

Team Gray's 2007 Urban Challenge Vehicle

Paul G. Trepagnier, Jorge E. Nagel, Powell M. Kinney, Matthew T. Dooner,
Sergey V. Drakunov, Aaron S. Lee, Michael T. Dewenter, Matt Hardey, Eric V. Gray
GrayMatter, Inc., Metairie, LA, 70001,
Email: teamgray@graymatterinc.com

DARPA Urban Challenge
Technical Paper – Team Gray

Submission Date: June 1, 2007

Executive Summary

It is the opinion of Team Gray that the Urban Challenge poses primarily a software engineering problem, since the individual hardware components needed for the Urban Challenge already exist in the commercial sector. These hardware components need to be reliably integrated with custom software logic to achieve the goals of the Urban Challenge.

Team Gray has addressed many of these concerns in the design of its Autonomous Vehicle System (AVS). The GrayMatter AVS™ is a commercially available autonomous vehicle platform designed as a generic framework for a variety of autonomous vehicle applications. The AVS provides significant improvements in the user interface, system reliability, and performance, and it is a substantially more compact and energy efficient evolution of the autonomous vehicle system employed in the team's 2005 Urban Challenge vehicle.

Through the development of its AVS system and its integration of advanced sensor technology, Team Gray has created an autonomous vehicle that it strongly believes can successfully complete the 2007 DARPA Urban Challenge.

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

1. Introduction

Team Gray was created to compete in the 2005 DARPA Grand Challenge. After the success of KAT-5 in the Grand Challenge, a new company, GrayMatter, Inc., was formed to commercialize and advance the technologies developed by Team Gray. GrayMatter, Inc. employees now comprise the majority of Team Gray, and Team Gray now relies on the stability and reliability of GrayMatter's AVS™ to provide the hardware and software platform for its Urban Challenge vehicle. Team Gray provides the medium for GrayMatter to develop and showcase the advances outlined in this paper.

Besides retaining all of the core members from Team Gray's original 2005 Grand Challenge team, GrayMatter, Inc. has added several more engineers and software developers to greatly broaden the scope of the team's capabilities. The company's additions have focused on bringing together members who are the best in their respective fields and whose skill sets create new possibilities and directions for the team to pursue.

2. Overview

The 2007 DARPA Urban Challenge poses an extremely daunting problem to be solved: autonomous navigation in a dynamic urban environment. The Urban Challenge retains all of the difficulties and technical challenges of the previous Grand Challenge, of which only four vehicles successfully completed, and adds many new difficult problems for autonomous vehicles to solve.

The most significant problem facing autonomous vehicles in the Urban Challenge is no different than in the previous two Grand Challenges. Each autonomous vehicle has exactly one opportunity to fully complete the mission; so a failure in any of the vehicle's critical systems will result in that vehicle losing the race. Since there is no opportunity for a second chance, all systems must be as reliable as possible. Analysis of the results from the 2005 DARPA Grand Challenge shows that either simple hardware or software failures ended the race for many of the teams that did not finish.

Besides the reliability problem, the Urban Challenge poses several significant technical challenges that must be solved through innovative hardware and software design:

- GPS data will be frequently unavailable due to the buildings and other obstructions that exist in an urban environment. Since all required elements of the autonomous vehicle's mission are specified via GPS coordinates, additional localization information will be needed to successfully complete an Urban Challenge mission.
- Along with static obstacles, many moving vehicles will be present in the urban environment. This requires the vehicle's software to track, interact with, and at times predict the movements of other vehicles.
- The autonomous vehicle must obey all applicable traffic laws at all times. This requires a complex software system to track the state of both the autonomous vehicle and the environment, particularly in intersections.
- The autonomous vehicle will be required to perform many advanced maneuvers, such as passing other vehicles, parking, performing a U-turn, performing a left turn through a

lane of oncoming traffic, and navigating through heavy traffic. This will require intelligent navigation software and precise driving capabilities.

- In certain areas of the course, the road will be specified with only a sparse set of waypoints, and the autonomous vehicle must use its sensors to detect the appropriate path to follow.

This list does not comprise all of the design problems faced by autonomous vehicles competing in the Urban Challenge, but it illustrates the most significant. The problems posed by the Urban Challenge can further be classified into three primary requirements: hardware and software reliability, accurate sensing capabilities, and an extremely complex software system.

Team Gray firmly believes that hardware and software reliability is the most important design problem that must be solved, so the team's approach to the Urban Challenge is centered on reliability. In addition, the team believes that the Urban Challenge primarily poses a software engineering problem, since the individual hardware components needed for the Urban Challenge exist in the commercial sector. These hardware components need to be reliably integrated with custom software logic to achieve the goals of the Urban Challenge.

The next sections will describe how Team Gray will approach the design requirements of hardware and software reliability, accurate sensing capabilities, and the development and ongoing management of an extremely complex software system.

3. System Design

3.1 Vehicle

Team Gray is using a 2007 Ford Escape Hybrid, which contains both a typical four-cylinder engine and a 330-volt electric motor/generator, as its vehicle platform for the Urban Challenge. This 330-volt electrical system provides a clean and sufficient source of power for the additional equipment installed in the vehicle. In addition, the vehicle's electrical system provides enough power for all of the additional equipment installed on the vehicle for the purposes of completing the Urban Challenge. The short wheelbase, narrow width, and Team Gray's familiarity with this platform made the Escape Hybrid the ideal solution to the Urban Challenge.

Team Gray worked with Electronic Mobility Controls (EMC) to interface with the primary vehicle systems (steering wheel, gear shift, accelerator, and brake) and secondary vehicle systems (turn signals, ignition, and emergency brake) via the AEVIT drive-by-wire system. This is a commercially available solution designed to outfit vehicles for handicapped drivers that has been specifically modified for use in autonomous vehicles. It consists of actuators and servos mounted on the steering column, brake pedal, throttle wire, emergency brake, and automatic transmission. It is also able to control the vehicle's turn signals and ignition. By using the AEVIT system, Team Gray is able to control all aspects of the vehicle via one fully integrated system, reducing overall complexity and eliminating points of failure. In addition, the AEVIT system is approved by the Department of Transportation and contains several layers of redundancy in each of the actuators and power systems. Team Gray has been using the AEVIT system for over two years, and it has proven to be an extremely reliable component of the team's autonomous vehicles.

The AEVIT system also provides the E-Stop mechanism for the autonomous vehicle. Team Gray worked with EMC to design a custom E-Stop implementation that satisfies the rules of the Urban Challenge. When an E-Stop is triggered, the vehicle's primary braking system is applied, and then the vehicle's ignition is turned off. Finally, after a slight delay, the vehicle's emergency brake is applied and held. This ensures that the vehicle will stop effectively when an E-Stop command is received and be able to remain stopped even if the vehicle is on an incline.

3.2 Hardware Platform

The hardware platform for Team Gray's autonomous vehicle is based on GrayMatter, Inc.'s Autonomous Vehicle System (AVS). The AVS is a commercially available autonomous vehicle platform that has been designed for a variety of autonomous driving applications. In addition to its use in the Urban Challenge, it is currently being marketed for autonomous testing of vehicles and tires. The AVS was designed initially as a more compact version of the systems in Team Gray's KAT-5, but it has been improved significantly over the past 18 months.

The AVS is comprised of a hardware layer and a software layer. The hardware layer consists of several custom-printed circuit boards that contain all of the wiring necessary to both provide power to and communicate with external sensors such as GPS receivers or obstacle sensors. GrayMatter's AVS provides all of the hardware required to integrate a wide variety of sensors. By implementing the majority of the physical wiring on printed circuit boards rather than wiring it by hand, the hardware layer in the AVS has become much more reliable than other approaches. These circuit boards also interface with E-Stop radios and the AEVIT drive-by-wire system. The hardware layer contains a programmable logic device that monitors the operation of the hardware and is capable of power-cycling failed components or even stopping the vehicle should a fatal error be detected.

3.3 Intra-system Communications

Communication within the individual components of the AVS system is segmented based upon the criticality and punctuality of the contained message. Vehicle control messages between the planning software and the AVS hardware are transmitted over an independent Controller Area Network (CAN) [1,2]. CAN has an integrated priority system, provides predictable real-time communications, and provides significant robustness against electro-magnetic interference. Therefore, it is an ideal choice for the most critical communications in the vehicle. Emergency control and, if necessary, stopping of the vehicle take the highest priority and will supersede any other communication on the CAN bus. Barring the infrequent presence of emergency messages, the control communications between the planning software and the AVS hardware are able to occur unhindered within a predetermined amount of time. As detailed in [3], CAN is well suited to complex real-time applications, particularly in the automotive fields.

A separate CAN bus is used for communication with sensors designed for automotive use, which may not be capable of other forms of communication. By separating the control network from the sensor network, control packets are prevented from preempting the sensor packets. This separation also prevents a malfunctioning device on the sensor network from disrupting the control CAN bus, as such a disruption could compromise the safe operation of the vehicle.

Higher bandwidth communication between the sensors and the planning computers occurs over Fast Ethernet. The high precision sensors incorporated into the AVS for this application generate large amounts of data that are well suited to the high bandwidth, low latency, and fault tolerance offered by Fast Ethernet. Both the position data from the localization sensor and object data from the obstacles scanners contain timestamps that negate the need for deterministic transmission of their data. The position and obstacle data can be reconstructed and reordered within the planning computers to rebuild the sensors' view of the world.

3.4 Sensors

An autonomous vehicle needs an accurate picture of the surrounding environment and its own global position to navigate safely in any environment. The added challenges of operating in an urban environment make the autonomous vehicle's sensors extremely important aspects of its successful operation. The following subsections describe the different types of sensors installed on the autonomous vehicle, and how they help solve the challenges that were identified in Section 2.

3.4.1 Localization Sensors

One of the challenges for a robot entering the Urban Challenge lies in building a map of the world around the robot and locating itself within that map. All of the data collected from the obstacle and lane detection sensors must either be referenced to some absolute location in the world or some location relative to the robot. Without accurate information about the location, heading, and velocity of the robot, other data can become useless. Planning a route within the world and in conjunction with traffic is simplified by translating all information gathered about the world around the robot to a set of global coordinates. Doing this translation requires exact knowledge of the location of the vehicle when the data were collected. From this information, a map of the area surrounding the robot can be created, and from this map the path of the robot can be planned.

Fundamentally, planning the path of the robot and synthesizing the data collected from the sensors requires very precise localization information. To obtain sufficiently accurate localization information, Team Gray utilizes the high precision RT3000 from Oxford Technical Solutions. The RT3000 incorporates two L1/L2 GPS receivers with Omnistar HP corrections to provide position measurements accurate to within 0.1 meters and heading measurements accurate to within 0.1 degrees. The RT3000 includes an inertial system to provide acceleration and roll information. In addition to the accelerometers and gyroscopes within the inertial system, Team Gray is providing wheel speed input from one of the Ford Escape Hybrid's rear ABS wheel speed sensors via GrayMatter's ABS Interface board. The RT3000 internally integrates the data from each source using a combination of Kalman filtering [4,5] and other proprietary algorithms.

Within the RT3000, a Pentium computer, running the QNX hard real-time operating system, monitors the measurements from the GPS and inertial systems. As this system collects the position and movement information, it transparently compensates for drift in the inertial system and outages in the GPS system before the data is transmitted to the AVS. The corrected data is then sent via Ethernet and CAN to the AVS, and it is transmitted via a dedicated CAN bus to the obstacle scanners. Both systems are then able to correlate obstacles, the robot's location, and mission waypoints to the same coordinate set.

The RT3000 proved its reliability throughout the 2005 Grand Challenge, and it continues to be a very capable localization sensor. Reviewing the data gathered by KAT-5 during the Grand Challenge Event, the team verified that the RT3000 had operated free of major error during the entire race. Even in instances when the GPS signal was partially or completely lost, the sensor compensated for the lost signal properly. Figure 1 shows data gathered from the steering controller during KAT-5's travel through the tunnel during the National Qualifying Event, in which GPS signal was completely lost for between 10 and 15 seconds. The fact that the vehicle was only 50 centimeters off of the desired path upon reacquisition of GPS signal speaks both to the reliability of the RT3000 sensor and to the ability of the control systems used by Team Gray to work well in conjunction with data emanating from the RT300 unit.

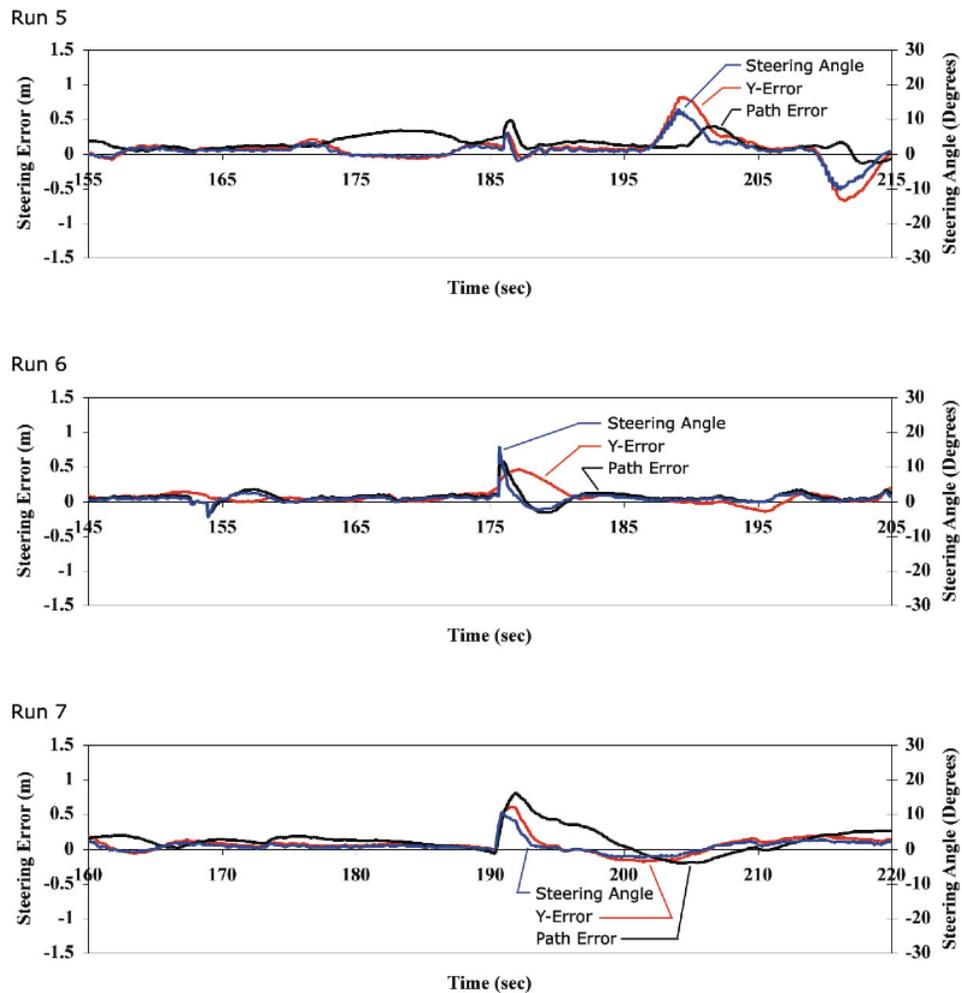


Figure 1: Upon exiting a 100-foot tunnel during the 2005 NQE, in which all GPS signal was lost, the RT3000 experienced a consistent jump of just over 0.5 meters. Even with this sudden change in the external state, the steering controller maintained control and was able to continue with its normal stable operation within seconds.

3.4.2 Obstacle Sensors

Team Gray has chosen to use two Ibeo ALASCA XT fusion systems for its primary obstacle avoidance sensors. Each ALASCA XT fusion system is comprised of two Ibeo ALASCA XT laser scanners and one Ibeo Electronic Control Unit (ECU). Each ALASCA XT laser scanner uses four eye-safe lasers in conjunction with a rotating mirror to sweep a 270° arc in front of the sensor. All four beams emanate from the unit on 4 different scanning planes (described in [6]), which are offset by 0.8° when the mirror is pointed directly ahead and 0° directly to the sides. By using multiple lasers in this manner, the ALASCA XT laser scanner is capable of maintaining a sufficient field of view even as the vehicle pitches and rolls during maneuvers.

Because of the flexibility of its field of view, the ALASCA XT laser scanners are rigidly mounted to the vehicle at a height of approximately 0.5 meters from the ground. By mounting the sensor at this relatively low height, the sensor can detect smaller obstacles more effectively than if it were mounted higher up on the vehicle. Most other horizontally mounted sensors are not as effective when mounted this low because their scanning plane is frequently obstructed by the ground when the vehicle is pitching.

The Ibeo ALASCA XT laser scanners can also operate in a wide range of weather conditions due to its ability to detect multiple echoes from a single laser beam. If the beam reaches a transparent object, such as a pane of glass or a raindrop, it will create a partial echo that is recognized and qualified as such by the laser scanner. This *Multi-Target Capability*, as termed by Ibeo, allows the ALASCA XT laser scanners to operate in many different types of inclement weather, including rainstorms.

Another advantage of Ibeo's advanced sensor technology is the ECU's ability to incorporate the laser angle and ranging information from two ALASCA XT sensors to create a map of the objects surrounding the vehicle. After filtering to remove the uninteresting laser echoes, such as raindrops and the ground, the Fusion system, as it is called by Ibeo, combines the data from both laser scanners and then fits a polygon around groups of echoes. Next, the software algorithms in the ECU calculate each obstacle's velocity vector and identify each obstacle with its own unique identification number. To reduce communications overhead, the ECU only transmits obstacles that satisfy a specific priority classification algorithm. Team Gray has chosen an algorithm based upon both object velocity and distance from the vehicle as the primary criteria for this classification. The resulting polygons are transmitted via CAN to the AVS. Since all of this processing is done locally on the ECU, the CPU cluster in the AVS is spared this additional processing overhead.

Most vehicles that need full 360° scanner coverage use one ALASCA XT Fusion system in the front of the vehicle and one standalone ALASCA XT laser scanner in the rear of the vehicle. This approach uses two Electronic Control Units (one for the fusion system and one for the rear sensor), but leaves the system vulnerable to single points of failure. As in the previous Grand Challenge, one of Team Gray's primary design philosophies is to remove single points of failure whenever possible. By using two ALASCA XT Fusion systems, each of which has a complete 360° field of view, Team Gray has two completely redundant views of the surrounding environment. Each fusion system has a sensor on one of the front corners of the vehicle and a sensor on the opposite rear corner of the vehicle, as illustrated in Figure 2, along with its own

ECU.

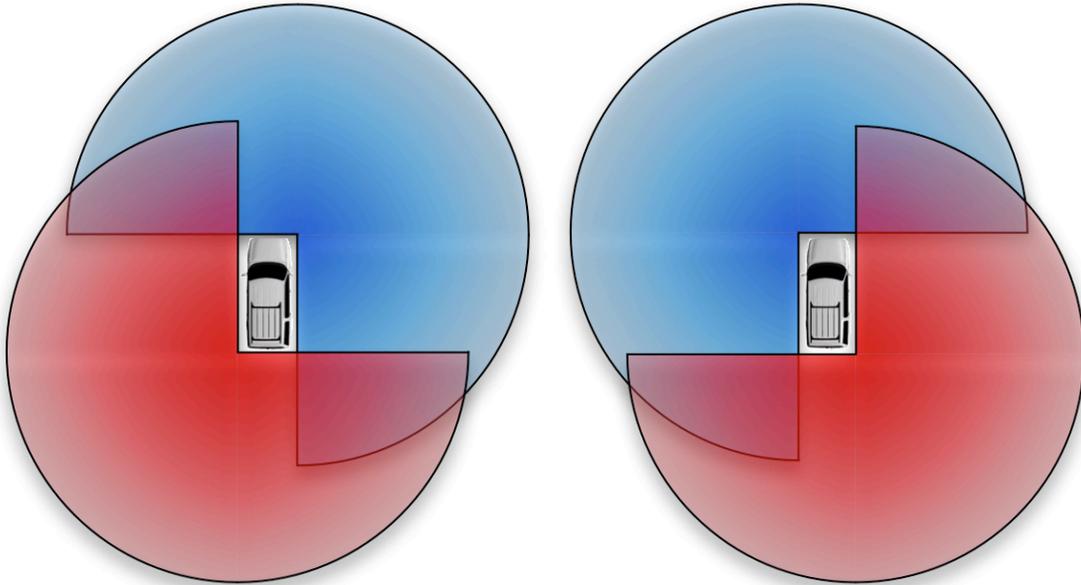


Figure 2: Team Gray’s sensor strategy incorporates two Ibeo ALASCA XT Fusion systems. Each system has a complete 360° view of the vehicle’s surroundings. With two such systems, not only is the view of an object more complete, but a failure in one of the Fusion systems will not affect the vehicle’s field of view.

The collection of obstacles returned from both fusion systems are incorporated into the vehicle’s obstacle repository. In the event that one fusion system fails to return a list of obstacles, the vehicle seamlessly continues operation with the other system without losing any of its field of view. When the failure of a fusion system is detected, the hardware layer of the AVS will attempt to reboot the faulty fusion system by powering down the sensor temporarily in hope that it will recover properly upon restarting.

3.4.3 Lane/Road Detection Sensors

Since the Urban Challenge rules specify that an autonomous vehicle must find and follow the proper lane/road in cases where only a sparse collection of waypoints identify the lane/road, an autonomous vehicle must be able to identify the proper lane or road in real-time. To solve this problem, Team Gray is using a video-based lane-detection system from Iteris, Inc. Iteris’ Lane Departure Warning (LDW) system is currently available in several consumer vehicles and as an installable option on many different models of heavy trucks. Iteris has supplied Team Gray with a modified version of their heavy truck LDW system for use in our autonomous vehicle.

The Iteris LDW system uses an optical sensor and image processing system to detect and track lane markings. A conventional imaging sensor in the LDW system creates a two dimensional digitized picture of the area ahead of the vehicle that the LDW system searches for lane markings. It is installed in the cabin of the vehicle, at the top of the glass windshield. The LDW system provides the autonomous vehicle with the location of the left and right lane markings, the type (solid, dashed, etc.) of the left and right lane markings, the angle of the vehicle within the lane, and the curvature of the lane. This information is provided to the AVS software at a rate of

25 times per second via CAN. The information from the Iteris LDW system is used to build a model of the current lane that can be used by the vehicle's software systems to adjust the vehicle's planned path to better adhere to the lane model.

Since it is primarily used in production vehicles sold in the United States, the Iteris LDW system reliably identifies all types of lane markings that are used on American roads. In addition, testing has shown that the Iteris LDW sensor can, in the absence of lane markings, also detect the edges of the road in most scenarios.

3.5 Software Platform

The Urban Challenge poses many difficult technical challenges, some of which require innovative, large-scale software solutions. The volume and complexity of the software needed for an autonomous vehicle to operate successfully in an urban environment can easily overwhelm software architectures that are not designed to handle it. Fortunately, the software architecture of the GrayMatter AVS helps reduce this development burden.

GrayMatter's AVS was designed as a generic autonomous application framework [7, 8] that can be used for many different types of autonomous vehicle applications. A framework is a software reuse technique where there is a skeleton of an application that can be customized by the application developer [9,10]. The first application developed on the AVS platform was an autonomous system for testing consumer vehicles and vehicle tires. This proved the reliability and flexibility of the software in the AVS as automotive testing facilities demand safe operation of autonomous vehicles even in the event of hardware or software failures. Significant emphasis will be placed on the AVS software architecture in the following sections since it enables Team Gray to develop higher quality and more complex software at an unprecedented pace. A detailed discussion of the AVS software framework is available in [11].

As with KAT-5, Team Gray has again decided to program a majority of the software using the Java Programming Language. While some studies point to Java's poor performance when compared to other lower-level languages for scientific computing (see analysis in [12]), Java quickly makes up for its deficiencies during development, where its strong type checking and built-in multithreading support save the programmer a great amount of effort. Java also lends itself to a modular and independent design scheme. Java's simple language semantics, platform independence, strong type checking, and support for concurrency make it a solid choice for high integrity systems such as autonomous vehicles [13]. Because of the platform-independence of Java, the same code base can be run on a varied number of hardware systems with reliable and repeatable results.

The AVS software framework uses several different software design patterns [14] or design principles to reduce the complexity of designing autonomous vehicle applications. Each of these design patterns has been proven successful at reducing complexity and improving the reliability of software development in enterprise application development, but they have yet to be applied to the development of autonomous vehicle applications.

One of the primary software design principles used by GrayMatter in the AVS software framework is the Separation of Concerns [15] paradigm, which reduces complexity in development by breaking a large problem into a set of loosely coupled sub-problems that are

designed to be easier to solve. The goal of Team Gray during the development process has been to separate the software system into as many distinct components as possible with minimal amounts of overlap. By separating the software into functionally separate components, a minor failure in one component should not adversely affect other components.

Because the software has been broken into separate modular components, the reliability of the software can be further improved by the use of Test Driven Development (TDD) [16], a software development practice in which unit test cases are incrementally written prior to actual code implementation. These unit tests are responsible for determining if the logic of a particular component is correct, and are run automatically by a server process each time the code base is changed.

These automatic unit tests serve two purposes. First, they help verify that a given software component is actually logically correct. For example, the obstacle repository component in the autonomous software platform is responsible for detecting intersections between a given polygon and the set of obstacles that it contains. To verify that this logic actually performs as expected, the unit test sets up several scenarios each with a given set of obstacles and then proceeds to verify that intersection tests with a set of test polygons perform as expected. If the unit test is sufficiently thorough and a given component passes its respective tests, developers can be fairly certain that the logic of the software component is correct.

Secondly, TDD ensures that changes made to the one part of code base do not affect other components that were previously working correctly. Such errors are referred to as regression bugs and can be very hard to track down and fix; however, since the entire code base is tested as a whole each time any change is made, such bugs should be revealed as soon as they are introduced.

To help ensure that the team's automated unit tests thoroughly test the Urban Challenge software, code coverage reports are automatically generated whenever the code base is changed. A code coverage report identifies each line of untested code in the Urban Challenge software and also provides statistics on the percentage of untested code. The developers examine these coverage reports frequently to determine what aspects of the code base need to be tested more extensively. This results in more thorough unit tests, which in turn help to ensure that the software performs as expected.

To make developing software for the AVS framework using these design methodologies more productive, the AVS software framework has been implemented using an architecture centered on an Inversion of Control (IoC) container [17]. Inversion of Control is a design pattern where the framework operates as a container that coordinates and controls execution of the individual application components. An IoC framework simplifies application design because the framework, rather than the application, links components together and is responsible for routing events to the proper components in the application [18]. In the AVS framework, the IoC container provides all of the services necessary for a proper real-time autonomous vehicle application, including thread scheduling, logging services, distribution of application assets across a computing cluster, fault tolerance, and network communications.

The thread scheduling capabilities of the AVS software framework significantly enhance the development of autonomous vehicle applications. For the Separation of Concerns paradigm to be most effective, components should be as isolated as possible. Ideally, the components should be executing in parallel, rather than sequentially, so that a failure in one component does not cause the execution of subsequent components to be aborted. The AVS software framework automatically executes each component as its own thread of execution, even across multiple computers, and transparently coordinates the sharing of data between the separate components. The AVS software can also execute these components at set frequencies under many different levels of processor load, which is beneficial for the many control systems needed for autonomous vehicle operations, as each of these control systems requires precise timing for accurate vehicle control.

For the Urban Challenge, the AVS software will run on three dual-core embedded computers operating as a distributed cluster. Each computer in the cluster runs a hard real-time operating system coupled with the real-time capabilities of the AVS software framework to support deterministic execution of autonomous applications within stringent time constraints. Figure 3 illustrates the improvements in time determinism shown by executing an autonomous vehicle application on the real-time AVS platform. The real-time capabilities of the AVS platform allow autonomous applications to behave more consistently and ensure that even in the case of software problems, higher priority components such as safety monitors and the low-level driving algorithms are allowed to properly execute.

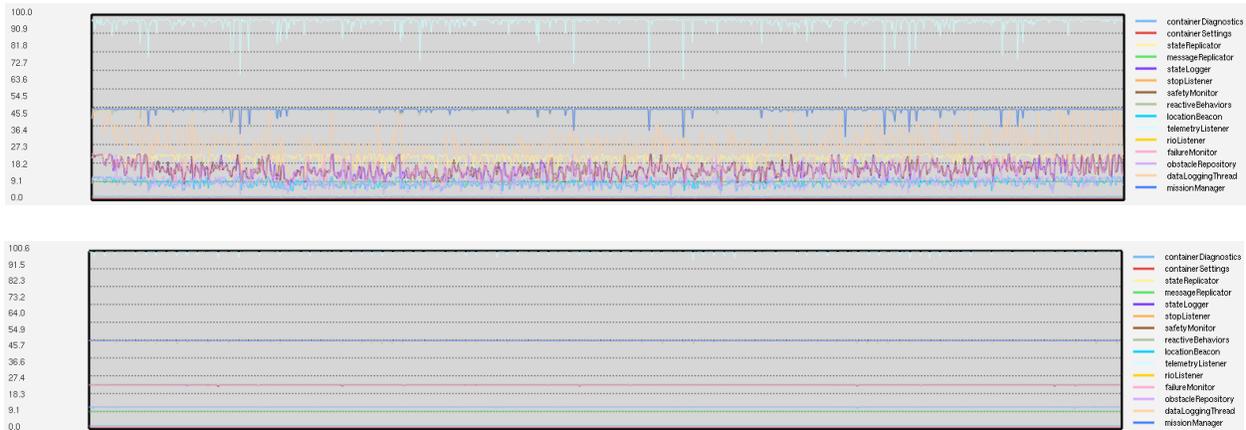


Figure 3: Using GrayMatter’s AVS software architecture significantly improves the stability of running processes. The top chart shows the operating frequency of several critical processes with real-time support disabled. Once real-time support is enabled, the operating frequencies of the processes stabilize drastically as can be seen in the bottom chart.

3.5.1 Urban Challenge Software Implementation

The software logic required to successfully complete the Urban Challenge has been implemented as modules for the AVS software framework. The AVS software framework executes each of these components with its own independent thread of execution, and it automatically manages dependencies between multiple components. The following subsections describe the most significant software logic components that Team Gray has created to solve the problems posed by the Urban Challenge.

3.5.2 Route Planning

Mapping of the environment and long distance route planning are important concerns in the design of a successful autonomous vehicle for the urban landscape. Team Gray's design model separates visual mapping from logical mapping and route planning. Human operators perform the visual mapping task of analyzing graphical maps, aerial photography, and landmark data with geospatial information system (GIS) tools to build logical maps that the robot's computer systems can process. This visual mapping process is used both to build RNDF maps for testing and to analyze RNDF maps provided by DARPA.

The logical mapping functions performed by the onboard computers include identification of intersection components, mapping of sensor-visible landmarks, and correction of under-defined map areas. Under-defined map areas consist of regions where the map provided to the robot is insufficiently correlated to the real-world environment. In this case the robot must explore and identify the area during its mission. Long distance route planning takes into account the constraints of the mission and the robot's constantly changing environment to find an optimal route that completes its mission. Team Gray's Urban Challenge solution seamlessly combines the provided mapping data with runtime logical processing to navigate a diverse range of urban environments.

For its own internal testing, Team Gray wrote an application to quickly create logical maps on top of publicly available aerial photography. This application allows Team Gray to test their vehicle with maps similar to those that would be constructed by an operator in an actual deployed environment. The team has found that vitally important to the success of an autonomous vehicle in the real world is the human operator's ability to create maps and missions for the vehicle to execute. In critical or large-scale installments, commercial GIS solutions and image processing can make the operator's task of defining missions much easier.

The logical map is provided to the onboard computers in DARPA's Route Network Definition File (RNDF) format. A two-pass parser identifies all of the waypoints and perimeter points before verifying that all of the `waypoint_id` and `perimeterpoint_id` references are valid. The map is stored in an object-oriented adaptation of the RNDF format and includes extensions for map features derived from the RNDF file.

The first derived feature gleaned from the data in the RNDF is the grouping of stop and exit/entry waypoints into intersections. An algorithm first picks any stop waypoint and then finds all of the exits leaving that point and entries leading to it. Next, for each exit in the intersection, if the waypoint following the exit is an entry, the entry/exit pair is added to the intersection. Likewise, for each entry in the intersection, if the waypoint preceding the entry is an exit waypoint, the exit/entry pair is added to the intersection. Finally, if any stops or exits are within a defined distance from the boundary of the intersection they are also added to the intersection. Provisions are made to ensure that each stop or exit only belongs to one intersection.

The second derived feature is the storage of the cost associated with traveling between waypoints. Since the Urban Challenge event will be scored based on the time taken to complete a mission, the time taken to drive from one waypoint to the next is a prime candidate for the metric

used to pick an optimal route. Time metrics are stored in waypoint, exit and zone objects. The initial cost for each waypoint is calculated optimistically by dividing the segment maximum speed limit by the distance between the waypoint and its previous waypoint. If the waypoint is at the beginning of a lane it has zero cost. The cost of an exit is calculated based on the speed of the entry's segment plus a fixed penalty.

Team Gray's route finding algorithm includes a learning component that enables the robot to become more efficient in its planning as it explores more of its environment. By recording the time it takes to travel between waypoints, through intersections, and across zones a route can be calculated that optimizes for travel time. A record of travel times is maintained for a given RNDF that is used across multiple missions. The traffic patterns should remain similar across missions, but to compensate for new areas of congestions, old observations are discredited as new ones are made. A weighted averages formula, as shown in Equation 1, with geometrically decreasing weights is used to calculate the cost of a specific travel unit. The most recent observation has a weight of 0.5 and the weight of each previous observation decreases by one-half.

$$\begin{aligned}
 S_n &= \text{samples} \\
 N &= \text{num samples} \\
 N = 1 &: \text{sum } S_0 \\
 N > 1 &: \text{sum} = S_0 * 1/(2^1) + S_1 * 1/(2^2) + \dots + S_{N-2} * 1/(2^{N-1}) + S_{N-1} * 1/(2^N)
 \end{aligned} \tag{1}$$

The optimal route between checkpoints is determined by an A* heuristic-guided search [19]. The A* search algorithm maintains a priority queue of explored paths. The priorities are determined by the current cost of the path ($g(x)$) and the estimated cost to the goal ($h(x)$). In Team Gray's implementation of A* for route planning, $g(x)$ is the sum of the observed average travel time for travel units already explored. The heuristic $h(x)$ is the straight-line distance to the goal checkpoint divided by the maximum speed limit for the course. This heuristic affects a behavior in which the most direct routes are explored first. The A* algorithm has proven to create optimal routes quickly and accurately in both simulation and actual testing by Team Gray during the development process.

3.5.3 Variable Structure Observer (VSO)

The main functions of the VSO are to provide information fusion and prediction of coordinates and trajectories for all of the stationary and moving obstacles in the autonomous vehicle's nearby environment (approximately 150 meters). The presence of the observer improves the situational awareness of the vehicle and provides the capability to maintain intelligent operation and navigation of the vehicle, even if sensor data is either temporarily lost or becomes temporarily unreliable. This is extremely useful in situations where one obstacle is temporarily hidden by another obstacle, such as when another vehicle is driving through an intersection in front of the autonomous vehicle.

The VSO principle is based on the idea that once the obstacle is detected by a sensor system that performs preliminary data processing to identify the obstacle's position, geometry, and velocity vector, the VSO will automatically create an identifier for the obstacle and its mathematical model of motion. The state vector of this model and its parameters will be constantly updated based on the incoming stream of sensor data, but if the sensor data is temporarily lost, the model

will continue to be simulated, thus, providing prediction of the obstacle's position and velocity to enable the temporarily blinded vehicle to safely stop until the sensor data is reacquired.

By running the obstacle model forwards into future time, the VSO is able to predict not only the current position, but also the future positions of this obstacle for the purpose of path-planning and speed-planning. The VSO combines the models of all obstacles in the nearby environment into one variable structure model of the vehicle's operating environment, which changes dynamically with the environment. The dimension of the state vector of the observer constantly changes since the VSO will add new models for obstacles that are entering this area and remove obstacles when they have left the area. The Variable Structure Observer is based on the theory of systems with sliding modes (see [20-28]). Such observers allow reliable reconstruction of the state vector from observation data for strongly nonlinear systems in the presence of uncertainties.

In actual practice, the VSO provides another significant benefit. By including a mathematical model of vehicle motion in its calculations, the VSO automatically filters out fluctuations that can occur in sensor data. This is particularly important with the Ibeo laser scanners, since the velocity vectors that they calculate can contain significant jitter. The VSO creates a much more accurate velocity vector that can be better used by the other modules in the system.

3.5.4 Path-Planning

The route-planning module (see Section 3.5.2) is used to create a global route that the autonomous vehicle should generally follow, but a local path planning module, named the Velocity and Path Planner (VPP), is used to translate from the global route to the current local path. The local path contains both the planned positions of the vehicle and the planned target velocities of the vehicle. The local path can be regenerated multiple times per second as both the vehicle's state and the surrounding environment changes.

The VPP uses the information from the Variable Structure Observer (VSO) to plan and update the time-space trajectory and velocity profile of the autonomous vehicle. The optimal path calculation is conducted in the extended domain, which includes time-space characteristics of the obstacles and their future positions. The trajectory calculation is done in three steps.

1. During the first step, the VPP calculates the (x,y)-space trajectory based on the provided GPS points from the global route. These points are then connected by a smooth curve, which is calculated using cubic or higher order spline interpolation. This is done using standard methods ([21,29]).
2. In the second step, the VPP calculates the time-optimal trajectory in the extended time-space domain $\{x^*(t), y^*(t), t^*(t)\}$, which satisfies the velocity constraints (maximum and minimum speed limits) and avoids any obstacles. The optimal and quasi-optimal trajectories with the aforementioned constraints are calculated using a combination of calculus of variations, Bellman dynamic programming, and Pontryagin's minimum principle. Pontryagin's minimum principle provides the necessary conditions for time-optimal control in the case of control and state variable constraints (see, [30-35]). The calculation is done using a novel sliding mode algorithm.

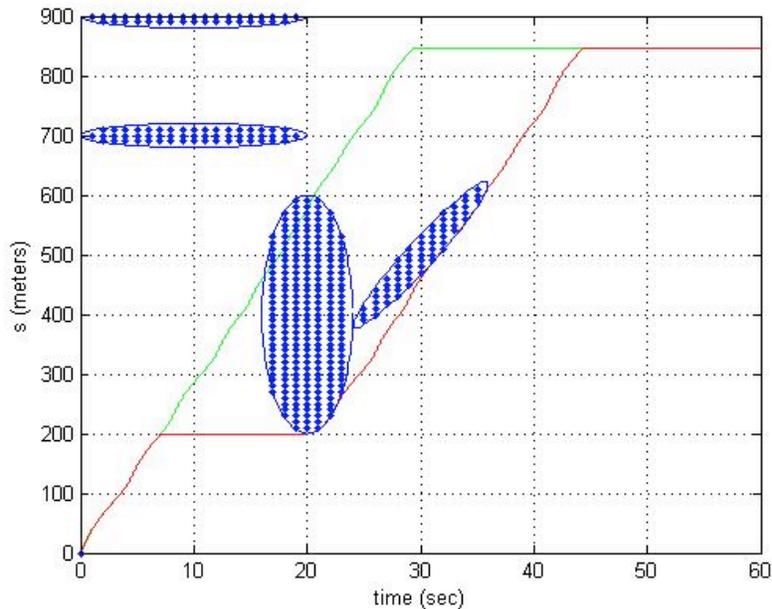


Figure 4: This example S-T diagram shows the original speed plan (green line) and the corrected path (red line) taking into account observed obstacles (blue shapes). In this chart, s represents distance along the path, and time is the expected time to be at that point along the path.

3. In the third step, the VPP uses an on-line quasi-optimal calculation of the trajectory that is closest to the trajectory calculated in Step 2 that satisfies acceleration/deceleration constraints while preserving velocity and vehicle safety. At this point, the ride comfort can be taken into consideration if it does not interfere with safe operation. This calculation is performed by the Velocity and Path Planner using novel on-line algorithms based on sliding mode of the higher order and its generalizations ([22,26,28]).

It was proven mathematically that the designed algorithm allows complete avoidance of all types of obstacles, in the space-time domain (S-T domain), by altering the vehicle's velocity. If there is no mathematical solution to this problem, then the vehicle cannot avoid the obstacle by slowing down, so a swerving maneuver will also be needed to avoid the obstacle, if the obstacle is capable of being avoided. This type of swerving maneuver will only be needed if another vehicle is behaving erratically.

The results of the VPP are constantly evaluated to determine if a lane-change or other path-altering maneuver should be attempted. The goal of the VPP is to achieve optimal velocity, so it will attempt to pass another vehicle if it is stopped for too long or even slowed down too much by another vehicle. The Mission Manager (see Section 3.5.6) is responsible for evaluating the results of the VPP and then determining if other path-altering actions are necessary.

3.5.5 Steering Control System

Team Gray spent considerable time during the development of KAT-5 developing accurate vehicle control algorithms. The team's strategy during both the 2005 Grand Challenge and the Urban Challenge is for the other software modules to be able to freely modify the desired path as

necessary to allow the vehicle to achieve its goal while avoiding any obstacles. By following this design strategy, the software planning modules can generate any paths that they desire, and as long as they satisfy certain constraints related to curvature and velocity, the vehicle will accurately follow them with an extremely high level of precision. Due to this high level of driving precision, the planning modules can generate paths that weave through tight fields of obstacles successfully.

The steering control system used for the Urban Challenge is a slightly improved version of the control system used by Team Gray in the 2005 DARPA Grand Challenge. The steering controller is a lead-lag controller based upon the on the classical single-track model or bicycle model developed by Riekert and Schunck [36]. In the lead-lag controller, the lead compensator increases the responsiveness of a system while the lag compensator reduces (but does not eliminate) the steady state error [37].

As Figure 5 illustrates, the vehicle's steering controller is capable of driving with extremely high accuracy, even in a rough environment like the 2005 DARPA Grand Challenge. The modifications made since then have significantly improved controller stability and controller accuracy at higher speeds, which will allow the vehicle to achieve higher stable speeds during a winding urban course.

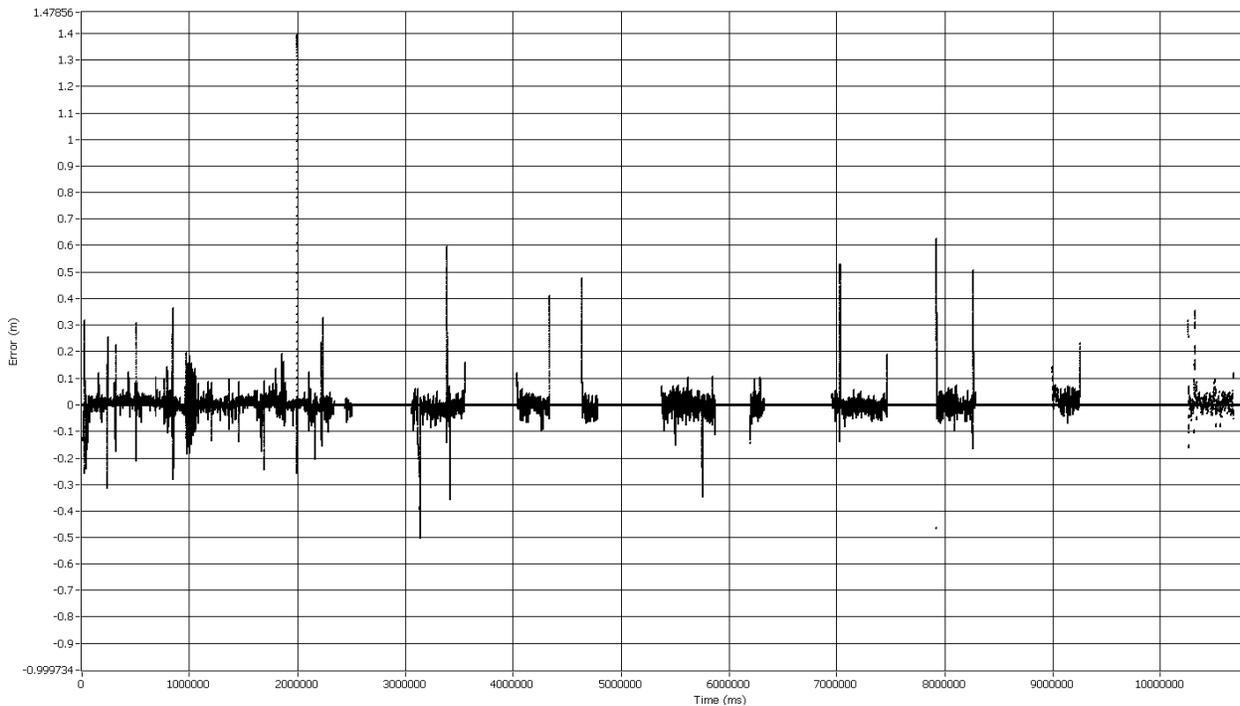


Figure 5. During the 2005 DARPA Grand Challenge, Team Gray's steering controller had a standard deviation of just 5 cm from the desired path. The gaps in the chart are from when the vehicle was paused by DARPA officials.

3.5.6 Mission Manager

Much of the higher-level processing and decision-making within GrayMatter's AVS is handled by the Mission Manager. The Mission Manager coordinates between all other components within the AVS architecture, in addition to monitoring each component for proper operation.

The Mission Manager itself is designed to operate independent of component implementation, so that replacing one type of sensor with one of a different design will not affect proper operation of the vehicle.

For the Urban Challenge, a specific implementation of the general purpose AVS Mission Manager was created to satisfy the unique requirements presented by the Urban Challenge. Additional capabilities were added, including the ability to handle large amounts of obstacle data, both moving and static, and the utilization of information from DARPA's E-Stop in compliance with the Urban Challenge Rules.

The primary element of the Urban Challenge Mission Manager is a Finite State Machine (FSM) [38] that is responsible for directing the vehicle through the sequence of events necessary for successful completion of the mission in accordance with the rules. An FSM is defined by a set of states that the vehicle can occupy and the transitions between states. These states include such events as driving, passing, waiting at an intersection, and so on. From each of these states, the developers define a set of "exits," which are transitions that the vehicle can perform to progress from one state to the other. Such an exit could occur when a vehicle blocks the desired path, which may cause the vehicle to change from a "driving" state to a "passing" state.

The architecture of an FSM lends itself well for adherence to traffic laws, as such rules generally contain very specific situations in which they are to be applied. The FSM also allows the developers to reconstruct whole chains of events quickly and easily during testing and after missions are completed. Since the actions of the vehicle can only be controlled by one state at a time, the FSM creates a chronological series of behaviors and reasons for initiating those behaviors that can later be analyzed for bugs and logical errors.

The actual implementation of the FSM has been implemented to allow for maximum flexibility during development. As opposed to implementations that require explicit coding of each state's transitions, which can make modification difficult (see [39]), the architecture of the FSM within AVS allows a set of loosely coupled states to be "stitched together" by the FSM at run-time. This means that adding new transitions or states to the FSM does not require the developer to explicitly link the exit point from one state to the entry point of another state. The FSM automatically handles these tasks by using the Java programming language's reflection and introspection capabilities [40]. (Reflection and introspection allow a piece of software's structure to be ascertained on-the-fly during runtime.) Due to this feature, the development time and complexity of development for the states within the FSM can be reduced substantially.

In addition to containing the FSM component, the Urban Challenge Mission Manager monitors the Mission Planner component, which performs high-level planning based on the provided Mission Data File (MDF) and Route Network Definition File (RNDF). The logic of the Mission Planner is described in Section 3.5.2. Once a global plan is created that navigates the vehicle from waypoint to waypoint along the MDF's prescribed checkpoints, modifications to this plan are tracked and verified by the Mission Planner. Finally, distribution of the most current plan to other components is the responsibility of the Mission Planner.

Yet another function of the Mission Manager is to ensure that requests from one component to another do not adversely affect the safe operation of the vehicle. For instance, steering commands that are sent from the steering control module are first verified as appropriate for the vehicle's situation (speed, roll, etc.) by the Mission Manager before being passed on to the vehicle's actuators. The Mission Manager also detects pause commands, and it coordinates the smooth stop of the vehicle.

4 System Testing

During the development process for the 2005 DARPA Grand Challenge, Team Gray developed an extensive set of system testing methodologies that helped it to succeed in the actual race. These testing methods were required to be as efficient as possible due to the team's relatively short development timeline of six months and the team's displacement due to Hurricane Katrina. These testing methods have been significantly improved during the team's development of its Urban Challenge vehicle.

A significant aspect of Team Gray's testing procedures is the use of the GrayMatter AVS Simulator. The GrayMatter AVS simulator is the successor to the simulation system used during the 2005 DARPA Grand Challenge and has been significantly improved with more accurate vehicle models and the support for many additional sensor types. The AVS Simulator operates by transparently replacing specific sensor components with corresponding simulated sensor components.

The simulated components are indistinguishable to the rest of the system from their real-world counterparts, so a majority of the software being tested is identical to the software that would be used during real-world autonomous operation. In fact, in the team's current Urban Challenge implementation, over 98% of the software system remains the same when running under simulation.

In most instances, the primary components that are simulated are the GPS sensor and the obstacle sensors. The simulated GPS sensor uses the output from the steering and speed control systems and a mathematical model of motion for the vehicle to calculate the vehicle's new position and orientation. Additional simulation options for the GPS sensor include the ability to add different types of noise or error into the simulated data or even random shifts in GPS output.

The simulated obstacle sensors generate simulated data from multiple sources. The simplest form of simulation replays sensor data captured during previous real-world testing. This type of simulation has limited usefulness, and so it is primarily used to verify that the obstacle sensor modules are operating properly. A more useful obstacle simulation allows the developer to graphically reposition obstacles in the environment in real-time or to specify a path or trajectory for the obstacle to follow. A mathematical model of motion is used to accurately move the simulated obstacles in the environment, based upon developer-specified parameters.

The most important type of obstacle simulation is based on the AVS software framework's support for the simultaneous operation and monitoring of multiple autonomous vehicles. In order to support running multiple autonomous vehicles safely at vehicle testing facilities, each AVS transmits its own location to any other AVS on the same network. This capability is used to simulate multiple autonomous vehicles in an urban environment by starting individual

simulations at different points on the route network. In addition, many different types of noise or error can be introduced into this simulated sensor and telemetry data to better test the software's ability to handle sensor interference and natural environmental noise.

The extensive simulation-based testing used during the development process is a direct result of Team Gray's experiences during the 2005 Grand Challenge. During that development cycle, the team realized that each field test became more expensive than the last in terms of time and resources and that a proper simulation performed at the team's offices would more often than not yield very similar results. Even with the significant emphasis the team places on simulated testing, it is accepted that actual field tests are irreplaceable and must occur frequently. In general, field tests are only performed after the desired test has already been proven successful in simulation.

The AVS Console was built to make the testing process (both actual field tests and simulations) as rapid and productive as possible. The Console is the primary means of communication with the autonomous vehicle and provides a usable interface for controlling and monitoring the behavior of the autonomous vehicle. Its graphical user interface is responsible for allowing users to launch autonomous missions, view the results of missions in real-time, and to replay previous missions. Figure 6 contains a screenshot of the AVS Console that was taken during an Urban Challenge run. In addition to the standard monitoring of vehicle telemetry, the Console give a user the capability to monitor state changes, detected obstacles, and the behavior of path- and speed-planning algorithms. Also, as demonstrated in Figure 3, the AVS Console displays real-time statistics on the average execution time and execution frequency of all of the threads in the system.

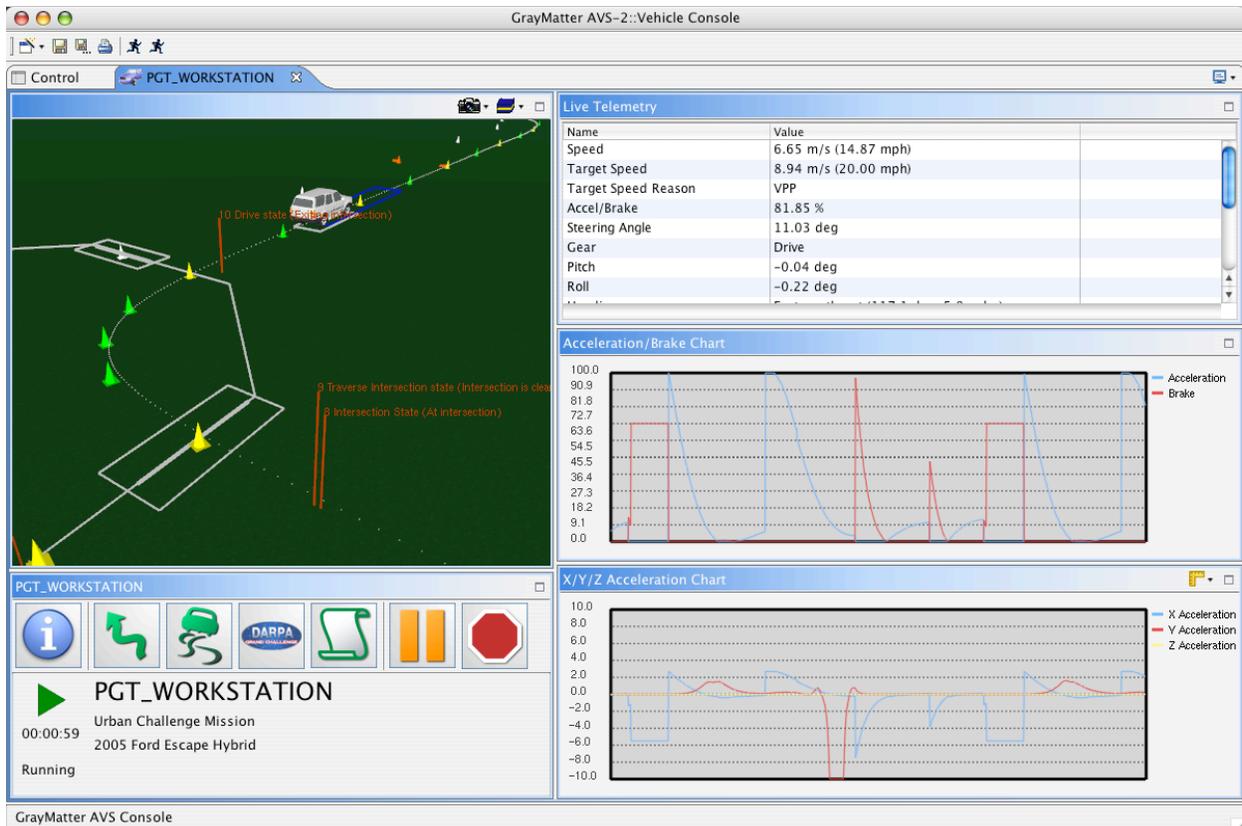


Figure 6. The AVS Console displays a real-time view of an autonomous mission, including a 3D environmental display and live telemetry data.

During extensive field-testing, Team Gray's autonomous vehicle system has proven itself capable of the precision driving needed for the Urban Challenge. Since the path planning components operate most effectively when the vehicle can drive the generated paths accurately, particular testing emphasis has been placed on the steering controller. Because the steering controller was already proven to drive with high accuracy at lower speeds during the 2005 DARPA Grand Challenge, testing has concentrated on stressing the steering controller with higher speed turns. The vehicle can now navigate turns successfully at much higher speeds due to the improvements in the speed controller, as shown in Figure 7.

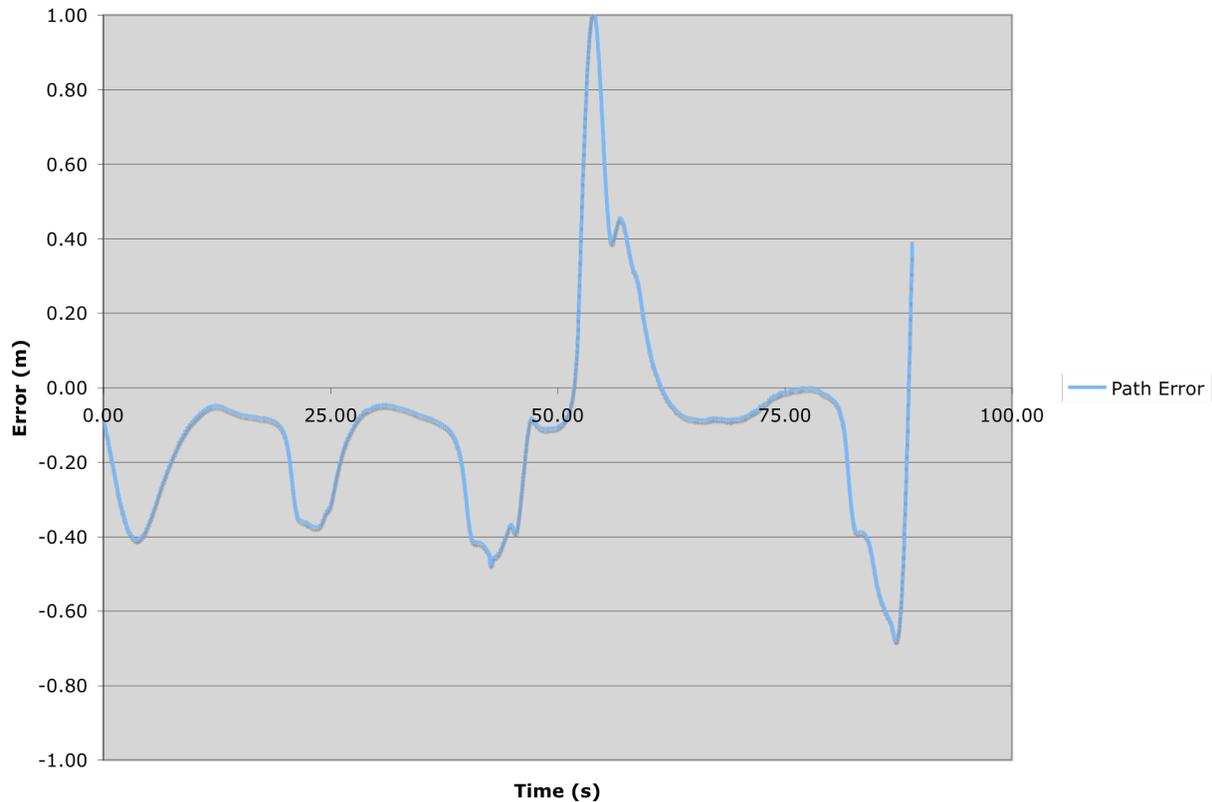


Figure 7: The steering controller can maintain a standard deviation from the path of under 25 cm even while driving a difficult slalom course containing hairpin turns at a constant speed of 30 km/hr.

In addition to the precision driving capabilities, testing has shown that the team’s data filtering algorithms are successful in improving the noise in its obstacle sensors. Figure 8 shows the results of filtering the velocity values from an Ibeo ALASCA XT Fusion system. The obstacle detection capabilities of these sensors have been extensively tested, which has shown that the vehicle can track moving obstacles reliably within an approximate 150-meter range. Smaller stationary obstacles such as utility poles can be tracked reliably within an approximate 75-meter range. These ranges will be more than adequate for avoiding obstacles during the Urban Challenge.

The autonomous vehicle software has been tested exhaustively in simulation to detect logic issues when interacting with other vehicles. These simulations have included large numbers of simulated vehicles interacting at intersections and sometimes behaving erratically. Numerous actual field tests have verified that the simulated performance is accurate, so the team is confident that interaction with other vehicles during the Urban Challenge will be handled properly.

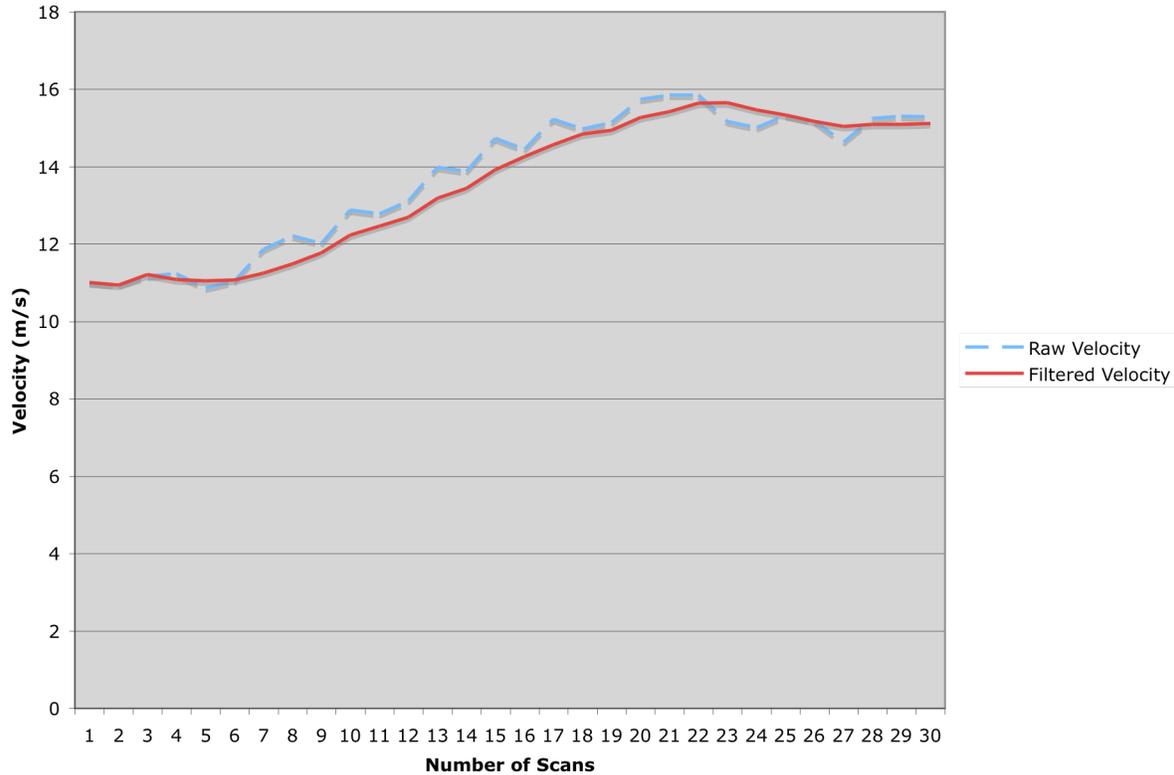


Figure 8. The velocity values provided by the Ibeo laser scanners contained significant noise that was successfully removed using custom filtering methods.

5. Conclusion

The Urban Challenge presents three primary challenges for autonomous vehicles: software and hardware reliability, accurate sensing systems, and the successful development of complex software systems. Team Gray has developed an Autonomous Vehicle System that is designed to fulfill these design challenges. The AVS hardware platform, which was proven successful during the 2005 Grand Challenge, has been improved significantly to increase its capabilities and reliability and to allow for simple integration of various commercial sensing systems into the autonomous vehicle. Team Gray has also tested each of the selected sensor systems extensively and has determined that they will be capable of fulfilling the sensing requirements of the Urban Challenge.

The AVS software framework was created specifically for the purpose of creating complex autonomous vehicle applications, and it has proven itself well suited to the difficult requirements of the Urban Challenge. All of the custom software logic for the Urban Challenge has been implemented as modules hosted by the AVS software framework, which allows them to be easily visualized and simulated by the AVS Console.

Significant testing of the vehicle has shown that the vehicle is capable of performing well in many difficult scenarios, and progress is still actively being made on improving the performance of the vehicle and successfully completing all of the requirements of the 2007 DARPA Urban Challenge.

References

- [1] Robert Bosch GmbH. CAN Specification Version 2.0, September 1991.
- [2] ISO. ISO International Standard 11898 – Road vehicles – Interchange of digital information – Controller Area Network (CAN) for high-speed communications, November 1993.
- [3] Livani, M.A., Kaiser, J. and Jia, W. Scheduling Hard and Soft Real-Time Communication in the Controller Area Network (CAN). 23rd IFAC/IFIP Workshop on Real Time Programming, Shantou, China, June 1998.
- [4] Kalman, R.E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering, Transactions ASME, Series D*, 82, 35-45, March 1960.
- [5] Kalman, R.E. and Bucy, R.S. New results in linear filtering and prediction theory. *Journal of Basic Engineering, Transactions ASME, Series D*, 83, 95-108, March 1961.
- [6] Dittmer, M. and Brumm, M. and Willhoeft, V. and Fürstenberg, K. Robust 360° environment observation using Laserscanners. 2002.
<http://citeseer.ist.psu.edu/dittmer02robust.html>.
- [7] Fayad, M. E., Schmidt, D. C., and Johnson, R. E., *Implementing application frameworks: object-oriented frameworks at work*. New York, NY, USA: John Wiley & Sons, Inc., 1999.
- [8] Fayad, M. E., Schmidt, D. C., and Johnson, R. E., *Building application frameworks: object-oriented foundations of framework design*. New York, NY, USA: John Wiley & Sons, Inc., 1999.
- [9] Schmidt, D. C. and Buschmann, F. Patterns, frameworks, and middleware: their synergistic relationships. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, (Washington, DC, USA), pp. 694–704, IEEE Computer Society, 2003.
- [10] Johnson, R. E. Frameworks = (components + patterns). *Communications of the ACM*, vol. 40, no. 10, pp. 39–42, 1997.
- [11] Trepagnier, P. G. A modular distributed software system for autonomous urban navigation. Ph.D. Dissertation, Tulane University, New Orleans, LA, 2007.
- [12] Bull, J. M., Smith, L. A., Pottage, L., and Freeman, R. Benchmarking java against c and fortran for scientific applications. In *Java Grande*, pages 97–105, 2001.
- [13] Kwon, J., Wellings, A., and King, S. Assessment of the java programming language for use in high integrity systems. *SIGPLAN Not.*, 38(4):34–46, 2003.
- [14] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison Wesley, 1994.
- [15] Mili, H., Elkharraz, A., and Mcheick, H. Understanding separation of concerns. In *Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, pp. 75–84, March 2004.
- [16] George, B. and Williams, L. An initial investigation of test driven development in industry.

- In SAC '03: Proceedings of the 2003 ACM symposium on Applied computing, pages 1135–1139, New York, NY, USA. ACM Press, 2003.
- [17] Johnson, R. E. and Foote, B. Designing reusable classes. *Journal of Object-Oriented Programming*, vol. 1, pp. 22–35, June/July 1988.
- [18] Schmidt, D. C., Gokhale, A., and Natara jan, B., Leveraging application frameworks. *Queue*, vol. 2, no. 5, pp. 66–75, 2004.
- [19] Hart, P., Nilsson, N., and Rafael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.
- [20] DeCarlo, R., Zak, S., and Drakunov, S.V.. *Variable Structure and Sliding Mode Control. The Control Handbook: a Volume in the Electrical Engineering Handbook Series*, CRC Press, Inc., 1996.
- [21] Dierckx, P. *Curve and Surface Fitting with Splines*. Oxford, England: Oxford University Press, 1993.
- [22] Drakunov, S. V., and Utkin, V. Sliding Mode Control in Dynamic Systems. *International Journal of Control*, v. 55, 4, pp. 1029-1037, 1992.
- [23] Drakunov, S.V. Sliding-Mode Observers Based on Equivalent Control Method. *Proceedings of the 31st IEEE Conference on Decision and Control (CDC)*, Tucson, Arizona, December 16-18, 1992, pp. 2368-2370.
- [24] Drazenovic, B. The invariance condition in variable structure systems,. *Automatica*, vol. 5, 1969, pp. ~287--295.
- [25] Filippov A.F. (1964), *Differential equations with discontinuous right hand side*, Amer. Math. Soc. Transactions , v. 42, Nb. 2, pp. 191-231, 1964.
- [26] Filippov A.F. (1988), *Differential Equations with Discontinuous Right Hand Sides* , Kluwer Academic Publishers, Boston, 1988.
- [27] Utkin, V.I. *Sliding Mode and their Application in Variable Structure Systems*, Moscow, Mir Publisher, 1977.
- [28] Utkin, V.I. *Sliding Modes in Control Optimization*, Springer-Verlag, Berlin, 1992.
- [29] Bartels, R. H., Beatty, J. C.; and Barsky, B. A. *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. San Francisco, CA: Morgan Kaufmann, 1998.
- [30] Gelfand, I.M. and Fomin, S.V. *Calculus of Variations*. Dover Publ., 2000.
- [31] Kirk, D. E. *Optimal Control Theory: An Introduction*. 2004.
- [32] Lebedev, L. P., and Cloud, M. J. *The Calculus of Variations and Functional Analysis with Optimal Control and Applications in Mechanics*. World Scientific. 2003.
- [33] Lewis, F. L., and Syrmos, V. L.. 1999. *Optimal Control*, 2nd ed. John Wiley & Sons.
- [34] McShane, E. J. *The Calculus of Variations from the beginning through Optimal Control Theory*. *SIAM Journal on Control and Optimization* 27 (5) (1989), 916-939.
- [35] Pontryagin, L. S. *The Mathematical Theory of Optimal Processes*. 1962.
- [36] Riekert, P. and Schunck, T. Zur fahrmechanik des gum-mibereiften kraftfahrzeugs. In *Ingenieur Archiv*, volume 11, pages 210–224, 1940.
- [37] Bernstein, D. A student's guide to classical control. *IEEE Control Systems Magazine*, 17:96–100, 1997.
- [38] Gill, A. *Introduction to the theory of finite-state machines*, NY, McGraw-Hill, 1962.
- [39] van Gurp, J., and Bosch, J. On the Implementation of Finite State Machines. *Proceedings of the 3rd Annual IASTED International Conference Software Engineering and Applications (SEA '99)* pp 172-178, October 6-8, 1999 Scottsdale, Arizona.
- [40] Sun Microsystems. *Reflection*. <http://java.sun.com/j2se/1.5.0/docs/guide/reflection/>, 2002.