TEAM OSHKOSH

DARPA URBAN CHALLENGE

Submission Date: April 13, 2007

Oshkosh Truck Corporation, Oshkosh, WI
Teledyne Scientific and Imaging Company, Thousand Oaks, CA
University of Parma, Parma, Italy
Auburn University, Auburn, AL
Ibeo Automobile Sensor GmbH, Hamburg, Germany

Corresponding Authors:

John Beck – Oshkosh Truck Corp.
jbeck@oshtruck.com

Alberto Broggi – University of Parma
broggi@ce.unipr.it

Venkataram Sundareswaran – Teledyne Scientific and Imaging Co.
sundar@teledyne.com

Benton Derrick – Auburn University
derrijb@auburn.edu

Executive Summary

Team Oshkosh understands the additional hurdles we must overcome by working with vehicles the size of TerraMax™, but feel it is the most relevant platform for development of autonomous systems which focus on providing logistics support. For this reason precision driving is necessary.

Building upon our previous successes, we have continued development in proven technologies such as the Vision, LIDAR and GPS/INS systems used in the Grand Challenge 2006 promise to allow TerraMax™ to detect and track static and moving obstacles with high precision.

A new service-based system architecture allows efficient path planning, trajectory algorithms and behavior modes ensure that sensor data is processed in a timely manner, while robust low level control and X by Wire components enable TerraMax™ to assess situations and react quickly.

Table of Contents

List of Acronyms

AVD	Autonomous Vehicle Driver
AVM	Autonomous Vehicle Manager
BML	Behavior Modes and Logics
CAN	Controller Area Network
CEP	Circular Error Position
COTS	Commercial Off The Shelf
DARPA	Defense Advanced Research Projects Agency
DPM	Drive Point Manager
ECU	Electronic Control Unit
ECBS	Electronically Controlled Braking Systems
EEG	Environmental Event Generator
FMVSS	Federal Motor Vehicle Safety Standards
FSM	Finite State Machine
FSMwP	Finite State Machine with Parameters
GPS	Global Positioning System
IMU	Inertial Measurement Unit
IRAD	Internal Research and Development
INS	Inertial Navigation System
LIDAR	Light Detection and Ranging
MDF	Mission Data File
MMBS	Mission Manager/Behavior Supervisor
MTM	Mode Transition Machine
MTVR	Medium Tactical Vehicle Replacement
RNDF	Route Network Definition File
RP	Route Planner
TSC	Teledyne Scientific Company
UF	Utility Functions
VSS	Vehicle State Server
VisLab	Vision Laboratory at the University of Parma
VMS	Vehicle Management System
WPS	World Perception Server

1 Overview

Team Oshkosh is comprised of Oshkosh Truck Corporation, Teledyne Scientific and Imaging Company (formerly known as Rockwell Scientific Company), the University of Parma’s VisLab, Auburn University, and Ibeo Automotive Sensor GmbH. Oshkosh Truck provides program management and overall design direction of the hardware, software and controls. Teledyne has developed the system architecture, mission and path planning, and much of the vehicle's high level control and programming. Auburn University is working to provide a well thought out and robust GPS/IMU package, VisLab is furthering development of vision capabilities, and Ibeo is providing software integration of the LIDAR system.

2 Hardware Design

2.1 Vehicle

Oshkosh Truck leverages a 4x4 variant of its Medium Tactical Vehicle Replacement (MTVR) platform, which has been designed for and combat-tested by the U.S. Marine Corps. Some capabilities of this vehicle have been enhanced to improve its performance to meet demands of the DARPA Urban Challenge. Oshkosh has decreased the turning radius, developed and installed low-level controllers and actuators for "by-wire" braking, steering and powertrain control. COTS computer hardware has been selected and installed for the vision system and vehicle control functions.

2.1.1 Rear Steer System

The nature of the DARPA Urban Challenge, particularly right-hand turns at intersections and various parking maneuvers, require higher precision maneuvers than previous Grand Challenges. To address these requirements TerraMax™ was designed to have improved turning capability over a standard MTVR by incorporating a 4 x 4 configuration instead of the standard 6 x 6 configuration. The reduced wheelbase and increased steering angle of the rear axle allows the turning radius to be reduced from 29 ft to 21 ft.

2.1.2 X by Wire Components

2.1.2.1 Braking

The standard air brake manifold of the MTVR has been replaced with a Bendix EBS5 Electronic Braking System. This brake-by-wire air brake system was chosen to allow full autonomous control via the J1939 bus and allows manual control in the event of human intervention or power failure.

2.1.2.2 Steering

The steering motor used to control TerraMax™ is a servo design originally used to automate clutch application for a manual transmission. It has been customized for this steering application and has been mounted on the column just below the dashboard. The motor input voltage is 28VDC, and it is controlled on CAN bus making it the right choice for vehicle application.

2.1.2.3 Engine

TerraMax™ uses the Caterpillar C12 11.9L inline 6 diesel engine normally found in MTVR trucks. This engine produces 425 HP 1515 ft-lbs torque. The software of this "throttle-by-wire" engine has been modified to allow it to receive commands over the J1939 CANbus, to change various parameters including throttle, "Jake" brake, and idle speeds.

2.1.2.4 Transmission

The transmission used in TerraMax™ is the standard Allison 4070 Transmission 7 speed transmission found in current MTVRs. This transmission is already a "shift-by-wire" transmission so no modifications needed to be made. The current Allison Gen IV software allows J1939 commands to emulate all of the commands found on the MTVR transmission panel (Reverse, Neutral, Driver Range [1-7])

2.1.3 Computer Hardware

The computation hardware had been sourced after a thorough search of the industry for the best performing units. Vehicle control PCs are (4) PC Advantech 3382 with Pentium M 1.4 GHz processors running Windows XP Pro. Due to the high processing demands, the 4 Vision PCs use SmallPC Core Duo computers running Linux Fedora. One PC is dedicated to each vision camera system (ie; trinocular, close range stereo, rearview and lateral). Vehicle Controller and Body Controller modules are custom Oshkosh embedded controllers and use the 68332 and HC12X processors respectively. All software is auto-code generated from Matlab and SimuLink at Oshkosh Truck.

2.2 Sensors

2.2.1 Vision System

2.2.1.1 Camera Selection

The specifications shared by all cameras are mainly dictated by the following considerations:

- Color, with the possibility to use also gray level
- External trigger for synchronization with other cameras and other sensors
- High speed FireWire (IEEE-1394b) connection for fast transfer of large amount of data
- Windowing and subsampling performed by the camera for increase frame rate
- Removal of the bayer pattern (if applicable)

The above considerations led to the choice of the following cameras:

- Medium resolution 1024 x 768, color camera with bayer pattern
- High resolution 1920 x 1080, 16:9 color camera with bayer pattern

Each system, given its specificity, needs a different optic. In the following all the vision systems are surveyed and the corresponding pair camera-optic is presented:

	Camera	**Optics**
Trinocular System	3 1024 x 768 color cameras	4.2 mm fl wideview
Rearview System	2 1024 x 768 color cameras	4.2 mm fl wideview
Front Stereo System	2 1024 x 768 color cameras	1.8 mm fl fisheye
Back Stereo System	2 1024 x 768 color cameras	1.8 mm fl fisheye
Lateral System	2 1920 x 1080 color cameras	2.8 mm fl wideview

Table 1: Vision camera and optics selection

2.2.2 LIDAR System

The Ibeo LIDAR system consists of three ALASCA XT laserscanners placed stragically to provide 360° total field of view. Two scanners located at the front corners of the vehicle, and one scanner in the center of the rear. As with all sensors and components, the mounting positions were detailed in 3D CAD for analysis prior to final mounting hardware fabrication.

3 System Architecture

3.1 Logical Architecture

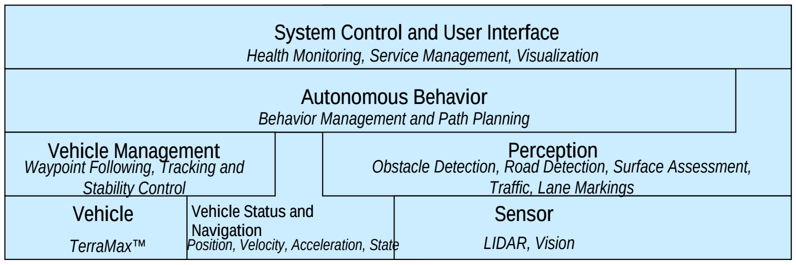

System Control and User Interface
Health Monitoring, Service Management, Visualization
Autonomous Behavior
Behavior Management and Path Planning
Vehicle Management
Waypoint Following, Tracking and Stability Control
Perception
Obstacle Detection, Road Detection, Surface Assessment, Traffic, Lane Markings
Vehicle
TerraMax™
Vehicle Status and Navigation
Position, Velocity, Acceleration, State
Sensor
LIDAR, Vision

Figure 1: Logical Architecture

The system architecture adheres to the layered architecture pattern. In the layered architecture pattern, dependencies flow downward – each layer depends on the layers below but not those above it. This architecture pattern was chosen because it facilitates a clean separation of functionality and development of well-defined interfaces between subsystems. This is particularly well-suited for Team Oshkosh as it has allowed team members to develop subsystems independently without sacrificing system cohesion.

3.2 *Deployment Architecture*

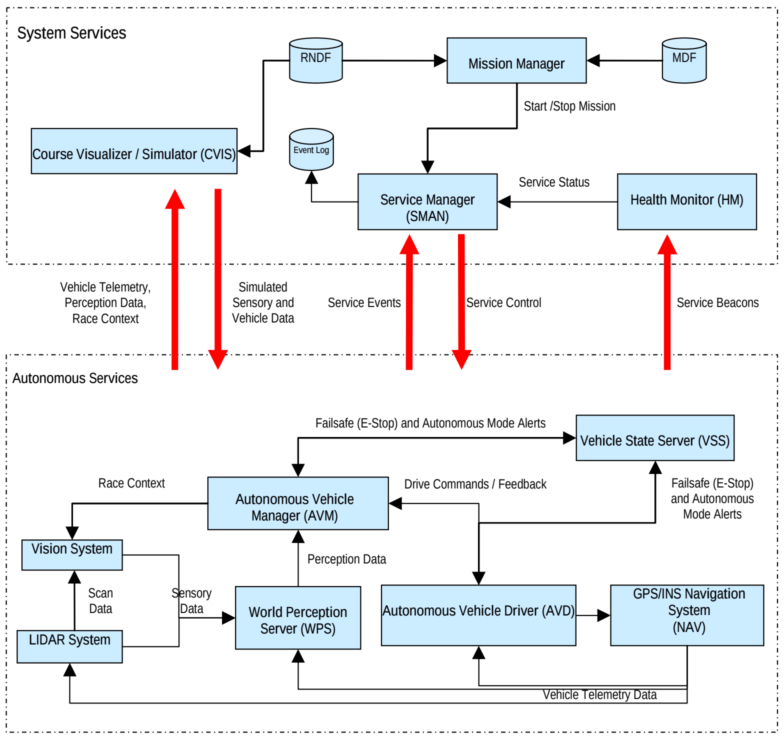

System Services
RNDF
Mission Manager
MDF
Start /Stop Mission
Course Visualizer / Simulator (CVIS)
Event Log
Service Manager (SMAN)
Service Status
Health Monitor (HM)
Vehicle Telemetry, Perception Data, Race Context
Simulated Sensory and Vehicle Data
Service Events
Service Control
Service Beacons
Autonomous Services
Failsafe (E-Stop) and Autonomous Mode Alerts
Vehicle State Server (VSS)
Race Context
Autonomous Vehicle Manager (AVM)
Drive Commands / Feedback
Failsafe (E-Stop) and Autonomous Mode Alerts
Vision System
Perception Data
Scan Data
Sensory Data
World Perception Server (WPS)
Autonomous Vehicle Driver (AVD)
GPS/INS Navigation System (NAV)
LIDAR System
Vehicle Telemetry Data

Figure 2: Deployment Architecture diagram

The deployment architecture consists of System Services and Autonomous Services. System Services include health monitoring, service control and configuration, and visualization. Autonomous services include sensor services, autonomous behavior services, perception services, and vehicle management services.

3.3 *Autonomous Services*

3.3.1 Autonomous Vehicle Manager

The Autonomous Vehicle Manager (AVM) manages the high level operation of the vehicle. It is primarily responsible for performing route planning, trajectory planning, and behavior management. The AVM receives perception updates from the World Perception Server (WPS) and uses this information to track the current vehicle state and determine the current behavior mode. The AVM continuously monitors perceived obstacle and lane boundary information and issues revised trajectory plans to the Autonomous Vehicle Driver (AVD) through Drive Commands.

3.3.2 Autonomous Vehicle Driver

The Autonomous Vehicle Driver (AVD) performs waypoint following, lateral, longitudinal and stability control by accepting messages from the AVM and commanding the lower level x-by-wire controls.

3.3.3 Vehicle State Server

The Vehicle State Server (VSS) monitors, manages and reports all system mode preconditions in the low level control that allow a transition from manual to autonomous operation. The VSS detects any faults, attempts to recover, and transitions the system into failsafe mode if necessary.

3.3.4 World Perception Server

The World Perception Server (WPS) publishes perception updates containing the most recently observed vehicle telemetry, obstacle, and lane/road boundary information. The WPS subscribes to sensory data from the LIDAR and VISION systems. The WPS combines the sensory data with the vehicle telemetry data received from the navigation service (NAV). Obstacles detected by the LIDAR and VISION systems are further fused in order to provide a more accurate depiction of the sensed surroundings. The AVM consumes the perception updates published by the WPS and uses this information to determine the next course of action for the vehicle.

3.3.5 Vision System

The vision system is composed of 11 cameras in order to sense the environment around the vehicle in the most important directions.
Vision must be able to provide sensing to implement the following functionalities:

- Lane/path detection for straight driving
- Lane markings detection for precise driving and maneuvering
- Obstacle detection for straight driving and maneuvering
- Curb detection for maneuvering
- Backward vehicle detection for possible lane changing
- Oncoming traffic detection when stopped at a stop line

All these functions need specific subsystems able to sense the relevant information and pass them on to the WPS in real-time. The vision systems receive map data and vehicle information from other vehicle subsystems, and the integration and fusion with raw data coming from three laserscanners is planned.

3.3.5.1 Trinocular

The sensing of obstacles up to a distance of 50m is of paramount importance when driving in an urban environment since other vehicles traveling in either direction, or immobile obstacles may be on the vehicle's trajectory. Stereoscopic systems offer great advantages, since they reconstruct the 3D environment and therefore help to detect anything above the road surface.
The innovative solution successfully designed and fielded in the 2005 DARPA Grand Challenge is used again: an asymmetrical trinocular system forms three stereo systems with three different baselines. According to the needed depth of perception (usually linked to vehicle speed) the software selects the pair of cameras to use. The two image streams are fed to the 'trino' PC which detects the road slope thanks to the use of the V-Disparity approach; it then detects any

kind of obstacle (defined as anything raising up from the surface determined as being the road surface by the previous step), performs a 3D reconstruction of the scene, and finally provides these information to the WPS. Both gray level and color processing have been developed, and we are investigating whether the higher computational time of color processing (almost doubled with respect to gray level) provides a better understanding of the 3D world.

The low-level fusion with laserscanner data is now currently being studied and implemented.

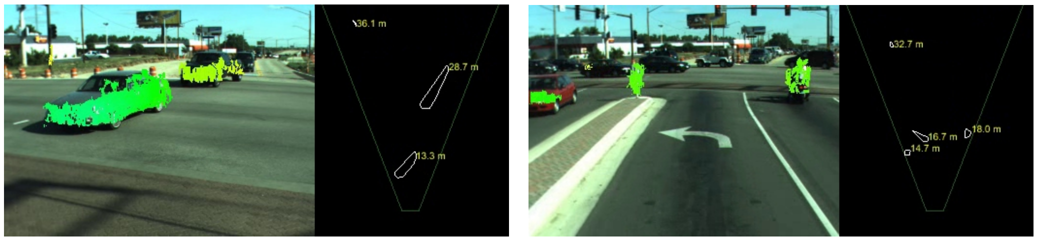


Figure 3: Examples of obstacle detection in an urban environment. Left: obstacles map (color encodes distance); right: top view of the scene in front of the vehicle

The trinocular system is also used to perceive the lane markings ahead of the vehicle. More precisely, a color processing detects the presence of candidates of white and yellow lane markings and groups them together according to their position, orientation, and probability. The lane markings detection software runs on the same PC used for obstacle detection since it can benefit from the results of obstacle detection. The position of obstacles is used to mask portions of the acquired image in correspondence to obstacles, so that the lane markings detection algorithm focuses on road texture only.

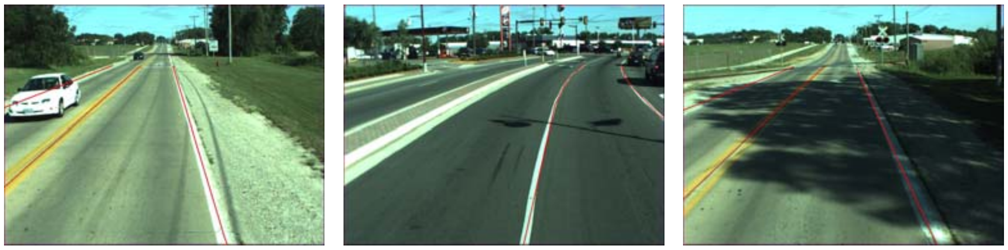


Figure 4: Examples of lane detection in urban and rural environments

In case lane detection does not provide robust and stable data as output, a new sequence of processing is triggered instead. Free Space Detection and Path Detection (similar to the ones successfully fielded in the 2005 DARPA Grand Challenge) are used to locate a portion of the area in front of the vehicle which is regarded as smooth enough to be safely traversed and has a shape that resembles a driveable path.

Both these systems have a minimum depth of perception of 7m, limited by the position of the cameras: in order to keep the cameras in a safe position, they are installed into the driving cabin; the presence of the hood limits the visibility in the close range.

3.3.5.2 Stereo Systems

The navigation in an urban environment must be sharp and very precise. The trinocular system described in the previous section can only be used for driving at medium-to-high speeds, since it covers the far range (7-50m). TerraMax™ includes two stereo systems (one in the front and one in the back) which provide a precise sensing in the close proximity of the vehicle. The two stereo systems are identical: the four cameras are connected to the same PC which selects which system to use based on driving direction (forward or backward). Images are acquired at 1024x768 pixels, with bayer pattern. Thanks to wide angle (fisheye, about 160°) lenses, these sensors gather information about an extended area of the immediate surroundings; the software is designed to improve the detection of the trinocular system. The stereo systems (based on software included into other intelligent vehicles prototypes) are designed to detect obstacles and lane markings with a high confidence on the detection probability and position accuracy.

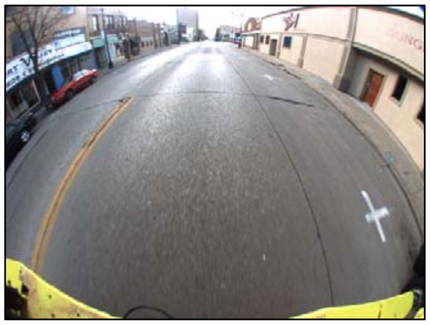


Figure 5: An image acquired by one of the cameras of the forward stereo system with the fisheye lens: image distortion is clearly visible.

Obstacle Detection is performed in two steps: first the two images, acquired simultaneously, are preprocessed in order to remove the very high lens distortion and the perspective effect, and then the two images are matched so that any appreciable deviation from the flat road assumption is labeled as an obstacle. The flat road assumption is considered valid since the area of interest is limited to a maximum of 8m. From this assumption the system:

- is able to detect any kind of obstacle without a-priori defined obstacle classes
- is not based on motion, which usually generates false positives when the vehicle moves
- is robust with respect to shadows on the ground since it performs a comparison between images taken at the very same time and since shadows belong to the road texture

The Lane Markings Detection module is based on the processing of a monocular stream. Following the same ratio used for the trinocular system, once obstacles are detected, their area is removed from the analysis, thanks to a masking operation. The image is then analyzed using a Hough transform so that all straight segments are detected; reasonably spaced parallel segments including an area brighter than the surroundings are paired, and a lane marking is detected.
The module is used to detect also stop lines, which are defined as thicker horizontal lane markings. The same functionality is also used to detect curbs for backward maneuvers.

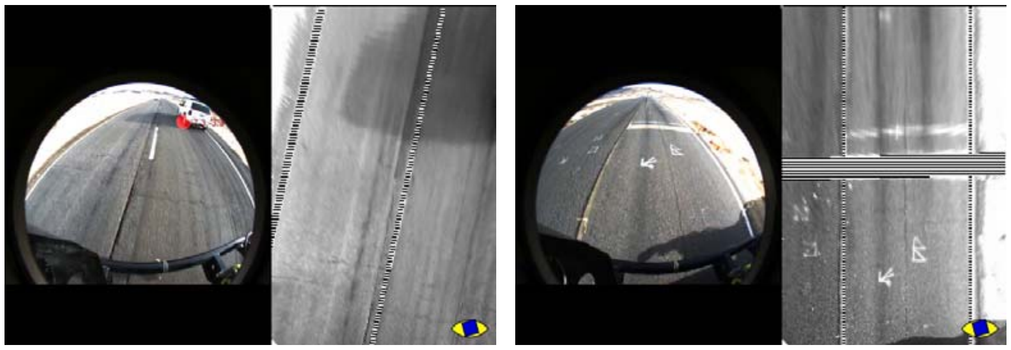


Figure 6:The StereoBox processing. Left: original image with obstacles identified by red markers; right: unwarped image on which lane markings and stop lines are identified by dashed lines

As the detected obstacles and lane markings get closer and closer to the camera, the evaluation of their position becomes more and more precise, yet remaining in the camera field of view thanks to the fisheye lens. In this way a precise localization of lane markings allows both to keep the vehicle inside the lane despite its large width, and stop the vehicle just in front the stop line in a junction. Plus, the capability of detecting lane markings at intersections helps in the turning maneuver. At the same time, a precise localization of the obstacle in front allows a sharp planning of obstacle avoidance maneuvers. Fusion with LIDAR raw data is now under study.

3.3.5.3 Lateral System

Lateral perception is important when facing an intersection. When the vehicle has to yield to oncoming traffic, it has to hold until all traffic has cleared the junction. The vehicle needs to perceive the presence of oncoming traffic, regardless of the junction layout. In other words, the intersecting road might be seen from an angle, which in general may be different from 90°. Therefore the lateral system must be able to perceive traffic coming from different angles.

We installed two high resolution cameras (1920x1080 pixels) on TerraMax™ – one on each side – for lateral view, together with 90° optics, the main advantages being:

- the camera covers a wide angle (90°)
- objects can be seen with a good resolution even when they are very far from the camera.

Indeed the handling of a high resolution image is generally regarded as a computational expensive problem. However, when the vehicle needs to yield to oncoming traffic (i.e. in the situations in which the lateral system is of use) the demand for real-time processing is generally low, being the vehicle stopped at a stop line. Furthermore an *attentive* processing allows to reduce the computational load: first, an image stream is loaded from the camera at a low resolution (480 x 270 pixel) and is processed with the aim of detecting the areas of interest, driven by the presence of motion; then, once the area of interest has been defined, a new stream of images (at the maximum resolution) is loaded from the camera, using the camera's capability of delivering windows instead of entire frames. In this way the system focuses its attention to an area which is dynamically defined as a function of the junction's layout, truck heading, and presence of oncoming traffic. Although using a single fixed camera, the system is able to handle different scenarios.

During the high resolution processing, the tracking of the movement provides the main direction of motion; at the same time a color analysis of the low resolution image stream (480x270 pixel) is used to detect the road location. A further step, still under development, is aimed at matching the motion and road structure in order to eliminate motion which is out of the road boundaries.

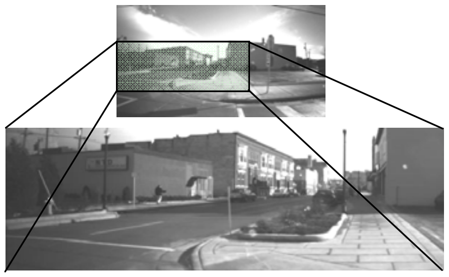


Figure 7: Lateral System. Top: original low resolution image, a rectangle determines the attentive area; bottom: high resolution image used to detect oncoming traffic

3.3.5.4 Rearview System

When driving on a road with multiple lanes, the lane change maneuver is a common task in order to overtake slower traffic. Two lateral cameras installed on top of the driving cabin acquire images of the road behind and on the vehicle side. The cameras are installed rotated by 90° and frame the scene in portrait mode so that, despite being installed at 2.8m, they can frame the area close to the vehicle and extend their vision over the horizon. The images are acquired at a 384 x 512 pixels resolution, in color, leaving the de-bayer processing to the camera.

The processing aimed at Vehicle Detection is based on color image segmentation, and tracking of the blobs. Blobs moving towards the truck represent vehicles, while all the other blobs that move away from the truck are generated by background features or static obstacles. The current system is able to detect approaching vehicles at about 30m.Currently the system does not use data coming from the vehicle (speed, yaw rate, steering angle), nor laserscanner raw data, and the processing rate is 20Hz. When these additional data will be included the processing cycle will be fixed to 10Hz.

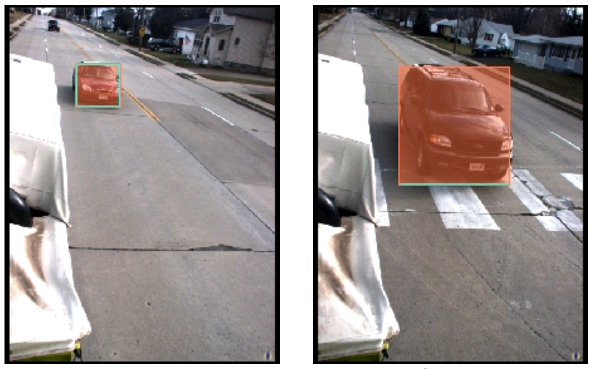


Figure 8: Rear-view System: two examples of vehicle detection.

We are also considering whether to include a further detection of lane markings behind the vehicle, in order to refine the area of interest for the vehicle detection step. Currently the whole image is processed, but a lower computational load may be needed if the vehicle detection step could be preceded by a precise localization of the nearby lane.

3.3.6 LIDAR System

Precise measurements of distance, velocities and headings of objects are paramount to the vehicle's perception and navigation of the urban environment. We have integrated a COTS LIDAR system that provides a 360° field of view. The system software has been adapted to operate within our system architecture messaging schema. Each laserscanner scans a 240° horizontal field. Two laserscanners are positioned on the front corners of the vehicle and one laserscanner is positioned in the center of the rear. The front scanners are fused at the low level and fusion of the rear scanner is being investigated. Obstacle detection and tracking is performed in the system ECUs and published to the WPS, while raw scan data will be used by the vision systems to enhance precision and aid in identifying interest areas.

3.3.7 GPS/INS

3.3.7.1 Oxford GPS/INS

The Oxford RT3052 GPS/INS solution combines a six degree freedom inertial measurement unit with a dual antenna GPS receiver and wheel odometry to produce a total navigation solution. In the Urban Challenge, there exists the necessity to have a good navigational solution in the presence of a GPS outage. In order to evaluate the performance of the Oxford, a Novatel GPS receiver with Omnistar HP corrections was used as a truth measurement. The Omnistar corrections guarantee that the receiver position is within 10 cm of the actual position 95% of the time. The trajectory used to evaluate the Oxford using IMU, GPS and odometry measurements is shown in Figure 9. The starting point is in the top right. In this test, the maximum error observed over approximately a minute is 1.3 m. The error over the test can be seen in Figure 10.

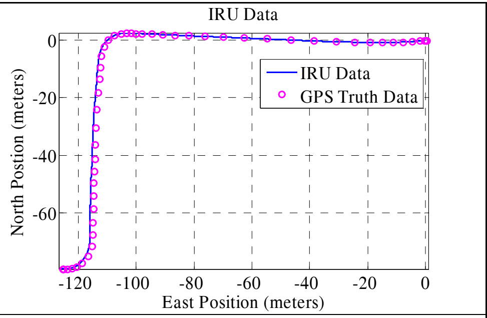


Figure 9: Trajectory with GPS

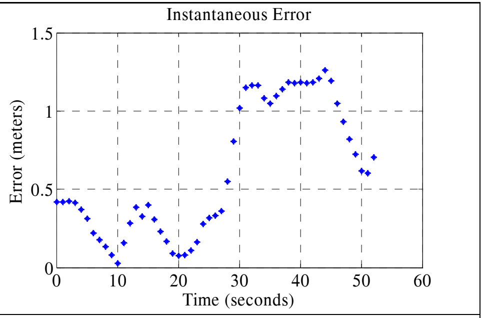


Figure 10: Absolute Error with GPS

A second test was conducted to evaluate the performance of the navigational solution in the presence of a GPS outage. In order to accomplish this part of the evaluation, the GPS antennas were disconnected from the Oxford after it was initialized. The path followed can be seen in Figure 11 As before, the path starts in the top right and then goes left. As can be seen in Figure 12, the error grows to about 8 m after 220 s.

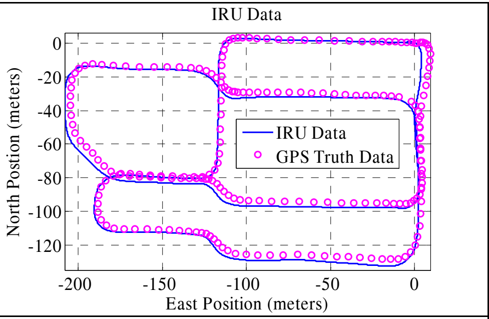


Figure 11: Trajectory without GPS

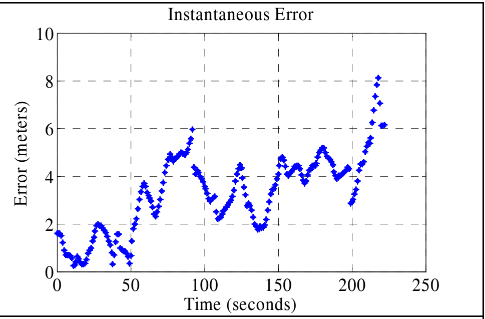
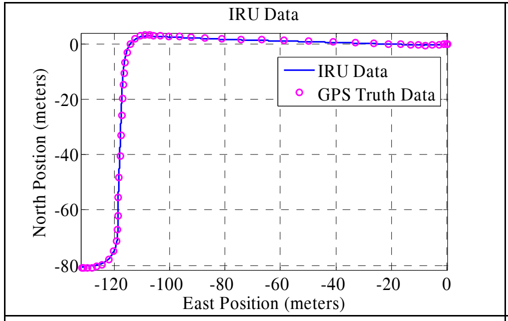


Figure 12: Absolute Error without GPS

3.3.7.2 Smiths GPS/INS

The Smiths Aerospace inertial reference unit, like the Oxford RT3052, combines a six degree of freedom inertial measurement unit with a single antenna GPS receiver and wheel odometry. Like the evaluation of the Oxford, the Smiths IRU was tested along the path shown in Figure 13. In this run IMU, GPS, and odometry measurements were all available to the Smiths IRU's Kalman filter. The same Novatel GPS receiver with Omnistar corrections was used as a truth measurement accurate to 10 cm 95% of the time. As can be seen in Figure 14, the maximum error incurred during the test period was 1.2 m.



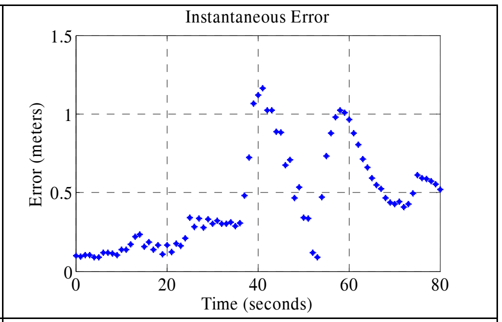
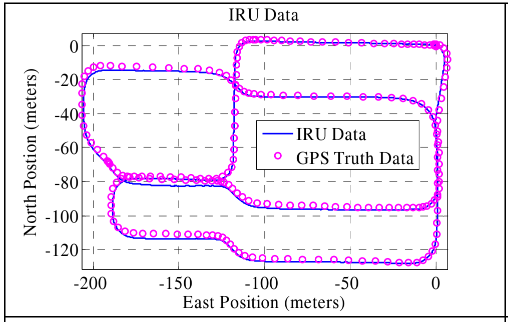


Figure 13: Trajectory with GPS | **Figure 14: Absolute Error with GPS**

To evaluate the Smiths IRU without GPS available, the path shown in Figure 15 was followed starting at the top right and going left. Before the test, the Smiths IRU was initialized with GPS and then GPS measurements were disconnected from the unit. As can be seen in Figure 16, the maximum error incurred over the 220 s test was about 4.1 m.



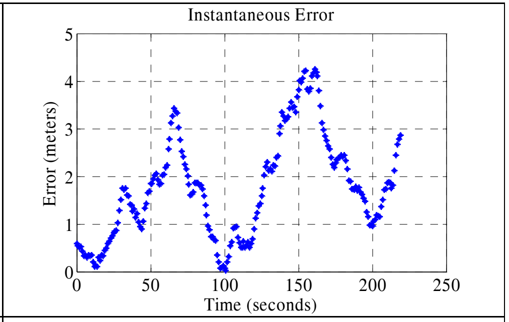


Figure 15: Trajectory without GPS | **Figure 16: Absolute Error with GPS**

Due to a more robust initialization routine and greater accuracy without GPS, the Smiths IRU is deemed a better choice for the GPS/INS solution for Team Oshkosh.

3.4 System Services

3.4.1 Health Monitor

The Health Monitor maintains the status of all services by monitoring service beacons that are sent as UDP multicast data packets. Each service is required to send a service beacon at a prescribed periodicity. The service beacon contains the services status, service port, and IP address. The Health Monitor sends alert notifications when a service has reported an error or when a service has not sent a beacon within the allotted timeframe. The Service Manager will respond to the health monitor alert notifications and take appropriate measures; such as restarting an unresponsive service.

3.4.2 Service Manager

The Service Manager manages the initialization, startup, and shutdown of all autonomous services. During system startup, the Service Manager orchestrates the startup process using service dependency information that is read from a configuration file. The Service Manager starts or stops the autonomous service in response to the mission start/stop notifications sent by the Mission Manager application. The Service Manager maintains the mission status so that autonomous services can be properly configured and restarted in the event of system reset.

The Service Manager monitors health alerts originating from the Health Monitor. When a service is no longer responding or has indicated an error, the Service Manager will take appropriate actions, such as attempting to restart the service or causing system reset.

3.4.3 Mission Manager

The Mission Manager provides the user interface for configuring and starting the autonomous services in preparation for autonomous mode. The application allows a user to load the Road Network Definition File (RNDF) and Mission Definition Files (MDF) files. The Mission Manager validates the RNDF and MDF files before loading them into the system. To start a mission, the user clicks on the “Start” button. This instructs the Service Manager to initialize and start all autonomous services.

3.4.4 NAV Service

The NAV Service manages communications to the GPS/INS unit and converts latitude/longitude outputs to UTM coordinates.

4 Planning and Behaviors

4.1 Overview of Vehicle Behaviors

We adopt a goal-driven / intentional approach in mission planning and generation of behaviors for our vehicle. The main goal for the mission and behavior generation is to navigate, sequentially, through a set of checkpoints as prescribed in the DARPA supplied MDF. Functionality related to autonomous vehicle behaviors are implemented in the Autonomous Vehicle Manager (AVM) described earlier. Main components in the AVM (as depicted in Figure 9) include:

- Mission Manager / Behavior Supervisor (MMBS), which manages the pursuit of mission goal (and subgoals), selects and supervise appropriate behavior mode for execution;
- Route Planner (RP), which generates (and re-generates) high-level route plans based on the road segments and zones defined in the (RNDF);
- Behavior Modes & Logics (BML), which contain behavior modes, the transitional relationship among them, and the execution logics within each behavior mode;
- Environmental Event Generator (EEG), monitors the vehicle and environment state estimations from the WPS and generates appropriate events for the behavioral modes when a prescribed circumstance arises (e.g. an obstacle in the driving lane ahead, etc.);

- Utility Functions (UF), provide common services (e.g. trajectory generation, etc.) for different behavior modes; and
- Drive Point Manager (DPM), which manages the set of imminent waypoints for driving and other interactions with the Autonomous Vehicle Driver (AVD).

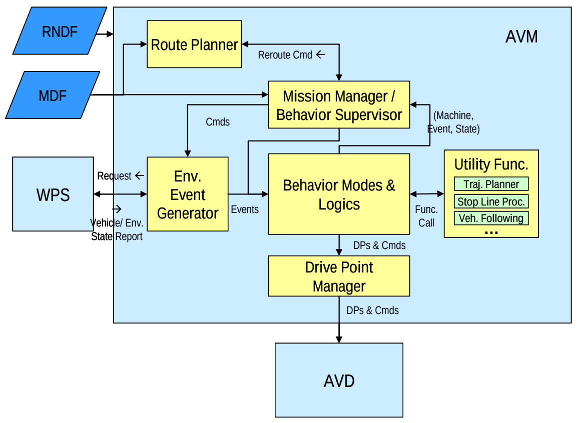


Figure 19: Major function blocks in the Autonomous Vehicle Manager (AVM)

An example of interactions among these components is as follows: Given the main goal of the mission in MDF, the MMBS decomposes the mission goal into a set of sequential sub-goals of reaching each checkpoint and calls the RP to generate high-level route plans. After the start of the mission, the MMBS selects the most appropriate behavior mode in BML based on the current mission / vehicle status. Utilizing appropriate utility functions, the BML executes the selected behavior mode according to the vehicle state, its behavior mode logics, and environmental events (generated by the EEG) and generates a sequence of waypoints for driving and corresponding speed limitations. These waypoints and speed instructions are managed by the DPM and sends to AVD for execution. The MMBS supervises the execution of the behavior mode and may switch to different behavior modes according to the transition conditions described in the BML. If the current sub-goal is not attainable through existing route plan, the MMBS may call the RP to re-generate an alternative route plan for the current road and environmental conditions.

4.2 Behavioral Modes and Supervision

We adopt a Finite-State-Machine (FSM) based discrete-event supervisory control scheme as our primary approach in generating and executing autonomous vehicle behaviors. The FSM based scheme provides us a simple yet structured approach to effectively model the race rules/constraints and behaviors/tactics, as opposed to the conventional rule-based approaches. This scheme allows us to leverage existing supervisory control theories and techniques [10,3,7,6] to generate safe and optimized behaviors for the vehicles.

Figure 20 illustrates major components for our approach. In this approach, we model autonomous behaviors as different behavior modes. These behavior modes categorize the potential situations the vehicle may encounter during the race and enable optimized logics and tactics to be developed for the situations. Examples of the behavior modes that we implement include:

- Lane Driving, where the vehicle follows a designated lane based on the sensed lane or road boundaries;
- Inspecting Intersection, where the vehicle observes the intersection protocol and precedence rules in crossing the intersection and merging with the existing traffic;
- Lane Changing, where the vehicle moves from the current lane to a designated lane;
- Passing, where the vehicle passes a sensed obstacle or (moving) vehicle in front by temporarily moving to a different lane and returns back to the original lane;
- Multi-point Turn, where the vehicle reorients itself to a prescribe lane or heading by making a series of multi-point, possibly, back-and-forth, maneuvers; and
- Zone Driving, where the vehicle drives through the set of waypoints in a prescribed zone.

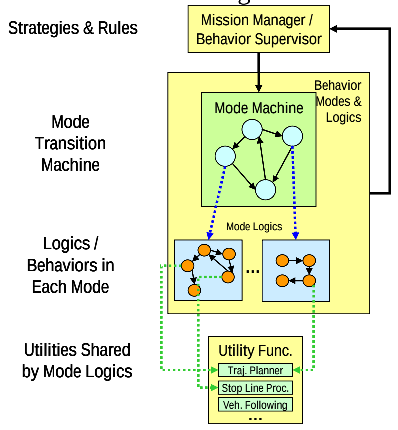


Figure 20: Components for Behavior Generation and Execution

For each behavior mode, a set of customized logics is modeled as an extended finite state machine (e.g., a Finite State Machine with Parameters (FSMwP) [4]), which describes the potential behavior steps, conditions, and actions to be taken. The behavior logics may employ different utility functions during execution. Note that, different behavior modes may share the similar behavior logics. For example, the behavior logics for the Passing mode can be viewed as an "aggregation" of the behavior logics for Lane Changing, Lane Driving, and then followed by another Lang Changing. However, by implementing the passing maneuver as a separate behavior mode enables us to explicitly specify conditions for checking and exception handling for this specific sequence of maneuver and may improve the performance in selecting the behavior modes during the execution.

Transitions among behavior modes are modeled explicitly as an FSMwP [4], named Mode Transition Machine (MTM), where guard conditions and potential actions / consequence for the transitions are expressed. The MTM is used by the Behavior Supervisor in MMBS in determining the appropriate behavior mode to transition to during execution.

With the autonomous behaviors categorized and implemented as a collection of behavior modes and the transitional relations among the modes modeled as an FSMwP (MTM), the generation and control of the vehicle behavior may be formulated as a supervisory control problem. However, unlike the original supervisory control approaches [10,3,7] which focused mainly on passively maintaining the system behaviors to remain in a certain prescribed “good sub-space”, the vehicle behavior generation requires actively driving the system behaviors toward a given “goal”. Hence, we adopt an approach that combines safety control [5] and optimal effective control [6] for FSMwP where we formulate the traffic rules and protocols as safety constraints and current mission sub-goal as the effective measure to achieve. To further improve the real time performance during execution, we are currently experiment with a simplified supervisor implementation that does not require explicitly expansion of supervisor states, by exploiting the structure of the MTM.

Our finite state machine based behavior generation scheme is intuitive and efficient. However, it may suffer from several potential drawbacks. Among them are the robustness concern in handling unexpected situations and the lack of “creative” solutions/behaviors. To mitigate the potential concern in handling unexpected situation, we will include an unexpected behavior mode and institute exception handling logics to try to bring the vehicle to a know state.

4.3 Route Planning

The objective of the route planning component is to generate an ordered list of road segments among the ones defined in RNDF that enables the vehicle to visit the given set of checkpoints in sequence with least perceived cost in the current sensed environment. We implement the route planner by modifying the well-known Dijkstra’s Algorithm [8], which adopts a greedy search approach in solving the single-source shortest path problem.

We use the estimated travel time as the *base* cost for each road segment, instead of the length of the road segment. This modification allows us to take into account of the speed limit of the road segments and the traffic conditions the vehicle might experience traveling through the same road segment previously (during the same mission run). Meanwhile, any perceived road information, such as road blockage, and (static) obstacles may be factored into the cost of the road. We plan to experiment on different approaches in adjusting the road segment costs to reflect these perceived road conditions. Future versions of the code could actually learn expected travel times based upon prior history of the vehicle driving a given road segment.

To further optimize the route planning results, we incorporate the cost of vehicle turning in our route planning calculation. Turning cost from a road segment to a neighboring road segment is analyzed and determined when the RNDF is loaded based on the turning direction and intersection condition (e.g. protected with stop sign, unprotected, etc.).

Constructing a graph from the RNDF is not as straightforward as one would expect as the data provided in the file only explicitly describes waypoints (represented as nodes in our graph). The edges connecting waypoints are only implicitly described in the RNDF; some of the rules we used to connect the waypoints were trivial such as connecting adjacent waypoints to the next waypoint in a road segment or connecting lane exits to their corresponding lane entrances. Others such as connecting road segment with multiple lanes traveling at the same direction required us to introduce fictitious “crossing segments” in the lanes to simulate the possible lane changes between the two lanes. This modification enables the route planner to properly plan for a necessary lane change between the lanes, which usually results from a subsequent turn at the intersection. The explicitly planned lane change also provides much easier guidance during the behavior generation. In future versions we envision adding other fictitious segments and even additional waypoints to support better navigation, parking and collision avoidance in RDNF zones.

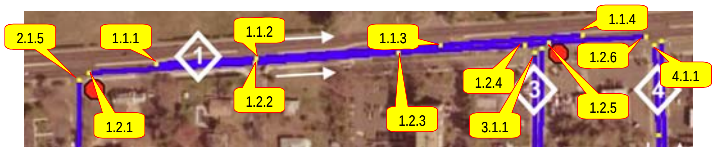


Figure 21: A partial view of the sample RDNF file with waypoint numbers marked.

$$\begin{tikzcd}
1.1.1 \arrow[r] & 1.1.2 \arrow[r] & 1.1.3 \arrow[r] & 1.1.4 & & \\
1.2.1 \arrow[r] & 1.2.2 \arrow[r] & 1.2.3 \arrow[r] & 1.2.4 \arrow[r] \arrow[d] & 1.2.5 \arrow[r] & 1.2.6 \arrow[d] \\
2.1.5 \arrow[u] & & & 3.1.1 & & 4.1.1
\end{tikzcd}$$

Figure 22: The resultant graph of the waypoints shown in Figure 4 without crossing segments.

4.4 Trajectory Planning

The trajectory planning generates a sequence of dense and drivable waypoints for the AVD to execute, given a pair of start/end waypoints and, possibly, a set of intermediate waypoints. We implement the trajectory planning capabilities as a collection of trajectory planner utility functions that may be called upon by different behavior mode logics depending on the current vehicle and mission situation.

Three different types of trajectory planners are implemented, which is summarized in the following:

- **Lane following trajectory planner**: this planner utilizes detected estimated lane and road boundaries to generate waypoints that follow the progression of the lane / road. The lane following trajectory planner tries to generate the waypoints close to the center line of the lane. It first generates the estimated centerline points using detected estimated lane boundaries on the both side of a designated lane, if both are present and supplementing that with other information such as road boundaries, default lane width, next given waypoint, etc. when either or both boundaries are missing. Then it checks and removes potential out-of-order waypoints and applies a cubic spline based algorithm to further smooth the generated trajectory waypoitns. Figure (a) shows a trajectory generated by the lane following trajectory planner.
- **Templated trajectory planners** are a set of trajectory planners that can quickly generate trajectory waypoints based on instantiation of templates with current vehicle and environment state estimates for common maneuvers such as lane changing, passing, and turning at intersections. The templates are developed using approaches similar to the simple geometric methods in [9] (without wrapping from lane space to real road space). Figure (b) shows a trajectory generated by the templated turning trajectory planner.

- **Open-space trajectory planner** provides general trajectory generation and obstacle avoidance in a prescribed zone where obstacles may be present. We leverage the open space/lane trajectory planner that we used during our previous participations in 2005 DARPA Grand Challenge (see Team TerraMax™'s work in [11,12]).

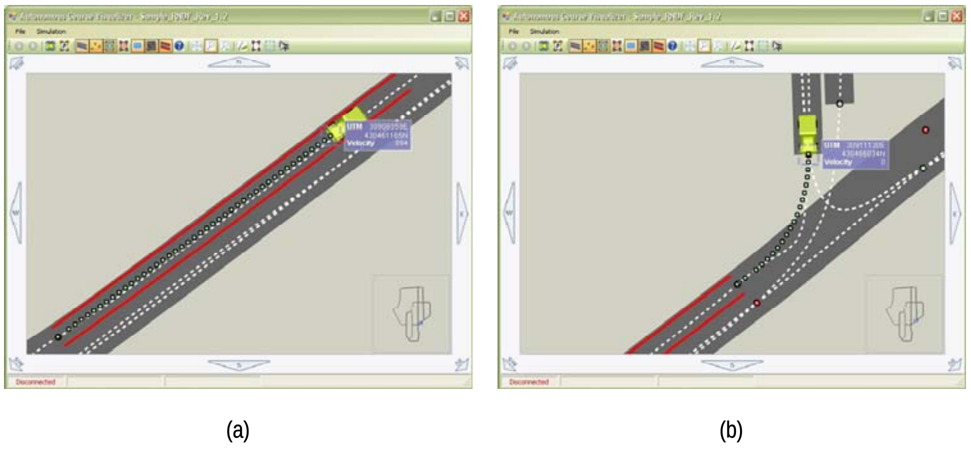


Figure 23: Trajectories generated from (a) lane following trajectory planner, (b) templated turning trajectory planner.

5 Results and Performance

5.1 Basic Navigation

5.1.1 Mission Start and Planning

Before entering into autonomous mode, the system must first locate the vehicle in the route network. A vehicle location algorithm is used that enables start or re-start of a mission from any road segment on the route network. This feature has proven useful when performing road tests on the TerraMax, as it allows the vehicle to reenter autonomous mode without having to drive to a predetermined course location.

The TerraMax™ system automatically plans a route through route network using the sequence of checkpoints provided in the MDF file. The current algorithm is based on the well-known Dijkstra's Algorithm [8], which adopts a greedy search approach in solving the single-source shortest path problem. The route planning algorithm, which uses travel time as a base cost, has been tested in simulation and on the TerraMax. Using turning cost for route optimization has not yet been tested.

5.1.2 Passing

The ability to pass a stationary vehicle has been tested in simulation. Testing on the TerraMax will commence shortly.

The system will exhibit passing behavior when it has come to rest behind a stationary obstacle in the planned path. Before initiating the passing maneuver, the system first inspects the current passing conditions to ensure that is legal and safe to begin the operation. During the passing operation the system monitors the return lane for obstacles and returns at the earliest opportunity, taking into account the required min/max distances from the vehicle being passed.

5.1.3 U-Turn

The ability to perform a U-Turn has been tested simulation and testing on the TerraMax vehicle has recently commenced.

A U-Turn will be attempted as required by the planned route or when a replan is required due to a road block condition. Before attempting a U-Turn the system first inspects the local area to ensure it is free from obstacles. The vehicle trajectory planner then generates a trajectory to accomplish the multi-point turn. Figure 1 shows a U-Turn trajectory that was issued during testing on the TerraMax.

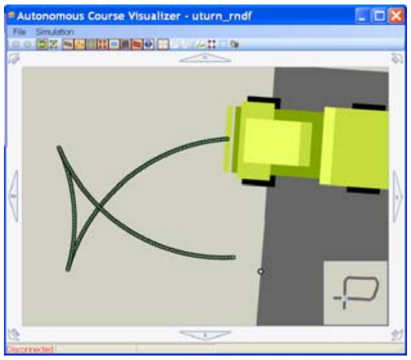


Figure 14 U-Turn Trajectory

5.1.4 Lane Keeping

The ability to stay in the current lane has been tested in simulation and on the TerraMax vehicle.

The TerraMax combines GPS-based position information along with lane boundaries sensed by the Vision System to generate trajectories that keep the vehicle centered within the driving lane.

5.1.4.1 Lateral Control

We began our development of lateral control using Pure Pursuit (13) as our vehicle path tracking algorithm and found that it produced poor results in tight corners, often cutting corners by more that a meter. Do to the width of TerraMax™ it was specified that the steering performance should not result in the vehicle deviating more than 25 cm from the intended path. We investigated numerous path tracking algorithms and found that Vector Pursuit (14), an algorithm that tries to reach the intended location and heading of the lookahead point, reduced lateral path error by up to 50% on tight corners. In both simulation and vehicle testing, Vector Pursuit has shown to be a vast improvement over Pure Pursuit and puts us much closer to our intended path error, but is still not within our specifications. Further work on a proprietary algorithm has shown exceptional results in simulation and initial vehicle trials (Figure 24). We are now in the process of ensuring the robustness of the new algorithm.

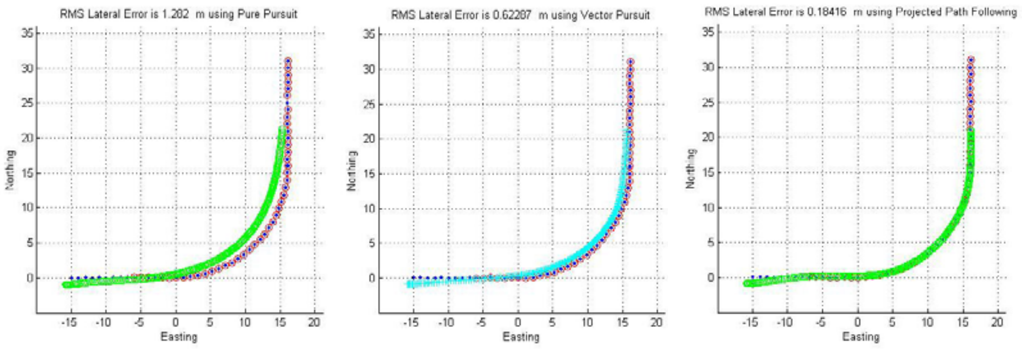


Figure 24: Plot data showing path tracking deviation

5.2 Basic Traffic

5.2.1 Intersection Precedence

During Simulation the vehicle properly observed the precedence order and did not attempt to proceed out-of-turn. Using the Course Visualizer, obstacles were placed at the various exit points before the vehicle arrived at the intersection. The TerraMax™ did not proceed until all vehicles had cleared the intersection.

Testing on the TerraMax vehicle demonstrated similar behavior, the TerraMax yielded right of way to a vehicle that was already present at the intersection. Additional tests scenarios will be excised in the upcoming weeks.

6 References

1. R. Arkin. Behavior-based Robotics. MIT Press, 1998.
2. R. Brooks. A robust layered control system for a mobile robot. *IEEE J. of Robotics and Automations*, RA-2(1): 14-23, 1986.
3. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. 2nd ed. Kluwer, 1999.

4. Y.-L. Chen and F. Lin. Modeling of discrete event systems using finite state machines with parameters. *In Proc. 9th IEEE Int. Conf. on Control Applications*, pp. 941-946, September, 2000.
5. Y.-L. Chen and F. Lin. Safety control of discrete event systems using finite state machines with parameters. In *Proc. 2001 American Control Conf.*, pp. 975-980, June, 2001.
6. Y.-L. Chen and F. Lin. An optimal effective controller for discrete event systems. In *Proc. 40th IEEE Conf. on Decision and Control*, pp 4092-4097, December, 2001.
7. S.-L. Chung, S. Lafortune, and F. Lin. Limited lookahead policies in supervisory control of discrete event systems. *IEEE Trans. on Automatic Control*, 37(12):1921-1935, 1992.
8. T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*, 2nd ed., Sec. 24.3: Dijkstra's algorithm, pp. 595-601. 2001.
9. J. Horst and A. Barbera. Trajectory generation for an on-road autonomous vehicle. In *Proc. of SPIE., Vol. 6230, Unmanned Systems Technology VIII*, May, 2006.
10. P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206-230, 1987.
11. V. Sundareswaran, C. Johnson, and D. Braid. Implications of lessons learned from experience with large truck autonomous ground vehicles. In *Proc. AUVSI 2006*, 2006.
12. Team TerraMax™. http://www.terramax.com.
13. O. Amidi, "Integrated Mobile Robot Control," M.S. Thesis, CMU. (May 1990).
14. J S. Wit; Vector Pursuit Path Tracking for Autonomous Ground vehicles, Ph.D thesis, University of Florida, 2000