

---

June 1, 2007

# DARPA Urban Challenge 2007

## Team Urbanator Technical Description



### Abstract

Military operations in urban environments, such as logistical resupply and transport are becoming some of the most dangerous operations that the military has to perform. The Defense Advanced Projects Agency (DARPA) is sponsoring the Urban Challenge Race in order to accelerate the development of autonomous unmanned ground vehicles to carry out dangerous operations in urban environments. Perceptek has developed an autonomous ground vehicle based on a commercially available sport utility vehicle, commercial off the shelf sensors and custom autonomous navigation software. The following describes Perceptek's team, UGV hardware design and autonomous navigation software design.

### Team Contact Information:

Mark Rosenblum  
Urbanator Team Leader  
5705 S. Danube Circle  
Aurora, CO 80015  
[mark.rosenblum@perceptek-robotics.com](mailto:mark.rosenblum@perceptek-robotics.com)

Prepared for:  
**DARPA Urban Challenge 2007**  
THE DEFENSE ADVANCED  
RESEARCH PROJECTS  
AGENCY  
3701 North Fairfax Drive,  
Arlington, VA 22203-1714

*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 Team Description

PercepTek's Team Urbanator is made up of individuals that are passionate about robotics. The members of our team have significant robotic experience that included the DARPA Grand Challenge 2 [1] and many other government sponsored robotics programs. The team skill set is well balanced and includes experts in robotic hardware, electronics, navigation systems, robot perception, and robot architectures.

## 2 Vehicle Description

### 2.1 Base Platform

The base platform for this effort is a white 2007 5.3L V8 Chevy Tahoe named Rocky as shown in Figure 1. An SUV platform was preferred over other vehicle types due to the extensive space to install equipment in a protected and air conditioned environment. The height of the SUV was also appealing because it allowed for mounting the perception sensors from a high view point which allowed for a greater grazing angle with the roadway. Our system architecture does not require an SUV platform, but it minimized issues related to packaging, power and thermal conditioning for this effort. Our system can easily be adapted to any standard vehicle platform.

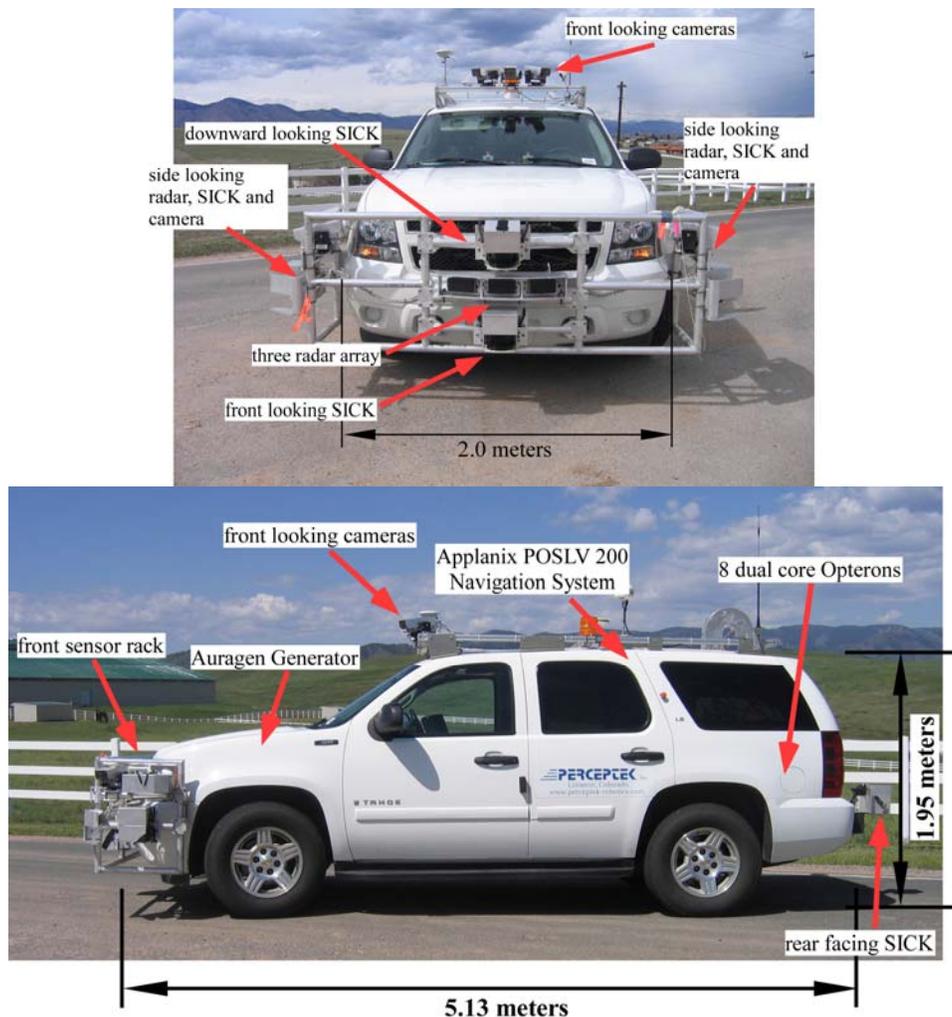


Figure 1: Team Urbanator 2007 Chevy Tahoe Platform called Rocky.

## 2.2 Actuation

The Chevy Tahoe was modified to provide actuation of steering, throttle, brake, parking brake, transmission and turn signals. Rather than design our own actuation package, our team chose to purchase the AEVIT® X-wire Primary RPV Control System. This system consists of the actuation and control for steering, throttle, brake, transmission, and ignition. This same control system is being used on many other DARPA Grand Challenge vehicles. The decision to purchase an actuation system rather than design a custom solution was based on cost, schedule and reliability issues. Our integration of the AEVIT® X-wire system allows for easy operational mode change between autonomous and manual control of the vehicle.



Figure 2: Primary AEVIT X-Wire System.

The AEVIT® X-Wire RPV Control System shown in Figure 2 consists of 5 primary subcomponents that include the main CPU, the DC servomotors for moving the brake, throttle, transmission and parking brake, the Primary and Secondary Displays and the EC Panel which acts as a driver interface through the AEVIT system. This special RPV version of the AEVIT® technology was developed specifically for autonomous vehicle applications in rugged "off-road" conditions.

## 2.3 Power System

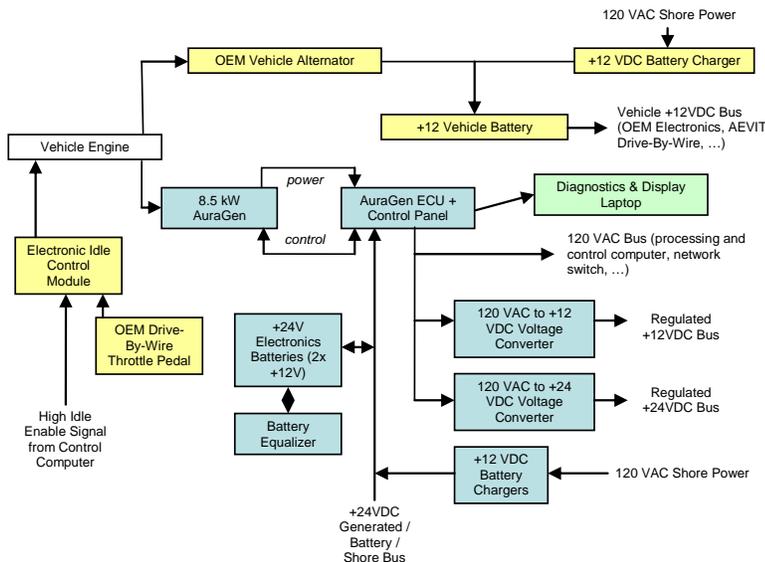


Figure 3: Rocky's vehicle and electronics power system.

Rocky's power system is composed of two distinct systems: the vehicle power system and the electronics power system. Figure 3 provides a block diagram of the power systems.

The vehicle power system provides power to OEM vehicle components, as well as the AEVIT® drive-by-wire system. This system primarily consists of the original OEM alternator and vehicle battery. In addition, a battery charger has been added to this system to permit operations while using shore power (120 VAC) for highbay development

and testing of the vehicle. It should be noted that the AEVIT® system includes an isolated battery backup to facilitate safe operations in the event of a failure of the vehicle power system.

The electronics power system provides power to all components added to the stock vehicle, other than the AEVIT® drive-by-wire system. These loads include all of the computing and sensing hardware added to the vehicle. As this power system is separate and distinct from the vehicle power system, the vehicle is easily returned to a configuration of a stock vehicle, augmented by the AEVIT® system, simply by disabling the electronics power system. The electronics power system is composed of a number of components, including an AuraSystems G8500XM 8.5kW AuraGen ICS system, an electronic idle control module, a set of 12V batteries, a 24VDC to 12VDC battery equalizer and 120VAC to 24VDC shore power source.

The AuraGen system consists of an under-hood generator driven by the vehicle engine, an electronics control unit (ECU), and a control panel. The Auragen ECU converts the unconditioned AC power from the generator into 24VDC which is used for battery charging and driving an internal 24VDC to 120VAC inverter. The 24VDC load/charging bus is bidirectional in that when a disruption of generator power occurs (e.g., UGV engine failure) 120VAC and 24VDC load power is not lost as the system will seamlessly fault over to the 24VDC battery array. Seamless transfer between generator (engine) and shore power operation is similarly handled by having the 24VDC shore power attached to the 24VDC load/charging bus.

The AuraGen generator must spin at a minimum rate to both provide power to the nominal system load to avoid discharge of the electronics batteries, as well as to provide cooling. With the nominal idle speed of the engine, and the ratio implemented in the pulley system for spinning the generator, the generator does not spin fast enough at idle to meet the minimum generator speed. Therefore, an electronic idle control module is utilized to increase engine idle speed. This module interfaces between the OEM drive-by-wire throttle pedal and the engine computer. With a software controlled enable signal, the module increases engine speed to meet the minimum speed required for the generator. Rocky's control computer enables this module when the vehicle is in park, and when traveling at low speeds.

For shore power operations, with the engine off, two battery shore power chargers provide up to 40A each at +24VDC, from 120 VAC shore power. In the shore power configuration, the chargers provide sufficient power to charge the electronics batteries, as well as provide sufficient power to maintain operation of the 24VDC and 120VAC Navigation Power System loads and charge the back up batteries.

To provide +12 VDC and +24 VDC power buses, two AC to DC power converters are utilized. The use of power converters, as opposed to tapping directly from the electronics batteries, provides filtered, well-regulated DC supplies to system components, and additionally provides graceful degradation of functionality when the electronics batteries are not otherwise sufficiently charged.

## **2.4 Safety System**

Our safety system has two modes of operation: manned and unmanned. In the manned mode of operation, a safety driver sits in the driver's seat and overtakes the autonomous system when necessary. In the unmanned mode of operation, a remote emergency stop system has been installed that when activated will cause maximum deceleration of the vehicle and kill the engine. In this section we discuss each of these safety modes of operation.

### 2.4.1 Safety for Manned Mode Operations

Most of our testing takes place in the manned mode of operation. In this mode of operation a trained human safety operator is sitting in the driver's seat of the robotic vehicle. The safety operator can either manually control the vehicle through the normal vehicle interface (steering wheel, brake, and throttle) or put it in autonomous mode with the flip of a switch.



**Figure 4: Manned Safety Overtake Mechanisms**

control degree of freedom. In other words, fight the actuation for the steering, brake and throttle. In all degrees of control, the low-level controller will attempt to overcome the resistance on achieving its goal and compensate by applying more power to the impeded or opposing actuator. In the case of overriding the brake or throttle actuator, the controller will respond with more throttle or brake, respectively. Therefore, this approach is considered a “first response” to a hazardous situation and must be followed by additional steps in order to disable the actuation system.

Unlike the first method of overtaking the autonomy, the second mechanism does not require any additional disabling to regain control of the vehicle. This second method of overtaking the autonomy involves disabling the autonomy through various interfaces in the vehicle. This disabling can occur through the toggling of either of two toggle switches mounted on the steering wheel for “finger tip” control or through the depression of one of two red mushroom buttons mounted on the dash of the vehicle. One mushroom button is mounted near the driver and the second is mounted near the passenger seat. Once either of these disabling pathways is activated, control of the vehicle actuators is returned to the safety driver who must safe the vehicle (i.e., correct trajectory and velocity to meet current situational need).

### 2.4.2 Safety for Unmanned Operations

The main responsibility of the safety operator is to ensure the robotic platform causes no harm to humans, property or the vehicle itself. This is our team's preferred mode of operation especially during integration and testing of new autonomous functions where unexpected results may occur. Our team has used this safety approach successfully across many ground robotics programs with no safety issues. The safety operator has two means to regain control of the vehicle while in autonomous operation. Figure 4 shows the driver access to the various mechanisms for overtaking computer control in autonomous operation. The simplest and most natural mechanism for overtaking the vehicle is to overpower the offending



**Figure 5: Omnitech-SR remote emergency stop system integrated into the Rocky safety system.**

The unmanned mode of operation is only used after extensive testing of the robotic system and it has proven to be reliable. This mode of operation was used for the filming of the video for the DARPA Urban Challenge video submission and is intended to be used for the NQE and the Urban Challenge race event. Our team also runs in this mode on a weekly basis with a safety operator in the driver's seat to validate that the remote emergency system is functioning properly and to keep the remote safety operators trained. Our remote emergency stop system is the Omnitech Two-way Safety Radio for Unmanned Ground Vehicle Operations (Omnitech-SR), shown in Figure 5, which is the same emergency stop system used for the previous Grand Challenge races. The Omnitech-SR consists of the Safety Receiver, which is integrated with the unmanned vehicle, and the Transmitter which is operated by a human safety monitor in a remote position relative to the robotic platform. The Omnitech-SR has two modes of remote intervention: Pause/Run and Enable/Disable. On the activation of a Pause, the robotic platform comes to a controlled stop and the actuation system is disabled until a Run command is issued. When a Disable is issued, the vehicle comes to an abrupt stop, the parking brake is engaged, the power is cut to the actuation system and the vehicle engine is killed.

## **2.5 Environmental**

All of the vehicle electronics are mounted in the rear cargo area of the Chevy Tahoe leaving the middle row of seats intact.

### **2.5.1 Shock Isolation**

The rack enclosures are supported on each corner using Lord Heavy Duty Platform shock isolation steel cup mounts. The selection of the mounts was based on analysis of the weight of the rack and components under urban driving conditions.

### **2.5.2 Thermal, Dust and Moisture**

One of the reasons our team chose an SUV as our base platform was that all environmentally sensitive hardware could be mounted in the vehicle, thus significantly reducing issues with dust, moisture and thermal conditioning. Through analysis, we determined that the stock air conditioner on the 2007 Chevy Tahoe could handle the thermal load of all the electronic hardware that was to be installed in the vehicle. Some modifications were made to the ducting in order to vent cool air directly into the enclosures containing the electronics.

## **2.6 Processing Architecture**

### **2.6.1 High Level**

The high level processing cluster is composed of 8 dual core 2.2 GHz Opteron 848HE processors in a single server enclosure. The system uses a Tyran Thunder S4882 Quad Opteron motherboard with daughter card. Four of the processors are on the main board and four are on the daughter card. All processors share 16 GB of RAM. The system also has two 500GB SATA hard drives for storage. This configuration and processor set was selected for both processing horsepower, low power usage, ease of use and operation under a standard Linux distribution. The high level processing cluster is built around the SUSE 10.1 Linux distribution running a 64-bit 2.6 kernel.

All processing in the cluster was designed and implemented using C and C++ in Linux. The processes responsible for sensor and navigation processing as well as the high level path planning are dynamically and automatically allocated across the processing cluster by the Linux symmetric multiprocessing configuration. The inter-process communication is handled using the Neutral Message Language (NML) developed at the National Institute for Standards and Technology. This approach provides a common memory-mapped interface both within each and across all of the processors in the cluster. The high level processing architecture is separate from the low-level control processing and the interface between the two is over a Gigabit ethernet network using a DLink Gigabit switch.

### **2.6.2 Low Level**

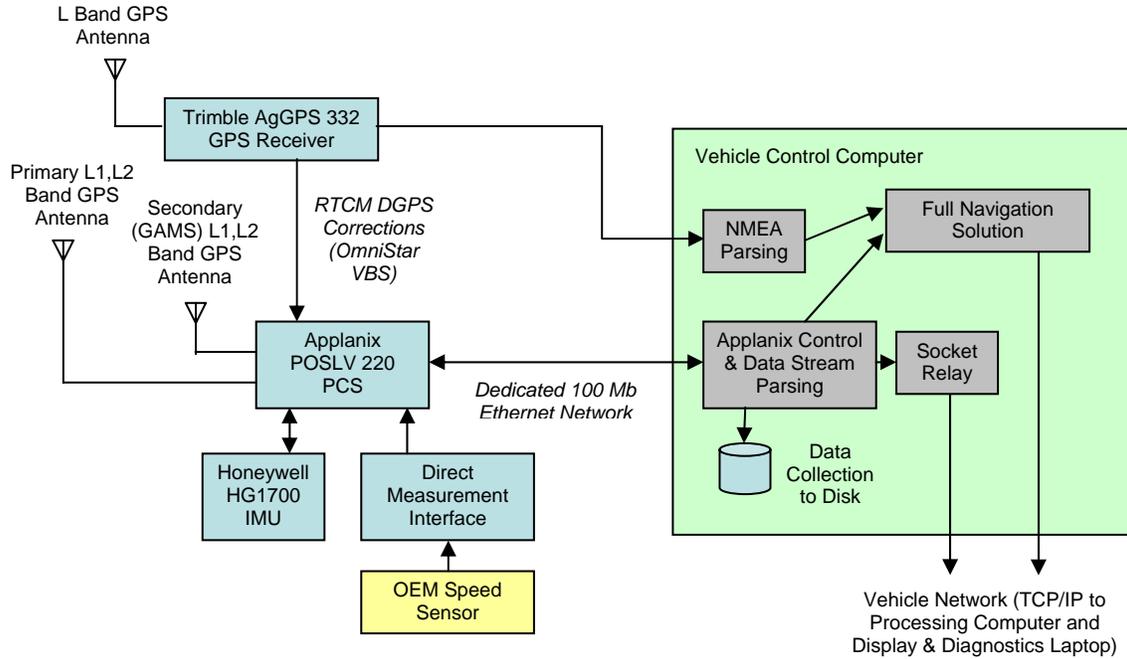
The low-level processing is the bridge between the high level behaviors and the vehicle hardware and is also responsible for managing the vehicle safety system. When the high-level architecture desires a specific steering command, it sends a command packet over the ethernet to the low-level controller for execution. The low-level controller has analog outputs to control the steering and brake/throttle actuators. The low level controller uses an RS232 serial bus to command the transmission and accessories (e.g., turn signals, head lights, etc.). The low-level processor is a VersaLogic Cobra embedded EBX board with a 1.6 GHz Pentium-M processor, 512Mbytes of RAM and an 80Gbyte hard drive for data storage. The processor operating system on the low-level processor is SUSE 10.1 Linux in a 32 bit configuration.

## **2.7 Localization**

Navigational position and orientation sensing is provided by an Applanix POS LV 220, a commercial off-the-shelf tightly coupled inertial and GPS navigation system. The Applanix system consists of several L1, L1/L2 and differential GPS receivers, an inertial measurement unit (IMU), and an interface to the OEM Speed sensor for odometry purposes. Figure 6 illustrates the different components of the POS LV system and the flow of POS data within our navigation system.

With the use of OmniStar VBS DGPS service, Table 1, below, provides accuracy specifications of the POS LV 220 system both during nominal operations, and through successively longer GPS outage durations.

The vehicle control computer is responsible for interfacing with vehicle navigation sensing, and providing the navigation solution to other consumers in the system. To accomplish this task,



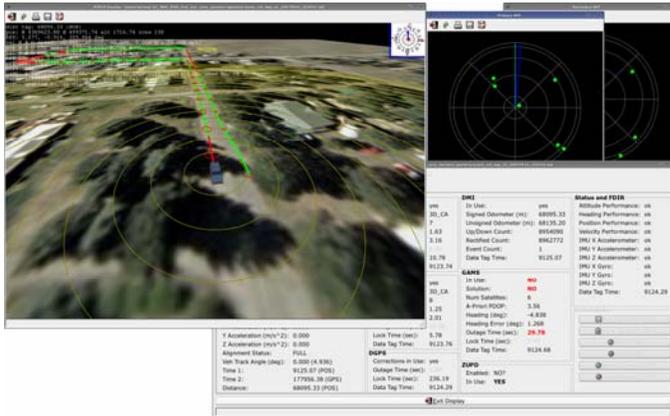
**Figure 6: Rocky's Navigation System.**

several software components are utilized. A software module is responsible for interfacing with the Applanix via TCP/IP sockets on a dedicated 100 Mbit Ethernet network. This software module is responsible for issuing control and configuration messages to the POSLV, and more importantly, parsing the navigation data stream from the POS LV and providing formatted data to other consumers in the system. The software module also provides a data collection capability that is always running, and can be saved off to disk when desired in order to capture significant events.

	0 sec GPS outage	15 sec GPS outage	30 sec GPS outage	60 sec GPS outage	120 sec GPS outage
X, Y Position RMS Accuracy (m)	1.0	1.13	1.25	1.5	1.75
Z Vertical Position RMS Accuracy (m)	1.5	1.63	1.75	2.0	2.2
Roll and Pitch RMS Accuracy (deg)	0.07	0.07	0.07	0.07	0.07
Heading RMS Accuracy (deg)	0.07	0.07	0.07	0.07	0.08

**Table 1: POS LV 220 Accuracies during nominal periods, and through GPS outages**

A dedicated software application has been developed for the Rocky vehicle for display of navigation data, as well as to provide diagnostic, analytic and calibration functions. This software application executes on a developer's computer, either from a live data stream provided by the socket relay function of the Applanix interface software module, or from stored data collections. This application provides a real-time display of navigation data, including the

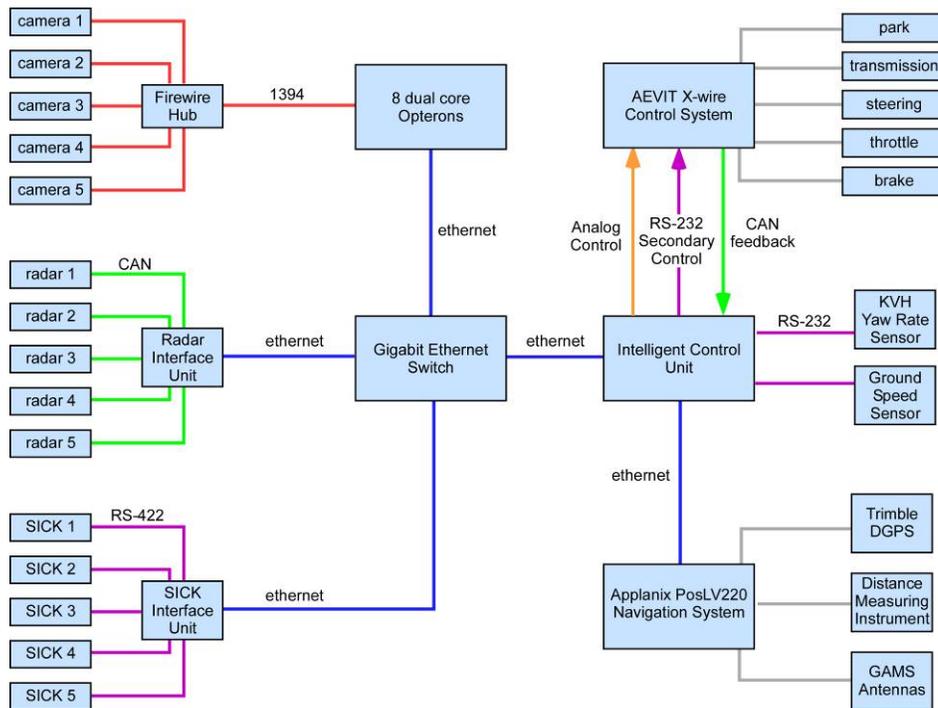


**Figure 7: Navigation display, diagnostics, analysis, and calibration application**

essential navigation solution, and other ancillary data, such as status of the POS LV and GPS satellites tracked. This application also provides analytic functions, such as GPS outage times, maximum error and so forth. Finally, this application also provides calibration functionality, such as correction of IMU mounting angles, and visualization of calibration parameters. Below, Figure 7 shows a sample screenshot of execution of this application.

## 2.8 Hardware Configuration

Figure 8 shows the interconnectivity of all of the hardware components on the Rocky vehicle except for the power system.

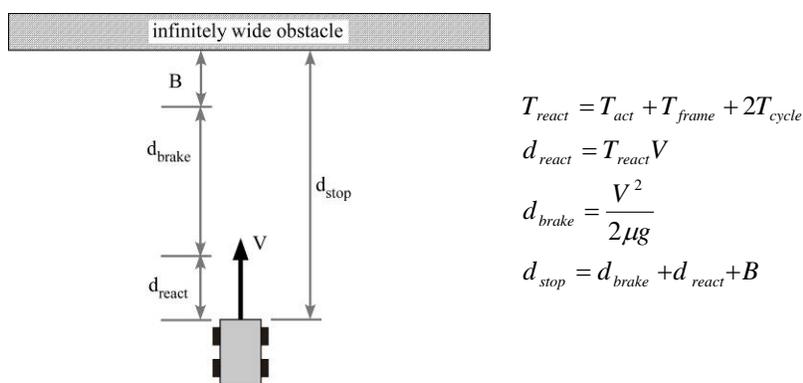


**Figure 8: Hardware Configuration**

### 3 Sensing

#### 3.1 Analysis Approach

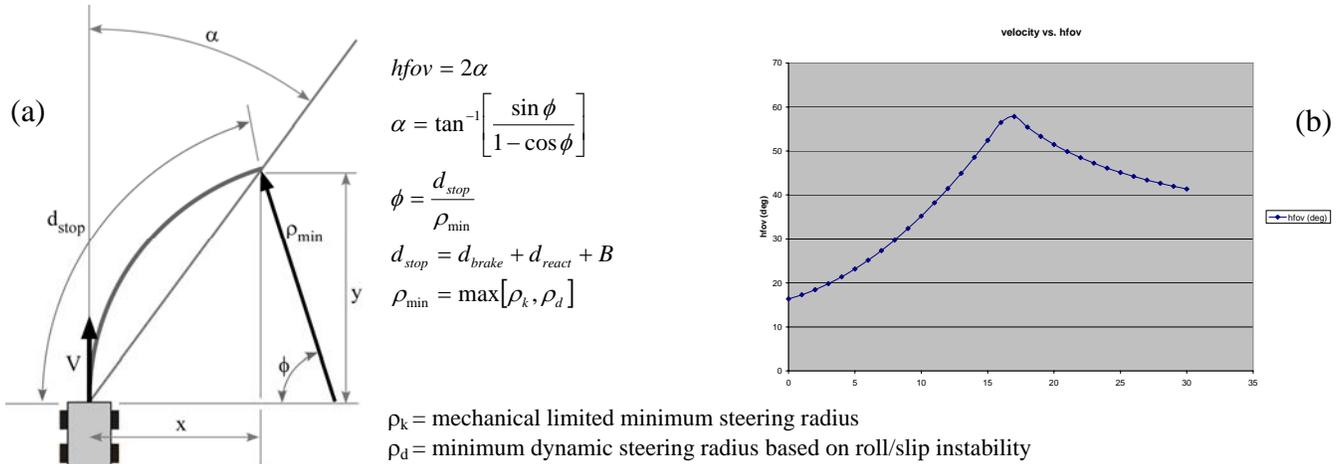
In order to generate the requirements for the sensor system, we analyzed the sensing requirements for each of the individual low-level robotic behaviors. The analysis was performed for maximum and minimum sensor range and horizontal and vertical fields-of-view for each sensor type. Much of the sensor analysis is based on techniques outlined in a paper by Alonzo Kelly of Carnegie-Mellon University [2]. In order to determine the requirements of a particular sensor, we used the most stringent requirement for each type of analysis across the set of behaviors which use the sensor. We used five forms of analysis to define requirements for the urban environment. Some of the analyses build on each other and some of the analyses are very specific to a particular environmental situation. Figures 9-13 show the five types of analyses used to define sensing requirements.



**Figure 9: Pure Stopping Maneuver:  $T_{react}$  is the reaction time based on the actuation time ( $T_{act}$ ), frame time ( $T_{frame}$ ) and cycle times ( $T_{cycle}$ ).  $B$  is a safety buffer.**

Figure 9 shows the Pure Stopping Maneuver analysis. This analysis is used to determine required sensing range and is based on the premise that a sensor must be able to see out further than the minimum distance it takes the vehicle to stop at full braking which is considered the most elementary means of collision avoidance. This idea becomes clearer if the inverse logic is considered where the sensor cannot see beyond the stopping distance of the vehicle. If a hazard is detected at the fringe of the detection range and the vehicle responds with full braking, it will collide with the hazard, where if the detection was made further than the stopping distance, the vehicle would stop before contact was made. This analysis uses standard kinematics equations to determine the stopping distance of the vehicle. For the pure stopping maneuver a worst case analysis was performed using the DARPA specified maximum mission speed of 30mph. Other term values were 0.1 seconds for actuation time, 0.1 seconds for algorithm cycle time, 0.03 seconds for data acquisition time or the frame rate of a standard camera, a coefficient of friction of 0.8 and a safety buffer zone of 2 meters. We also assumed a 0% grade of the terrain. This analysis was used to determining the range of our obstacle avoidance trajectory planning system and our road and lane tracking algorithms.

Figure 10a shows the Horizontal Field-of-View analysis. The idea here is that the sensor coverage must cover the full area in which the vehicle can travel in a Pure Stopping Maneuver, i.e., the vehicle is full braking while at the same time, steering away from the hazard. With the



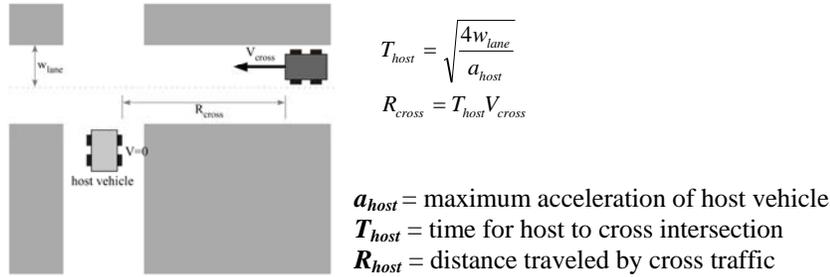
**Figure 10: Horizontal Field-of-View Analysis: (a) The sensor system must cover the full envelope where the vehicle could drive to a distance of the full braking stopping distance; (b) horizontal field-of-view versus vehicle speed.**

added steering maneuver the analysis gets more complex because at higher speeds while turning, two forms of instability can be induced: 1) the vehicle rolling or 2) the vehicle sliding sideways. This analysis takes into account the roll/slip instability and restricts the horizontal field-of-view of the sensor to deal only within the stability limits of the vehicle. Figure 10b shows a plot of the horizontal field-of-view requirement versus speed. At the lower speed ranges as the speed increases, the horizontal field-of-vehicle requirement also increases. However, above 17 mph, the vehicle cannot make hard turns due to the roll/slip instability, so the horizontal field-of-view requirement begins to drop at these higher speeds. For our requirements, we used the most stringent requirement or the peak of the field-of-view curve in Figure 10b.

Assumptions	Value
Host Vehicle Is Stopped	N/A
Perpendicular Intersections	N/A
Cross Traffic Speed	30 mph
Time Margin	2 seconds
Intersection Width	2 lane width wide (~7meters)
Acceleration of Host	0.25g's
Intersection is Orthogonal	

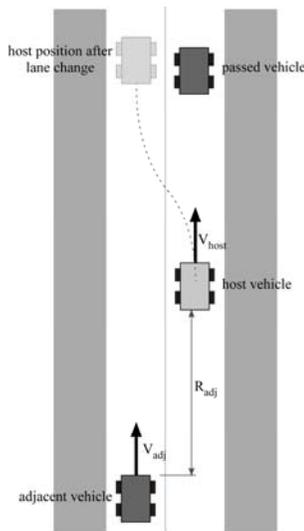
**Table 2: Intersection Analysis Assumptions**

Where the previous two analyses were based on emergency maneuvers of the vehicle to avoid a collision, the intersection analysis in Figure 11 is based on being able to see all vehicles in the crossing lanes of traffic that could interfere with a safe crossing of the intersection. The analysis assumes that the host vehicle is stopped before the intersection and the cross traffic is moving at the maximum allowable speed of 30mph. Other assumptions for this analysis are listed in Table 2. The first step in determining if it is safe for the host vehicle to cross the intersection is to determine how long it will take the host vehicle to cross the intersection with a margin of safety of 2 seconds. Our assumption is that the intersection is two lane widths wide and the host vehicle starts from a stopped position and has an acceleration of 0.25 g's. The intersection crossing time is multiplied by the speed of the cross traffic to get the distance traveled by the cross traffic in the time it takes for the host to pass through the intersection. This distance then



**Figure 11: Intersection Analysis: host vehicle must be across intersection from a stopped position before the crossing vehicle arrives at the intersection. Therefore, the host vehicle must see out further than the cross traffic can travel in the time it takes the host to cross the intersection with some safety margin.**

becomes the detection range for the sensor used for detecting cross traffic. A conservative analysis for the required field of view would assume that the host needs to see infinitely far to both the right and left of the intersection leading to a field-of-view requirement of 180 degrees for an orthogonal intersection.



**Figure 12: Merge/Lane Change Analysis: The detection range for sensing vehicles in the adjacent lane must be greater than the stopping distance of the vehicle in the adjacent lane.**

Two other forms of analysis: Merge/Lane Change and Hard Turn Analyses shown in Figures 12 and 13, respectively. These, like the intersection analysis are very specific to the urban environment. The results of all five analyses with respect to the relevant behaviors are shown in Table 3.

The resulting sensor configuration for our Urban Grand Challenge vehicle is shown in Figure 14 where each sensor meets the sensing requirements across the set of relevant behaviors to which it is assigned. The forward looking cameras will be used for lane tracking and making a left turn. The front looking radars as well as the front looking SICK laser will be used for forward obstacle detection. The front looking radars will also be used for safe gap maintenance between the host vehicle and any vehicle in-front of the host vehicle. The downward looking SICK will be used for the hard right turn around a corner with a curb. The side looking SICKs will be used for lane changing and merging. The side looking radars and front looking SICK will be used to determine intersection clearance for crossing.

The backup SICK will be used for detecting hazards behind the vehicle when backing out of a parking spot or the reverse portions of a three point turn. The summary of sensor ranges and fields of view are shown on Table 3 along with the behavior the sensor supports and the analysis types used to drive its geometry requirements.

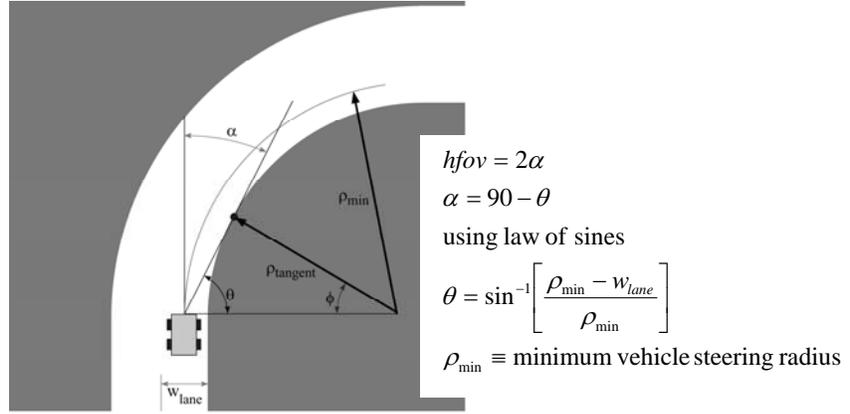


Figure 13: 90° right turn analysis

behavior	Sensor Type	hfov (deg)	vfov (deg)	maximum range (m)	analysis type
Lane Tracking	camera	58	15	17	hfov analysis, stop distance analysis
Paved/Unpaved Roads	camera	58	15	17	hfov analysis, stop distance analysis
Safe Gap Maintenance	radar, camera	58	15	17	hfov analysis, stop distance analysis
Lane Change	side looking laser	180	NA	17	merge/lane change analysis
Merge	side looking laser	180	NA	17	merge/lane change analysis
Intersection Crossing	side looking radars, front looking laser	15 (radars), 180 laser, 100 camera	NA	120 (radar), 15 (laser)	intersection analysis
90 degree Right Turn at Curb	downward looking laser	90	NA	5m	curb analysis
Left Turn Across Traffic	side looking radars, front looking laser	15 (radars), 180 laser	NA	120 (radar), 15 (laser)	intersection analysis
Obstacle Avoidance	laser, radar	58	NA	17	hfov analysis, stop distance analysis
Pullin Parking Spot	laser	180	NA	3m	hfov analysis, stop distance analysis
Pullout Parking Spot	backup aid radar	180	NA	5m	hfov analysis, stop distance analysis
U-turn (5 mph)	laser	23	NA	3m	hfov analysis, stop distance analysis
Intersection Queing	laser	23	NA	3m	hfov analysis, stop distance analysis

Table 3: Limiting Sensor Configuration

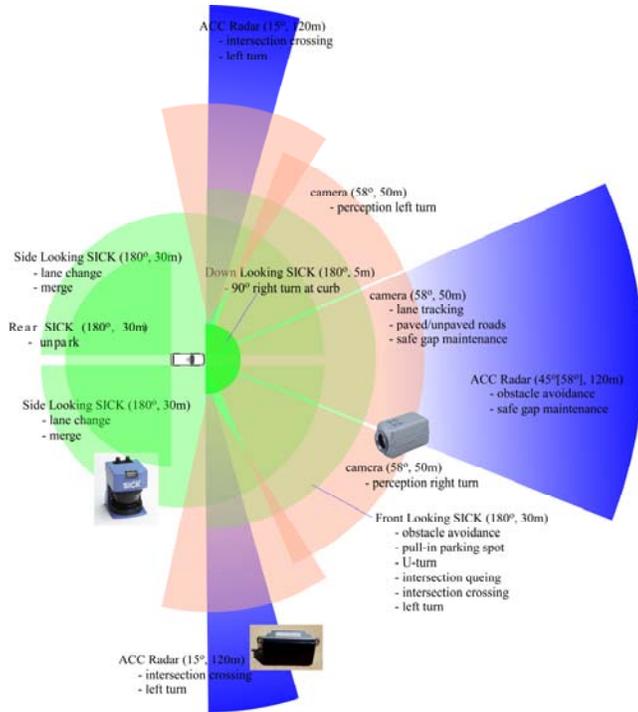
### 3.2 Sensor Specifics

#### SICK LMS Line Scan LIDAR

A combination of SICK LMS line scan lidar units like the one shown in Figure 15 are used for close range terrain sensing on the Rocky vehicle. The LMS 290-S14 (90 deg at 0.5 deg/sample with 75Hz scan rate) mounted on the front sensor rack looking downward is used for curb detection and determining drivable surfaces. A LMS 211-30106 (180 deg at 1 deg/sample with 75Hz line scan rate) mounted on the front sensor rack looking forward is used for obstacle detection. The same SICK is mounted on both sides of the vehicle for lane change maneuvers.



Figure 15: SICK Laser Scanner



**Figure 14: Sensor Configuration based on sensor analysis.**

Delphi ACC3 Radar

The Rocky vehicle is equipped with five production Delphi Forewarn® Smart Cruise Control radars (ACC-3) like the one shown in Figure 16. These radars were designed for adaptive cruise control and automated safety controls in passenger automobiles. Smart Cruise Control radars use a mechanically-scanning, 76 GHz FMCW, long range radar sensor to detect objects in the vehicle’s path up to 500 feet (152 meters) ahead. The ACC radars have a 15 degree field of view. For the Rocky vehicle, three ACC radars are configured across the front of the vehicle in a manner to provide a 45 degree field of view. This configuration is critical in order to detect potential obstacles in the opposing lane before attempting a lane change maneuver. An ACC radar is mounted on each side of the vehicle angled to detect cross traffic out to greater than 100 meters at an intersection. Data from the five radars is transmitted via a CAN interface to a Radar Interface Microcontroller which converts the CAN signal to Ethernet where the data is then read by the main navigation computing cluster and is made available for behavior analysis.



**Figure 16: Delphi ACC3 Radar**

Sony FireWire Camera

All of the cameras on Rocky are Sony DFW500VL color firewire cameras. The cameras have auto-exposure, auto-gain and auto-iris. This improves the cameras ability in diverse lighting conditions. The camera produces imagery in multiple formats and frame sizes. For this application all of the cameras are set to provide 320x240 color imagery at frame rate.

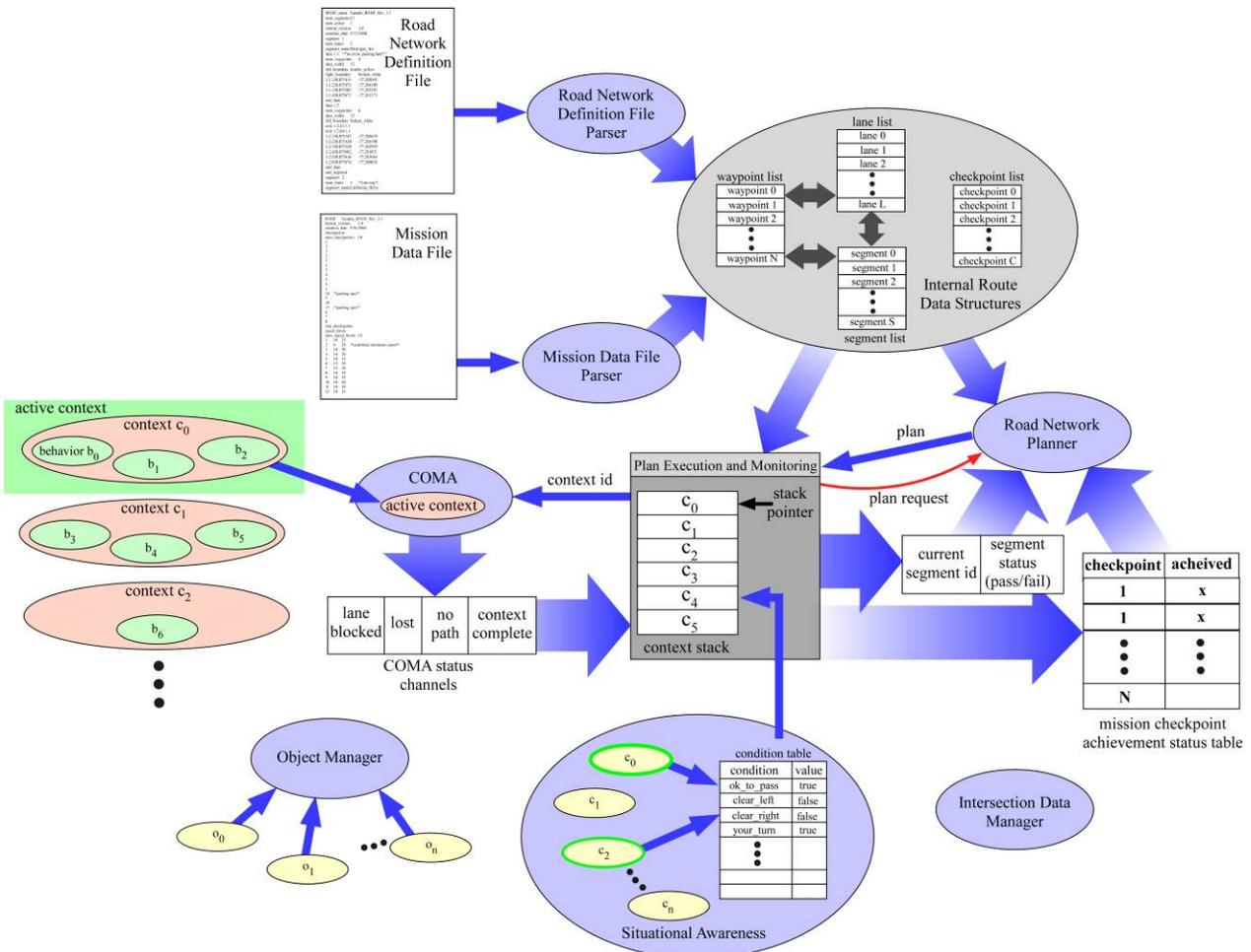


Figure 17: Team Urbanator Urban Challenge Software Architecture.

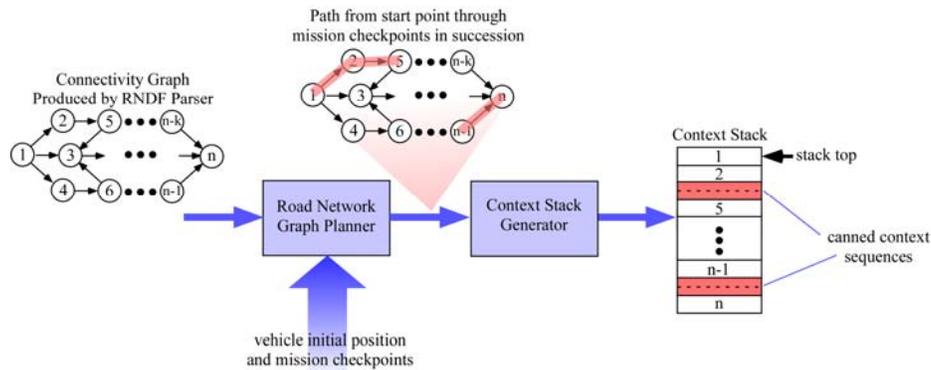
## 4 Control/Software Architecture

### 4.1 Architecture Components

The software architecture is shown in Figure 17. The key responsibilities of the software architecture are: 1) reading the desired mission file, 2) planning a route to achieve all of the mission checkpoints, 3) combining behaviors and sequencing these combinations in an optimal way to achieve the mission, 4) providing data access for all components of the system, and 5) coordinating an appropriate response to dynamic changes in the environment. Our Urban Grand Challenge software architecture grew out of the architectural approach we used for our Grand Challenge 2 system, which made extensive use of environmental contexts. In our system, an environmental context encapsulates a particular environmental scenario to which we assign sensors, algorithms and parameter settings which are best suited for the context. In a Grand Challenge 2 mission, we would dynamically switch between contexts, thus activating and deactivating algorithms and sensor streams to best handle the current context. In studying the Urban Grand Challenge problem, we determined that the use of environmental contexts could be extended in order to achieve the complex missions characterized by the DARPA Urban Challenge.

The main components of the architecture, as shown in Figure 17, are the robotic behaviors that are part of behavior sets associated with specific environmental contexts. A single behavior can be part of multiple environmental contexts and multiple instances of a particular behavior type may exist with each tied to a different sensor input. For instance, we may have an obstacle avoidance algorithm coupled with a radar, and a different instance of the same obstacle avoidance algorithm taking in data from a laser. The outputs from the current active set of behaviors are presented to an arbiter called COMA (Contextual Operating Mode Arbiter) that combines the outputs of the set of behaviors into a single resultant control response for speed and steering. The current environmental context is maintained by the context stack manager. The stack internal to the stack manager is a set of ordered contexts that were pre-planned based on the desired set of mission checkpoints. The stack data structure is used so that the architecture can deal with dynamic environmental events by pushing new contexts onto the top of the stack. For instance, if a slow moving vehicle is encountered in the host vehicle’s lane of traffic a “pass-vehicle” context is pushed to the top of the stack. After the completion of the pass, the pass-vehicle context is popped off and the old context is resumed.

In order to deal with blocked routes and replanning, the context stack manager updates a globally accessible set of buffers. The first buffer contains the current segment identification number and status of progress on that segment. On a blocked route where a replan operation must be invoked, the planner will know what segment caused the failure and will not incorporate that segment into the new plan. The second buffer maintains a status of the mission checkpoints and whether they have been achieved or not. The planner uses this information to prevent the generation of a new plan that attempts to achieve checkpoints that have already been achieved through the execution of the previous plan. In the following sections we describe our planning approach.



**Figure 18: The High Level Planning approach consists of two stages. The first stage produces the optimal path through the directed graph and the second stage produces a context stack augmented canned behavioral sequences.**

#### 4.1.1 Planning

Our planning system shown in Figure 18 consists of two stages. The first stage called the Road Network Planner uses a constraint based planner to generate a path from a start node to a goal node through the directed graph. The second stage of the planning called the Context Stack

Generator traverses the path through the graph producing the context stack that will be sent onto the context stack manager for execution. In the process of constructing the context stack, the second stage planner inserts specific context sequences that have been formulated to handle very specific environmental situations such as stopping for a stop sign. In the following sections we describe each stage of the planner in more detail.

### 4.1.2 Road Network Graph Planner

A directed connected graph is constructed by the Road Network Data File Parser (RNDF Parser) from the Road Network Data File provided by DARPA. The nodes in the graph correspond to waypoints in the RNDF and the directed edges between nodes are established by applying a simple set of rules based on keywords and format of the RNDF. For example, a sequence of waypoints making up a lane in the RNDF are sequentially connected with directed edges in the order they are presented in the RNDF. Keywords such as “exit” are also used to establish links between nodes. Zones are incorporated into the connectivity graph with each parking spot being represented by two nodes that correspond to the two waypoints that make up a parking spot. The parser also stores attributes of waypoints specified from the RNDF such as whether a waypoint corresponds to a stop sign, checkpoint, and/or parking spot. In addition, the parser collects or infers attributes on edges based on whether the edge is an exit or entry into or out of a parking spot. The exit edges can further be refined into whether the exit represents a left turn, right turn or straight connection between map entities. A connected graph, based on a small portion of the sample RNDF provided by DARPA, is shown in Figure 19 along with node and edge attributes. The stage one planner uses the directed edge connectivity to generate a path from a start point to a goal point based on the A\* constraint based planner. In generating our optimal plan, we will be able to use any combination of a distance, time or route complexity constraints. Since it will be

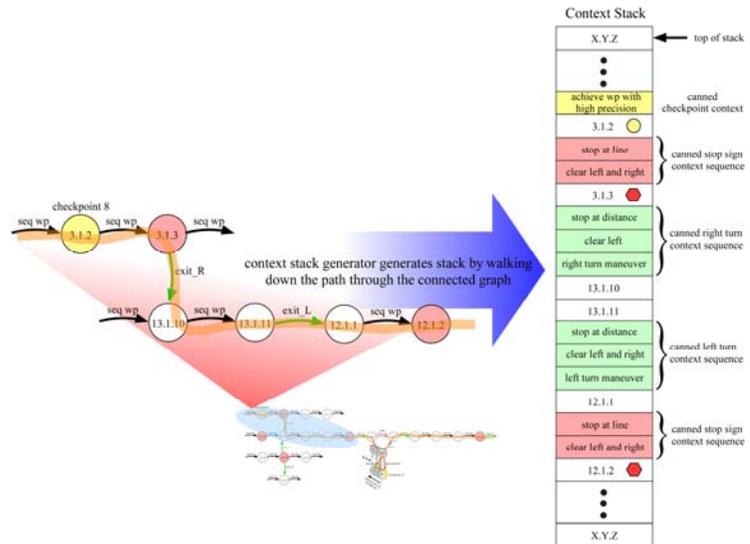


**Figure 19: Connectivity Graph produced from the sample RNDF**

necessary to achieve the set of checkpoints provided in the Mission Data File in the correct order, the Road Network Graph Planner produces sub-plans from the start location of the host vehicle to the first checkpoint, and between subsequent checkpoints. The final plan provided to the Context Stack Generator is a concatenation of the set of sub-plans.

### 4.1.3 Context Stack Generator

The purpose of the Context Stack Generator is to convert the path through the directed graph produced by the stage one planner into a “robotic” plan that is of a form that can handle dynamic environmental events, and has been augmented with specialized context sequences designed to handle upcoming environmental situations known to occur along the route from the attributes obtained from the RNDF such as a stop sign or intersection. Figure 20 shows the augmentation of the context stack for a small portion of the path from the directed graph in Figure 19. The augmented context stack contains canned context sequences designed to handle specific environmental situations such as stop signs, left turns, right turns and achieving checkpoints. The individual environmental contexts that make up the context stack are in a form that can be executed by the rest of the architecture. As each context is completed, which is indicated by one or more of the behaviors running in the context, the context gets popped off the context stack and the subsequent context becomes the current active context.



**Figure 20: Generation of the context stack from the path through the connected graph representing the sample RNDF.**

### 4.1.4 Context Stack Manager

The high level management of the plan execution is handled by the Context Stack Manager. The Context Stack Manager has several responsibilities: 1) control the sequencing of contexts in the stack, 2) handling dynamic environmental events, and 3) forcing a replan if the current executing route is blocked. The sequencing of contexts in the stack is handled in the following way. The current active context that gets executed by the rest of the architecture is the context at the top of the stack. On the completion of a context, which is indicated by the Context Complete event from one of the behaviors in the current context, the current context is popped off the stack and the next context on the stack becomes the active context. The ability to handle dynamic environmental events is dealt with in two ways: 1) the context itself can handle the event such as avoiding a hazard, or 2) a specific dynamic event is detected requiring a specialized event resolution context sequence to preempt the current active context and handle the situation. The preemption of the current active context is achieved by pushing the event resolution context sequence to the top of the context stack. On completion of the resolution sequence, the system

reverts back to the prior context before the dynamic event occurred. An example of one of these specialized events is the requirement to pass a slow moving vehicle in the host vehicle lane. A replan occurs when a specific behavior determines that it has no path. This could be an obstacle avoidance algorithm seeing all candidate steering directions blocked by hazards. The offending behavior signals the COMA arbiter that the path is blocked. COMA then sets the No Path buffer to true which is in turn seen by the Context Stack Manager causing it to generate a replan signal to the planner.

#### **4.1.5 Context-Based Arbitration**

Our software architecture design is based on the premise that not all sensors and algorithms are optimal for all environmental situations. For instance, an autonomous vision-based road-follower will not work well in an off-road situation where there is no road. It is even possible that if the algorithm is allowed to influence vehicle control in this situation, it can generate the incorrect response for the situation leading to navigation failure. Our software architecture design philosophy is to be able to dynamically and optimally reconfigure the software architecture based on the current environmental situation. At the heart of our context based approach is COMA (Contextual Operating Mode Arbiter). COMA is responsible for monitoring the current environment context and dynamically reconfiguring the behavior sets based on changes in context. The association of sensors to behaviors and behaviors to contexts is maintained in configuration files and can be easily modified.

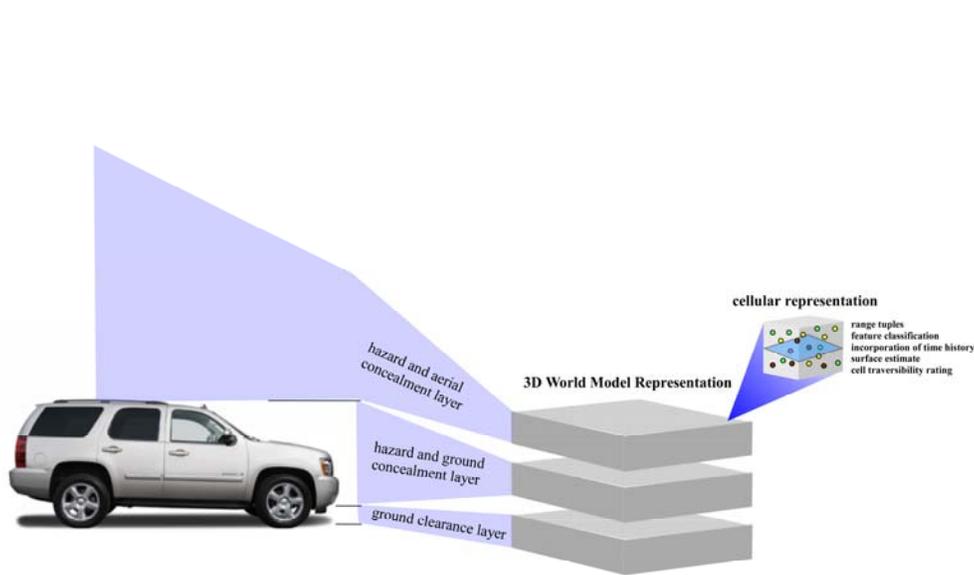
At the lowest level of COMA is the reactive layer of behaviors. Behaviors are categorized into positive and negative behaviors. A positive behavior is one that has a specific direction it desires to travel. A road-follower is an example of a positive behavior because it is attempting to steer the vehicle to the center of a road. A negative behavior is one that blocks candidate steering directions based on the hazard level of those candidates. The obstacle avoider is an example of a negative behavior. Based on the importance of the behaviors relative to the other behaviors in the context and the instantaneous confidences from the active behaviors in the context, COMA combines the steering and speed responses from these behaviors into a resultant speed and steering response that gets passed on to the controller for execution. If a behavior within the current context determines that there is no path, COMA performs several mitigation steps before passing the No\_Path event up to the planner for a replan. COMA was the basis of our architecture for the Grand Challenge 2 and we exploited the context-based philosophy extensively in our Urban Grand Challenge architecture.

### **4.2 Behaviors**

The role of a behavior in a reactive robot control architecture is to generate a control response on a very specific input type. If it is required that the system respond to additional input types, then additional behaviors are required in the system. Therefore, it is typical that multiple behaviors are executing concurrently in a reactive system. In this section we describe the behaviors used in the architecture.

#### **4.2.1 Waypoint Following**

The waypoint following behavior steers the vehicle along a sequence of waypoints attempting minimize the vehicle's normal distance from the segments connecting the adjacent waypoints. The only sensing modality used by the waypoint follower is the navigation system.



**Figure 21: The Layers of the obstacle avoidance map correspond to relevant vehicle dimensions and capabilities.**

#### 4.2.2 Geo-Maneuvers

These are navigation only behaviors and they include “Right-Turn”, Left-Turn”, Geo-Stop, and U/K turn. For these precanned trajectories, the algorithm servos around the navigational position of the vehicle relative to the trajectory generating the appropriate steering command to minimize offset from the planned path.

#### 4.2.3 Obstacle Detection and Avoidance

The function of this behavior is to detect and avoid hazards that occur along the desired route of the vehicle. The algorithm uses a 2.5 dimensional map as shown in Figure 21 to represent the environment around the vehicle. The map can be populated by any combination of radar, laser or camera data. The map is vehicle centered which means as the vehicle moves, the objects in the map are translated relative to the vehicle. The planning component of the map can function as either a positive or negative behavior. In order to function as a positive behavior, the algorithm requires a desired route in the form of waypoints. The plan generation is based on an A\* constraint based trajectory planner that plans around hazards in the map and produces a trajectory that achieves the goal point from the desired route. If the behavior is functioning as a negative behavior, it costs a set of candidate steering directions represented as steering radii. As a negative behavior, the behavior determines where the system should NOT drive. The algorithm can dynamically switch between a positive and negative behavior if there is no desired route to follow as a positive behavior.

#### 4.2.4 Safe Gap Maintenance

This behavior maintains a safe gap distance between the host and a lead vehicle in the predicted path of the host by controlling the host vehicle’s speed and acceleration. It detects if there is a lead vehicle in front of the host vehicle in the predicted path of the host using the front facing 45 degree radar array mounted on the front of the vehicle. The radar provides range and angle to a set of potential targets. The algorithm determines which targets are relevant to its predicted path and how to respond to the target if it is relevant. Safe gap maintenance is considered a positive speed behavior in the COMA arbiter.

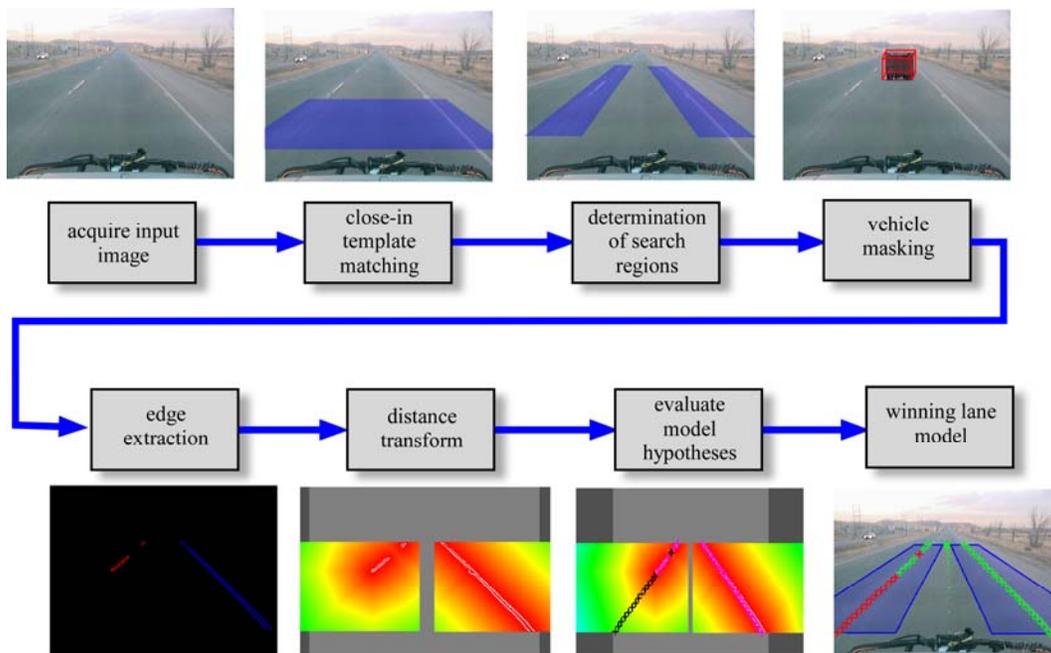
### 4.2.5 Unstructured Road-Following

The purpose of this algorithm is to keep the vehicle a minimum distance from a road edge for unmarked roads (no lines). This algorithm will detect contrast based road edges such as a boundary between a paved road and gravel shoulder in a color camera image. The algorithm will make use of a set of visual and geometry cues to produce the best contrast edge. The set of cues used cycle to cycle can change dynamically to optimize the contrast edge. Once the contrast edge is determined, it is fit with a cubic spline and from this a drive-to point along the curve is computed for a lookahead distance that is based on vehicle speed. In the Urbanator architecture, this algorithm acts as a negative behavior essentially pushing the vehicle away from the boundaries of the center of the driving lane. Figure 22 shows the output from the output from this algorithm. The left image shows the estimated road edges and the right image shows the classified road as a green overlay in the scene used to generate the road edges.



**Figure 22: The left image shows the original color image with the road boundary shown as connected yellow line segments. The right image shows the classification of the road as green.**

### 4.2.6 Structured Lane-Tracking



**Figure 23: Algorithmic stages for our structured lane-tracking algorithm.**

The purpose of this algorithm is to keep the vehicle laterally centered in a marked lane through straight-aways and curves. The algorithm uses the imagery from a front-mounted camera, and extracts the road lines from the scene. The algorithm then estimates the upcoming road

curvature and generates a steering command to maintain the vehicle's lateral position in the lane. This algorithm will accurately steer the vehicle at speeds up to 70mph and at curvatures up to  $0.02 \text{ m}^{-1}$  or 50 m radius turns but at slower speeds. Figure 23 shows the algorithmic stages of our structured lane-tracking algorithm.

#### 4.2.7 Profile Following

The profile follower uses a downward facing laser scanner mounted on the front of the vehicle to determine the position of the geometric edge of the road and generates steering arcs to steer the vehicle away from it from the edge. This behavior is primarily used to push the vehicle away from curbs.



**Figure 24: Visual Stop line algorithm.**

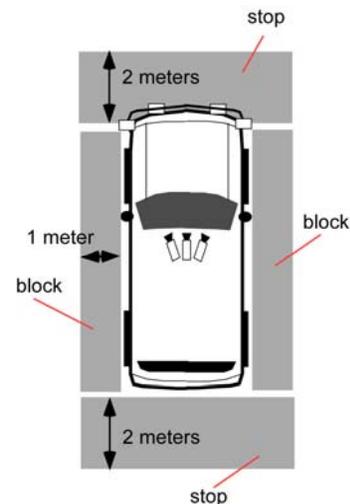
#### 4.2.8 Visual Stop Line

This behavior controls the speed of the vehicle so that it stops the vehicle so the vehicle's front bumper is directly over a physical stop line on the ground. The algorithm uses camera imagery and bounds its search for the stop line to a subregion based on the location of the stop waypoint from the RNDP. The stop line is detected using a profile over the summation of rows in the subregion. The stop line location is determined by finding a peak in the profile over the rows and making sure the peak meets minimum thresholding requirements. Figure 24 shows the output of our visual stop line

algorithm. The bounded search region is indicated by the transparent blue trapezoid and the scan line profile is shown on the left side of the search region. In this case the peak of the scan line profile directly coincides with the stop line on the ground. The purple horizontal line indicates the algorithms estimate of the stop line in the scene.

#### 4.2.9 Virtual Bumpers

Our architecture has multiple built-in layers of protection against collisions. At the highest level is the obstacle avoidance algorithm. At the lowest level are the virtual bumpers. The virtual bumpers provide a protective boundary around the vehicle as shown in Figure 25, and act as a last layer of defense for an impending collision. For front and rear breaches of the virtual bumper boundary, the vehicle is immediately stopped. This capability is only active below host vehicle speeds of 10mph. In the side zones, the virtual bumper algorithm blocks the vehicle from steering in the direction of the zone breach. The side zones are active at all speeds.



**Figure 25: Virtual Bumper Zones**

### 4.3 Condition State Evaluators

In order for the control architecture to make logical decisions based on the state of the urban environment, a set of condition evaluators have been implemented that provide the truth values for a wide range of conditions. For instance, if the reasoning system is attempting to cross an

intersection, it needs to know if the intersection is clear before proceeding. Unless the intersection is clear, the host vehicle's progress will be blocked. A typical set of condition evaluators used to determine intersection clearance are CLEAR\_LEFT, CLEAR\_RIGHT and CLEAR\_FRONT as shown in Figure 26. If all of these are satisfied, the vehicle is allowed to cross the intersection. In this section we describe the condition evaluators in our system.

### 4.3.1 Intersection Left/Right Clear

These condition evaluators make use of the data coming from the side looking radars and cameras. The radars in addition to providing target range and angle also provide a range rate for each potential target. The cameras are used with image differencing in localized portions of the image. By combining the radar information with the localized image differencing an accurate assessment of left and right clearance can be performed.

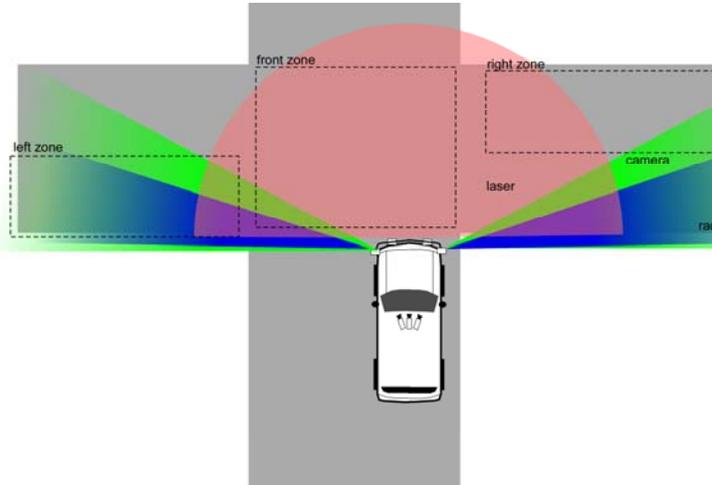


Figure 26: Intersection Clearance Zones.

### 4.3.2 Intersection Front Clear

The front clear determines if the center portion of the intersection is clear. This condition evaluator makes use of the front facing laser scanner and knowledge of the intersection geometry that is posted by the mission planner to determine if this region of the intersection is clear.

### 4.3.3 Intersection Precedence

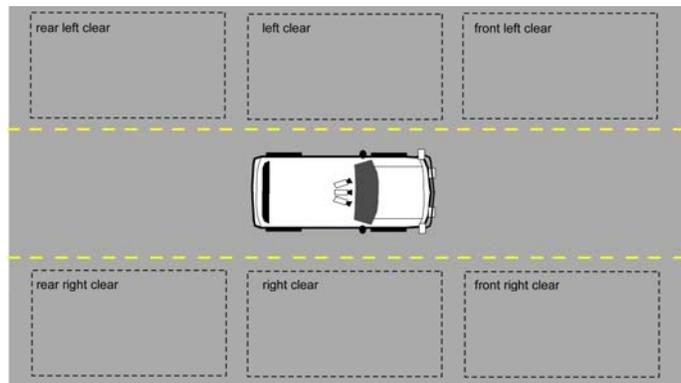
This condition evaluator determines when it is the host vehicle's turn to cross an intersection, if there is other traffic at the intersection. This algorithm looks at both the laser and camera data for the different zones of the intersection. The search zones for this algorithm are dynamically determined when the vehicle arrives at the intersection based on geometry deduced from the RNDF file and the current vehicle position relative to the intersection. In order to separate vehicle detections from clutter, templates on the laser data and refinement is done using image processing on the projected laser hits in the camera image. Figure 27 shows the situation where the host arrives at an intersection where another vehicle already was situated and waiting to cross the intersection. The upper panel shows the other vehicle in the imagery from the left looking camera and a host vehicle centered map showing laser data relative to the host vehicle. The detection in the map is indicated by red cells with a red bounding box. The search zones are shown as blue rectangles. The host vehicle's progress is blocked and this is indicated by the solid red square in the lower left of the upper panel. After the other vehicle clears the intersection, the algorithm determines it is the host's turn to cross and this is indicated by the solid green square in the lower left of the lower panel.



**Figure 27:** The top panel show the scenes from three separate cameras with laser data projected into the scene with a two dimensional map showing the laser data relative to the host vehicle. In the top row, since a vehicle has been detected in the left zone at the time the host arrived at the intersection, the algorithm determines that it is NOT the hosts turn to cross. After the other vehicle crosses the intersection, the algorithm determines it is the host’s turn to cross the intersection which is shown in the lower panel.

#### 4.3.4 Passing Condition evaluators

These condition evaluators determine if the passing zones around the host vehicle are clear for a passing or lane change maneuver. These passing zones are shown in Figure 28. The appropriate set of passing zones considered in the decision to pass is based on the direction of the pass maneuver. For instance a pass maneuver to the left lane requires clearance in the REAR\_LEFT, LEFT and FRONT\_LEFT zones. The clearance of zones is determined using a combination of laser and radar data.



**Figure 28:** Passing Zones for the Passing Condition Evaluators.

### 5 System Testing

Our team uses three forms of testing that includes a cluttered urban environment, a sparse urban environment and a well controlled regression test.

#### 5.1 Cluttered Urban Environment

Our cluttered urban environment testing takes place in a small town called Louviers Colorado shown in Figure 29 with the RNDF overlay. Louviers is particularly difficult for robot navigation due to the heavy tree cover that causes prolonged GPS outages and stark shadowing on the roadways which provide challenges for vision-based perception algorithms. In addition, Louviers has narrow roads with lots of road side clutter which challenge all of the sensing modalities. It is our belief that by overcoming the difficulties associated with Louviers, our



**Figure 29: Louviers Colorado cluttered urban environment testsite with RNDF overlay.**



**Figure 30: Keenesburg Colorado sparse urban environment testsite with RNDF overlay.**

system will be more reliable and robust in all urban environments. We frequent this test site on a daily basis.

## 5.2 Sparse Urban Environment

Our second urban test site is Keenesburg, Colorado shown in Figure 30 with the RNDF overlay. Keenesburg is a much more benign environment with much less tree cover making both the navigation and perception problems easier. This testing occurs on a weekly basis. The Keenesburg testsite is also the location of our DARPA site visit.

## 5.3 Regression Test

The regression test is based on the ISO-3888-1 standard with modifications specific to robotic testing [3]. This ISO test was originally designed to evaluate human drivers. We have setup this course in a large flat parking lot. The purpose of this regression test is to make sure that new modifications to the system have not negatively impacted the low-level controllability of the robotic platform. This test focuses mostly on low-level control, navigation and simple perception. The course is intended to be run at progressively higher speeds starting at 5 mph up to 30 mph. For each run a detailed account of performance is maintained.

## References

- [1] Klarquist, William, "Intelligent Vehicle Safety Technologies I Technical Description.", DARPA Grand Challenge II Technical Paper, August 2005.
- [2] Kelly, Alonzo, "A Partial Analysis of the High Speed Autonomous Navigation." ,CMU Robotics Institute Technical Report, CMU-RI-TR-94-16, May, 1994.
- [3] Koon, Phillip, "Evaluation of Autonomous Unmanned Ground Vehicle Skills.", CMU Robotics Institute Technical Report, CMU-RI-TR-06-13, March 2006.