DARPA Urban Challenge Technical Report

Austin Robot Technology

<u>Team Leader:</u>
Dave Tuttle {dave@austinrobot.com}

<u>Project Management:</u>
Dave Tuttle

<u>University of Texas at Austin Contact:</u>
Prof. Peter Stone {pstone@cs.utexas.edu}

<u>Tech Report Authors:</u>
Peter Stone
Patrick Beeson {pbeeson@cs.utexas.edu}
Tekin Meriçli {tmericli@cs.utexas.edu}
Ryan Madigan {r_madigan@hotmail.com}
with input from many other team members

6/1/2007

Executive Summary

Austin Robot Technology's (ART's) entry in the DARPA Urban Challenge has two main goals. First and foremost, the team aims to create a fully autonomous vehicle that is capable of safely and robustly meeting all of the criteria laid out in the Technical Evaluation Criteria document [1]. Second, and almost as important, the team aims to produce, educate, and train members of the next generation of computer science and robotics researchers. This technical report documents our significant progress towards both of these goals as of May 2007 and presents a concrete plan to achieve them both fully by the time of the National Qualifying Event (NQE) in October. Specifically, it presents details of both our complete hardware system and our in-progress software, including design rationale, preliminary results, and future plans towards meeting the challenge. In addition, it provides details of the significant undergraduate research component of our efforts and emphasizes the educational value of the project.

1 Introduction

The starting point for ART's entry in the Urban Challenge is the same vehicle that was designed and engineered as a part of the 2005 DARPA Grand Challenge. The vehicle qualified for that year's NQE and continues to meet all of the basic safety requirements that are uncompromisingly essential for participation in the Urban Challenge, including a purely mechanical E-stop that, with no software in the loop, is able to bring the car to a safe and quick stop. The main hardware enhancements since 2005 are 1) an Applanix POS-LV inertial navigation unit[1] that provides sub-meter GPS accuracy, and 2) a Velodyne HDL-64E high-density lidar sensor[2] that provides a 360° high-density point cloud of range data at a frequency of 10 Hz. Other key modifications include the introduction of the ability to shift into and drive in reverse, the ability to control the turn signals from software, improved braking controls, and the repositioning of the existing SICK lidars and visual sensors so as to provide close-range sensing and road marking information that complements the information provided by the Velodyne. As fully described in Section 3.1, we believe that the vehicle *hardware* is already fully capable and robust enough to support successful completion of the Urban Challenge.

As such, ART's main focus this year has been, and continues to be, on the software required to interpret the rich and voluminous sensor data and to safely, accurately, and robustly control the vehicle in an urban environment. This software is being developed in close partnership with The University of Texas at Austin (UT Austin) as a part of the Freshman Research Initiative (FRI) within the College of Natural Sciences (CNS). The aim of the FRI is to provide undergraduate students, including but not limited to freshmen, direct hands-on exposure to science research during their undergraduate careers. FRI is supported by the National Science Foundation (NSF). As a part of this program, CNS supported a spring 2007 class taught by Professor Peter Stone that was devoted entirely towards developing software for ART's Urban Challenge entry.[3] The course included 25 undergraduate students ranging from freshmen to seniors with widely varying backgrounds. In addition to approving this course as Professor Stone's primary teaching responsibility for the semester, CNS is supporting a senior graduate research assistant (GRA) – Patrick Beeson – to help manage and oversee the ongoing undergraduate research. CNS has also contributed the crucial new Applanix and Velodyne hardware to the vehicle. This deep institutional commitment has enabled the team to place a heavy focus on education and training

[1] Applanix POS LV: http://www.applanix.com/products/poslv_index.php

[2] Velodyne HD Lidar: http://www.velodyne.com/lidar/home.html

[3] Course website: http://www.cs.utexas.edu/~pstone/Courses/378spring07/

while also moving steadily forward towards the primary goal of meeting the challenge. Indeed, the control software that drove the car during the autonomous run shown in ART's qualification video was written by undergraduates as part of this course.

The two main problems we are trying to solve for the Urban Challenge are 1) the interpretation and integration of high-density sensory data towards a complete, real-time model of the world that includes an awareness of local terrain and any obstacles or other vehicles in the vicinity and 2) precise, safe control that takes into account this world model, including reaction to observed and predicted behaviors of other vehicles. In pursuit of these two challenges, the three key themes that drive our design choices, as well as central lessons in the UT Austin class, are as follows:

1. *Safety first*. All software needs be designed in such a way that the worst case scenario is mission failure. If there is any doubt regarding the safety of a maneuver, then it should not be undertaken. For example, this criterion indicates that in sensing obstacles, to the extent that trade-offs are necessary, false positives are preferred to false negatives: the vehicle must never fail to recognize an obstacle in its vicinity. Sensing "phantom" obstacles may degrade performance but will not compromise safety. With the maintenance of no false negatives as an invariant, the incidence of false positives should be reduced as much as possible.
2. *Controlled complexity.* While it is important to be fully aware of and knowledgeable about state-of-the-art robotics research, for any given problem the simplest possible solution should be tried first. Given the current state of the art and the time scale involved, we view the Urban Challenge as mainly an integration challenge. That is, most of the technologies to succeed already exist. Thus the research to be done is not in the development of completely new algorithms and approaches, but rather in tuning and integrating existing algorithms. For the sake of robustness and transparency, straightforward solutions are highly preferable.
3. *Frequent and incremental testing.* To succeed in the Urban Challenge it is not sufficient to successfully navigate an urban environment once. Rather, the car must be able to safely complete every mission it is given, no matter what the weather conditions or the behaviors of the other vehicles in the environment. As such, every individual software component must be tested, first in simulation but also in the real world, in as many possible conditions as possible, and every change to a component needs to be tested in the context of the complete system to ensure that it doesn't adversely affect other interacting software components.

   Though we do not explicitly refer to it again in this document, this theme is pervasive throughout our entire design and development process. We regularly test on long straight-aways and parking lots on the UT Austin Pickle Research Campus (PRC)[4], on curvy roadways at Driveway Austin[5] (see Figure 7), and occasionally at the Southwest Research Institute (SwRI) site visit location[6] in San Antonio.

The remainder of this document is organized as follows. Section 2 provides an overview of the main components of our system and team organization, beginning with the system architecture and including the team composition and further details about our approach to education and training. Section 3 presents and analyzes our system design, focusing first on hardware and then dividing the software into the sensing, modeling, and control components. Section 4 includes our performance results to date, and Section 5 concludes.

[4] PRC satellite photo: http://www.wikimapia.org/maps?ll=30.3866,-97.7269&spn=0.008592,0.005932&t=k
[5] Driveway Austin Motorsports Academy and Retreat: http://www.drivewayaustin.com
[6] SwRI site visit: http://urban.challenge07.googlepages.com/sitevisitcourse

2 Overview

This section begins with an overview of our system architecture. Full details of its various components are provided in Section 3. We then present an overview of our team's composition with emphasis on the key areas of expertise that are needed to succeed in the Urban Challenge, and elaborate on the ways in which we are achieving our goal of educating and training future computer scientists.

2.1 System Architecture Overview

An abstract view of the system architecture consists of the following four components, as illustrated in Figure 1.

1. The vehicle **hardware**, shown in blue, includes an electro-mechanical E-Stop; actuators for steering, throttle, shifting, signaling, and braking; computing hardware; and sensing hardware.
2. The **sensing** subsystem, shown in purple, takes raw sensor data as input and interprets it into a form that can be used for world modeling. This module includes coordinate transforms into a single frame of reference, ground plane removal, and other related pre-processing steps.
3. The **world model**, shown in orange, merges the fused sensory data into a single, coherent representation of the state of the vehicle and its environment.
4. The **control software**, in green, itself takes a common layered approach, being divided into a pilot that is in charge of low-level actuator commands, a navigator that is in charge of local obstacle avoidance and lane following, and a commander that is in charge of high-level planning based on the assigned mission.

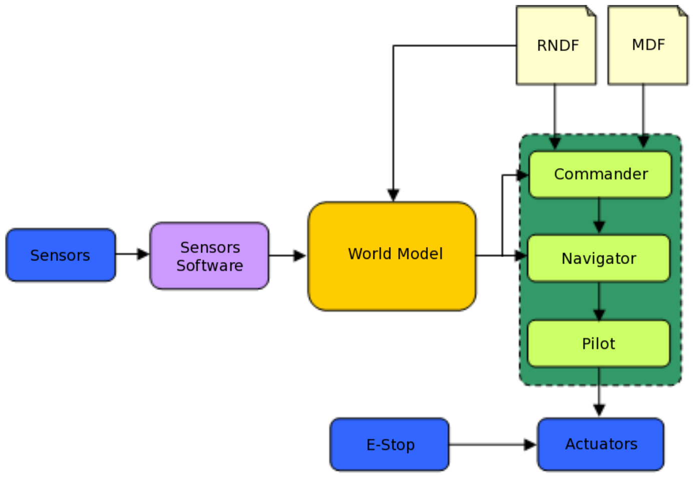


Figure 1 - System architecture overview.

Though fairly generic at this level of abstraction, a key feature of this architecture that aligns with our “safety first” design principle is that the E-stop is completely independent from the software. That is, no error in the software can affect the E-stop. Further detail on each of the above four components is presented in detail in Section 3.

2.2 Team Composition

In order to complete an integration project as complex as creating an autonomous vehicle, expertise in a wide variety of areas is needed. Specifically, every effective team needs deep expertise in mechanical engineering, electrical engineering, computer science/robotics, software development, and project management. In this section, we summarize the qualifications of some of the key personnel on the ART team with specific attention to how they fill the above needs.

Mechanical engineering: A prerequisite for any successful Urban Challenge entry is a vehicle that has been engineered mechanically to be robust and failsafe. Our base vehicle has been created largely by Juan Martin-de-Nicolas who brings to the team over 20 years of experience working with machinery and mechanical systems. For ART's entry in the October 2005 DARPA Grand Challenge NQE, Juan was responsible for all mechanical work done to convert a passenger vehicle into a competitive autonomous robot. Juan continues to maintain the vehicle hardware and make any necessary modification as new sensory systems become active.

Electrical engineering: As a close counterpart to the vehicle's mechanical systems, its electrical system also requires significant engineering and maintenance. Our electrical engineering is led by Don McCauley, who has over 25 years of experience designing hardware systems architecture and microprocessors for IBM, Intel, and AMD. Don has led the effort to re-engineer all our vehicle's sensors and actuators for the Urban Challenge. This effort dramatically improved the reliability and fault tolerance of all onboard systems. Skilled in interfacing analog and digital systems, Don designed most of the analog and digital filters and custom hardware interfaces required within our vehicle. Don also designed and integrated the various computer systems, micro-controllers and networks used within our vehicle.

Computer Science and Robotics: UT Austin contributes world class expertise in computer science and robotics via the leadership of Professor Peter Stone and GRA Patrick Beeson. Professor Stone, recent winner of the prestigious IJCAI Computers and Thought Award, brings with him 13 years of experience in CS robotics, with specific focus on machine learning and multirobot systems. In the course of developing several championship RoboCup robot soccer teams, he has contributed novel algorithms for sensor/actuator calibration [2], robot vision [3-5], localization [6], agent modeling for prediction of other agents' behaviors [7], reasoning under uncertainty [8], and multiagent reasoning [9], all of which are directly applicable to this project. At least as significantly as his technical algorithmic contributions, Professor Stone has extensive experience fielding complete, robust working multirobot systems, as is necessary for successful participation in international research competitions such as RoboCup and the Trading Agent Competition [7, 10-14]. This competition expertise will be invaluable as we prepare to field a complete, robust working system in the Urban Challenge.

Patrick Beeson is working closely with Professor Stone on this project with regards to software development, sensor calibration, and education of the students involved in the project. Patrick is within a few months of completing his Ph.D. on topological navigation to facilitate human-robot interaction.[7] His published research contributions are directly relevant to this project [15-21].

Software development: With such a large team of contributors at various levels of expertise, it is essential that our team use software development best practices in order to succeed. Jack O'Quin brings to the project 30 years of experience in contributing to development of large, complex software systems. He worked for IBM and the T.J. Watson Research Center leading development of, among other things, AIX (IBM's version of UNIX) and the microprocessor architecture for the PowerPC. Since retirement, he has helped pioneer open-source, real-time audio applications for Linux. Jack has been a member of ART since 2004, and developed much of the software for our 2005 DARPA Grand Challenge entry. His contributions include: selecting

[7] Patrick is delaying his thesis defense until after the competition so as to be able to participate fully in this project.

and deploying programming tools for build and change control; packaging stable versions of all external software dependencies, including the Linux operating system; improvements in on-board serial device error handling; PID control for braking and throttle to achieve requested vehicle speeds; and much vehicle integration testing and debugging.

Project management: In addition to software development expertise, the size and scope of this project requires expert project management. Dave Tuttle is a technical and management consultant based in Austin, team leader of the ART team, and a team member of the UT Challenge-X hybrid vehicle project. He started Sun Microsystems' Austin Microprocessor Design Center from scratch. Over a 5 year period he built one of Sun's most effective CPU design teams. He was one of the key designers and project leaders of the teams which launched the IBM Power1, Apple PowerMac, the "Deep Blue" chess playing supercomputer, and two generations of the world's fastest supercomputers for the ASCI projects.

In addition to the key personnel listed above, the team includes several other past and/or part-time contributors from ART and dozens of students from UT Austin.

2.3 Education and Training

As noted in Section 1, one of the two central goals of ART's Urban Challenge entry is to produce, educate, and train members of the next generation of computer science and robotics researchers. The currently declining enrollments in computer science undergraduate programs across the country is a potential crisis for the nation's future IT industry, as well as for government research agencies such as DARPA. Inspiring challenge problems may be essential drivers towards reversing this enrollment trend. As such, we believe strongly that the long-term impact of competitions such as the Urban Challenge will be as much in education and training as it will be in technological innovation.

Acting on this belief, our team has already used the project as a tool for educating 25 UT Austin undergraduate students about the world of computer science research and continues to work directly with these students towards successful completion of the Urban Challenge. With full participation and cooperation from ART members, Professor Stone designed and taught a spring semester course called "CS 378 -- Autonomous Vehicles: Driving in Traffic."

As a part of the Freshman Research Initiative described in Section 1, Professor Stone structured the class as a single, unified software development project. Rather than being assigned papers, the students read and summarized research papers of their own choosing (at least two per week), chose component subprojects in teams according to their own interests and expertise, and used class sessions to discuss their progress and brainstorm future directions. In contrast to more standard courses, they worked collaboratively rather than individually; read and evaluated cutting edge research rather than reading pre-digested textbook information; worked on a single, large, open-ended project rather than a series of constrained programming assignments; and presented their work to the class rather than listening to lectures.

The students were presented with this contrast as representative of the difference between undergraduate and graduate education with the goal of encouraging them to explore the possibility of proceeding to graduate school and eventual careers in research. Feedback from class surveys indicates that indeed several of the class members are newly interested in pursuing such a path, and one class member used his contribution to the project as the basis for his undergraduate honor's thesis. Although the class itself ended at the beginning of May 2007, almost half of the students are continuing on with the project in some capacity over the summer and, should we qualify for the NQE, into the fall of 2007.

From the perspective of education, our project has already been a great success. Students in the course have directly contributed to many aspects of the autonomous vehicle. They have helped make hardware modifications such as enabling software control of the turn signals. They have made needed extensions to the Stage simulator for more realistic offline testing. They have developed low-level software control devices for braking and speed control. They have developed prototype vision algorithms for stop-line and lane detection. They have developed parsers and visualization tools for RNDF and MDF files. They have developed capabilities to generate and follow smooth curves for path-planning. They have developed drivers for the Velodyne and our other sensors. And they have developed the complete high-level planner for vehicle control and obstacle avoidance that controlled the car during the autonomous run shown in ART's qualification video. This last development achieved the explicit goal of the course, which was to collaboratively create software with the capability of passing the video qualification and to thereby make significant progress towards the capabilities needed for the site visit.

Nonetheless, the project is still very much a work in progress and the students are eager to see it through to completion. As noted above, several of them are continuing to contribute to the project, many on their own time, and plan to continue through the site visit. In fact, we plan to allow students from the class to handle most of the duties during the actual June site visit. Should we qualify for the NQE, they will be energized to continue their work through October and will get to experience the ultimate satisfaction of creating a complete working autonomous vehicle. As such, qualifying for the NQE would dramatically increase the educational impact of the project.

3 Analysis and Design

This section expands in detail on the four system components presented in Section 2.1, namely the hardware, sensing, world modeling, and control, in more detail. As appropriate, we sketch the reasoning behind the design decisions and analyze their effectiveness. More detailed testing and evaluation of some of the key system components is presented in Section 4.

3.1 Hardware

Our base is the same vehicle that participated at the National Qualifying Event of the 2005 DARPA Grand Challenge. Figure 2 includes a recent picture of our vehicle, a 1999 Isuzu VehiCROSS *"Ironman"* edition which is described in detail in our 2005 technical paper.[8] Our complete hardware system upgrades our 2005 entry to handle urban environments, which require higher precision and more robustness. Most significantly, we have augmented the vehicle to include a high precision inertial navigation system that provides sub-meter localization accuracy and a high-density lidar that provides 3D range data at a frequency of 10 Hz. Both of these sensor additions are described in detail below after a brief overview of our existing E-stop system and computing hardware. The overall hardware system is illustrated in Figure 2.

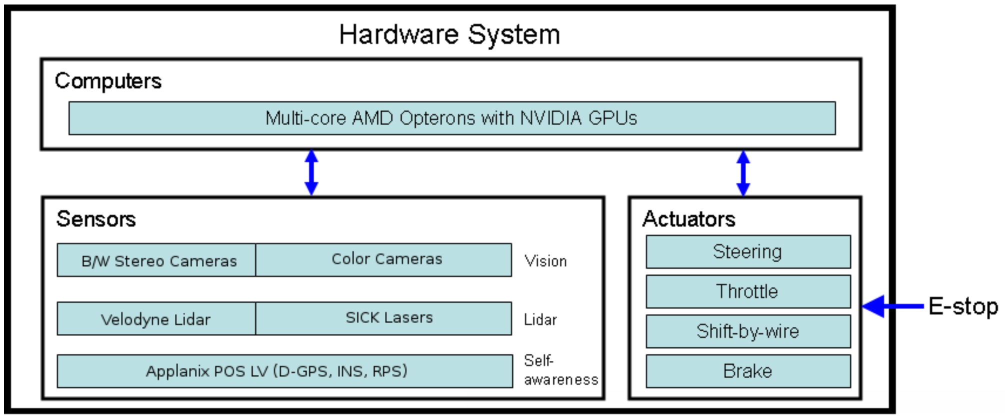
3.1.1 Actuators Hardware and E-Stop

Our actuators control the vehicle's steering, throttle, shifting, signaling, and braking. Steering, throttle, and braking all were in place and performed well during the 2005 Grand Challenge

[8] ART's 2005 Tech Report: http://www.darpa.mil/grandchallenge05/TechPapers/Austin_Robot_Technology.pdf

NQE. Since then, we have installed a custom shift-by-wire system to control forward, park, and reverse, and we have installed relays to do signaling. By monitoring the vehicle brake pressure, we now have a much more precise braking system than in 2005. All actuators have performed extremely reliably throughout the spring semester, creating a stable platform for the undergraduate class to test their software.

Our highly reliable electro-mechanic reactive E-Stop system is connected to the brake and the throttle, providing instant response to a disable signal with no software in the loop so that it works even if our vehicle loses power. The disable signal cuts current to a relay, which in turn activates the brake in full, disengages the throttle, and simultaneously interrupts the ignition circuit. The main idea behind the braking component is that a brake cable is connected to an engaged electromagnet pulling against tension coils. When the circuit to the electromagnet is broken, the tension coils pull the brake cable, causing the brake to engage fully.



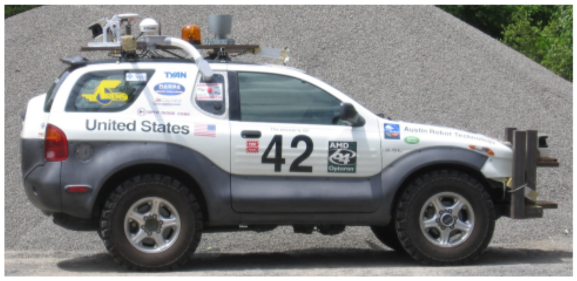


Figure 2 - Hardware overview.

3.1.2 Computing Hardware

Our computing hardware is a mobile TeraFLOPS supercomputer. It consists of a pair of computers to be used for vision and a third computer used for sensor fusion, world modeling, navigation, and planning. Each of the computer systems uses two shock-mounted SATA hard disk drives. The computers are connected with dual Gbps Ethernet links. The vision computers connect to the cameras with dual IEEE-1394a PCI cards and the other computers use multi-port RS-232 PCI cards to connect other sensors and actuators with RS-232, RS-422, RS-485, and USB 2.0 connections. These computer systems were designed for redundancy and high reliability. Parts — and even entire systems — can be swapped, if necessary.

Though our current system does not rely heavily on vision, we expect to focus on improving and extending vision capabilities towards the semifinals and finals, as described in Section 3.3.3. Thus the vision system is configured for maximum speed and reliability. Each vision system includes two low-power dual-core AMD Opteron 64-bit microprocessors (4 cores total) and two 16-lane PCI-Express based NVIDIA 7900 GT Graphics Processing Units (GPUs) running stereo vision software. Each of the other computers includes two single-core Opteron processors (2 cores total) with the same GPUs for visual display and debugging. Each of the NVIDIA GPUs is capable of approximately 250 GFlops. Combining 8 GPUs with 12 Opteron cores gives our vehicle well in excess of 2 TeraFLOPS of processing power.

3.1.3 Sensor Hardware

For navigation and maneuvering, two of the most crucial pieces of information needed by the vehicle are its current location and velocity. We use the POS LV system from Applanix, a high quality inertial navigation system (INS) coupled with GPS [22], to provide reliable location, heading, and speed information even during GPS outages. To improve our dead-reckoning ability and provide redundancy for the INS, our vehicle is fitted with a Sauer-Danfoss Hall effect rotary position sensor (RPS), which provides a very accurate independent confirmation of the steering angle and the wheel rotations.

For local sensing of obstacles and other vehicles, we rely most heavily on lidar sensors due to their reliability, precision, and the fact that they are not greatly affected by lighting conditions. However, we see vision as a necessary component of the system for its ability to recognize road markings that cannot be picked up by lidars. To the extent that vision is also able to provide information about obstacles and other vehicles, we will use it as a redundant information source to verify and calibrate information from lidar.

Because we believe high-density sensors will be vital to winning the 2007 DARPA Urban Challenge, we have invested in a Velodyne HDL-64E rotating high-density lidar, which gives a 360° real-time view of the environment around the vehicle, and thus is a perfect choice for this purpose. We also use low-facing SICK brand lidars to cover blind spots of the Velodyne lidar, to provide reactive collision avoidance for immediate threats, and to look for curbs and other geometric lane boundaries visible to lidars. As detailed in Section 3.3.2.2, bringing the Velodyne sensor online has been an extensive focus of our efforts during the spring of 2007. It is now operational and will provide the bulk of our local sensory information during the site visit.

At this writing, our vision hardware, along with prototyped software, is in place, but we do not use vision yet for vehicle control. Our vision system consists of a set of four stereo cameras and additional short-range color cameras. The housings for the left two cameras of our four-camera stereo rig are visible in Figure 2. These high resolution (1280x960) cameras from Sony deliver quality images through a Firewire (IEEE-1394) interface to our multi-processor computing systems, where dedicated GPUs process the data in real time and output depth information similar to the data from lidars. The stereo cameras are black/white and front-facing. We will mount multiple short-range low-quality color cameras around the vehicle to fill gaps in close-range information. They will be used to recognize road markings and other vehicles' signals and to cover blind spots of the stereo cameras.

3.2 Robot Control Interface

We utilize the Player robot server[9] as our hardware control interface. In doing so, we get well-defined interfaces of commonly used robot data types, a distributed object system, message passing, thread handling, and seamless integration with the Stage 2D simulation backend. All

[9] Player project: http://playerstage.sourceforge.net/

software modules, except for the Commander module, are written as Player drivers, which allow for clients written in several languages to connect to the drivers for testing. This platform has been invaluable for allowing students to make progress in the class, as they can focus almost entirely on the problems of programming the vehicle, while issues like threading and message passing are mostly handled "behind the scene" by Player.

Figure 3 illustrates the current robot architecture as Player drivers. This is a more detailed explanation of the overall software design that combines elements of Figures 1 and 2.

3.3 Sensing Modules

The sensing subsystem interfaces individual sensors. The sensing modules correspond to the three main types of sensing on our vehicle – global positioning (localization), lidar, and vision.

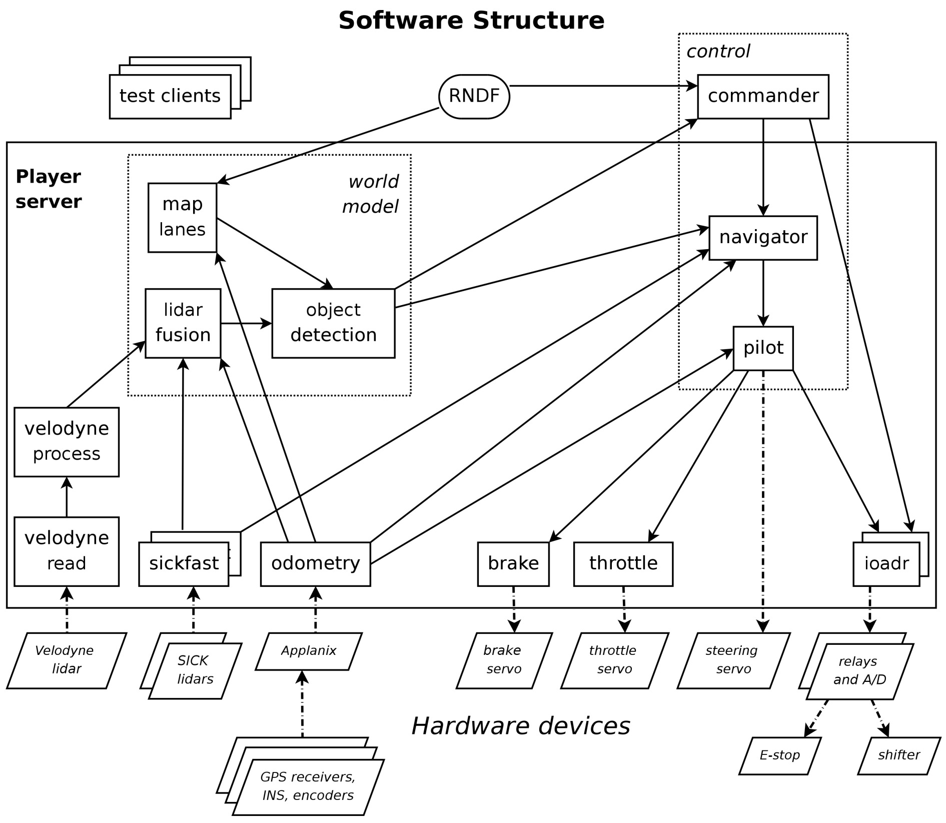


Figure 3 - Detailed system architecture, as Player drivers/clients. Note the left side of the figure denotes sensor fusion, while the right side lays out safe control.

3.3.1 Odometry

The Applanix POS LV sensor provides continuous position, heading, and speed information for navigation and control. The Applanix sensor provides this sub-meter position and orientation in the Lat/Long coordinate system used by the RNDF. For ease of development, the ***odometry*** module transforms this data from Lat/Long coordinates to a Cartesian (x,y,θ) coordinate system whose origin is centered on the vehicle's rear axle at the start of a mission. This Cartesian coordinate system is used by the *lidar fusion* module (Section 3.4.1) and the *map lanes* module (Section 3.4.2) to produce a world model.

The *odometry* module is also responsible for converting the Applanix velocities (in a Lat/Long coordinate system) into translational and rotational velocities (m/s). This information is used by both *Navigator* and *Pilot* (see Section 3.5) for reactive speed control.

3.3.2 Lidars

3.3.2.1 SICK Lidars

Our use of the ***SICK lidar*** units is relatively straightforward. We have mounted our units upside down to reduce sunlight interference by keeping the rotating mirror inside the unit angled away from the sky. We have had problems with the front SICK frequently overheating due to the Texas climate combined with the heat coming off the engine block. We have ordered thermal coolers that, once installed, will keep the lidar below 100° Fahrenheit. Our *sickfast* player driver improves on the standard Player driver by supporting the SICK S14 FAST and by setting the device to run in "high availability mode", which eliminates most laser shutdown conditions.

3.3.2.2 Velodyne Lidar

The ***Velodyne lidar***, which is newer technology that arose from Team DAD's 2005 Grand Challenge entry,[10] has been more challenging to use effectively. Figure 6(left) illustrates the 3D point cloud provided by the Velodyne unit, which provides information between +2° and -25° from the horizontal plane. Much work was needed to obtain these high quality results. Specifically, our unit arrived lacking calibration and returned drastically incorrect distances, adding 2.6 cm to every meter of real world distance, so that an object 40 meters away would show up as 41.04 meters away (not including the additive distance errors discussed below).

Unfortunately, this small distance-dependent error was not the only source of error in the unit. The device itself contains 64 individual lasers, each of which has a large constant error that is initially unknown and apparently unique to each unit. Calibrating these was straightforward once the distance-dependent error discussed above was discovered and solved for. To calibrate the unit, we first took several indoor data sets and focused on the few lasers that have pitch near 0°. Given that our data sets had unique obstacles that were a known distance away (e.g. a piece of plywood held vertically in a long corridor), we found the offsets to subtract from each of these near horizontal lasers in order to obtain the correct distance information.

Once we had several lasers calibrated, we then took several sets of data logs with the unit mounted at +45° and aimed at tall, flat buildings on the UT Austin campus. By using the few calibrated lasers to determine the ground truth distances of these buildings, we were able to find the distance offsets needed for the rest of the 64 lasers. Most of these constant distance errors are between .3 and .5 meters - quite a significant distance from the perspective of world modeling.

To take advantage of the calibrated Velodyne sensor and following our design principle of trying the simplest algorithms first, we use "height-difference" maps to identify vertical surfaces in the environment without the need for cutting-edge algorithms for 3D, real-time modeling [23-24]. We take the 3D Velodyne data, and at each cycle (i.e. every complete set of 360° data), we create a 2½D "height-difference" map. Our solution can be thought of as a "slimmed down," thus computationally efficient, version of the *terrain labeling* method performed by the 2005 Grand Challenge by the Stanley team [25].

[10] Team DAD's 2005 tech report: http://www.darpa.mil/grandchallenge05/TechPapers/TeamDAD.pdf

In our solution, we have a Cartesian grid (similar to the occupancy grid in Section 3.4.1) that is populated, analyzed, and then cleared at each processing cycle. Instead of each cell modeling the probability of occupancy as in our *laser fusion* module, each cell tracks the max and min Z value (height) of lidar scans that fall into the cell for the current set of 360° range data. After all range data for a revolution is added to the occupancy grid, a *simulated lidar scan* (described in detail in Section 3.4.1) is produced from the grid – the algorithm casts rays from the Velodyne origin, and an obstacle is "detected" whenever the difference between the max and min Z values is above a threshold. The result is a 360° 2D simulated lidar scan, which looks very similar to the data output by the SICK lidar devices. However, this lidar only returns the distances to the closest obstacles that have some predetermined vertical extent.

This simplified approach lends itself to speedy operation, allowing us to efficiently process the two million points per second produced by the Velodyne. Each of these points is transferred from the device in a UDP packet. The act of capturing and processing these packets, for reasons described in Section 4.2, is split into two modules. The 'reading' module serves only to capture data packets from the Velodyne, while the 'processing' module performs the actions necessary to transform the Velodyne output into useful information.

3.3.3 Vision

Figure 3 omits any reference to vision because we currently do not use vision information in the vehicle's navigation. Instead we rely on lidars and on the implied lane boundaries extracted from the RNDF waypoints (see Section 3.4.2 for details). Upon completion of the vision modules explained below, the ***World Model*** module (Section 3.4) will subscribe to the visual sensors and incorporate visual data, along with RNDF, lidar, and odometry, to improve our current model of the vehicle's local surroundings.

The ***Stereo Vision*** module has been in development since the 2005 Grand Challenge NQE. Once operational, it will complement the Velodyne lidar output by providing texture information (such as road markings) in addition to range data. Using GPUs to process stereo vision faster than using CPUs, the output of vision is an orthographic projection of the road ahead of the vehicle, viewed from overhead. Lane markings, obstacles, and vehicles appear in a Euclidean reconstruction that preserves relative distances. The stereo vision outputs depth information, similar to that from lidar, to ease fusing of sensing data. Figure 4 shows an example of the stereo vision: the left image is obtained by superimposing left and right camera data; the right image shows a recovered bird's-eye view of the scene. Note that the computed positions of double lines, the traffic cones, and the trash can correspond to their real-world positions.

The ***Short-Range Color Vision*** module is a new addition to the system, intended to recognize road markings and other vehicles in traffic. For the Urban Challenge, the vehicle needs to be careful to recognize and stop at stop lines and to stay in its lane. It is also important that the vehicle be able to recognize other vehicles in traffic as well as their intentions, including from their past trajectories and from their signals (braking, backing up, and turning) when available. Unlike the stereo vision, this module uses color cues for these purposes. Distance information for objects of known dimensions is recovered based on sizes of monochromatic colored objects. This module is designed with special-purpose shape and color heuristics specialized to recognize each expected type of road mark or signal. Such a design requires low computation overhead and is appropriate for the auxiliary roles played by these sensors. This work was initiated as a class project. Preliminary results for color-based road detection are shown in Figure 5.

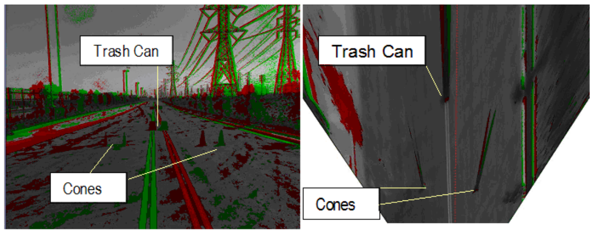


Figure 4 - Stereo Vision.

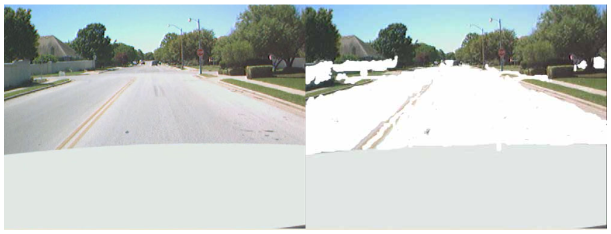


Figure 5 - Color-based road surface detection.

3.4 World Model

As introduced in Section 1, we believe that one of the two fundamental problems to be solved as a part of the Urban Challenge is the interpretation and integration of high-density sensory data towards a complete, real-time model of the world that includes an awareness of local terrain and any obstacles or other vehicles in the vicinity. Our current software does this using a combination of ***lidar fusion*** that provides distances to the closest obstacles seen by any lidar in the recent past, ***map lanes*** that provides a predictive representation of lane locations generated from the input RNDF, and ***object recognition*** that fits obstacles that fall into lanes to rectangular models and tracks each distinct object over time.

3.4.1 Lidar Fusion

The ***lidar fusion*** module is responsible for integrating information from the Velodyne and the SICK lidars. Currently, this is done as an extension to the commonly-used occupancy grid structure [26-27] that allows noise to be handled by accruing evidence of obstacles over time and allows blind spots to be handled by remembering information in portions of the environment not immediately visible to the vehicle. Figure 6(right) illustrates a short-term occupancy grid created using the raw Velodyne lidar illustrated by Figure 6(left).

To produce its output, this module pretends that a lidar unit is located at the origin of the vehicle (center of the rear axle), and it then projects a simulated laser ray from the vehicle into the grid for each degree of rotation around the vehicle's location. This simulated lidar works like a real 360° laser range finder, returning the distance to the nearest obstacle; however, because the grid is created by multiple lidar devices, the output of this *simulated lidar scan* is a single representation that returns the closest obstacles seen by any of the physical lasers in the recent past.

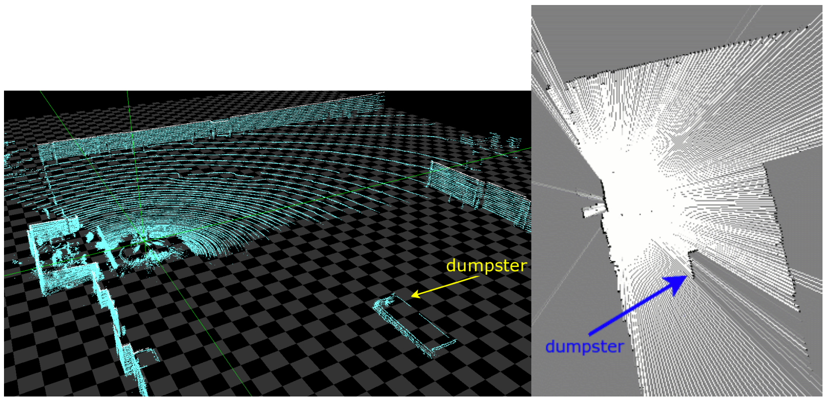


Figure 6 - Left: Velodyne lidar snapshot at SW corner of SwRI track. Right: Lidar fusion grid from Velodyne lidar information.

As a result, a nearby obstacle detected by the front SICK lidar, even if not detected by the Velodyne, is nevertheless included in the simulated range scan, while obstacles behind (in the same direction, within ±0.5°) are not represented. We have introduced logic to prevent the "free" evidence of one lidar unit from overwriting the "occupied" evidence of another lidar since, as mentioned in Section 1, we prefer false positives to false negatives for safety reasons.

In addition to filtering noise and keeping around state, we plan to extend our current implementation to also provide height estimates of the obstacles by utilizing the 3D information provided by the Velodyne lidar unit. With this information we can estimate both the location and height of obstacles, potentially allowing us to tell the difference between cones (small obstacles) and the concrete posts (large obstacles), both present at the SwRI track.

3.4.2 Map Lanes

Initially conceived as a temporary substitute for visual lane recognition and stop line detection, the ***Map Lanes*** module has become an important piece of our current software infrastructure. Map Lanes is designed to parse an RNDF and to create a lane map in the Cartesian coordinate system that the odometry provides (Figure 7). Its purposes are both to create a path for the car to follow in the absence of visual lane detection and to distinguish areas in the environment where objects detected by the sensors are relevant (within the lanes of the road where the car intends to drive) from those areas where objects should be ignored (trees or buildings outside of the lanes), no matter how close they are to the vehicle.

For relatively straight segments, this is simple line fitting with the lane widths defined by the RNDF. Map Lanes estimates the locations of lane boundaries along curves by interpolation of the location and heading. If RNDF waypoints are located exactly at the beginning and ends of the curves, these hypothesized lanes line up with the real world lanes; however, in some scenarios, particularly when the lanes swerve irregularly, these curves can vary slightly from the real world lane markings.

Though Map Lanes is only a heuristic of where the lanes may be, it is quite useful for filtering out the large amounts of lidar data coming into the World Model. However, it is important to note that if the vehicle ignores an obstacle deemed by Map Lanes to be outside the lanes, our reactive obstacle avoidance routines will nevertheless prevent collision with the obstacle, should it in fact be in the roadway (see Section 3.5.2).

As our vision-based lane detection improves, we will incorporate visual lane detection into the World Model using the same interfaces currently defined for Map Lanes. In this case, we plan to use Map Lanes as a failsafe for when visual data is unreliable (illumination changes, rain) or unavailable (unforeseen hardware failures).

3.4.3 Object Modeling

Using locations of obstacles from the lidar fusion grid, filtered through lane boundaries as given by Map Lanes, is useful for reactive, local controllers like *Navigator*. However, for high-level planning, we want to have object models to be able to make intelligent decisions. At the *Commander* level of control, the robot should know that the lane is blocked by orange cones (re-planning is necessary), a moving vehicle (queue up behind the vehicle), a stalled vehicle (go around), or even people (stop completely if within a safety radius).

Using the *simulated lidar scan* from the ***lidar fusion*** module, we are currently producing and testing code to build dynamic models of the obstacles nearest to the vehicle. We do this by first filtering out points that do not fall within lane boundaries as defined by Map Lanes. Next, we leverage the assumption that most obstacles are vehicles in order to cluster points to fit rectangles. With these rectangles, we then have estimates of the size of obstacles, and what lane (or lanes) these obstacles occupy. We are currently implementing the code to track these rectangles over time which will allow us to complete the intersection precedence portion of the Urban Challenge criteria. Figure 8 illustrates our progress.

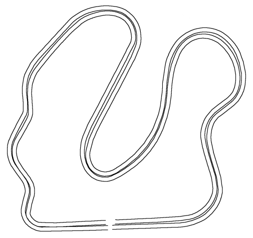


Figure 7 - Map Lanes representation of an RNDF created at a "Grand Prix" style racetrack in Austin.

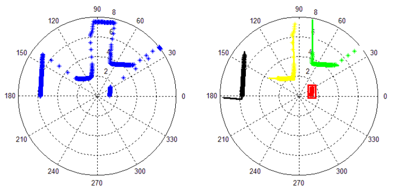


Figure 8 - This figure shows our current progress in object modeling – clustering lidar data and fitting rectangles to clusters that may be vehicles.

3.5 Control

At the highest level of reasoning – taking input from the world model – exists our software control system consisting of three primary modules: *Pilot*, *Navigator*, and *Commander* as illustrated in Figures 1 and 3. The modules are defined by their relative level of reasoning about the world and the position of the vehicle in it.

3.5.1 Pilot

The Pilot is the closest module to the physical hardware of the vehicle. It is responsible for converting a desired speed and heading into appropriate throttle, brake, and steering corrections through adjustments of the actuators of the vehicle. As such, Pilot essentially acts as a single interface between our software controllers and our hardware.

The Pilot uses a physical model of the vehicle to control its speed and heading. The model includes safe operating limits for the braking, acceleration, and steering controls. Using these limits, a safety check filters the speed and heading requirements from the Navigator before sending them to the PID controller which drives the physical actuators.

3.5.2 Navigator

The Navigator fills a crucial middle ground between high-level reasoning and low-level control. It accepts orders from Commander and combines this with local sensory understanding of the surroundings of the vehicle - the world model information along with raw range readings from the front and rear SICK lidars - to decide on an appropriate speed and heading for a given situation.

Navigator implements a collection of behaviors such as 'Follow Lane', 'Stop at Intersection', and 'Pass Left' from which Commander selects when sending orders. For instance, if Commander gives a general "Follow Lane" instruction, Navigator uses information from the world model to maintain the current lane of travel while simultaneously performing any minor obstacle avoidance within the current lane. Alternatively, if the vehicle detects that the lane is blocked ahead, Navigator reduces its speed appropriately, eventually coming to a complete stop when necessary. Commander then solves the problem of how to handle this blockage at a higher level and issues updated orders to Navigator.

3.5.3 Commander

The Commander module operates at the highest level of reasoning. Navigator and Pilot, then, act as implementers of the plan generated by Commander, which includes determining an optimal route from the current location of the vehicle to the desired goal, maintaining an awareness of the current state of the vehicle, and sending the resulting desired short-term behaviors to Navigator as instructions. In this section, we give an overview of three key Commander functions – large-scale path planning, behavior selection, and speed control – as well as an example of a higher level functionality, namely intersection precedence.

3.5.3.1 Large-scale Path Planning

Commander's first priority is to plan a distance-optimal route from the robot's current position to the goal position. This is implemented as an A* search in which every waypoint in the RNDF acts as a node and the exits between them act as edges. The start position for the search is defined as the last waypoint that the vehicle was near and the goal position is defined as the next *checkpoint* specified in the MDF. The search heuristic is Euclidean distance between the current position and the goal position. For efficiency, the route is only re-planned when the vehicle has visited one of the nodes along the planned path or when the vehicle's situation has changed. Situation changes occur when a new behavior is necessary, such as the realization that the vehicle's lane is blocked ahead by a stalled vehicle.

3.5.3.2 Behavior Selection

Another of Commander's responsibilities is the selection of a behavior for Navigator to follow. Example behaviors include 'Follow Lane', 'Turn Around', 'Stop At Line', and 'Park'. A more complete list of behaviors is presented in Figure 9.

The behavior selected is a function of Commander's state and the high-level situation of the vehicle. Commander's states currently include the following:

1) *Road* – Normal operation. Road following.

2) *Lane Blocked* – Our current travel lane is blocked ahead.

3) *Dodging Blocked Lane* – We are in the process of avoiding a blocked lane.

4) *Road Blocked* – The entire road is blocked.

The number of states will continue to increase as functionality is added to the vehicle.

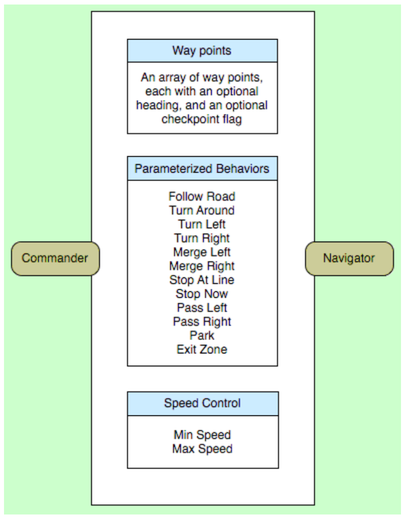


Figure 9 - Commander's interface to Navigator.

As an example, let us suppose there is a stalled car blocking the vehicle's lane of travel. Upon recognizing the lane blockage, Commander changes its own state to *Lane Blocked* and begins issuing orders with the 'Stop Now' behavior to stop the vehicle as fast as possible. Once the vehicle has stopped for an appropriate amount of time, Commander changes its own state to *Dodging Blocked Lane* and plans an alternate route through another available lane. Commander then begins issuing orders with the 'Pass Left' behavior and specifying high-level waypoints in the alternate lane to seed Navigator's behavior. Once Commander determines that the obstacle blocking our original lane is safely behind (*Road* state), it plans a new route through the original lane of travel and orders the 'Follow Road' behavior, or perhaps a 'Merge Right' behavior if a right hand turn must be made soon. Once our vehicle has returned to its original lane of travel, Commander resets its own state back to *Road* and continues on.

This finite state machine approach to a high-level control module has advantages and disadvantages. Advantages include high predictability and reliability, as well as high performance since state transitions are well defined and generally easy to compute. The primary disadvantage is an inability to handle situations that were not defined as part of the state machine. For the Urban Challenge, our state machine approach follows our "controlled complexity" theme while still allowing the vehicle to complete its mission.

3.5.3.3 Speed Control

The final responsibility of Commander is to provide Navigator with parameters controlling minimum and maximum speeds. In typical lane-following situations, Commander passes along the values defined explicitly in the MDF for the current Segment or Zone. For some situations, however, Commander lowers the maximum speed limit to ensure reliable control during higher-precision maneuvers. For example, when the vehicle is performing a maneuver to dodge a stalled vehicle blocking our lane, Commander ensures that we are not moving faster than 5 mph.

3.5.3.4 Intersection Precedence

At the time of this writing, Commander is capable of performing all basic navigation behaviors from the technical evaluation criteria. Our main focus leading up to the site visit is on the basic traffic functions of queuing at intersections and intersection precedence. Some of the advanced navigation and advanced traffic capabilities, such as dynamic re-planning by Commander in response to blocked roads, are already in place. However most will be the ongoing focus of our efforts leading towards the semifinals in October.

For intersection precedence, the key information is whether there is a car in each lane at the moment that we arrive at the stop line (i.e. with no more cars in front of us). Knowing that, the precedence algorithm is straightforward: we need to let exactly one car go from each occupied lane, and then it is our turn. It doesn't matter if there are any cars left in the other lanes, since they will have arrived after we have. This will be straightforward to implement once we complete our dynamic object modeling (see Section 3.4.3) so that we can recognize vehicles stopped at the intersection when we arrive and track them as they traverse the intersection.

4 Results and Performance

Because our system is under continual development with a view towards being complete by the NQE in October 2007, performance evaluation is ongoing. In this section we summarize performance tests done to date of some of the key modules in our system, including analysis of their strengths and currently needed directions for improvement. Specifically, we analyze the performance of the vehicle hardware, the Velodyne lidar, the Applanix, Map Lanes, and Commander.

4.1 Vehicle Evaluation

Our core vehicle hardware has logged hundreds of hours of testing over the spring of 2007 without major problems. Steering, shifting, and throttle have had no electrical or mechanical malfunctions. The braking system has had periodic difficulties that were diagnosed as a bad electric motor. It has worked as designed since the motor was replaced. The computers underwent a recent OS change – from Fedora Core 3 Linux to Ubuntu Dapper Linux (with a custom built kernel that includes a faster clock cycle) – with no adverse effects to the overall system.

4.2 Velodyne Evaluation

As discussed in Section 3.3.2.2, in order to integrate the Velodyne into our system in time for the site visit and to run it at its full 10Hz capacity, we use it to contribute to the 2D lidar fusion grid described in Section 3.4.1. In this section, we evaluate both the speed of our Velodyne sensor processing loop and the effectiveness of our integration of the Velodyne data into our occupancy grid representation.

4.2.1 Velodyne Read

Achieving 10Hz processing with the Velodyne was not as straightforward as expected. Specifically, our initial attempt at consuming data from the various lasers revealed that any delay in reading the UDP packets sent by the Velodyne quickly caused an overflow of the read buffer and thus packet-loss. As a result, the range data actually processed appeared to be arriving out of order. For example, if an Ethernet packet representing the range-reading at 30 degrees (base orientation) was received, the next packet might have a base orientation of 35 degrees, when information regarding range at 31 degrees would be expected. Figure 10(left) illustrates this problem by showing the base orientation of processed packets (in radians) plotted as a function of time when an eight-microsecond sleep is inserted between each packet read. Even such a brief pause quickly causes data to be lost.

The solution was to separate the processing of Velodyne data from the input reading. Figure 10(right) illustrates the result of allocating a separate thread for packet input. In this case, we package the 255 packets which compose a full Velodyne revolution as one message, and send the resulting information to a processing module. Using this approach, we are able to process all Velodyne input at the full 10Hz.

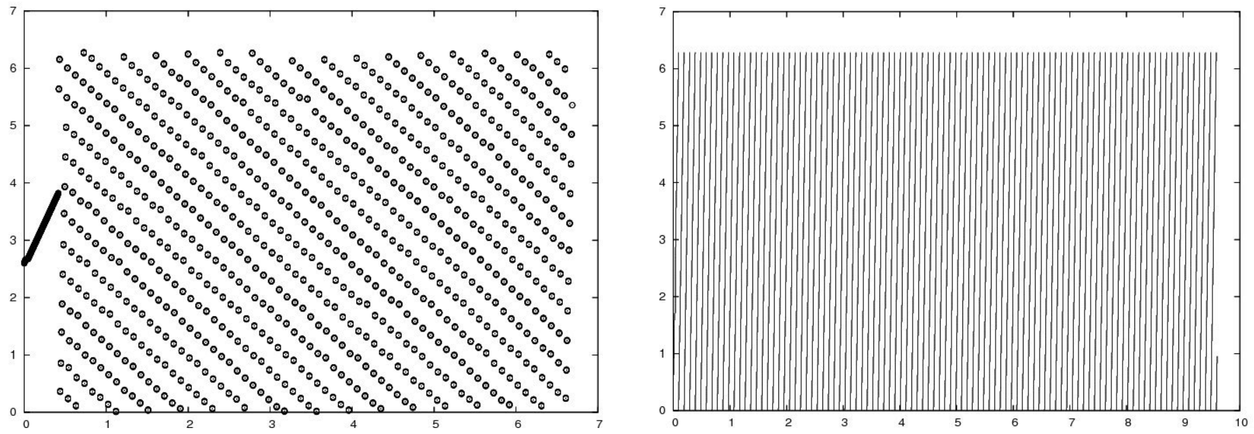


Figure 10 - Left: Graph of the base orientation (radians) of packet data as a function of time. Notice that packets are being lost before .5 seconds have past. Right: The same thing (circles replaced by dots for easier viewing) but with the processing done in a separate thread, packets are no longer lost.

4.2.2 Velodyne Process

Rather than creating a 3D (X,Y,Z) mesh or voxel-based model and rather than dealing with ground-plane removal, which may not easily generalize to hills or uneven roads that may be present in future competitions, instead we take the 3D Velodyne data and, at each cycle (i.e. every complete set of 360° data), we create a 2½D "height-difference" map, described in Section 3.3.2.2.

Results are shown in Figure 11 using one of the data sets used to calibrate the lidar. Notice the piece of plywood, in the office hallway. From our "height-difference" map, we get a 2D lidar scan that sees only the walls, the vertical plywood, and some carts in the hallway, while ignoring the ground and all other non-vertical objects.

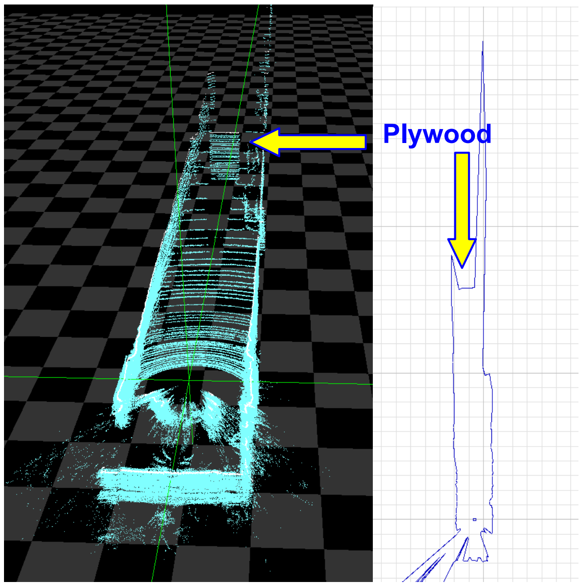


Figure 11 - Simulated lidar scan (right) created by ray casting in a "height-difference" map, where only vertical objects above a predetermined height are considered objects.

To test our calibration, and to verify Velodyne's claims of 5cm accuracy, we performed analysis of distances to various known obstacles. The plywood in Figure 11 was measured by hand to be exactly 16.22m from the center of the lidar unit. Figure 12 shows the range readings (after distances have been calculated from raw ranges) returned by the 6 lasers that have pitch angles within one degree of horizontal. We can see that though the readings do underestimate the overall location of the object, it is generally within the 5cm error claimed by Velodyne.

Laser ID	Pitch	Mean range	Median	Max	Min	Std. deviation
34	0.32°	16.16m	16.16	16.18	16.14	0.01
35	0.66°	16.19m	16.2	16.25	16.01	0.052
56	1.0°	16.15m	16.15	16.18	16.12	0.014
9	-0.7°	16.19m	16.19	16.20	16.17	0.009
62	-0.36°	16.13m	16.13	16.14	16.12	0.007
63	-0.02°	16.25m	16.25	16.29	16.22	0.014

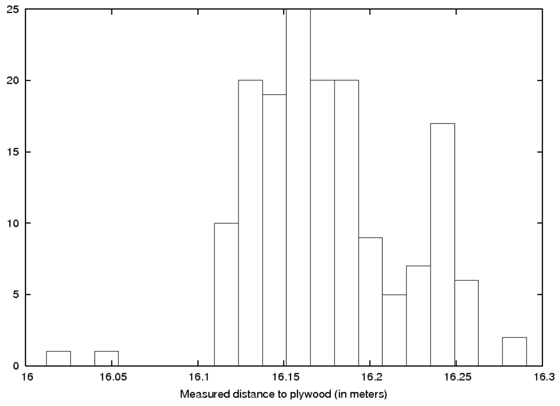


Figure 12 – After calibration using large building at distances above 30m, out unit tends to underestimate closer objects; however the error is well within acceptable limits. Left: Recorded distance to obstacle 16.22m away from 6 near horizontal lasers. Right: Histogram of the distances calculated from these 6 lasers.

Though in keeping with our principle of "controlled complexity", reducing the 3D Velodyne information to a 2D representation prevents us from making full use of the 3D information provided by the lidar at this time. Expanding our use of the 3D information, for instance to detect obstacles that are behind other obstacles, while still maintaining a 10Hz processing loop is to be a main focus of our efforts between the site visit and the NQE.

4.3 Applanix Evaluation

Along with the Velodyne sensor, the Applanix POS LV is one of the most significant hardware improvements to our vehicle since the 2005 Grand Challenge. When utilizing D-GPS from 6 satellites, the Applanix returns positions with sub-meter accuracy, and we have observed up to 0.02° angular accuracy.

To test the performance without the GPS signals, we covered both GPS antennas with aluminum foil, and verified that we had no signal. We then drove the vehicle in both forward and reverse, circled around traffic circles, and performed U-turns and three-point turns. The position was estimated with inertial and wheel odometry only (dead reckoning) for 3 minutes before returning to the exact starting position. The reported position was off by only 1 meter.

In contrast, before installation of the Applanix unit, localization accuracy was significantly worse. In particular, we observed very noisy angular pose information – heading swings of ±5° when accelerating or stopping were not uncommon. Loss or re-acquisition of GPS satellites resulted in lateral position discontinuities of several meters and driving under trees or an underpass always resulted in position discontinuities. Vertical errors of several thousand feet were also observed under these conditions! Thus, the Applanix POS LV was an essential upgrade to our vehicle for the Urban Challenge.

4.4 Map Lanes Evaluation

As described in Section 3.4.2, Map Lanes was originally conceived as a temporary placeholder for visual lane detection. However, its effectiveness at providing Navigator and Commander with useful lane information has been such that we plan to use it as the default lane modeling module. Vision information, when available and highly reliable, can then be used to fine tune the details.

As an informal evaluation of Map Lanes' effectiveness, we present Figure 13, which overlays its output on top of an aerial view of our site visit test course at Southwest Research Institute (SwRI). Note that even though its input is a sparse RNDF representation that does not precisely model the curves at the corners of the track, the resulting lane markings line up quite closely with the actual lanes.

4.5 Commander Timing Evaluation

In order to react in real time to incoming sensor data, it is important that none of the modules in the control loop takes longer than the sensor cycle time. The undergraduate thesis that focused on the creation of our Commander module includes an evaluation of Commander's speed, both on average and in the worst case.

Commander's fast execution is a result of its underlying finite state machine implementation. State transitions are triggered by simple Boolean expressions representing situations encountered by the vehicle. The route adjustment itself is inexpensive computationally since it only requires small modifications to the RNDF structure and an execution of the A* path planner. A performance graph of Commander's execution time is presented in Figure 14.

Analyzing Figure 14, we can confirm several required evaluation criteria. In the graph, the dark blue and yellow lines (bottom two lines in the plot) represent the execution time of Commander for two consecutive test runs of our vehicle. The turquoise and pink lines (top two lines in the plot) represent the execution time of Navigator for the same two test runs. Observe that Commander operates very quickly under the common case, with most control cycles completing in less than 1 ms.

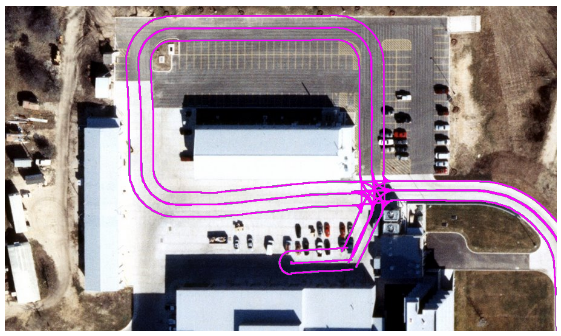


Figure 13 - An example of our Map Lanes algorithm run on the SwRI version 2.2 RNDF. Lane markings are overlaid onto an aerial view of the course.

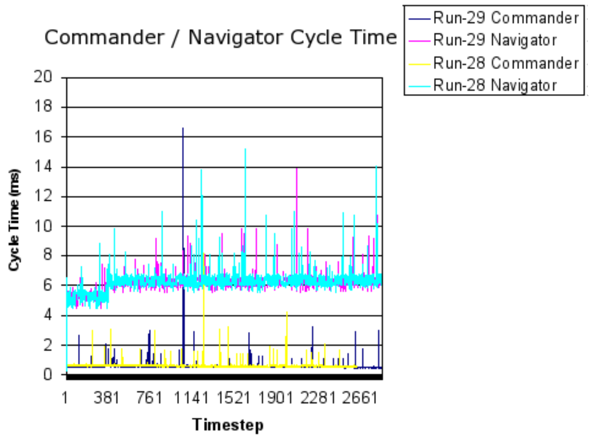


Figure 14 - Performance graph comparing Commander to Navigator.

More specifically, we have two main evaluation criteria with regards to Commander timing. First, we want to ensure that Commander executes faster than Navigator so that Navigator is always operating with up-to-date situational information from Commander. Each test run contains just one time step for which the Commander execution time shoots above the corresponding Navigator execution time, probably due to random processor scheduling. Given its rare and temporary nature, this overshoot does not present a significant problem with regards to this evaluation criterion. Second, Commander should not spike its execution time above 50 ms (the cycle rate of our sensors) so that it can keep up with incoming environmental data. Evaluation of this graph shows that even the most extreme spike in execution time was below 18 ms.

5 Conclusion

As documented throughout, our team has made significant progress towards the Urban Challenge goals. Table 1 addresses the status of our work towards each of the technical evaluation criteria, including target completion date and the high-level approach for tasks that are in progress or completed. We aim to have all tasks completed by September 1st at the latest in order to leave a month and a half for only bug fixes and an emphasis on enabling hundreds of miles of failure-free driving leading up to the NQE.

In summary, Austin Robot Technology is well along the way towards achieving both of its goals, namely 1) successfully meeting the Urban Challenge technical evaluation criteria and 2) educating new young Computer Science researchers. With regards to our second goal, we have already succeeded significantly by inspiring a class of students to work productively on the project and to become involved in the world of academic research. With regards to our first, and primary, goal, our hardware is in place and we have made significant strides in our software development. Following the design principles of safety first, controlled complexity, and frequent, incremental testing, we are well-positioned to complete the site visit tasks and are eager to continue towards the final Urban Challenge.

Task	*Status*	*Timeline*
Basic navigation		
Preparation for run	Mission/Path Planning reads in RNDF and MDF, plans routes between checkpoints.	Completed
Mission start	World model determines the position and heading, proceeds to the first checkpoint.	Completed
Checkpoints	World model determines the position and heading; Commander ensures that a checkpoint can be traversed.	Completed
Stay in lane	Using Map Lanes / Using vision	Completed / July 1st
Speed limits	Commander determines the speed; Pilot enforces the speed limit.	Completed
Excess delay	Commander maintains timer.	Completed
Collisions	Lidar detects close obstacles; Pilot avoids immediate collisions; Object detection predicts trajectories of self and other vehicles to avoid predicted collisions.	Completed July 15th
Stop line	World model determines presence at stop line; Color vision verifies.	Completed July 1st
Vehicle separation	Commander issues immediate stop in response to obstacle detected in lane.	June 15th
Leaving lane to pass	Commander plots extra waypoints to pass.	Completed
Returning to lane after pass	Commander plots extra waypoints to return.	Completed
U-turn	Commander plans for and navigator executes 3-point turns as needed for mission.	Completed.
Basic traffic		
Basic navigation	See above.	
Intersection precedence	Object modeling determines presence of other vehicles at intersection upon arrival. World model tracks their passage through intersection.	June 8th
Minimum following distance	Navigator dynamically adjusts speed to maintain following 2-3 s following distance.	June 15th
Queuing	Happens based on vehicle separation and minimum following distance.	June 15th
Advanced navigation		
Basic traffic	See above.	
Obstacle field	Path Planning determines a route through the field; Immediate Collision avoids collisions	Completed.
Parking lot	Not started.	August 1st
Dynamic re-planning	Word model recognizes road blocks; Commander re-plans.	July 15th
Road following	Using Map Lanes / Using vision	Completed / July 1st
GPS outage	Applanix POS LV maintains localization	Completed
Advanced traffic		
Advanced navigation	See above.	
Merge	Not started.	August 15th
Vehicle separation during merge	Not started.	August 15th
Left turn	Not Started.	August 1st
Lane changes	Commander issues lane change command; navigator plots course.	Completed
Vehicle separation during left turn	Not started.	August 1st
Passing moving vehicles	Not started.	September 1st
Zones	Not started.	September 1st
Emergency braking	World model recognizes the need to brake; Pilot performs safe emergency braking using the E-stop hardware.	Completed
Defensive driving	Not started.	August 1st
Traffic jam	Not started.	September 1st

Table 1

References

[1] DARPA Urban Challenge Technical Evaluation Criteria Document.
http://www.darpa.mil/GRANDCHALLENGE/docs/Technical_Evaluation_Criteria_031607.pdf

[2] D. Stronger and P. Stone. Towards Autonomous Sensor and Actuator Model Induction on a Mobile Robot. *Connection Science*, 18(2):97–119, 2006.

[3] M. Sridharan and P. Stone. Autonomous Planned Color Learning on a Mobile Robot Without Labeled Data. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, 2006.

[4] M. Sridharan and P. Stone. Autonomous Color Learning on a Mobile Robot. In *Proceedings of the National Conference on Artificial Intelligence*, 2005.

[5] M. Sridharan and P. Stone. Real-Time Vision on a Mobile Robot Platform. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005.

[6] M. Sridharan, G. Kuhlmann, and P. Stone. Practical Vision-Based Monte Carlo Localization on a Legged Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[7] G. Kuhlmann, W.B. Knox, and P. Stone. Know Thine Enemy: A Champion RoboCup Coach Agent. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[8] P. Stone, M. Sridharan, D. Stronger, G. Kuhlmann, N. Kohl, P. Fidelman, and N.K. Jong. From Pixels to Multi-Robot Decision-Making: A Study in Uncertainty. *Robotics and Autonomous Systems*, 54(11):933–43, November 2006. Special issue on Planning Under Uncertainty in Robotics.

[9] M.L. Littman and P. Stone. A Polynomial-time Nash Equilibrium Algorithm for Repeated Games. *Decision Support Systems*, 39:55–66, 2005.

[10] P. Stone. Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, 2000.

[11] M. Veloso, M. Bowling, and P. Stone. The CMUnited-98 champion small-robot team. In *Advanced Robotics*, vol. 13, no. 8, pp. 753-766, 2000.

[12] P. Stone, M.L. Littman, S. Singh, and M. Kearns. ATTac-2000: An Adaptive Autonomous Bidding Agent. *Journal of Artificial Intelligence Research*, 15:189–206, June 2001.

[13] P. Stone, R.E. Schapire, M.L. Littman, J.A. Csirik, and D. McAllester. Decision-Theoretic Bidding Based on Learned Density Models in Simultaneous, Interacting Auctions. *Journal of Artificial Intelligence Research*, 19:209–242, 2003.

[14] D. Pardoe and P. Stone. TacTex-2005: A Champion Supply Chain Management Agent. In *Proceedings of the National Conference on Artificial Intelligence*, 2006.

[15] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proceedings of the National Conference on Artificial Intelligence*, 2002.

[16] P. Beeson, M. MacMahon, J. Modayil, J. Provost, F. Savelli, and B. Kuipers. Exploiting local perceptual models for topological map-building. In *Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robotics,* 2003.

[17] J. Modayil, P. Beeson, and B. Kupiers. Using the topological skeleton for scalable global metrical map-building. In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2004.

[18] P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the Extended Voronoi Graph. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 2005.

[19] P. Beeson, A. Murarka, and B. Kuipers. Adapting proposal distributions for accurate, efficient mobile robot localization.In *Proceedings of the IEEE International Conference on Robotics and Automation,* 2006.

[20] B. Kuipers, P. Beeson, J. Modayil, and J. Provost. Bootstrap learning of foundational representations. Connection Science, 18(2), June 2006, pages 145-158.

[21] P. Beeson, M. MacMahon, J. Modayil, A. Murarka, B. Kuipers, and B. Stankiewicz. Integrating multiple representations of spatial knowledge for mapping, navigation, and communication. AAAI Spring Symposium Series, Interaction Challenges for Intelligent Assistants, Stanford, CA. AAAI Technical Report SS-07-04, 2007.

[22] W. Whittaker and L. Nastro. Utilization of Position and Orientation Data for Pre-planning and Real Time Autonomous Vehicle Navigation. In *Proceedings of IEEE/ION Position Location and Navigation Symposium*, 2006.

[23] M. Yguel, C. Tay M. Keat, C. Braillon, C. Laugier, and O. Aycard. Dense Mapping for telemetric sensors: efficient algorithms and sparse representation. In *Proceedings of Robotics: Science and Systems Conference*, 2007.

[24] D. Wolf, A. Howard, G.S. Sukhatme. Towards geometric 3D mapping of outdoor environments using mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2005.

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

[26] A. Elfes. Occupancy Grids: A Probabilistic Framework for Robot Perception and Navigation. PhD Dissertation, Carnegie Mellon University, 1989.

[27] H.P. Moravec. Sensor fusion in certainty grids for mobile robots. *Artificial Intelligence.* Summer, 1988.