of turn at a stop sign despite arriving at the intersection after Odin.

5.3.2. Passing and Blocked Roads

At no time during the UCE did Odin encounter a disabled vehicle outside of a safety area or blocked road requiring a replan.

5.3.3. Parking Lot Navigation

Odin performed extremely well in the zone navigation/parking portions of the UCE; however, the missions were very easy in comparison to prechallenge testing performed by the team. Whereas each of the three UCE missions contained only one parking spot (and the same one each time), Odin was prepared for much more complicated parking lot navigation. The team had anticipated missions with multiple consecutive parking checkpoints in the same zone but in different rows, requiring intelligent navigation from spot to spot, traveling down the accepted patterns (parking rows). A special strategy was even implemented to navigate parking lots with diagonally oriented spots, traveling down the rows in the correct direction.

Although Odin was overprepared for more complex parking lots, dynamic obstacle avoidance in zones was weak, having seen far less testing. For this reason, the route planner gave zones a higher time penalty.

5.4. Motion Planning

One of Odin's key strengths in performance was smooth motion planning that maintained the maximum speed of a segment or a set global maximum of 10 m/s. The motion planning used on Odin was flexible enough to be used for all situations of the challenge, such as road driving, parking, and obstacle avoidance. By handling the problem of motion planning in a general sense, goals and weightings could be adjusted for new situations, but the core motion planning would ensure that no obstacles would be hit.

Reviewing race logs, there was one situation in which more robust motion planning would have been required during the UCE. This occurred when Odin was traveling east on Montana and was cut off by Stanford's Junior taking a left off of Virginia. Because Odin had the right-of-way, the motion planning algorithm would not slow down until Junior had mostly entered and been classified in Odin's lane. This situation was tested prior to competition, and Odin would have likely stopped without hitting Junior, but it would have been an abrupt braking maneuver. However, before the speed limiter of motion planning engaged, the safety driver paused Odin to prevent what was perceived as an imminent collision. More reliable classification of vehicle orientation and speed would allow motion planning to consider a wider range of traffic obstacles and apply brakes earlier in a similar situation.

6. OVERALL SUCCESSES

All of the teams in the Urban Challenge produced amazing results considering the scope of the project and short timeline. This section provides examples of the characteristics and tools of team VictorTango that increased productivity to accomplish the required tasks to successfully complete the Urban Challenge.

6.1. Base Vehicle Design

The Ford Escape hybrid platform used for Odin proved to be an excellent selection. The vehicle was large enough to accommodate all of the required equipment but was not so large that tight courses posed maneuvering difficulties. The seamless integration of the drive-by-wire system proved to be highly reliable and was capable of responding quickly to motion planning commands. The power available from the hybrid system enabled the vehicle to easily power all systems and to run for more than 18 h continuously. Good design combined with attention to detail in execution produced an autonomous vehicle that offered great performance and suffered very few hardware failures, enabling the team to focus testing time on software development.

6.2. Software Development

This section describes the features and tools used by the team for rapid software development, testing and debugging, and error handling. These attributes helped Odin emerge as a successful competitor in the Urban Challenge.

6.2.1. Software Architecture

During the initial planning and design phases of the project, team VictorTango spent a significant amount of time breaking down the Urban Challenge problem

into specific areas. The resulting subset of problems then became the guiding force in the overall software architecture design. This yielded an extremely modular architecture and also ensured that a clear approach to all the main problems was addressed early on. Evidence of the usefulness of this foresight is the fact that the software architecture is almost exactly the same as it was originally conceived in early January 2007. Slight modifications were made to some of the messages between software components, but in general, the original approach proved to be very successful.

Another benefit to the modular architecture was that it suited team VictorTango's structure very well. The size and scope of each software component could be developed by one or two principal software developers. Along with open communication channels and strong documentation, the team was able to tackle all of the Urban Challenge subproblems in a methodical, efficient manner. Furthermore, by avoiding any sort of large, global solver, or all-encompassing artificial intelligence, team VictorTango's approach provided more flexibility. The team was not pigeon-holed into any one approach and could find the best solution for a variety of problems. Finally, the modular architecture prevented setbacks and unforeseen challenges from having debilitating effects, as they might with less-diverse systems.

Finally, the decision to implement the JAUS for interprocess communications in Odin offered several key advantages. Primarily, JAUS provided a structure for laying out software modules in a peer-to-peer, modular, automatically reconfigurable, and reusable fashion. It provided a framework for interprocess communication inside and across computing nodes and a set of messages and rules for dynamic configuration, message routing, and data serving. Finally, the implementation of JAUS ensured that software developed under the Urban Challenge is reusable in future robotics projects, a critical step in accelerating the progress of unmanned systems.

6.2.2. Simulation

A key element of the software development was a custom simulation environment, shown in Figure 20, presenting Odin with a simulated road blockage. The simulator was used in all phases of software development, including initial development, software validation, hardware-in-the-loop simulation, and even as a data visualization tool during vehicle runs. An easy-to-use simulator allowed software developers to obtain instant feedback on any software changes made, as well as allowing other members of the team to stress test new software before deploying it to the vehicle.

Figure 20. Screenshot of simulated Odin encountering a roadblock.

Testing also involved more stringent validation milestones. Typically the first milestone for a new behavior involved a software validation. Software validations consisted of running software components on the final computer hardware with all perception messages being supplied by the simulator and all motion commands being sent to the simulator. These validations were run by the test team that would provide a set of different scenarios saved as separate scene files. After a successful software validation, some behaviors required hardware-in-the-loop simulation. In these simulations, the software was running on Odin in a test course and was configured to send motion commands to Odin as well as the simulator. The simulator would produce obstacle messages, allowing Odin to be tested against virtual obstacles and traffic, eliminating any chance of a real collision during early testing. Finally, in final validation, the simulator was used as a visualization tool during real-world testing.

6.2.3. Data Log Replay

Instrumental in diagnosing and addressing failures was a data-logging and replay system integrated at the communications level. Called *déjà vu*, the system amounted to logging all input JAUS messages between modules for later playback in near real time.

Figure 21. Odin crosses the finish line.

During diagnostics, the component under analysis operated just as it did on the vehicle, only with the messages replayed from the déjà vu files. Additional features allowed logged data and Odin's position to be visualized in the simulator's three-dimensional view as well.

Déjà vu was critical in solving the problems encountered during NQE. In a typical scenario, the logged messages were played into the software as it ran in source code form, allowing diagnostic breakpoints and probes to be used during playback. Once the problem had been diagnosed and a solution implemented, the new software was verified against the same logged data to verify that the software made the intended decision.

Finally, by integrating déjà vu logging directly into the TORC JAUS Toolkit, it remained independent of the primary software module's functionality. As such, déjà vu is an immediately reusable tool for future use in other projects.

7. CONCLUSIONS

Team VictorTango successfully completed the DARPA Urban Challenge final event, finishing third, as shown in Figure 21. During the competition, Odin was able to drive several hours without human intervention, negotiating stop sign intersections, merging into and across traffic, parking, and maintaining road speeds. Heightening the challenge was a very aggressive development timeline and a loosely defined problem allowing for many unknown situations. These factors made development efficiency as well as testing key components for success.

The aspect of the challenge that gave team VictorTango the most difficulty was environmental sensing. Unreliable sensing at longer distances was a major factor in limiting the vehicle maximum speed to 10 m/s. This speed limit and especially delays due to falsely perceived obstacles added significant time during the final event. The vertical field of view of sensors on the market today is a major limiting factor, especially in laser range finders. New technology such as the IBEO Alasca XT sensors and the Velodyne laser range finder are beginning to address these issues but are relatively new products that are still undergoing significant design modifications. The use of prototype products in a timeline as short as the Urban Challenge introduces risk, as functions may be unreliable or settings may change.

When evaluating the performance and design of vehicles participating in the DARPA Urban Challenge, it is important to consider the short development timeline of 18 months. Owing to funding and organization, team VictorTango had closer to 14 months of actual development time. For example, road detection algorithms using vision were developed, and good results were achieved in certain conditions, but the team felt that the software was not mature enough to handle all the possible cases within

the scope of the Urban Challenge rules. A LIDAR road detection algorithm gave more consistent results over a wider variety of terrain but had limited range requiring a reduction in travel speed. These results are more a product of the development time and number of team members available to work on road sensing, rather than the limitations of the technology itself. With the short timeline, the team chose to use roads defined entirely defined by GPS, causing failures on at least three occasions during the race and NQE.

The Urban Challenge event demonstrated to the world that robot vehicles could interact with other robots and even humans in a complex environment. Team VictorTango has already received feedback from industry as well as military groups wanting to apply the technology developed in the Urban Challenge to their fields immediately.

ACKNOWLEDGMENTS

This work was supported and made possible by DARPA track A funding and by the generous support of Caterpillar, Inc., and Ford Motor Co. We also thank National Instruments, NovaAtel, Omnistar, Black Box Corporation, Tripp Lite, IBEO, Kairos Autonomi, and Ultramotion for sponsorship or other support.

REFERENCES

Cacciola, S. J (2007). Fusion of laser range-finding and computer vision data for traffic detection by autonomous vehicles. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Currier, P. N. (2008). Development of an automotive ground vehicle platform for autonomous urban operations. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Duda, R. O., & P. E. Hart (1972). Use of the Hough transform to detect lines and curves in pictures. Communications of the ACM, Vol 15(1), 11–15.

Eren, H., Fung, C. C., & Evans, J. (1999). Implementation of the spline method for mobile robot path control. In V. Piuri & M. Savino (Eds.), Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference, Venice, Italy (volume 2, pp. 739–744). IEEE.

Fuerstenberg, K. C., Dietmayer, K. C. J., & Lages, U. (2003). Laserscanner innovations for detection of obstacles and road. In Proceedings of 7th International Conference on Advanced Microsystems for Automotive Applications, Berlin, Germany.

Fuerstenberg, K. C., Linzmeier, D. T., & Dietmayer, K. C. J. (2003). Pedestrian recognition and tracking of vehicles using a vehicle based multilater laserscanner. In Proceedings of 10th World Congress on Intelligent Transport Systems, Madrid, Spain.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Transactions on Systems Science and Cybernetics, SSC4(2), 100–107.

Hurdus, J. G. (2008). A portable approach to high-level behavioral programming for complex autonomous robot applications. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Kelly, A. J. (1994). A 3D state space formulation of a navigation Kalman filter for autonomous vehicles (Tech. Rep. CMU-RI-TR-94-19). Pittsburgh, PA: Robotics Institute, Carnegie Mellon University.

Konolige, K., & Myers, K. (1998). The Saphira architecture for autonomous mobile robots. In D. Kortenkamp, R. Bonasson, & R. Murphy (Eds.), Artificial intelligence and mobile robots. Cambridge, MA: MIT Press.

Lacaze, A., Moscovitz, Y., DeClaris, N., & Murphy, K. (1998). Path planning for autonomous vehicles driving over rough terrain. Proceedings of the ISIC/CIRA/ISAS Conference, Gaithersburg, MD, September 14–17, 1998.

Maes, P. (1989). How to do the right thing (Tech. Rep. NE 43–836). Cambridge, MA: AI Laboratory, MIT.

Milliken, W. F., & Milliken, D. L. (1995). Race car vehicle dynamics. Warrendale, PA: SAE International.

Murphy, R. R. (2000). Introduction to AI robotics. Cambridge, MA: MIT Press.

Pirjanian, P. (2000). Multiple objective behavior-based control. Robotics and Autonomous Systems, 31(1), 53–60.

Pirjanian, P. (1999). Behavior coordination mechanisms—State-of-the-art (Tech. Rep. IRIS-99-375). Los Angeles, CA: Institute for Robotics and Intelligent Systems, University of Southern California.

Rosenblatt, J. (1995). DAMN: A distributed architecture for mobile navigation. In AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents, Stanford, CA. Menlo Park, CA: AAAI Press.

Russel, S., & Norvig, P. (2003). Artificial intelligence—A modern approach. Upper Saddle River, NJ: Pearson Education, Inc.

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., 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, Vol. 23(9), 661–692.

Urmson C., Ragusa, C., Ray, D., Anhalt, J., Bartz, D., Galatali, T., Gutierrez, A., Johnston, J., Harbaugh, S., Kato, H., Messner, W., Miller, N., Peterson, K., Smith, B., Snider, J., Spiker, S., Ziglar, J., Whittaker, W., Clark, M., Koon, P., Mosher, A., & Struble, J. (2006). A robust approach to high-speed navigation for unrehearsed desert terrain. Journal of Field Robotics, Vol. 23(8), 467–508.