A Proposed Standardized Testing Procedure for Autonomous Ground Vehicles

Thomas James Alberi

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Mechanical Engineering

Dr. Alfred L. Wicks, Chairman
Associate Professor of Mechanical Engineering

Dr. Charles F. Reinholtz, Co-Chairman
Alumni Distinguished Professor, Virginia Tech
Chair, Mechanical Engineering, Embry Riddle Aeronautical University

Dr. Dennis W. Hong
Assistant Professor of Mechanical Engineering

April 29, 2008
Blacksburg, Virginia

Keywords: Autonomous Vehicle, Software Validation, DARPA Urban Challenge

A Proposed Standardized Testing Procedure for Autonomous Ground Vehicles

Thomas James Alberi

ABSTRACT

Development of unmanned vehicles will increase as the need to save lives rises. In both military and civilian applications, humans can be taken out of the loop through the implementation of safe and intelligent autonomous vehicles. Although hardware and software development continue to play a large role in the autonomous vehicle industry, validation of these systems will always be necessary. The ability to test these vehicles thoroughly and efficiently will ensure their proper and flawless operation.

On November 3, 2007 the Defense Advanced Research Projects Agency held the Urban Challenge to drive the development of autonomous ground vehicles for military use. This event required vehicles built by teams across the world to autonomously navigate a 60 mile course in an urban environment in less than 6 hours. This thesis addresses the testing aspect of autonomous ground vehicles that exhibit the advanced behaviors necessary for operating in such an event. Specifically, the experiences of Team Victor Tango and other Urban Challenge teams are covered in detail. Testing facilities, safety measures, procedures, and validation methods utilized by these teams provide valuable information on the development of their vehicles. Combining all these aspects results in a proposed testing strategy for autonomous ground vehicles.

Acknowledgments

There are many people who have contributed to my research and my success. First, I would like to thank my family. Without the help and support of my parents, Mike and Lori Alberi, I would not have accomplished all that I have. My sister, Kirstin, has been a tremendous role model for me. She exemplifies how successful one can be through hard work and dedication. Finally, I would like to thank my extended family for their support throughout my life.

I would like to acknowledge the influence my advisors have had on my research. Dr. Reinholtz, Dr. Wicks, and Dr. Hong have provided an atmosphere conducive to learning and achievement. Throughout my undergraduate and graduate years, they have always been available to answer questions, provide guidance, and supply the support I needed. Many of the design projects that secured my decision to come to Virginia Tech would not be in existence had it not been for these three individuals. The success of many of the unmanned systems projects at Virginia Tech is due in large part to these outstanding people.

Odin's success in the Urban Challenge would not have been possible without the hard work and dedication of Team Victor Tango. Graduate and undergraduate students came together to develop this outstanding vehicle and provided a basis for my research, as well as others'. The employees at TORC Technologies are owed just as much gratitude for the time and effort they invested in the project. It was an honor to work along side all of these smart, talented individuals and I wish them all the best of luck in the future.

Finally, several individuals from other Urban Challenge teams must be recognized for their valuable input. I would like to thank the testing leaders from Team CarOLO, Intelligent Vehicle Systems, Team MIT, Stanford Racing, and Tartan Racing for their detailed responses to my questions. Their answers provided valuable information necessary for the development of this thesis.

Contents

List of Figures

Chapter 1

Introduction

The autonomous vehicle testing process discussed in this thesis was developed through the testing of Team Victor Tango's Odin, an autonomous vehicle that placed third in DARPA's 2007 Urban Challenge. Through months of software and vehicle systems testing, the process has been refined and improved. The experiences of Team Victor Tango and other Urban Challenge teams have been used to propose an effective testing strategy for autonomous vehicles.

1.1 Thesis Overview

An effective testing strategy is an important asset in any autonomous vehicle development program. This thesis goes through the experience of developing and testing an autonomous vehicle and proposes a methodology for doing so. Chapter one explains the DARPA Urban Challenge and the objectives it presented to the participating teams. A

brief overview of Team Victor Tango's vehicle, Odin, will also be given.

Chapter two describes the safety measures Team Victor Tango put in place during real life vehicle testing. In any program where the potential for dangerous accidents is high, such as one testing a full size autonomous vehicle, the need for safety is paramount. Chapter 3 describes the various facilities used by Team Victor Tango on which Odin's systems were tested. Chapter 4 covers the testing experiences of other Urban Challenge teams during the development of their respective vehicles. Chapter 5 describes the evolution of Team Victor Tango's testing process. The experience on each site will be covered, as well as the positive and negative aspects of the various strategies used. Finally, in Chapter 6, a general testing method will be proposed, taking into account the experiences reviewed in previous chapters.

1.2 The 2007 DARPA Urban Challenge

Currently, there is a great need for autonomous ground vehicle research and development. In 2006, there were 38,588 fatal car crashes in the United States. These accidents resulted in a total of 42,642 deaths, which includes individuals inside and outside of the vehicle [1]. It is estimated that 45% to 75% of vehicle accidents are caused by driver error [2]. The implementation of a safe, proven autonomous transit system would eliminate the human driver aspect and reduce yearly accident totals dramatically.

The military has a more immediate need to remove humans from its vehicles. As of January, 2008, there have been 1,702 coalition deaths in Iraq due to IED attacks on ground vehicles (1,619 from the United States) since the start of the start of the war in Iraq in March of 2004 [3]. Many of the vehicles being attacked are supply vehicles that have no other purpose but to bring goods from point A to point B. Utilizing autonomous vehicles in their place will result in decreased casualties. The United States government

has recognized this need, represented in a congressional mandate stating:

> It shall be a goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such thatby 2015, one-third of the operational ground combat vehicles are unmanned [4].

The Defense Advanced Research Projects Agency, DARPA, held the first Grand Challenge in March of 2004 to drive the development of fully autonomous vehicles for military use. The competition consisted of a 142 mile autonomous vehicle race from Barstow, CA to Primm, NV. The only form of navigation given was a list of waypoints, in latitude and longitude, to be reached by each vehicle on its way to the finish line. In addition, vehicles utilized various sensors to observe the world around them and avoid obstacles. The $1 million grand prize went unclaimed in this event, as the team that went the farthest, Carnegie Mellon University, only completed about 7 miles of the course. However, DARPA held a second challenge in 2005, which consisted of a 132 mile race across the desert Southwest. A total of five teams completed the course, with Stanford winning the $2 million grand prize [5].

Next, DARPA set its sights on navigating an urban environment with the announcement of the 2007 Urban Challenge in the spring of 2006. In order to participate in the final event, each team was required to pass several DARPA-judged objectives. Rules and expected vehicle behaviors were given in the Technical Evaluation Criteria document [6]. Demonstration videos from each team were reviewed by DARPA, as well as technical papers that outlined the vehicle and software architectures. DARPA also made scheduled site visits to the teams that showed their vehicle had the potential to compete in the final event, based on their videos and technical papers. During these visits, DARPA ran each vehicle through a series of tests on a standardized course to evaluate whether or not each team had reached the prescribed milestones. Out of the initial eighty nine participating teams, only thirty five were invited to Victorville, CA.

The event in Victorville consisted of a series of three qualification tests for each team, culminating in the final race on November 3, 2007. Each qualification run focused on different aspects of urban driving, including intersection precedence, merging, left turns across traffic, and obstacle avoidance. Eleven teams passed the qualification process to continue on to the final event.

To complete in the final event, each robot was required to complete three missions, each consisting of six or seven sub-missions, on an urban course. DARPA set a time limit of 6 hours on the roughly 60 mile course. Six teams finished the course, with Tartan Racing taking first place, Stanford taking second place, and Team Victor Tango coming in third.

1.3 Urban Challenge Navigation Format

The navigation standard used in the first two challenges consisted of a list of GPS waypoints, with associated lane widths and speed limits. Each vehicle had was required to navigate these waypoints on their way to the finish line. This simple format was all that was needed to navigate across the desert. An urban environment consisting of intersections and stop signs required a much more complicated navigation standard.

DARPA created a navigation format consisting of a Route Network Definition File and a Mission Definition File. Each file is a tab delimited text document that contains information needed to navigate the course. The RNDF provides information regarding streets, lane markings, stop line locations, entrance/exit pairs for intersections, as well as parking lots. Specified as ”zones” in the RNDF format, parking lot information contains boundaries and parking space locations. The RNDF can be thought of as a detailed map of the area in which the vehicle will be traveling. The MDF provides speed limits for each segment, or road, and a list of prescribed checkpoints that are to be reached in order [7].

The mission is accomplished when the last checkpoint is reached.

As a result of creating the RNDF and MDF format for the Urban Challenge, DARPA had developed a well planned navigation standard for autonomous vehicles. The RNDF provides a map to follow, much like a map a human would read, or similar to those preprogrammed in retail GPS navigation units. The MDF plans the route to be taken and serves as the speed limiter for each segment on the map.

As great of a format as it is, there are still some improvements that could be made. For many civilized areas, especially more urban environments, roads are well mapped with GPS. A major fault in DARPA's format arises in areas where GPS street mapping is limited or non-existent. Given that military operations take place in mostly undeveloped areas, there should be a decreased reliance on GPS waypoints. As sensor technology continues to improve, lane markings and road coverage detection can be used to stay on the road. Cameras can also be used to read signs and traffic signals, eliminating the need for stop signs to be positioned with GPS. The MDF format can still be used to help plan a path for the vehicle, so checkpoints must be positioned using GPS.

Chapter 2

Team Victor Tango's Odin

Some background on the vehicles that participated in the Urban Challenge is helpful in understanding the procedures used to test and validate their software. Team Victor Tango's entry in the 2007 Urban Challenge, Odin (named after the Norse god of wisdom), was developed on a 2005 Ford Escape Hybrid base platform.

2.1 Base Vehicle

Odin was converted to be completely drive-by-wire, but still retains the ability to be driven manually, making it an easy vehicle to test on. Many steps were taken to integrate the added autonomous system component into the vehicle while still maintaining a stock look. Much of the actuation needed to drive autonomously is taken care of by the vehicle's stock components. Shifting and throttle control are operated by tapping into the vehicle's stock system, which was by-wire to begin with. By intercepting the vehicle's voltage signals

and sending our own, the team was able to control the shifting and throttle components. Power steering on the vehicle is taken care of by an electric motor that uses signals from a torque sensor on the steering column to apply an appropriate amount of assistance when the wheel is turned. Autonomous steering is accomplished by replacing the torque sensor signals with our own. Because of the high level of complexity of the brake system, and the fact that it was very easy to fault out, an actuator was added to control braking.

The hybrid model of the Escape also comes equipped with a stock high voltage battery pack, which provided power to all the components without the need for a generator or an aftermarket alternator. The voltage was stepped down through 48V and 24V DC-DC converters. Power for Odin's computers was run through a Tripp-Lite Uninterruptible Power Supply. Power for all other components was sent through custom power distribution boxes. Finally, cooling for power distribution and computing components in the back was provided by the vehicle's stock A/C. To cut down on fuel consumption, the rear cargo bay temperature was monitored and air flow was electronically controlled. Figure 2.1 shows the layout of Odin's rear cargo bay.

2.2 Safety & Sensor Components

Odin was equipped with an emergency stop system capable of manually disabling the vehicle through the actuation of buttons on the interior and exterior of the vehicle. Door sensors were monitored to disable the vehicle from autonomous operation unless all the doors were closed. A wireless E-stop transmitter also provided the ability to stop the vehicle remotely. The vehicle's sensor array consisted of two IBEO Alasca XT Fusion laser rangefinders, an IBEO Alasca A0 multiplane laser scanner, four SICK LMS laser rangefinders, two Imaging Source color monocular cameras, and a NovAtel Propak LB+ GPS/INS system coupled with Omnistar HP differential corrections. External sensors

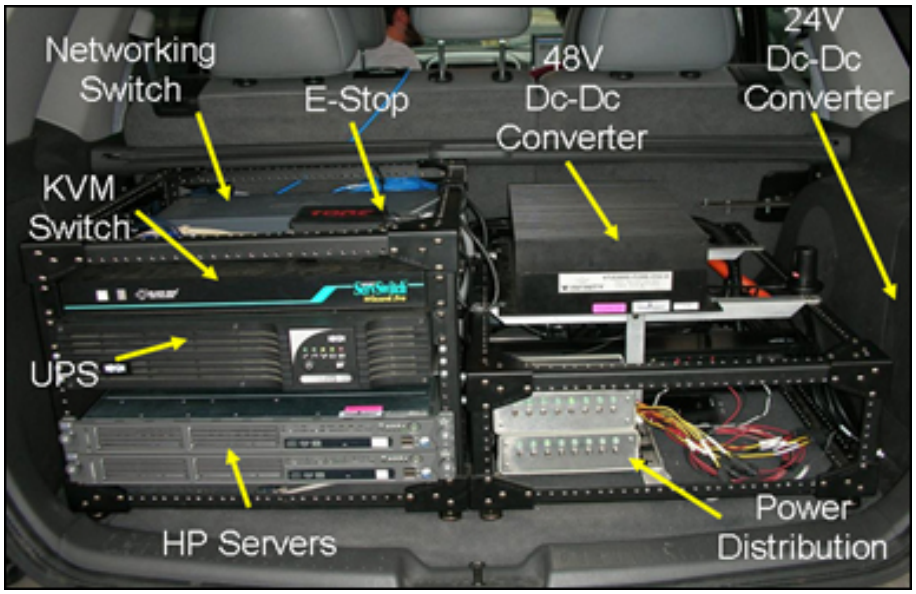

Figure 2.1: Power and computing components were placed in Odin's rear cargo bay [8]. Figure used with permission from Dr. Reinholtz.

(excluding the IBEO A0 in the rear of the vehicle) are labled in Figure 2.2. Figure 2.3 depicts the coverage and range of each sensor. The sensor suite was used by Odin to navigate the world, follow roads, recognize lane markings, and avoid obstacles.

2.3 Computing & Software

All high level software ran on two HP servers equipped with a total of sixteen processor cores. One server, "Bill", ran Windows while the other, "Linus", was equipped with the Linux operating system. Both computers could be logged in and viewed remotely through VNC. A National Instruments Compact RIO control and acquisition system, located in the glove box, ran *LabVIEW Real Time* and was responsible for sending actuation signals to the vehicle's drive by wire components. The computers, along with power distribution and networking components, were located in the cargo area of the vehicle. Due to the high level of integration of Odin's hardware, the team retained the vehicle's capacity to seat five people.

Figure 2.2: Odin's external sensor suite provided the software with valuable obstacle and positioning data during the 2007 Urban Challenge.

The majority of the code was developed in National Instruments *LabVIEW* software, a graphical programming language that has been used by Virginia Tech and its autonomous research programs for many years. *LabVIEW* code can be changed on the fly without the need for recompiling, saving testing and debugging time. The ability to "probe" data connections between different components is also a valuable debugging tool.

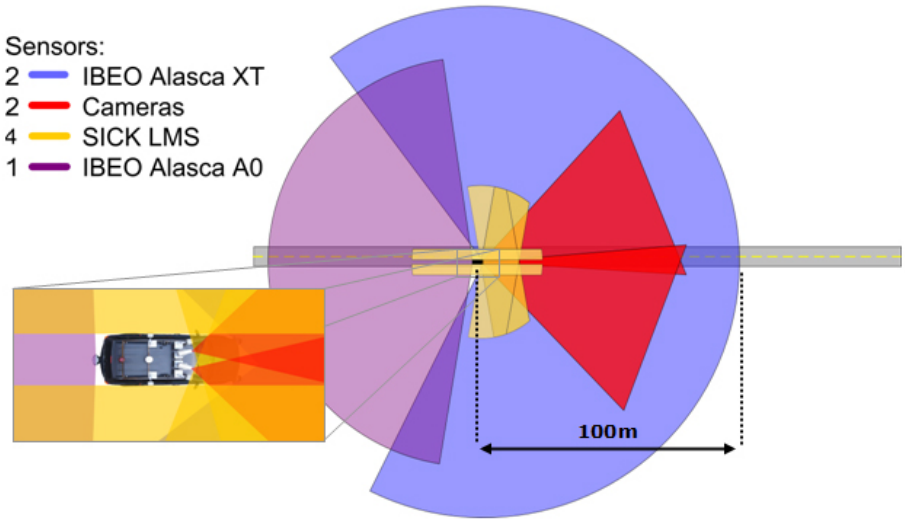

Figure 2.3: Odin's sensor suite provided proper coverage and range to alert the vehicle of obstacles in time to react to them [8]. Figure used with permission from Dr. Reinholtz.

Chapter 3

Team Victor Tango Safety Measures

Before testing could be carried out on the vehicle, safety measures were put in place to minimize the probability of accidents occurring. Team Victor Tango implemented several rules and procedures that were followed while the vehicle was in use.

Testing on Odin was conducted on closed courses with only authorized personnel and approved traffic vehicles and drivers on site. The courses were blocked off and signs were posted that alerted the general public to the operation of autonomous vehicles in the area. A siren and amber warning light came to life when Odin was put into autonomous mode, informing everyone to pay attention. Having total control over the test course was necessary to avoid risk to those not on the project.

All testing was conducted with a safety driver seated behind the wheel. Odin was designed to operate autonomously only while the shifter was the neutral position. If, at any point, the driver needed to take control of the vehicle, the shifter was simply moved to the drive position. This provided a fast method of gaining total control of the vehicle

with little effort. The driver also had the ability to step on the brake at any point during autonomous testing, although this was rarely used. The door sensors on the vehicle were also monitored and used as a software pause, preventing Odin from going into autonomous mode unless all the doors were closed. The last resort to stop Odin involved the use of the emergency stop system. Pressing any of the two buttons on the exterior of the vehicle, the button in the interior, or using the remote e-stop commanded the vehicle to stop and lock the parking brake. Only by manually restarting the vehicle and releasing the parking brake could an individual take control again. The ability to override the vehicle instantly proved to be a valuable asset during vehicle testing, further reducing the risk to humans and the vehicle.

In addition to a closed course and safety equipment, a set of procedures was put in place to further reduce any safety hazards. No individual was allowed on the test course while Odin was in motion, either autonomously or manually. Drivers and passengers in traffic vehicles utilized their seat belts at all times, even during low speed testing. Individuals not inside vehicles stood behind protective concrete barriers or inside a portable office trailer. The vehicle could not be run autonomously until the remote e-stop operator cleared the course and put the unit into run mode. After an autonomous run, Odin was programmed to beep its horn twice to signal that the mission file had been run successfully. This extra measure let people outside the vehicle know when the run was concluded. Once an autonomous run was over, the safety driver would signal all clear and the e-stop would be put back into pause before anyone could enter the course. At this point, the vehicle could not be put into autonomous mode accidentally. It is virtually impossible to be completely safe while testing a vehicle of this nature, but the combination of a closed course, safety equipment, and a specially designed testing procedure helped keep the risk at a minimum.

Chapter 4

Team Victor Tango Testing Facilities

During Odin's development, several different sites were used to test and validate the vehicle's code. Each course was accompanied by its positive and negative aspects, but required a list of traits conducive to successful testing. It was required that each site have sufficient area to drive the vehicle around, whether that consisted of a large parking lot, a road, or a combination of the two. Lane markings were also helpful to test vision line recognition and give traffic drivers visual confirmation of the course layout in parking lots. Because testing was often carried out in large groups, not everyone could be testing on the vehicle at the same time. The team required a building or base of operations on site to use when the vehicle was not available. Amenities included power, air conditioning, and restroom facilities. Testing was also conducted at night, so ample area lighting was helpful. The following is a description of the testing sites used by Team Victor Tango.

4.1 The Cage

The team's primary testing facility was a large parking lot used during the summer of 2007. Otherwise known as "the Cage", the parking lot is normally used by resident students. However, in the summer fewer people live on campus, leading to a large vacancy. Figure 4.1 shows an aerial view of the parking lot with the site visit course layout. The team was allowed to use half of the Cage for the entire summer, blocking it off with signs, saw horses, and caution tape. Because much of the summer was used preparing for DARPA's site visit, the required test course was surveyed and marked with paint over much of the area. A portable office trailer was rented, placed on one side of the course, and surrounded by protective concrete barriers. The trailer contained desks, chairs, power, wireless internet, and air conditioning, making it a comfortable place for team members to work while the vehicle was in use. A portable restroom was also placed on site near the trailer.

4.2 Kentland Farm

Once school was back in session, the Cage was no longer available, leaving the team searching for a new location. During this time, three separate sites were used. Kentland Farm is a 3,200 acre farm land owned by Virginia Tech and primarily used by the College of Agriculture and Life Sciences [9]. Several paved roads exist on the property, which the team was allowed to use for testing. The farm also contains restroom facilities on site.

4.3 Police Training Facility

The Roanoke County Police Department allowed the team to use their driver training facility, asking only to cover the cost of paying one of their officers to be present while

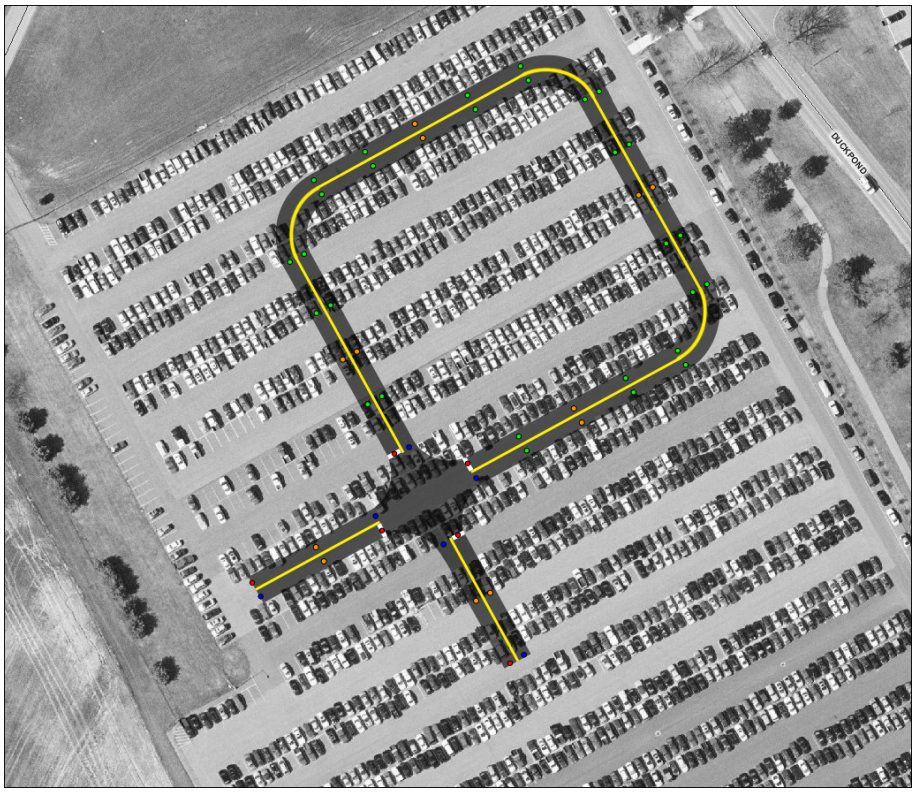

Figure 4.1: The Cage was used during the summer of 2007 to test Odin on the site visit course.

the site was in use. The driving area consists of a curving road up a hill ending in a large paved area with various lane markings. A structure next to the paved area housed power, restrooms, a refrigerator, and a microwave. Figure 4.2 shows Odin maneuvering around some traffic cones on the open paved area.

4.4 Corporate Research Center

The last site used for testing was Virginia Tech's Corporate Research Center, also referred to as the CRC. The CRC is a collection of corporate offices, roads, and parking lots located next to the Virginia Tech campus. The roads combine straight and curved sections covering several elevation changes. Parking lots are scattered throughout the area, with ample nighttime lighting. Figure 4.3 shows an aerial view of the CRC with a practice

Figure 4.2: The Roanoke County Police training facility was used during the fall of 2007.

RNDF on top. The offices of TORC Technologies, LLC, an engineering firm that partnered with the university on the Urban Challenge project, are also located in the CRC. The offices are furnished with desks, power, internet, and restroom facilities, providing a comfortable home base for team members to work. The combination of these four sites resulted in a valuable testing atmosphere for the development of Odin. While a dedicated autonomous vehicle testing site would have been ideal, it was not absolutely necessary.

Figure 4.3: Parking maneuvers and endurance runs were primarily tested at the CRC.

Chapter 5

Testing Experiences of Other Urban Challenge Teams

Before Team Victor Tango's vehicle testing is explained in depth, this section will cover the experiences of several other Urban Challenge teams. A short questionnaire was sent out to several teams in order to gather information on their testing facilities and methods. A copy of the questionnaire and a list of the responding teams can be found in Appendix B. A broad understanding of various autonomous vehicle testing methods is essential for compiling an effective testing strategy.

5.1 Testing Facilities

For the most part, other teams used similar testing locations to those of Team Victor Tango. Given the scope of the contest, urban areas were prevalent. Locations such as

parking lots and abandoned neighborhoods provided teams with paved areas to operate their vehicles.

Princeton's team had use of several sites at which testing and vehicle development were conducted. A garage housed their vehicle and provided a place to test equipment. An adjacent parking lot was used to test non-autonomous systems and a nearby field was used for control systems testing. Another area at the University's Forrestal campus was blocked off and used for autonomous testing [10].

MIT leased the main hangar tarmac of the decommissioned South Weymouth Naval Air Station in Weymouth, MA. Due to the lack of available office space on site, MIT rented a construction trailer outfitted with air conditioning, a generator, and wireless internet access. Their vehicle, Talos, was also outfitted with internet access, giving the team the capability to log onto the vehicle's computers from the trailer and other locations. With this service, even software developers in Cambridge could access the vehicle to correct software problems that could not be fixed by team members on site [11].

Stanford Racing utilized three different test sites, often without access to amenities that most teams had. The Shoreline Amphitheatre in Mountain View, CA provided a large empty parking lot, free internet access thanks to Google, and portable lavatories on site. Their vehicle, Junior, was also tested in abandoned suburban neighborhoods in Fort Ord in Monterrey, CA and Alameda Naval Air Station in Alameda, CA. While these two sites presented the team with an urban environment to test in, they offered no facilities [12].

Tartan Racing utilized numerous sites to test their vehicle, Boss. Their primary site exists at an abandoned steel factory in Pittsburgh, PA. Their vehicles were stored and maintained in a railroad roundhouse, previously used to house locomotives. Because of a leaky roof and the generally dirty environment in the roundhouse, software development was conducted in a much cleaner location known as the "Square House". Located within one hundred yards of the roundhouse, the Square House consists of three 57' trailers

placed next to each other with the adjacent walls removed. The Square House contains a central area in which the programming took place, while technical leads and the program manager were located in offices around the perimeter. While each facility has a refrigerator, microwave, internet access and lavatories, the Square House is also carpeted and has air conditioning.

Vehicle testing was conducted on several paved roads and intersections about a mile from the roundhouse in an area the team named "Robot City". Even though the roads were paved and professionally painted, their surfaces were uneven and contained pot holes. Vegetation also posed a problem as it grew through cracks in the pavement and on the edge of the road.

Two other facilities were also used before the team shipped their vehicles out to California. The Beverun Motorsports Complex in Big Beaver Borough, PA contains several race tracks and is located about 40 minutes away from Tartan Racing's headquarters. The facility was previously used by the team to conduct endurance runs with their Grand Challenge vehicles Sandstorm and H1lander. Boss was tested on a curved track at the site. The team also used a General Motors test site in Scottsdale, AZ during the winter. However, no details on the facility could be disclosed.

The team traveled out to test in California about 45 days before the Urban Challenge qualification runs, to continue testing on their vehicle. Testing took place in Northern California at the Castle Commerce Center, where the team leased a hangar. Streets and intersections, similar to those in Victorville, along with a curving road cut into tall grass helped prepare the team for what was to come at the Urban Challenge [13]. Figure 5.1 shows the team testing at the facility at Castle.

Figure 5.1: Tartan Racing used the Castle Commerce Center to test their vehicle a little over a month before the final competition [13]. Figure used with permission from Tartan Racing.

5.2 Safety

As Team Victor Tango did, each other team took precautions during testing to ensure a reasonable level of safety. Each team practiced on closed courses, blocking any unauthorized personnel from entering the area. For most tests, teams used a safety driver to take control of the vehicle if the need arose. Each vehicle had a fast and easy method for transferring control from the computer to the human. MIT used a toggle switch [11] and Tartan Racing used a button [13]. Stanford regained control of their vehicle through the actuation of a button or the steering wheel [12]. All teams utilized commercial wireless emergency stop systems in the event the vehicle had to be stopped during an autonomous run with no safety driver present in the vehicle.

5.3 System Monitoring

Each team had a system in place to monitor their vehicle's status. MIT developed a software module named "The Sheriff" to accomplish this task. All other software modules communicated to The Sheriff to provide it with their status in regards to memory leaks,

core dumps, and processor load. The Sheriff had the authority to shut down and restart any process deemed to be out of control. Due to a specially developed data transfer and communication protocol, restarting one process did not affect any others on the vehicle. The team had the ability to record all data from each process and play it back, including sensor data. As far as bug tracking and recording, MIT used *Bugzilla* to accomplish the task, even though many bugs were found and fixed before they could be reported [11].

Stanford's team had the same ability to monitor software modules, restart them when they failed, and respond to known issues as they arose. The team also recorded all data with the ability to play it back in special visualization software. Several different bug tracking methods were used until Stanford settled on the *Trac* wiki with a built-in bug tracker [12]. Team CarOLO also used *Trac* and its included wiki for bug management. Version control for CarOLO was taken care of by *Subversion 16* [14].

Tartan Racing developed the *Tartan Racing Operator Control Station* to start up software modules, monitor the vehicle's health status, and debug software problems. The status of each module was color coded to provide visual feedback on which tasks were running normally, running slow, or if they had failed. Each software module was integrated with *TROCS* through the use of widget plug-ins created by each module's developer. Logs were kept in centralized storage for later playback through the team's simulator. The ability for the vehicle operator to annotate logs during autonomous runs was helpful to provide feedback during simulator playback. *Bugzilla* was used to log bugs and create reports, which were reviewed by the software lead every week [13].

5.4 Testing Processes and Development

Even though every team had the overall goal of reaching DARPA's specifications, the requirements were vague. This led to different approaches to testing and evaluating code

between teams. MIT began their overall test plan by developing qualification tests for the entire project, based on DARPA's requirements. However, due constant additions and changes to code, earlier prepared tests could not be run. Rather, tests were planned as new behaviors were added to the software. Each behavior was tested by itself, rather than integrating it into a larger, more elaborate test as was previously planned [11].

In contrast, Stanford's independent test team was able to create a test plan based on DARPA's requirements and follow it throughout the project. The document consisted of over one hundred pages of tests used to validate correct software operation. Changes were made to the software as tests were failed and bugs were found. The team also made use of simulated DARPA official visits to evaluate the vehicle's performance [12].

Team CarOLO tested the abilities of their vehicle, Caroline, against requirements dictated in "story cards". Each story card depicted a driving scenario, based on DARPA's required behaviors, in which the vehicle had to perform correctly. An independent test team was used to check the abilities of the vehicle every week after it had been updated with the latest software revision. Due to time constraints, CarOLO was not able to test every traffic situation in real life with traffic vehicles; so much of the testing was also done on their simulator [14].

Tartan Racing began their testing development with the creation of a requirements document. Based on these requirements, the team's "play book" was created. This document listed every test needed to evaluate the vehicle's performance, rated in importance and difficulty. Traffic drivers used the play book to coordinate themselves and run consistent tests. Proceeding DARPA's site visit, the team began performing regression tests. These black box tests were run for four hours every Friday and were designed to confirm that the abilities of the vehicle matched those set forth in the requirements document. The software developers were given a verbal summary one hour after each regression test and an executive summary 24 hours after. Endurance tests consisting of autonomous operation

of the vehicle for 6 hours or 60 miles were also performed [13].

5.5 Experiences in Simulation

Many teams at the challenge used virtual vehicle simulators to aid the development of their software. One can suggest that the use of a simulator is almost necessary to be successful in a project such as the Urban Challenge. Safety concerns and strict time constraints must be overcome by testing in the virtual world to validate code quickly and with minimum risk. This section details the experiences several teams had with their simulators.

Princeton's team developed their own simulator in which to test their software. The program displays a 3D interface, simulated dashboard, and a map of the desired RNDF. Figure 5.2 shows Princeton's simulator in action. Through the use of the Microsoft Robotics Studio framework, the team was able to test their code in the simulator and then transfer it to the vehicle without the need to recompile [10].

MIT's simulator consisted of a dynamic model of their vehicle running in a virtual environment in which random or scripted traffic vehicles could be added. Through the use of sim-modules integrated in their software architecture, the vehicle could be run in the simulator in real time as if it were running autonomously in the real world. Through testing in the simulator, the team checked for bugs in their software and ran stress tests to find problems such as core dumps and memory leaks in software modules. As was the experience for other teams, MIT's simulator was not able to simulate realistic sensor data. It could play back data recorded in real life test runs, but simulated obstacles reflected perfect data, something that actual sensors cannot obtain [11].

Stanford's software does not distinguish between driving the vehicle, replaying a log file, or driving in the simulated world. Therefore, almost every part of their code could

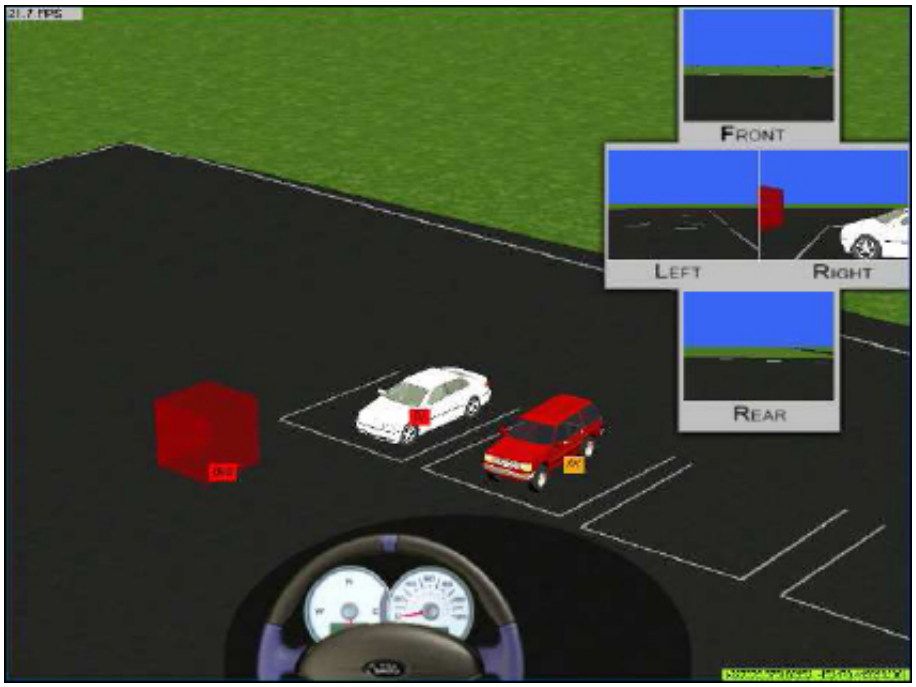

Figure 5.2: Princeton tested their software in a 3D virtual environment [10]. Figure used with permission from PAVE.

be tested in simulation. Their simulator was used to save time and reduce safety risks during testing. The team was able to test situations too dangerous to involve humans and test more scenarios than would have been possible in the real world. Simulation was used to find obvious problems with their software, but this was always followed by testing on the vehicle [12]. Figure 5.3 shows Stanford's simulator replaying data from the Urban Challenge final event.

For some teams, simulation was available but not used enough be effective. This was the case for Intelligent Vehicle Systems and their vehicle, XAV-250. Their simulator was used to review data sets from real world tests and point out problems with their software. However, it was only used for the final month of testing and did not provide them with enough data to reveal all the situations in which their vehicle would have become stuck. One such case occurred when their vehicle perceived a deep rain gutter as an obstacle, causing it to sit at a stop sign during the final event. DARPA retired their IVS's vehicle

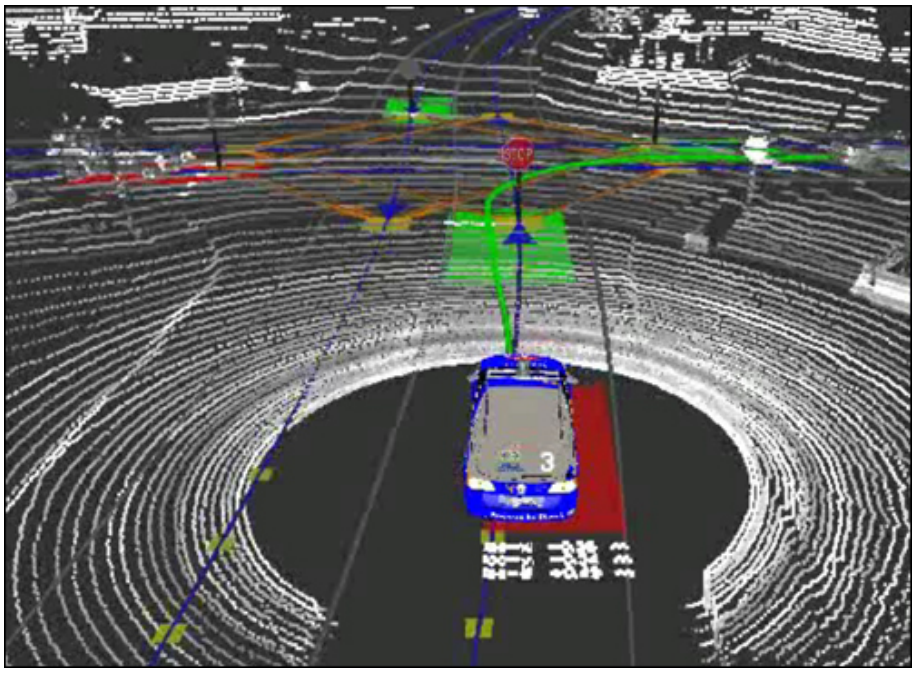

Figure 5.3: Stanford's simulator has the ability to display recorded data. On the right, a car can be seen pulling through the intersection. On the left, Odin can be seen stopping at the intersection [16]. Figure used with permission from Stanford Racing.

from the race due to this flaw. The team is confident that this situation would not have presented a problem had they run similar data through their simulator during testing [15].

CarOLO used their simulator to test new software implementations before adding them to the vehicle, as well as confirming bugs found during real world tests. Further development of their simulator has yielded a version in which multiple instances of their autonomous vehicle could be operated. In doing this, their software could learn efficient driving behavior in an environment in which multiple traffic vehicles exist. In addition, different versions of code could be run from the same starting point, running the same mission file, in order to compare their performance [14].

Tartan Racing made the decision early to spend time developing their simulator before vehicle code was created. Doing this saved the team time and effort in the end. Simulation was used to test hazardous behaviors before real world tests, and to play back recorded data. Tartan's simulator also had the ability to add virtual obstacles to a real world envi-

ronment during testing. In doing this, the vehicle was made to think there were obstacles in front of it even though there were none [13].

Chapter 6

Team Victor Tango's Urban Challenge Testing Experience

The testing methods to validate Odin's development were far from being consistent. Testing and validation strategies varied depending on the location, the project timeline, and lessons learned from previous experiences. As the project progressed, the team recognized flaws in its testing strategy and evolved their methods to eliminate them.

From the onset of the project, it was clear that the team required an independent testing group to evaluate and validate the performance of the software developed for the vehicle. It was known from previous experiences in similar projects that the developers themselves cannot exhaustively test their own code. The purpose of the test group was to subject the vehicle to thorough tests throughout its development to determine if problems in the code existed.

The primary strategy for evaluating code consisted of individual testing combined

with comprehensive validation tests. Individual tests were carried out by software developers to find obvious errors and to determine if their code functioned properly. Validation tests were conducted by the testing team to find any and all problems with the code that were not found during individual testing. Schedules spanning several months were created and included milestones for validation tests. The team's progress on these milestones was evaluated during weekly meetings to make sure development was on track. Online scheduling software was used to schedule time with the vehicle and other resources to ensure there were no conflicts.

To track code changes and documents used throughout the project, team Victor Tango used *Subversion*, a version control software that utilizes a central repository. As individuals added to their code and made changes, it was important to keep track of new versions and make them available to others on the team. *Subversion* provides a method for doing so by keeping software modules in a central repository to which all team members have access. The program also provided the ability to revert to an older version of code in the event that a new version exhibited any drastic problems. By tracking changes made to software by different individuals, *Subversion* was able to eliminate any discrepancies in code on different computers.

Extensive data logging was conducted during all tests. During each test, data from every sensor and software component was logged and stored in a central server, and could be played back for further evaluation. Due to the large amount of data coming from the vehicle's sensors and software modules, only so much can be viewed in real time during tests. The playback of data logs in the simulator allowed for more in depth analysis of testing events, leading to the discovery of errors that were not apparent during the actual tests.

To save time creating new courses and RNDFs, geo-referenced aerial imagery was used. Figure 6.1 shows the roads and a parking lot plotted in the *RNDF Tool*. Courses

could be plotted by simply clicking on the image to obtain the coordinates of a desired waypoint. Existing lanes and boundaries were also edited using the software. Due to the resolution of most available aerial imagery, this method was better suited to large paved areas and parking lots. Much time and effort were saved by eliminating the task of surveying courses with GPS. The extent to which the vehicle was tested would not have been possible had the multitude of testing courses not be created in software.

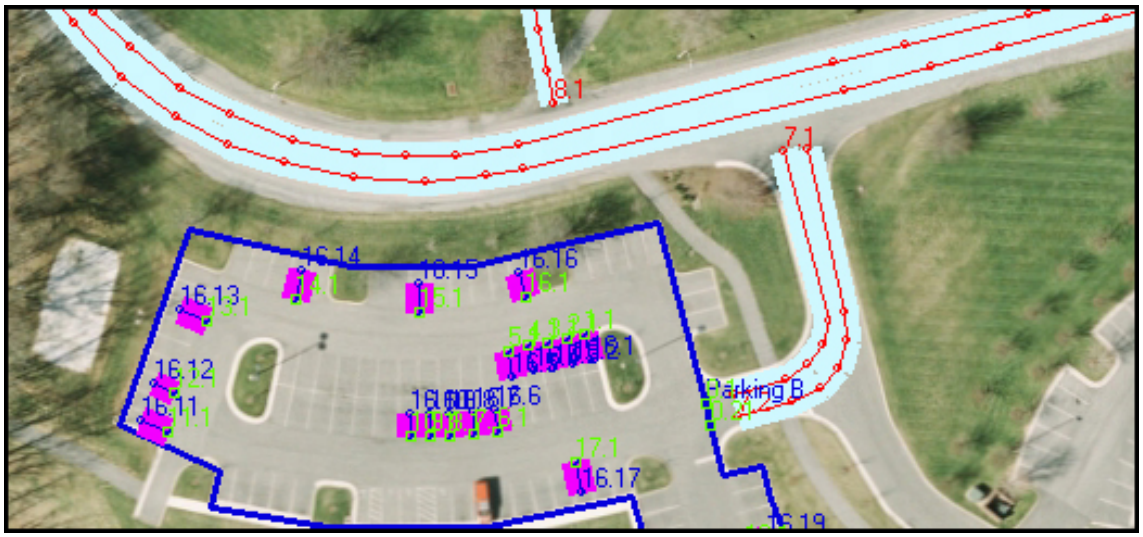

Figure 6.1: Team Victor Tango used aerial imagery to plot RNDFs without the need to survey points manually.

6.1 Winter & Spring of 2007

During the early vehicle development stages, the team did not have a designated testing facility. At this point, Odin's drive-by-wire conversion was taking place in a shed on the Virginia Tech campus. Normally, this shed was used for storing gardening and lawn care equipment. But, thanks to the horticulture department, space was made to house and work on the vehicle.

At this point in Odin's development, the steering and speed control software were the

primary focus of testing. These simple tests were usually conducted on a relatively level road near the university-owned cow and horse pastures. Given its location, geometry, and low traffic, the road was well suited for low level control strategy development.

As the software progressed, new behaviors such as waypoint following were ready for testing. For these more complicated tasks, a parking lot on campus was closed off on the weekends. Courses were surveyed and marked with cones. Although not as large as the Cage, this parking lot was certainly useful in the validation of more basic navigation software.

Testing during this time was very simple. Test plans were written up the week of the tests and carried out at the end of the week. Tests did not include traffic vehicles or stationary obstacles, as the software was not ready to handle these challenges. Bugs were often found and corrected after several hours of testing. Tests concluded with a brief test report to provide the team with information on the vehicle's performance.

6.2 The Cage

By this point, the team had moved into its new lab and office space. The former automotive repair facility, given the name "The Bot Cave" by the team, provided plenty of space to store the vehicles and equipment. A room in the back was used as office space and contained a lavatory. The lab was located off campus, but only a three minute drive from the Cage.

During the team's time in the Cage, the primary goal was to prepare Odin for DARPA's site visit. The vehicle was required to perform several different behaviors set forth by the agency before it could be considered for entry in the Urban Challenge. Code development primarily consisted of testing by individuals during the week, followed by several planned validation tests at the end of the week. The site visit course configuration was used for all

the testing, as DARPA was to run their tests on the same course.

Individual testing was conducted with one to three software developers in the vehicle, accompanied by a safety driver. After code was written and deemed safe to use on the vehicle, the individuals responsible for that particular module performed their tests. These tests were often short, simple runs demonstrating the particular behavior was performing as expected. The vehicle often did not travel any more than a few hundred yards before accomplishing its task. For example, during testing of Odin's passing capability, a disabled vehicle would be placed in Odin's lane with the intention that Odin would pass it after waiting ten seconds. Odin was started about forty yards behind the vehicle, would successfully pass it, and return to its lane. The safety drive would then take control of the vehicle and reposition it for another run, while any necessary changes were made to the code.

In contrast, the software validation tests were more complicated. The point of these tests was to attempt to put Odin in situations in which the vehicle would become stuck or would perform incorrectly. Test plans were well thought out and often involved many team members running Odin and traffic vehicles, coordinated by a single test lead through the use of radios. Several scenarios were presented to the vehicle, often involving more complicated maneuvers than those used during initial testing. For example, where Odin's original passing test may have consisted of a single vehicle, a passing validation test would involve passing multiple vehicles one at a time or in succession. In these initial tests, it was found that placing disabled vehicles at certain distances from each other would cause Odin to get stuck returning to the traveling lane. Because of observations like this, the code was changed to allow Odin more freedom in its passing distances.

As the code evolved, the use of traffic vehicles on the test course increased. Behaviors such as passing disabled vehicles, following, and intersection precedence all required manned traffic vehicles. During these tests, it was necessary for the traffic drivers to be

cautious, but to act as if Odin would perform correctly in every case. This meant that Odin's safety driver was primarily responsible for stopping the vehicle in the case of an incorrect action on Odin's part. In this respect, the fact that Odin could be switched into manual operation quickly was imperative.

The experience in the Cage was very positive overall. The strategy of individual testing followed by software validation tests worked well. However, as circumstances changed, the strategy changed. First, the code was still in a simple form, when compared to that used during competition. This allowed people to make changes to their code without strongly influencing the behavior of another module. Second, the situation being prepared for (site visit) took place on a simple track. As the requirements became more complicated, the course had to include more aspects such as zones, traffic circles, and four lane roads. An attempt at adding these factors to the original course was made after site visit, but was seldom used because the scale was not large enough. Third, because the behaviors being tested were relatively simple, deadlines for software validation could be met without having to push the timeline back. Deadlines became more of an issue when more complicate behaviors were added to Odin's repertoire later in the project.

6.3 Kentland Farm

The majority of the time spent at the farm was used for individual testing. Given the narrow width of the roads, it was difficult to test any situations involving more than one lane. However, basic behaviors such as road coverage and sparse waypoint navigation were still tested.

The use of the farm, although limited, was a valuable experience. The team learned to cope with limited resources, which would be an asset while testing at the Urban Challenge event. The long, winding roads were used to work on sparse waypoint navigation, even

if only involving one lane of travel. Several intersections were used to practice merging using different intersection configurations. Another useful trait of the site was the absence of paved roads in certain sections, which was used to test road following with sparse waypoints. The team was also unsure how well the vehicle would handle on loose ground, so the gravel roads were helpful in that respect.

The individual testing strategy worked well at the farm. The team was allowed access at all hours, making it easier to schedule vehicle use. Private vehicles were also allowed on site, so testing with traffic could be carried out.

6.4 The Corporate Research Center

The CRC contains a network of roads and parking lots, making it an excellent site for testing in an urban environment. TORC's offices on site provided a place to work on code between testing times on the vehicle. The main catch to testing at the CRC was that the offices on the grounds were in use during the day, so night testing was the only option.

A mix of individual tests and validation tests were performed at the CRC. Groups were able to test numerous behaviors including passing, merging, and parking during the week. Validation tests were performed at the end of the week to ensure that the code was being developed on time and worked properly.

Testing at the CRC was often unproductive. At this point in the project, the code was reaching a much more complicated level. Making small changes in one piece of code sometimes resulted in malfunctions in another module. Often, changes made to Odin's code were not tested immediately after they had been made. This resulted in groups going out to test, but not being able to due to malfunctions in software changes made prior. Many hours of valuable testing were lost because of these problems, resulting in missed deadlines and schedule changes.

6.5 Roanoke County Police Training Facility

The team was allowed the use of a police training facility in Roanoke County on several occasions for a few days at a time. The site was conducive to testing several different software behaviors, given the layout and features it provided. The main road climbs several hundred feet before exiting onto a large paved area. The use of the road helped to test the driving control software, focusing on driving straight, climbing hills, and negotiating sharp curves. Extensive tests involving line detection and following were also performed using the clearly painted line markings on the road.

Depending on the behavior being tested, several RNDFs were created utilizing the large paved area at the top of the facility. Courses involving intersections were used when testing merging and left turn behaviors. Disabled vehicle passing tests were conducted on large loops of two opposing traffic lanes. The large open area was also conducive to acceleration testing. By placing lines of cones on either side of a lane, the vehicle was tested exiting a start chute onto the main course, a necessary behavior of the Urban Challenge final event. Different course configurations were constantly being created, providing new road geometries for groups to test on.

Traffic vehicles were essential in testing intersection, merging, and passing behaviors. At this point in Odin's development, the intersection precedence and merging modules were being fine tuned. Tests involved measuring the range of the IBEO sensors and timing merges with moving traffic, driven by humans. The disabled vehicle passing behavior was developed enough to deal with oncoming traffic. Human driven traffic vehicles were used to test Odin's ability to sense an oncoming vehicle and wait for it before proceeding to pass a disabled vehicle.

Overall, testing at the facility was very successful. The ability to use geo-referenced imagery to create many course RNDFs was essential to software debugging and testing. Had the RNDF repertoire for the site not been so diverse, some software behavior prob-

lems may not have been found. Each course had its own purpose in exposing a certain flaw in Odin's software. While the vehicle performed well in the majority of these tests, there were several that posed problems that it was unable to overcome, leading to improvements in the software. The fact that the entire team was present on these days made fixing software problems much easier. When software modules exhibited problems, there was always someone there to fix it. Because of this, testing was conducted continuously throughout the day without serious delays. After the time spent at the facility, it was clear that testing should be conducted with most of the team on site to ensure that things ran smoothly and efficiently.

6.6 Pre-Compeition Testing

The final week of testing was mainly spent at the CRC during the night. The team spent this time finalizing the code before the vehicle was shipped to Victorville. The CRC contained many of the elements the team thought would be present in the final event. Some of the main goals of testing at this time were to finalize the zone navigation software and to run some endurance runs with traffic vehicles.

Zone navigation and parking behaviors were tested in a parking lot down the road from TORC's offices. Parking behaviors were first tested without traffic in the zone to make sure the parking and backing out behaviors were working correctly. Several runs were made, commanding the vehicle to parking in different parking spots during each run. Approaches from different angles were executed to determine if the software would have any trouble. With every test, the vehicle was always commanded to enter the zone first and exit the zone when the parking maneuvers were complete. In doing this, the team made sure that the zone navigation behavior was tested in its entirety, eliminating the starting position variable from the equation. This consistency also made it easy to

compare new tests to previous ones as changes in the software were made.

Zone navigation testing continued with the addition of traffic vehicles. Unlike vehicle avoidance on roads, vehicles in zones could be anywhere within the boundary, traveling in any direction. The one assumption the team had been given by DARPA was that vehicles in head-on trajectories would turn to the right to avoid each other. Odin's ability to wait for a vehicle after parking was testing by driving traffic vehicles behind it after Odin had parked. This did not pose much of a problem for the vehicle. Dynamic obstacle avoidance in zones was preliminarily tested by driving a traffic vehicle head on towards Odin to test the "turn right" avoidance behavior. Not only was this a test of Odin's avoidance capabilities, but it also served as a safety test. Satisfied with Odin's ability to avoid head on collisions, the vehicle was then commanded to perform several parking maneuvers in the same zone as a moving traffic vehicle. The traffic vehicle driver drove in random patterns throughout the zone, utilizing several approach angles and distances.

Endurance tests throughout the site were conducted utilizing as many as four traffic vehicles. Scenarios involving intersections, zones, following behavior, and disabled vehicle passing were all used. Odin's performance in these tests was excellent and few problems were found.

This week of testing was very successful. The team learned from their experience at the police facility that testing in a large group led to less delays and more efficient use of time. When problems with another individual's code did arise, they were sorted out on site within minutes, rather than left to be fixed the following day. The CRC became an excellent test site given its layout and geography. The site contained roads, parking lots, intersections, and dead ends. All of these aspects were necessary to test Odin's ability to navigate an urban environment. However, additional aspects such as buildings, hills, bushes, trees, and signs completed the successful testing atmosphere. Buildings and trees posed the potential for occluded GPS signals. Hills provided elevation changes that tested

sensor ranges. Sensors were also tested by objects next to road such as bushes, signs, and road cones.

6.7 Testing at the Urban Challenge

Through the Urban Challenge qualifying event, each team was required to successfully complete three qualification courses. Between runs, teams were allowed to make changes to their code and test their vehicle in a designated area. Odin did not perform perfectly during all its qualification runs, so testing for Victor Tango did not stop once reaching Victorville.

To test code changes on the vehicle, the team was allowed to use two specially designated dirt practice areas. The coordinates of the boundaries to these areas, along with 6-inch resolution aerial imagery were provided to the teams before the start of the qualifying event. In this case, it was necessary to have a program in which RNDFs could be laid out using the provided aerial imagery. Limited access to the course prevented detailed GPS surveying every time a new practice RNDF was to be made. The time it would take to gather GPS points would have taken much too long. RNDFs for various testing scenarios were created within minutes using aerial imagery.

In a situation where resources were limited, improvisation proved to be a key factor for success. A certain obstacle observed on one of the test courses looked like it would cause some trouble for Odin. The obstacle consisted of a horizontal suspended pole, about 4 feet off the ground, blocking the road. Six stop signs were bolted along the length of the pole. In previous tests, only obstacles sitting on the ground had been avoided, never something suspended in mid air. Unsure how the vehicle would react to, or even be able to see such an obstacle, Team Victor Tango constructed a mockup of it out of wood and cardboard. The mockup was placed in the practice area on an RNDF consisting of a straight road to

test how the vehicle would deal with it. At first, Odin was not able to see the obstacle and react to it in time. However, after some adjustments to the code and the front sensor mounts, Odin recognized the blockage in time and performed a u-turn. This simple test proved to be successful when Odin was able to do the same thing in the qualifying course when the situation arose.

Once again, traffic vehicles were needed during testing. Although competition safety rules prevented human drivers from being in traffic vehicles while Odin was in autonomous mode, they played an important part as large obstacles to navigate around. One of the qualification courses included a large obstacle field in the middle of the road in which the autonomous vehicle was required to navigate through. On its first attempt, Odin was unable to navigate the field, resulting in the vehicle becoming stuck in the middle of the road. It was determined that Odin's vehicle passing behavior was too cautious, and changes were made relax Odin's passing standards. Tests were conducted in which many obstacles, including cars and traffic cones, were placed on an RNDF with a similar geometry to that in the qualification course. With a few changes, Odin was able to navigate around the obstacles at speed.

Making changes and testing software during the Urban Challenge presented various problems due to the limitations imposed. The test areas were large, but only consisted of a dirt lot. Even though they were both rectangles, the orientation of each practice area was different, making the creation of two practice RNDFs for each scenario necessary. Practice areas were only available for half an hour at a time. Because of this, testing had to be planned out before hand and carried out efficiently.

The team was able to overcome these challenges and test its software successfully. The willingness to be proactive and to improvise led to this. One key tool used during all of Odin's testing was the simulator. The next section covers simulation testing in detail.

6.8 Simulation

A large part of testing was done in simulation. Team Victor Tango developed its own simulator in which almost all aspects of the vehicle's code could be tested. This section goes over a description of the simulator, its positive aspects, and its drawbacks.

The simulator was mainly used to test code changes before they were committed to the vehicle. The program also provided the ability to play back data logs to view real world tests in the virtual environment. The program consisted of a graphical display in which a 3D model of Odin navigated a virtual environment. RNDF's, obstacles, traffic vehicles, and aerial imagery are added to the environment by editing a simple xml file. Static obstacles ranged from trees and traffic cones to houses, and could be placed anywhere in the scene. Traffic vehicles followed a predetermined route by specifying waypoints for them to navigate to. Aerial imagery could also be added to each scene, overlaying a proper RNDF on top. Figure 6.2 shows the simulator in action.

Odin's software modules communicated between each other through the use of the Joint Architecture for Unmanned Systems. This messaging standard was designed to improve code interoperability between different autonomous systems [17]. This aided in simulation testing by allowing the code to run on laptops just as it would have on the vehicle. Through the use of the JAUS architecture, setting up tests in the simulator was very simple and required only two computers to run the proper software modules. During much of the project development timeline, the simulator was mainly used by the code developers. However, about a month before the vehicles shipped to Victorville, more team members were running various test scenarios in simulation to look for small problems in Odin's code. Finally, at the Urban Challenge, simulation was constant. Knowing the situations the vehicle was to go through during the qualifying tests, the team was able to test them out in the virtual world before the vehicle was tested. After the final RNDF was given 24 hours before the final event, simulation was conducted in shifts up until a few

Figure 6.2: Static obstacles, traffic, and RNDFs can be displayed over aerial imagery in Team Victor Tango's simulator [18].

hours before Odin was sent off. This was done to find any last minute problems that might pop up due to the course geometry. Testing this way made a big difference when it came to zones and parking situations. DARPA had marked every parking spot in the zones, making for some tricky geometry when maneuvering around the lanes. Certain changes were made to the code allowing Odin a little more freedom to move around. Had these changes not been made, the chances Odin would have become stuck were very good.

The ability for the simulator to play back data logs from real world tests was very important. It was difficult to spot problems in the software in real time during testing, but the simulator provided the team with a tool for replaying real world tests to view the data. In playback mode, the simulator displayed Odin's vehicle model, the RNDF, and

any aerial imagery that was applied to the scene. Virtual obstacles were replaced with data obtained from the vehicle's sensors, showing objects and their boundaries. Data could be played back and paused at any point within the test to view the world as Odin saw it.

There were many reasons why simulation was a necessity during the vehicle's development. Software changes had to meet a certain level of operation before they were put on the vehicle. It would have been inefficient and dangerous in some cases to add code to the vehicle before it was tested to some degree. At some points during development, time on the vehicle was at a premium. Some tests sites were only available at certain times (such as the CRC and the police facility) and many people had code that needed to be tested.

Testing in simulation was also a lot safer, efficient, and cheaper than testing on the vehicle. There was no need to worry about the safety of pedestrians and traffic drivers in a simulated environment. The team was also able to place virtual obstacles in the real world, much like Tartan Racing did. No time was spent setting up and marking off a course, directing traffic drivers, and carrying out safety procedures. Virtual testing also resulted in the use of less gasoline and reduced wear on the vehicle.

While perfect sensor data is necessary to evaluate the behavior of a vehicle in ideal situations, these scenarios do not reflect the real world accurately. Object classification is a tricky subject when dealing with obstacles that are oddly oriented to the vehicle's sensors. Some surfaces do not reflect laser pulses well, where others reflect too well. For example, well painted road lines can be picked up with an IBEO XT system and reported as an obstacle, when in fact the vehicle can drive over them.

GPS signals in the real world are not constant and are often occluded by tall buildings and trees. A bad fix resulting from signal occlusion can result in position drift up to several meters. Uncertainty in Odin's position would increase from 0.1 meters to 2.87 meters RMS after just one minute of signal loss [8]. These jumps can cause a vehicle to

radically change its heading to correct its course when it is truly in the correct position in the first place. The ability to test these situations in simulation must be present, but can be very difficult to implement. In order to represent a GPS blockage in simulation, satellites and their orbits would need to present in simulation. A position outage for the vehicle would exist when line of sight between it and several satellites is broken. To be completely comprehensive, even signal reflections off buildings and other large objects would need to be simulated. This is something that Team Victor Tango did not have the means to implement, and would be a daunting task for other groups. The computing power needed to introduce such simulation would be quite large.

Perhaps the most difficult sensor to test in simulation is vision based. Cameras are used in the autonomous ground vehicle industry mainly to detect roads and lane markings. While vision algorithms can be tested using simulated roads and markings, testing using actual cameras introduces new variables. In the real world, vision algorithms must deal with shadows from overhanging trees, direct sunlight, and weather phenomena like fog and rain. Once again, these factors could be introduced into simulation, but would require large amounts of processing power and development time.

While stationary obstacles are easy to simulate, traffic is a different issue. The traffic vehicles in Team Victor Tango's simulator are able to perform simple behaviors. These behaviors are limited to waypoint navigation, stopping at stop signs, and following. Simulated vehicles lack the ability to follow intersection precedence, avoid obstacles, and merge correctly. If traffic vehicles were simulated perfectly, they would act as if a human driver were in control. There are two methods to accomplish this. The first would be to assign a human control of each vehicle. This would require more individuals to be involved with simulation testing, one of the factors simulation is aimed at eliminating. The second method to creating realistic traffic vehicles is to make software responsible for controlling them. However, the Urban Challenge project was aimed at developing

software for exactly this task, so this is currently not an option.

Finally, terrain must be taken into account. Even though the Urban Challenge was held in an area with relatively small changes in elevation, much of the real world covers area where changes in terrain are much more drastic. Once realistic sensor simulation is implemented, changes in terrain must be simulated to test the limits of these sensors to see down hills and around corners. The effects of gravity must also be simulated to assess the vehicle's performance climbing and descending hills. Even different ground surfaces such as loose gravel and dirt can be simulated on which the vehicle's control can be checked.

The idea of a perfect simulator may never be realized. To have the perfect simulator would mean having a program the can create a virtual world that is indistinguishable from the real world. By the time the technology exists to accomplish this goal, autonomous vehicles may very well be developed and used mainstream.

Chapter 7

A Proposed Autonomous Ground Vehicle Testing Strategy

This section proposes an autonomous vehicle testing strategy based on the information presented above. Facility requirements, safety measures, simulator abilities, and an overall method for autonomous vehicle testing utilizing these assets will be covered in detail. It is hoped that this proposal will be used as a guide by other groups to aid in testing and validating similar vehicles.

7.1 Facility Requirements

In testing an autonomous ground vehicle, it is important to have a facility that contains features conducive to validating every behavior the vehicle must perform. Autonomous vehicles have evolved from the days of traversing the desert on a single lane path into nav-

igating complex urban environments. The facility in which these vehicles are put through their paces should most closely resemble such an environment, testing every sensor capability the vehicle has.

As was the case in the Urban Challenge, global positioning is used as the main mode of navigation for an autonomous vehicle. Traveling from one point to another becomes very difficult when the software does not know where the vehicle is in relationship to those two points. GPS signals are often taken for granted as being consistent, when in fact many things can cause degradation or even complete loss in signal. A vehicle's position accuracy can be degraded when GPS signals are reflected from large objects like buildings. In an area where tall buildings exist on both sides of a road, often referred to as an "urban canyon", GPS signals can be blocked completely. Even different times of day can result in fewer satellite signals available. Autonomous systems must have the ability to deal with such challenges in order to continue to navigate to their destination. It becomes necessary for an adequate testing facility to contain tall structures such as office buildings and trees to provide a vehicle with such situations.

Cameras offer another method to help vehicles navigate their environment. Currently, cameras are used to detect road coverage and lane markings. In the future, they may be required to recognize stop lights and road signs. In the case of road coverage detection, a facility must have many different kinds of road surfaces. Asphalt, cement, gravel, and dirt roads are some of the most common surfaces on which a vehicle may drive. All must be tested. Lighting also affects a camera's performance. Often, high contrast can affect a camera's ability to adjust its aperture. Due to this, areas of dark shade between areas of direct sunlight must exist to test on. Operating at night, with and without street lights, will test the other extreme. The environment in which the facility exists also plays an important role in testing optical systems. Weather such as rain, snow, and fog will inhibit the camera's ability to view the world. Operating in an area where these conditions exist

will lead to more thorough testing of optical sensors and recognition software.

Dealing with static objects and other traffic vehicles is necessary in order to navigate an urban environment. The primary concern with operating autonomous vehicles on roads with other vehicles lies with the sensor array's ability to detect vehicles and other obstacles. A proper testing facility should have many different configurations of roads, intersections, and parking lots for the vehicle to travel. Intersections of varying dimensions will test the vehicle's ability to detect traffic at other entrances and determine precedence. Roads with tight curves will test the vehicle's reaction to obstacles that are not visible until after turning through the curve. Hills also help to obscure obstacles when a vehicle is climbing and descending. It is impossible to test a vehicle on every possible road configuration during a project's timeframe, but testing on the more prominent ones should reveal most problems a vehicle's detection system will have.

Not only do road geometries test a vehicle's sensing ability, but so do the characteristics of the obstacles themselves. Different surface finishes reflect LIDAR differently. Smaller objects are harder to detect than large ones. Sometimes obstacles are suspended above the ground, such as an overhanging sign. Negative obstacles such as pot holes and storm drains can put a vehicle out of commission, and so must also be detected.

Not only must a proper test facility have sufficient features to test an autonomous vehicle, it also must have features to keep the developers productive. Collaboration on such projects is key to successfully integrating a cohesive software architecture into the system. Working in a group also improves efficiency and lowers development time. A central office location keeps everyone in the same area and allows ideas to flow freely. If at all possible, the office area should be located on site. Problems that arise during testing can be fixed faster if the necessary personnel are near the vehicle. In the same respect, the vehicle maintenance facility should be located on or near the test site to ensure faster turnaround when fixing mechanical problems.

Amenities in the office space should include electricity, ample seating and table space, heat, air conditioning, lavatories, and internet access. White boards and projectors also help convey ideas and keep track of daily tasks. Internet access in the office and on the vehicle is essential. Software problems can be fixed wirelessly and centralized software storage can be updated.

Of course, the odds that a facility like this actually exists are very small. When looking at testing sites for an autonomous vehicle, evaluating aspects ranked in terms of importance can help make the decision easier. Choosing the proper facility can save much time over a project's lifetime, so it is important to get it right from the start.

7.2 Safety

Safety is paramount when testing an autonomous vehicle. In such a new field of research where a simple sign error in the code can cause drastic problems, it is important to be prepared for the worst. There are several steps that can be taken to minimize risk during testing.

The easiest safety precaution to take is to keep personnel away from danger. This means keeping the test course closed to non-team members, keeping team members away from the vehicle when it is in autonomous mode, and keeping traffic drivers in their cars when the vehicle is in autonomous operation. Radios should be used to communicate team member positions and to coordinate autonomous runs. While personnel are at the test site, a building can provide a safe place to observe the vehicle. Obstacles such as concrete barriers can also provide protection against a rampaging vehicle.

The next step in safely testing an autonomous vehicle is to provide several methods for disabling the system when problems arise. Unless there is an official demonstration being conducted, there is no reason not to have a safety driver in the vehicle. A safety

driver should have the ability to take control of the vehicle with a simple action such as pushing a button or flipping a switch. Disabling the vehicle through an emergency stop inside the vehicle is also an option. E-stop buttons inside and outside the vehicle should be placed for easy access. Finally, a wireless transmitter should always be on hand to stop the vehicle in the case where the safety driver does not realize there is a danger. However, the emergency stop is used to disable the vehicle rather than merely transferring control to a human. Due to this, the E-stop system should only be used during an actual emergency in order to use testing time more efficiently.

Finally, operating procedures and safety briefings should be used to keep everyone informed. Procedures can be used to clear people from the course before autonomous operation, notify the team and traffic drivers of the start of an autonomous run, and to inform the team when an autonomous run has concluded. Briefings will inform team members of what to expect during a day's testing so that there are no unexpected surprises.

7.3 Testing Strategy

This section proposes an overall strategy for testing and validating the operation of an autonomous vehicle. Using the experience and knowledge gained from working with Team Victor Tango, as well as input from the other teams mentioned earlier, suggestions are made to improve vehicle testing and software validation. Following these guidelines will result in effective testing and efficient use of time.

First and foremost, an independent test team should be created. The purpose of this group is to undertake all of the tasks of testing and validating the vehicle software. It is important for this group to follow up any preliminary testing done by the developers with detailed validation tests covering any and all situations the vehicle may be involved in. It is almost guaranteed that an independent test group will find problems with the software

that were not found by the developers.

Before any testing can start, there must be a clear schedule for the vehicle's development. The schedule should detail the vehicle design, construction, software development, and testing phases of the project. Task leaders should be assigned to keep their respective portions of the project on schedule. Creating a full, well thought out schedule at the beginning of a project will keep development on track and minimize delays due to unforeseen circumstances.

It is essential to have a central repository to keep track of software versions, bug changes, and documents. Keeping track of software changes will eliminate discrepancies between different computers. Since different people often work on their own software modules, it is important that new versions are updated on a central server and verified to work with other modules. Programs such as *Subversion* and *Trac* provide easy to use, automated methods for version control and bug management. A central repository also provides easy access to documents such as wiring diagrams, software tutorials, images, and other technical documents. Easy access to such documents can save an individual time from tracking them down. Also, a program used for scheduling and resource management will eliminate conflicts and overlaps.

It is important to adopt a standardized navigation format before testing begins. Such a standard should include a primary mode of navigation, such as GPS, supplemented with other guidelines such road markings, stop points, and speed limits. The RNDF and MDF formats used by DARPA for the 2007 Urban Challenge served this purpose very well. Using a standard, unchanged navigation format will help keep tests consistent and meaningful.

As several teams have found out, the importance of a simulator can not be overstated. Simulators can provide valuable feedback on software changes even before the vehicle is ready for software implementation. Unless one is readily available, it is important to

spend time developing a fully functional simulator before starting work on vehicle software. Although simulation cannot include every aspect present in the real world, there are minimal requirements. Obstacles, traffic (random, scripted, or human-controlled), aerial imagery, elevation changes, and a simulated vehicle model should all be implemented in simulation. Creating an effective simulator early on will provide the team with a valuable tool to validate software operation to a degree before it is implemented on the vehicle.

Once all of these preliminary measures have been set in place, a test plan spanning the entire project schedule should be created. This plan should describe, in detail, every test to be carried out on the vehicle and software modules to validate their correct performance. Simulation and real world testing should both be covered based on prescribed project goals. Taking time to produce this document early on will result in more productive testing later on. This ensures that every subsystem, software module, and overall vehicle performance is thoroughly tested through the project timeline.

As far as actual testing goes, the process should combine simulation and real world testing for a number of reasons. Simulation provides a method for testing software easily and safely, without having to use time and resources to test on the vehicle. Due to safety concerns, software changes can be tested in the simulator first to validate a minimal level of operation. Any obvious bugs that cause undesired performance should be found in this step. Since testing time on the vehicle can be at a premium, simulation tests can be run quickly without eating up time on the vehicle. As with real world tests, simulation tests should be set up beforehand so that they can be executed in succession when called for in the project time. Often, simulation tests are run by the test team and not the developers. All data should be logged during each test run so that it can be replayed and scrutinized by the developers later if problems arise.

Finally, due to limits in the virtual world, no amount of simulated testing alone can fully validate the successful operation of an autonomous vehicle. Real world tests must

be conducted to bring out all the nuances of the real world that a simulator cannot imitate. Because real world tests need time to set up and run, it is important to have them well planned out ahead of time. A document, such as Tartan Racing's "play book", can provide all team members with details on the course, mission, obstacle placement, traffic vehicle patterns, and overall goals of each test. Such plans will save time in setting up and running tests. Thorough real world validation will put the vehicle in every situation it is designed to handle. However, due to time constraints, this may not be possible. Ranking tests in levels of importance will ensure that more common situations will be tested first, regardless of their difficulty.

When conducting tests on the vehicle, taking the time to set them up and carry them out methodically will result in fewer test runs and a more efficient use of time. Radios should be used to coordinate traffic driver positions, obstacle placement, and safety confirmations. To speed up the correction of software bugs that could cut testing short, team members responsible for software modules should have access to the vehicle's computers. Equipping the vehicle with wireless internet will provide these individuals with the ability to fix problems remotely, rather than on site.

Sufficient data must be collected to be able to replay every test. The software developers must be able to see the obstacles that the vehicle saw as well as how it reacted to them. Appending notes at certain points in the replay data will better explain the situation to someone who was not present during the test. The best way to make sure data is collected is to make the process automatic. Every time the vehicle is put into autonomous mode, data logs should be created by software. This eliminates the possibility that a human could forget to start the logging process. Once all tests are concluded, replay data should be uploaded to the central server for access by all other team members.

Preceding every software validation, the test group must evaluate the vehicle's performance based the test's goals. Follow-up reports covering the test, the behaviors that

were tested, and the vehicle's performance should be sent to every member of the team. In depth analysis of each test is just as critical as carrying out the tests themselves.

7.4 Conclusion

With advances in technology and the increasing need to keep transportation safe, the development of fully autonomous ground vehicles will become prominent in the unmanned systems industry. Testing these vehicles thoroughly will ensure that they operate and respond safely and efficiently. By taking time to evaluate goals, create and stick to a schedule, and test every aspect of code, bugs and problems will be found. Simulation is a great tool for development, but nothing can replace testing in the real world. Finally, it is important to put as much effort into analyzing the results of each test as was put into creating and carrying them out.

References

[1] National Highway Traffic Safety Administration. *Fatal Analysis Reporting System Encyclopedia.* 2006. Available Online: http://www-fars.nhtsa.dot.gov

[2] "Incident Clustering: Diagnostic Approach for Assessing Usability of Intersections and Other Road Sites." *Transportation Research Record.* Vol. 1897 (2007): 173 - 179.

[3] iCasualties.org. *IED Fatalities by Month.* January, 2008. Available Online: http://icasualties.org/

[4] Floyd D. Spence National Defense Authorization Act for Fiscal Year 2001. *Section 220. Unmanned Advanced Capability Combat Aircraft and Ground Combat Vehicles.* Public Law 106-398, October 30, 2000.

[5] Defense Advanced Research Projects Agency. *Overview* Available Online: http://www.darpa.gov/

[6] Defense Advanced Research Projects Agency. *Urban Challenge Technical Evaluation Criteria* March 16, 2006. Available Online: http://www.darpa.gov/

[7] Defense Advanced Research Projects Agency. *Route Network Definition File (RNDF) and Mission Data File (MDF) Formats.* March 14, 2007. Available Online: http://www.darpa.gov/

[8] Reinholtz et al. "DARPA Urban Challenge Technical Paper." 13 April, 2007.

[9] Virginia Tech College of Agriculture & Life Sciences. *Kentland Farm Operation.* November, 2001. Available Online: http://www.vaes.vt.edu/

[10] Kornhauser et al. "DARPA Urban Challenge Princeton University Technical Paper." June, 2007. Available Online: http://pave.princeton.edu/

[11] Jones, Troy B. "Testing answers from Team MIT" Email to Thomas Alberi, January 31, 2008.

[12] Montemerlo, Mike. "Re: Questions about your UC testing experience" Email to Thomas Alberi, March 14, 2008.

[13] M. N. Clark. "Re: Thesis questions"' Email to Thomas Alberi, March 13, 2008.

[14] C. Basarke, C. Berger, and B. Rumpe. ”Software & Systems Engineering Process and Tools for the Development of Autonomous Driving Intelligence.” *Journal of Aerospace Computing, Information, and Communication* Vol. 4 (2007): 1158 - 1174.

[15] McBride, James. ”Re: Help for a fellow UC ceontestant’s thesis work” Email to Thomas Alberi, March 17, 2008.

[16] junior-mission2-vtnearmiss.avi. Stanford Racing, 2007.

[17] Ruel Faruque. ”A JAUS Toolkit for LabVIEW, and a Series of Implementation Case Studies with Recommendations to the SAE AS-4 Standards Committee,” Master’s Thesis, Virginia Tech, 2006.

[18] Reinholtz et al. ”DARPA Urban Challenge Final Report.” 30 December, 2007.

Appendix A

Acronyms

CRC Corporate Research Center

DARPA Defence Advanced Research Projects Agency

E-stop Emergency stop

GPS Global Positioning System

IED Improvised Explosive Device

INS Inertial Navigation System

JAUS Joint Architecture for Unmanned Systems

MDF Mission Definition File

RNDF Route Network Definition File

TROCS Tartan Racing Operator Control Station

VNC Virtual Network Computing

Appendix B

Testing Questionnaire

The following is a copy of the testing questionnaire sent to other Urban Challenge teams to collect information. Also included is a list of the teams who responded.

B.1 Testing Questionnaire

1. Describe your primary testing facility. What services and technologies were available (internet, air conditioning, seating and desk space, etc)?

2. Describe your process for developing tests for your vehicle? Were these tests thought out thoroughly before hand or were they thrown together at the last minute?

3. Was a simulation program used in your testing? Was it made by your team/school? How important was simulation testing for your team? Could everything be tested in simulation? If not, what are the most necessary steps that need to be taken in order to be able to do full software testing in simulation? Do you see simulation testing being more dominant than real life testing in the future when it comes to these types of projects?

4. Did you have software in place that monitored the health/status of your various software components? Did you have a system for data logging and reporting software bugs?

5. What safety measures and procedures were used during testing?

6. Do you have any more comments on your testing strategy that would be helpful?

B.2 Responding Teams

Team CarOLO

Intelligent Vehicle Systems

Team MIT

Stanford Racing

Tartan Racing