DARPA Urban Challenge

What Makes the Armadillo Go?

Team Cybernet
Charles Jacobus, PhD, Team Leader
Steve Rowe, Principal Architect
Douglas Haanpaa, Principal Vision Systems Architect
Prof. CJ Chung, Student Project Coordinator
Gary Siebert, Hardware Engineer
Ben Michael, Chris Wagner, PhD, OCU Development
Kevin Tang, Test Engineer
Pavan Namineni, Location Sensors Engineer
Gary Moody, Glenn Beach, Sensors and Vision Consultants
Charles Cohen, PhD, Robotics Consultant
Joe Tesar, Project Coordinator
Kirk Fifer, Technician & Driver

Draft per 3/29/2007 as first submitted
Final updates per 5/31/07

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.*

Executive Summary

The Team Cybernet approach to the Urban Challenge is completely driven by requirements as set forth in the rules and requirements defined by DARPA and published at http://www.darpa.mil/grandchallenge/rules.asp and our status as "Track B" team.

Our goal is to meet the requirements at the lowest cost using the simplest architecture possible in the hardware, while providing software functionality that meets navigation, safety, and control requirements. Our approach uses a test vehicle, which is approximately the same age as most of the US Army's present vehicle fleet, makes maximum use of low cost computer vision, and integrates automotive grade navigation elements with precision GPS (as opposed to high cost state-of-the-art military grade inertial measurement componentry).

The goal is to prove out a man-safe autonomous vehicle appliqué kit that could be deployed in HMMWVs and FMTVs for less than $10,000 per vehicle.

Introduction and Overview

This paper describes the analysis, design, results, and performance of the Team Cybernet Armadillo vehicle as of May 31, 2007. As this is work in progress, and additional important effort will be expended through NQE, we expect to get smarter as we go. This is especially true pertaining to basic and advanced driving behaviors that we have not yet fully implemented and tested by the date due for this draft technical paper.

The primary problems for us as first time challengers has been to:

(1) Get a vehicle reliably automated so that we can assume that it is basically a "sloppy and big pen plotter," i.e. when higher level command and control calls for the vehicle to execute a speed, position, orientation maneuver, it does it reliably enough that the plan does not fail or fails in a determinable manner.
(2) Provide GPS fault tolerant self-location to sufficient precision so that point-driving behavior is stable and accurate.

Figure 1. The Armadillo test vehicle – standard 1996 Chrysler Town & Country minivan previously used for vehicle microwave soft stopping tests[1] -- also shows forward ladar, camera and inertial measurement unit

[1] ***Non-cooperative vehicle stopping***, Department of Homeland Security, NBCHC060019. We used tuned and directed modulated microwave energy on this vehicle to insert spurious vehicle control computer commands into the target vehicle through its buses to jam or "crash" on-board vehicle controllers and stop the vehicle at power levels that do not cause permanent damage.

Figure 2. Steer and Brake Actuators – also shows placement of forward and side looking cameras

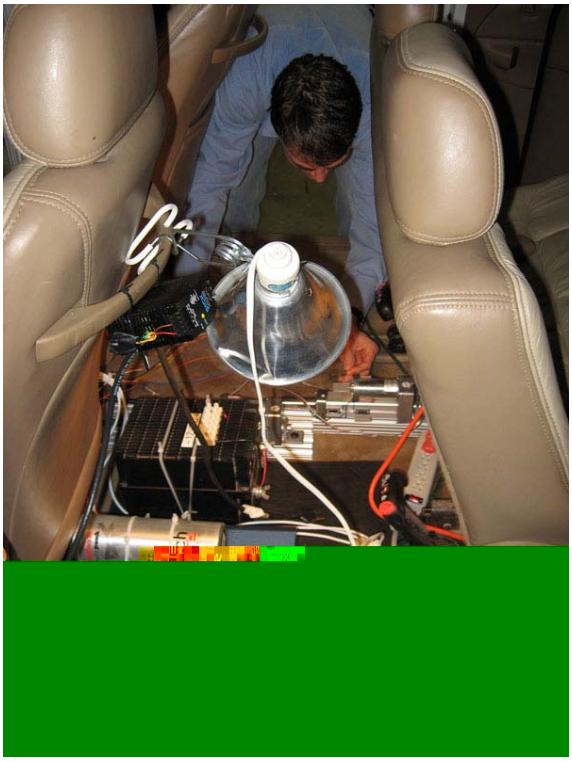

Figure 3. Discrete control electronics and Accelerate and Shift actuators – Blue box at the bottom is the primary 120VAC power inverter that supplies PCs, GPS, wireless, etc.

Figure 4. Dual controller PCs – also shows interior, side and back looking cameras

Then we have focused on proximate obstacle detection and avoidance, which presently depends heavily on a high update-rate laser radar (75 Hz, 180 degree FOV, approximately 3-4 cm range accuracy to ½ degree precision).

The ladar system is augmented with an 8 camera, 360 degree view, real-time color computer vision system (Figure 5). This system includes auto iris adjustment, fixed focus and zoom, real time compression to disk (so that all video data can be later used for presentation footage or computer vision algorithm testing), and real time processing for road features (the front facing driving camera), motion/proximity-based course ranging and obstacle speed detection (all cameras, periodically sampled based on higher level vehicle behavior requirements), and road motion/white line detection (camera sighted on the same FOV as the ladar).

The final component is the system higher-level control architecture. This provides safety monitoring, deadman/run/pause features, route planning, specific navigation/driving behaviors, direction of the lower level point drivers, and support functions (road map and mission read-in, logging, map edit and display, etc.).

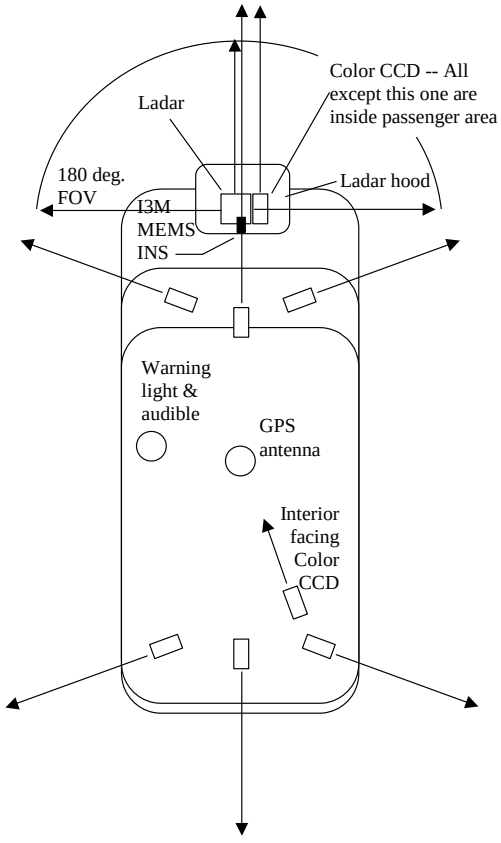

Figure 5. Primary sensors placement

By our standards the Armadillo is less AI oriented and more algorithm oriented – that is, each subsystem is built to perform a given function within acceptable and predictable performance bounds. What this means as far as computer vision, controls, etc. is fairly intuitive, but we will discuss further how it pertains to higher-level vehicle control as well in the sections focused on these functions.

We chose an Open JAUS architecture[2] implemented in JAVA as our backbone. One of the architects is a member of the JAUS working group and we presently have a program out of the US Army to implement a JAUS operator control unit and JAUS to JVMF gateway, which has provided on-going familiarity with this approach.[3]

Analysis and Design

In the JAUS architecture,[4] each object in the robot's software behavior is an object. Objects communicate to one another via JAUS messages. The JAUS specification provides a defined set of objects as shown in Figure 6. We implement our functionality within this framework to provide easy JAUS compliance at Level I (inter-subsystem control messages -orange) and Level II (inter-nodal software messages - yellow).

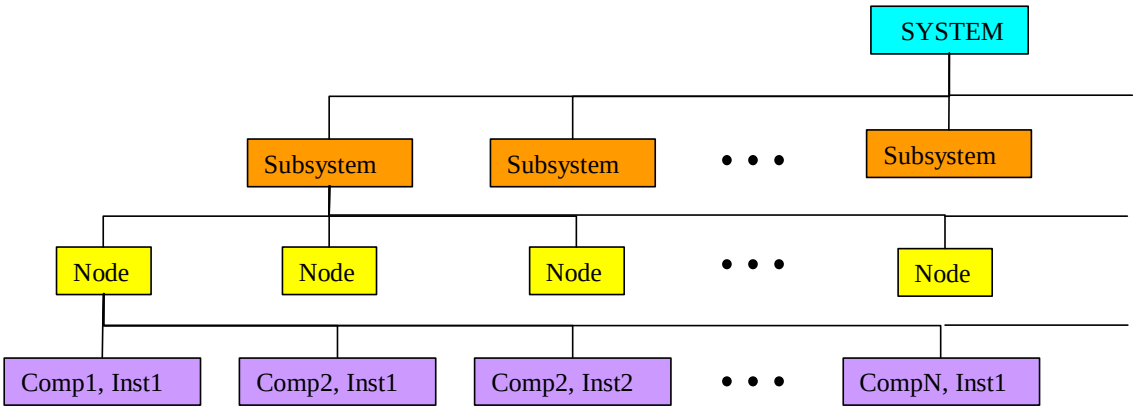

Figure 6. JAUS Architecture Topology

[2] http://www.openjaus.com/ derivative of the system used in a summer of 2006, University of Florida offered a class on unmanned systems, taught by the authors of the initial OpenJAUS source. These are Archit Baweja, Tom Galluzzo, Alex Gizis, Danny Kent, Brian Prodoehl, and Bob Touchton**.**

[3] Steve Rowe, Joe Tesar, ***Modular Agent-Based Component (Robotics) Architecture***, W15QKN-06-C-0076, US Army TACOM-ARDEC, COTR: Zbigniew Bogdanowicz, Picatinny Arsenal.

[4] ***The Joint Architecture for Unmanned Systems, Reference Architecture Specification***, V1-3, Version 3.2, August 13, 2004

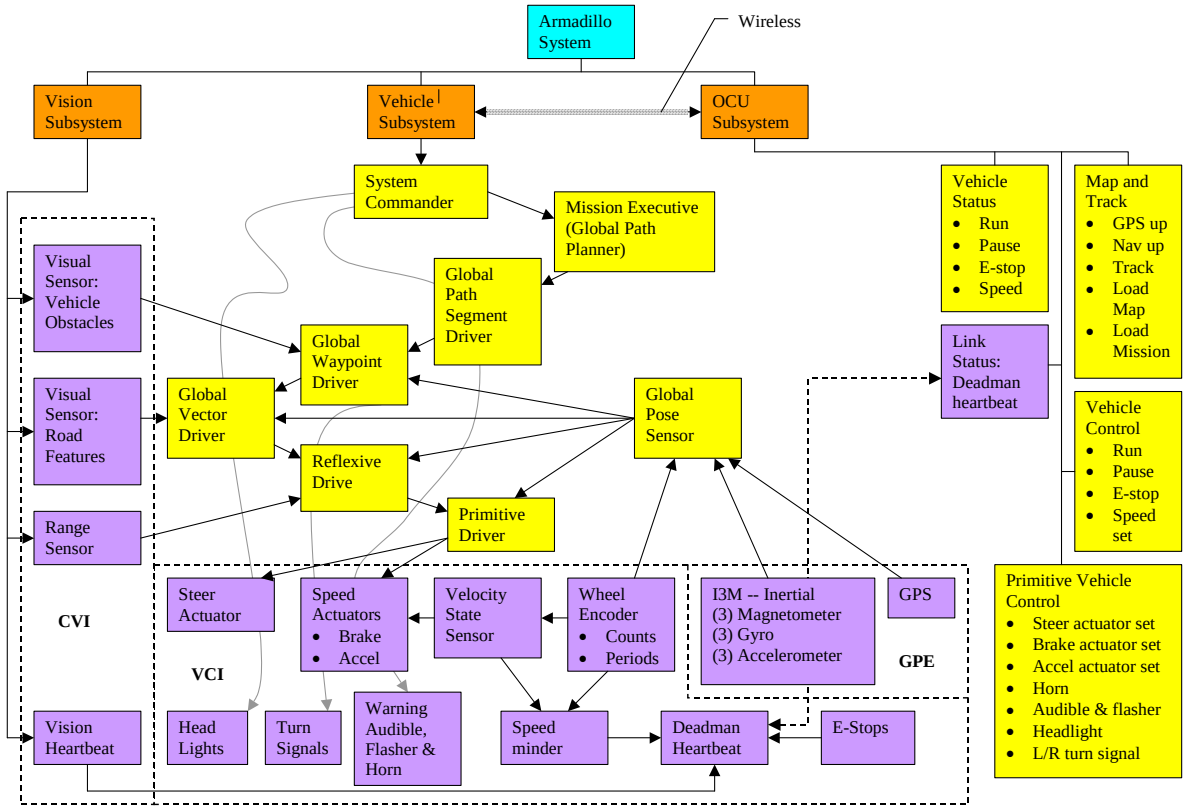

Figure 7. How JAUS maps to Armadillo

We decided that JAUS does not provide a good set of primitive objects for direct vehicle control, localization sensor fusion, range processing, or computer vision for road features, motion/proximity, or safety at the present time.[5] For this reason, the lowest rung of the architecture is implemented as software APIs: on the control side via the vehicle control interface (VCI), on the localization side via the Global Pose Estimator (GPE), and on the sensor side via the computer vision interface (CVI).

The VCI provides the low-level driver API that interfaces between servo actuators, speed sensors, steering servos, safety devices and the vehicle hardware interfaces. We implemented the vehicle hardware interface through a USB digital/analog/counter module, an encoder interface (for the steering wheel), and vehicle-specific electrical taps. Figure 8 shows the basic arrangement. Due to timing and safety considerations, a redundant set of hardware interfaces have been inserted for reliable wheel speed encoder reading.[6]

[5] All of these functions are areas of current JAUS standard improvement efforts by the working group.
[6] Implemented using the embedded PIC processor core of a Cybernet Medical MedStar data hub because it has the facilities needed for communications, signal reading, and timing: www.cybernetmedical.com.

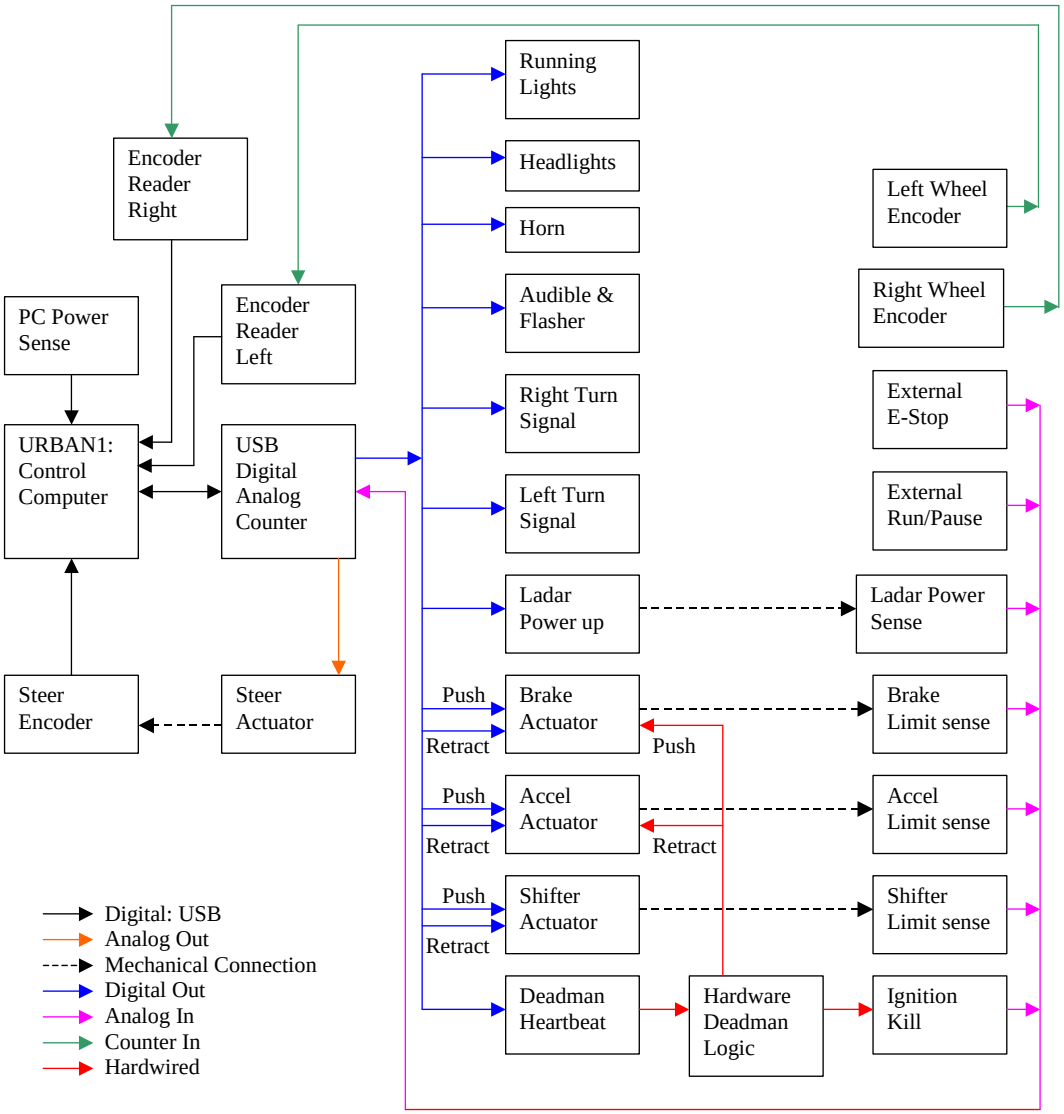

Figure 8. Interfaces through the USB digital/analog/counter module

We interface to the localization sensors via serial ports to the GPS and Cybernet I3M MEMS Inertial/Magnetometer module (Figure 9). We measure wheel motion (for travel distance when GPS is out and for dynamic wheel diameter calibration) by tapping into the wheel encoder system provided by the VCI.

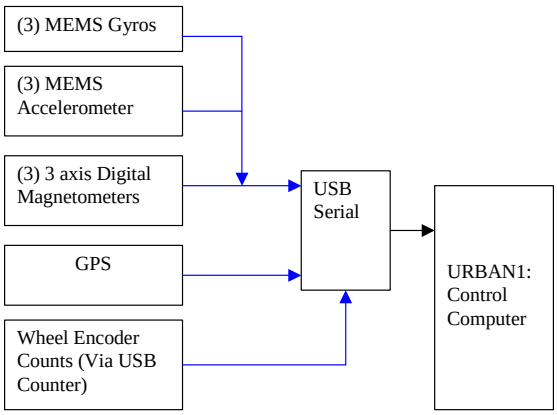

Figure 9. Localization sensors

The computer vision subsystem is handled by a separate computer core because of workload associated with capturing, compressing, storing (to disk), and processing eight real time video

channels. Each video channel is calibrated to a flat Earth perspective model, runs its own automatic iris routine, and is continuously captured, compressed, time stamped, and placed on disk store – the system can keep more than 12 hours of continuous video on the vehicle. Remarkably, the cost of this system is less than $2000, driven down by the proliferation of high-speed processors, low cost security cameras, and emerging MPEG/JPEG conversion chipsets.

The core system is a basic real time 8-channel digital video security system (Figure 10). What makes this one a bit unique is that we have APIs that allow any channel of video (compressed or uncompressed) to be selected for real time processing. *The following table shows the way each camera is processed presently in the system.*[7] The CVI is in the Armadillo control computer and provides an API interface that tasks the computer vision subsystem computer and its software suite (and thus provides the JAUS to CVI dedicated software interface). The laser radar is conceptually part of the CVI, but operates as a thread within the Armadillo control computer to keep interprocess inefficiency to a minimum.

Table 1. Camera Function Allocation

Camera	Function	Algorithm	Duty Cycle
1	Backward obstacle detection (Pointing out the rear window)	External Vehicle Motion Detection	When backing up
2	Side lane obstacle detection (pointing to vehicle port blind spot)	External Vehicle Motion Detection	When backing up When changing lane
3	Side lane obstacle detection (pointing to vehicle starboard blind spot)	External Vehicle Motion Detection	When backing up When changing lane
4	Cabin inside view	Video Capture	Continuous
5	Forward driving camera (points out of the windshield from vehicle center line)	Road Edge and Feature Finding/Tracking	Continuous
6	Front Side looking (pointed to the port side intersection area to find distant cross traffic)	External Vehicle Motion Detection	When changing lane or before entering intersection
7	Front Side looking (pointed to the starboard side intersection area to find distant cross traffic)	External Vehicle Motion Detection	When changing lane or before entering intersection
8	Forward Looking Ladar Co-mounted (Points where the Ladar Points from Bumper Level)	Road Motion & White Line Finding	Continuous

[7] As of this date, this processing is being still developed and does not run outside of the vision system developer's desktop PC. Transfer to the vehicle will be done after site visit deadlines are achieved.

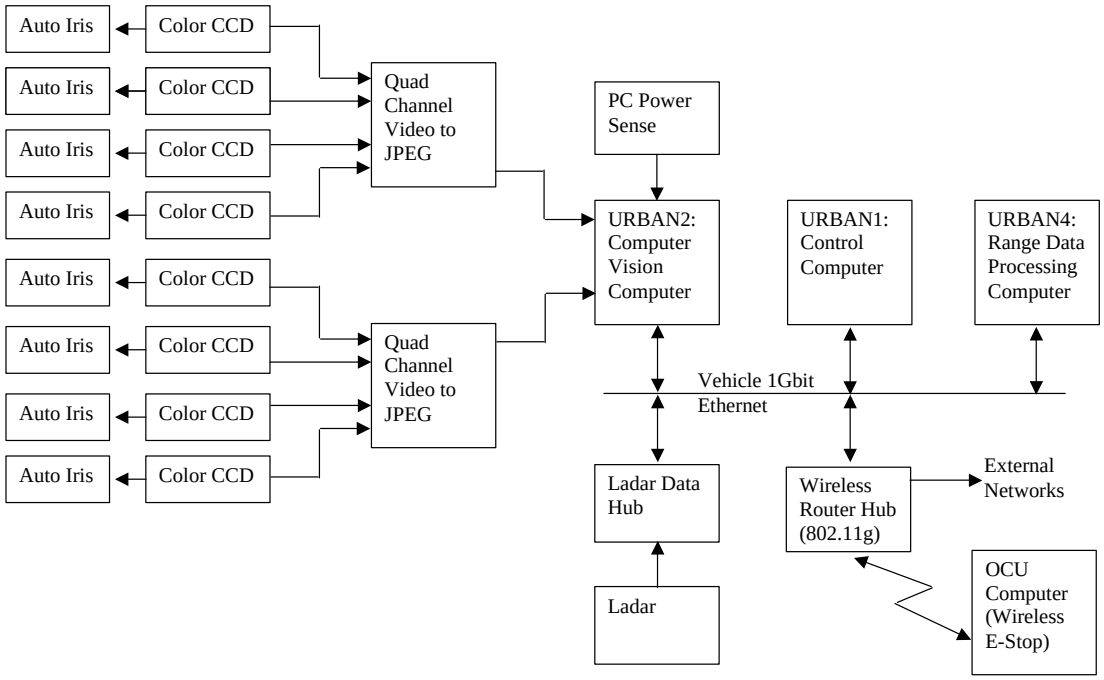

Figure 10. Computer Vision interconnect (and network between Computer Vision computer, Control computer, Range Processor, and external networks)

Computer vision algorithms are still under development, but the preliminary approach is to unwrap JPEG compressed 8x8's to acquire the upper diagonal coefficients in real time (JPEG hardware determines these coefficients as part of its real time image compression). From these we can identify 8x8 predominant color, edges, and changes frame-to-frame that support determination of optical flow. Optical flow images are converted via a flat earth model into pseudo range maps that can be merged with LADAR data for collision detection and avoidance of moving and still obstacles.

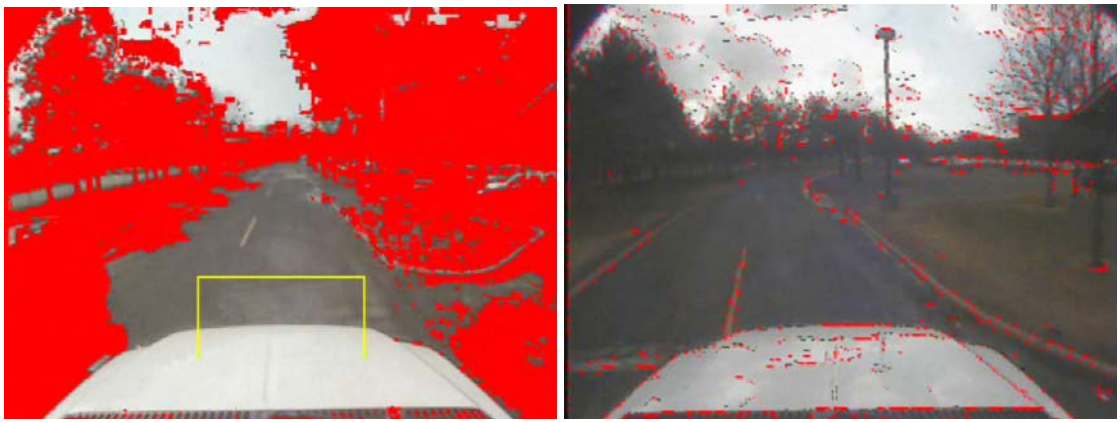

Figure 11. Two alternative road edge finding approaches; (left) the road is uniform color and non-uniform areas are color coded read; (right) the road is relatively smooth and the boundaries are not – thus image edges include road edges (high likelihood edges are color coded red).

Edge and color data provide two alternative means for detection of road edges (or conversely traversable areas out from the vehicle hood). Figure 11 shows two processed images showing where edge algorithms and color-based algorithms find edge or discontinuity information. This work is just getting started so more data and results will be coming later.

Core E-Stop, Pause and Run modes are controlled via (a) hardware fail safes and kill buttons, (b) deadman signals exchanged between the hardware kill system, each processor (the vision and main controller processor), and (c) the OCU connected to net via wireless link (if this link fails, or if the E-Stop behavior is commanded, the E-Stop behavior is triggered). E-Stop ties into the hardware kill system and triggers when any communications or inter-processor message fails, or if a hardware kill is initiated. The E-Stop circuits kill the ignition, retract the accelerator, and push the brake to full stop immediately. E-Stop state can only be overridden by manual access to the vehicle.

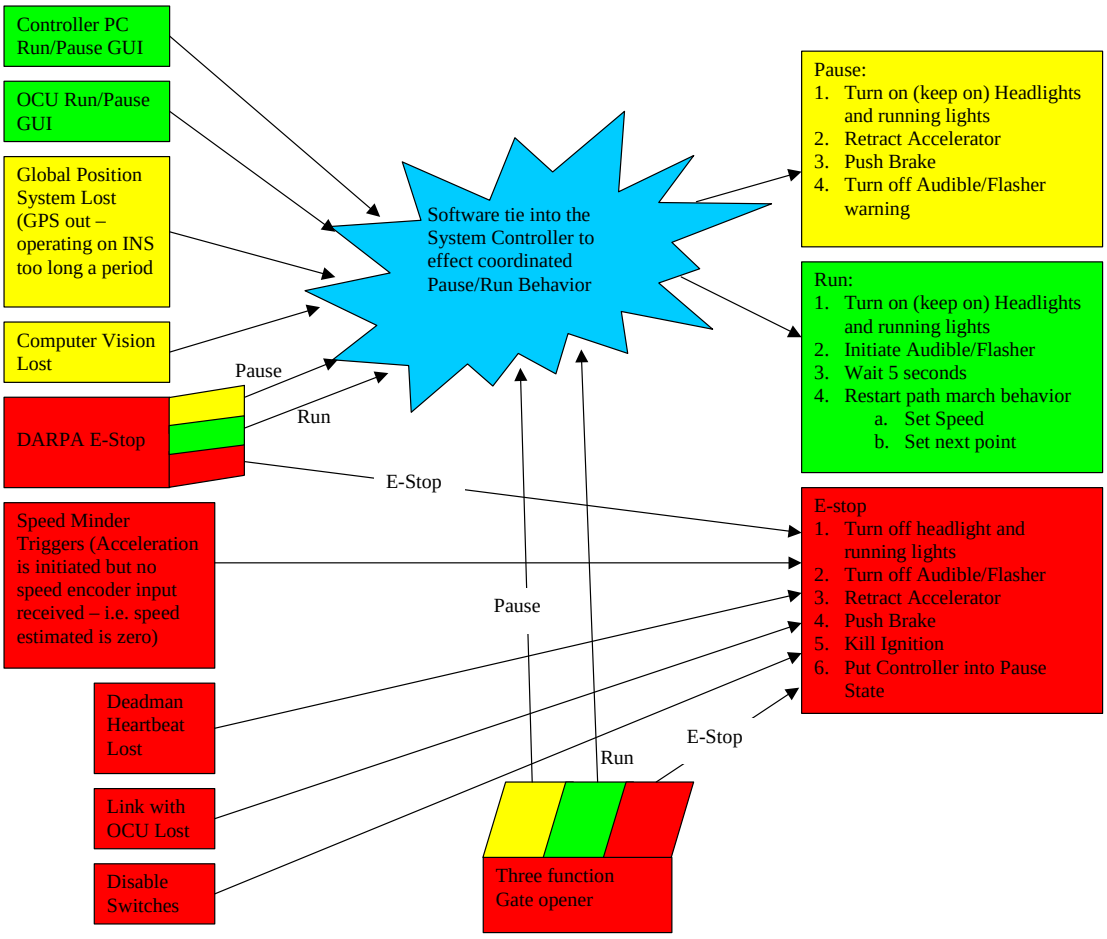

Figure 12. Run, Pause, and E-Stop Logic

Run and Pause behavior is triggered and mitigated by software. In our controller arrangement this can be done by direct access to the control GUI presented by the Controller PC, access to the GUI on the OCU which is wirelessly linked to the vehicle, or through an external signal line read by the digital/analog subsystem. The last option is to

support external connections that trigger Run, Pause, and E-stop (i.e. the DARPA wireless controller).[8]

The power subsystem (Figure 13) could be simple because we had kept the computing, control and sensor hardware systems simple. The overall power budget for the vehicle is nominally 600 watts, which is provided as a combination of the direct 12 VDC from the vehicle power system (for critical hardware safety controls), and 120 VAC generated by two sinusoidal inverters driven by 12 VDC and vehicle generator (600 watt peak for computers and sensors and 400 watt peak for steering servo controls). The vehicle standard generator (which puts out nominally 1192 watts) can power the entire system and the original vehicle electronics, keeping power accommodations to a minimum. For added power reserves for peak loads and when then engine is off we added a second 12 VDC battery and a peaking ultracap to the system. It presently powers the entire electronics load for over 1 hour without the engine and alternator up.

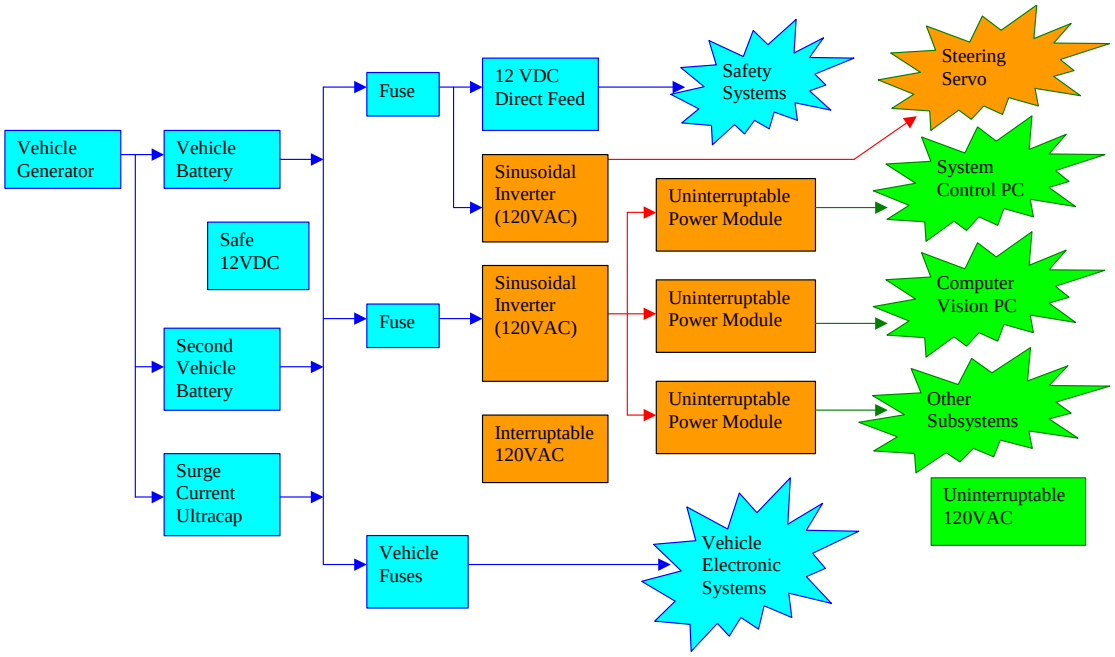

Figure 13. Power System

The vehicle we had planned to used was a Stewart & Stevens LTV. However, because we could not get that in house quickly enough to meet April 13 video deadlines, we decided to use the back-up test vehicle instead. It is a 1996 Chrysler Town and Country minivan. As a commercial vehicle it has a well-understood safety record, but also because it is an old vehicle it roughly matches the wear and repair status of most of the vehicles in the US Army inventory.[9]

[8] As a back-up E-stop we have also included a three function wireless gate-opener to also activate E-stop and Run/Pause through remotely toggled relay contacts.

[9] Even with consideration for continuous repair and maintenance, as of a 2006 Army Science Board study into US Army modularity, the US Army average vehicle fleet age is 13 years – our Chrysler is 11 years old

Because of vehicle age and our plan to support kits that will retrofit present Army fleet vehicles, we elected to bypass internal CAN bus interfaces and go directly into the driver control interfaces (turn signal, head lights, horn, ignition system, accelerator, brake, steering wheel, and front wheel rotations - right and left). We did professional grade tie-ins, but we purposely did not embed wiring and controls inside the vehicle behind veneer panels because (a) we wanted this thing up and running ASAP for testing, and (b) we wanted to have easy access to everything for replacement, debugging, and later on reliability enhancement.

Our overall approach is to drive towards maximum simplicity to keep cost down, reduce software complexity, support sufficient testing, and enhance system reliability. The authors' previous robotics experience indicates reliability and performance has more often been a result of simplicity rather than over sophistication. On a constrained budget this is even more important.

Results and Performance

The majority of analysis has focused on vehicle controls, computer vision processing, and the software command and control required to create driving behaviors. However, before addressing these most important topics, we will touch briefly on supporting hardware and vehicle system performance analysis.

We were determined to keep power requirement below what could be put out by the unmodified minivan generator/battery system. We ran an approximate power budget roll-up that determined that the two PCs, related electronics, and actuators would draw approximately 300 watts. The power inverter was sized for 600 watt output, but we were somewhat concerned whether the vehicle could generate the required energy. The vehicle system is rated to provide 90-110 amps at 12 VDC, which is at least 1080 watts, so the present system plus internal vehicle electronic systems is well within the rating. Testing to date appears to confirm this.

We both designed and ran trials on the vehicle control system because a complete vehicle transfer function was not available. This vehicle had been run through an extensive series of tests at the GM Test Center at Milford, Michigan in the last six months to support development of a microwave vehicle stopper. These tests placed the vehicle on a program controlled dynamometer set-up within an anechoic chamber used for EMI testing. The vehicle was run at speeds between 0 and 40 mph while being assaulted with high powered microwave energy in the 10 meter to 1.6 Ghz range. The vehicle was soft stopped and restarted several times at specific energies, frequencies, and modulation methods proving that this type of application is feasible. This work also gave us preliminary data on vehicle response to controls.

Our analysis of vehicle response to accelerate commands is that it can track updates in the 0.1 to 1 second range. We experimented with up to 100 Hz input changes with limited effect at input speeds faster that 0.1 seconds. The Brake response is

approximately twice as fast as the accelerate response, but brake stroke length (on the linear actuators) is nominally 10 times longer. We also developed a calibration procedure to electronically take out the slack range at the beginning of both the accelerator and brake linkages.

The servo method to hold commanded speed is implemented in three alternative ways depending on route plan requirements. The first is PauseStop control[10] – in this method, the accelerator is retracted and the brake is applied more or less as quickly as possible to the preset limit stops for full braking and zero acceleration. The notion is to get to the commanded zero speed while maintaining vehicle control as quickly as possible for emergency reasons.

The second is BrakeOnly mode – in this method we control speed only through Brake actuator operation. It is used to bring the vehicle to a controlled stop before obstacles, stop signs, when there is temporary navigation failure, or possibly when very slow parking lot speeds are needed. As with most automatic transmission cars, the Armadillo can move at slow speed over flat terrain with the engine idling, so BrakeOnly can provide control of speeds slower than the default idle speed. Basically this mode is used if you want to go very slowly, or if you know you are too fast and want to slow down or stop ASAP in non-emergency situations.

The third and most general mode is BrakeandAccelerate mode. In this mode, using both accelerator and brake controls the speed. It is applicable for most normal speed control on road marches. It prefers to regulate speed by small changes to the accelerator, but if the accelerator retracts completely to the limit stop, it will then initiate speed reduction control via the brake using servo algorithms that approximate the BrakeOnly mode.

A third linear actuator that couples to the shift lever accomplishes shift from Forward to Reverse. The brake and accelerator actuators are hybrid bang-bang, proportional servos that operate on difference between measured wheel speed and commanded speed. The shifter actuator is two state: up to Reverse and down to Forward. Park and Neutral are simulated by retracting the accelerator and fully pressing the brake.

The foundation of the speed control is read out of the wheel speeds. This is accomplished by magnetic encoder pickups mounted on the front drive/steer wheels, left and right. The encoders provide a rising edge every 11.25 degrees of wheel rotation. This is approximately one event every $1/10^{th}$ of a second at 1 MPH and $1/100^{th}$ of a second at 10 MPH. Speed is estimated independently on each wheel by timing and smoothing the periods between each edge.

If speed control is lost, the speed servo would compensate by requesting more acceleration – thus the vehicle would run away if speed feedback failed and thus produced false zero speed estimates. To guard against this eventuality, the Speed Minder

[10] PauseStop and BrakeOnly have been combined into the standard BrakeAccelerate mode – to guarantee a full stop, command zero (0) mph. It goes into an embedded BrakeOnly when the vehicle is commanded to slow down.

task requires that speed encoder readings are being generated from both wheel encoders and that speed estimates be within approximately 30% from left to right sensor system whenever acceleration is being commanded. If this condition fails, E-Stop is activated. *Later on, we anticipate also including pavement motion detection from the vision system into the minder so that we can operate safely even with one failed speed encoder channel.*

The steering system is a conventional motor servo that chain drives the steering column. The servo method used for this application is a conventional PID operating from the difference between encoder feedback and commanded steer angle. The servo operates at approximately 200 Hz.

The Primitive Drive layer directly controls steering angle and speed by continuously adjusting speed and steering angle. Steering angle is set to bring the vehicle heading towards the next "generated' waypoint[11] (the command heading is in the direction of the vector formed from vehicle's present position to the next "generated" waypoint). The speed is set to get the vehicle there approximately at the time commanded.

The Primitive Driver depends upon accurate vehicle self-location or Global Position feedback. This is accomplished by fusing of MEMS Gyro, MEMS Accelerometer, magnetometer, wheel encoder counts, and precision DGPS. The first three items are read at approximately 100 Hz from the Cybernet I3M electronic module mounded on the front bumper of the vehicle. The wheel encoder data is acquired in the speed control as a by-product of measuring encoder periods. The DGPS provides a submeter accuracy position fix approximately once per second to nominally 5-10 cm accuracy. From this DGPS stream we also apply a quadratic extrapolator to determine heading and position data between each DGPS fix.

Fusion of this navigation data is done by the Global Pose Sensor (Estimator). Two alternative methods have been reported by our team and others in the past. The first, and most written about, is using Kalman filter estimation. Our first use of this approach was in 2000 for a vehicle self-location system based on GPS augmented by MEMS accelerometer, gyro, and magnetometer.[12] Because the sensor fusion computation is nonlinear, the Extended Kalman filter is required and, while that approach works, it is very time consuming to properly search for and validate the linearized sensor operating models and other Kalman coefficients. For this reason we looked for an alternative "engineered" approach.

The alternative fusion approach is sometimes called fuzzy or "expert systems"-based. The notion is that we have good approximate ideas of where and when each sensor component is accurate and therefore can create a time-based fusion means that could be characterized as dynamic sensor recalibration based on known sensor noise model and

[11] The terminology "generate" waypoint indicates that this point is computed as the next point to steer towards. It is generated in the Global Waypoint Driver as a series of points that if followed, will drive to the next RNDF defined waypoint.

[12] Enhanced Accuracy INS/GPS System Utilizing Low-Cost Sensors and Geophysical Models, DACA42-00-C-0042, ACA42-00-C-0008, U.S. Army TEC, End date 9/28/2002.

drift rates. The version of this model we have used has two principle modes: DGPSgood and DGPSnotgood.

Before navigation starts, we require that DGPS operate long enough to become "good" – that is that we get DGPS link and GPS solutions that meet the GPS system minimum figure of merit. This reports out at a rate of approximately 1 reading per second as GPS fixes that vary, but only within the expect CEP, and result from a sufficient number of satellites tracked. In this DGPSgood mode we compute heading and quadratic position estimator parameters every time a new GPS value is read. The heading is used to reset and, if necessary, rescale the heading estimator (which is based on integration of the MEMS gyro heading differentials). Between DGPS readings, we estimate vehicle heading as the integration from the last MEMS Gyro heading fix (which was initially set to the last DGPS computed heading) and vehicle position as the likely position based on extrapolation using the quadratic position estimator which was computed at the last DGPS fix. As a side effect we also dynamically recompute front tire diameter based on accumulated wheel encoder ticks and distance traveled as reported by successive DGPS fixes to support more accurate DGPSnotgood dead reckoning.

When DGPS becomes "not good" – no new DGPS fixes are reported because the GPS system internal figure of merit deteriorates – we revert to DGPSnotgood mode. In this mode, filtered absolute heading references generated at an approximately 100 per second rate from the I3M magnetometer system are used to update the MEMS Gyro heading fixing system. Distance traveled is estimated by wheel encoder ticks read at a variable rate depending on wheel speed and scaled by estimated tire diameter. This distance metric is nominally good to 2.5" (6.5 cm). At the MEMS Gyro update rate of 100 Hz, dead reckoning is done based on Gyro heading angle and incremental distance traveled from the last known good DGPS fix. We have estimated that the resulting Global position estimate is good to approximately 1%-2% of the distance traveled or +/- 6.5 cm which ever is larger.

Because at higher levels navigation decisions may be affected by Global Position accuracy, we also compute an estimate CEP for the system that is bounded when DGPSgood to nominally the DGPS CEP and dynamically grows when DGPSnotgood based on estimated error due to distance traveled and time since the last magnetometer fix.

Because the magnetometer is affected by mounting on the car (i.e. proximal ferric metal structures), and the location of true north versus magnetic north in and around the course location, we provide a calibration mode that can be executed during vehicle set-up. The calibration mode collects GPS points and headings through a set of preplanned vehicle maneuvers, and then computes (using a LSQ estimation algorithm) calibration parameters for the magnetometer/MEMS accelerometer sensors. This polishes the calibration on the I3M that is done in laboratory benchtop facilities at the operating site for enhanced accuracy.

The Reflexive Driver operates on next "generated" waypoint data that is identical to that used by the Primitive Driver, but where the Primitive Driver just tries to execute the speed and heading commands to hit the point, the Reflexive Driver also checks feasibility. This is done by examining a vehicle centered "free space" map that shows all detected impediments to travel in directions from +90 to –90 degrees from the present vehicle heading. This map is populated with impediment location and relative velocity at approximately 75 Hz, the frame rate of the laser radar. *Data from the computer vision system is also included at nominally the frame rate of the color CCD cameras (30 Hz).*[13] The Reflexive Drive checks speed and heading commands to determine that executing them will be safe through the vehicle's stopping distance before passing the commands down to the Primitive Driver.

If an unsafe condition is detected, the Reflexive Driver will either reprogram the "generated" waypoints to steer around the detected impediment so as to maintain a safe stopping distance (defensive driving) or will command a controlled stop until a safe steer around path becomes available. If after a nominally 10 second wait time no free path is present, the Reflexive Driver will fail back to the Global Vector Driver (and possibly cascading from there back up to the Master Executive to (a) turn the vehicle around, and (b) re-plan a route to accomplish moving to the mission checkpoint through an alternative route).

The Reflexive Driver gets its next points from the Global Vector Driver. The Global Vector Driver breaks waypoint-to-waypoint vectors into smaller segments that can be achieved by the Primitive Driver. It operates on two alternative path generation models. The first is to keep to generated waypoints that follow the line between successive waypoints – this effectively is to drive blindly between waypoints with reflexive path changes that avoid obstacles. The second alternative is to make generated waypoints that follow high certainty road edges detected by the computer vision subsystem. The latter is the preferred mode when the computer vision system has high certainty, but we flip to the former mode when this is not true (or in intersections and zones where we know road edges will not be informative). The Global Vector Driver uses a b-spline fitting method to create the optimal path between waypoint specified by the Global Waypoint Driver so that it can match waypoint positions and headings in a smooth manner. This helps to properly move through intersections from exit to entry waypoints with the proper turning radius and vehicle orientation change required.

The Global Waypoint Driver selects the next waypoint-to-waypoint path from the waypoints listed in the lanes encapsulated in the segment provided by the Global Path Segment Driver. Special behaviors like path changes to hit checkpoints, lane changing, stopping and going at intersections, and turning around before failing to the Master Executive for route re-planning are closely related to waypoints as defined in the RNDF.

[13] Computer vision portion is not yet integrated at the time of this draft. It will consist of motion objects and, in special cases, vehicle target identifications located in range based on size and elevation in the image calibrated to the camera flat Earth perspective model. Range is less precise using this approach, but since the ladar provides accurate range only within about 30m or so, the computer vision has the primary task of finding earlier detections of fast moving objects in the far field at intersections and when crossing lanes.

Thus, logic for these activities is housed in the Global Waypoint Driver object. *At the time of this draft, road march behavior and zone march behavior are essentially the same with minor differences. Both proceed to get from one waypoint to the next allowing the Reflexive Driver to accommodate to obstacles that might perturb a smooth path march. In the ultimate system, this simplistic approach will be replaced by an area-based frontier exploration-based path march approach that has been proven in an on-going project we are doing for Picatinny Arsenal.*[14] *This approach will provide particularly better performance in zone maneuvering.*

The Global Path Segment Driver is given a list of road and zone segments by the Master Executive based on a known starting point (i.e. where the vehicle is positioned and oriented on the road network now) and the next checkpoint that must be hit. Re-planning occurs when (a) the present plan fails due to an unexplained path blockage, or (b) when the checkpoint is hit. The mission is completed when the last checkpoint is hit. The mission is aborted (i.e. re-planning back to the start point) if all segments to the checkpoint have been explored without success (i.e. re-planning that reflects all known broken segments and open segments yields no possible segment list). Because the present expectation is that the road network will be relatively simple, the planning process used is an exhaustive Dijkstra's algorithm.[15] When zone behavior is fully added this may be replaced by an A* algorithm.

The System Commander is effectively the vehicle user interface. It allows output of tracking data, control inputs to the low-level vehicle functions, initiation of higher level commands like Pause, Run, E-Stop, display of vehicle status indicators, and support functions like I3M recalibration, load/store RNDF data, load/store MDF data.

The System Commander is available on the on-board vehicle control computer and from the wireless OCU/kill system. The DARPA wireless kill system effectively plugs into this subsystem via the hardware Run/Pause and E-Stop inputs.[16]

Results To-Date and Conclusion

To support tracking, decision making, and control systems performance evaluation, the system logs time stamped location, heading, individual position determination sensor streams, computer vision captured data, laser range data, and decision status data. Presently we are still tuning vehicle control behaviors, so we can report tracking errors only approximately.

[14] Previous footnote 3.
[15] http://en.wikipedia.org/wiki/Dijkstra's_algorithm
[16] Recall that the E-Stop function operates both at a low-level hardware level and is sensed by the System Commander at a higher level, so that related software status can be brought in alignment with hardware E-Stop status.

The vehicle tracks commanded speed to approximately +/- 2MPH in accelerator control mode and better than 1 MPH in BrakeOnly mode, however large swings in commanded speed can over or undershoot by as much as 5MPH.

It follows point tracks as commanded to approximately +/- the CEP of the GPS (+/1 5-10cm).

It keeps accurate heading information to +/- about 0.2 degrees.

Wheel encoder accuracy is to about one encoder uncertainty (+/- 6.5 cm), *but how this combines with heading certainty for accuracy when DGPS is not good is not yet known – past systems have given us +/- 1%-2% of distance traveled. We expect that the higher level planning functions will be near real time with the complexity of road network expected. This is presently the case with the complexity of the site visit map.*

We have had good meantime between failure for the system to-date, but because most of the runs have been debugging runs, it is too early to make any representations in this area.

Because higher-level behaviors are still under development, is not possible to make final conclusions about this part of architecture as yet. However, site visit basic navigation and traffic understanding fits into this framework well.

The Global waypoint driver does most of the real work but when it fails out due to obstacle detection, segment completion, or stopping, the higher level, Global segment driver which houses intersection, go-around, lane merge, U-turn, etc. behaviors fires up and provides the interface between the mission and vehicle behavior. The mission planner is called up whenever the Global segment driver indicates that the current plan is no longer valid.

These components run real time in three control computers (not including the lightly loaded OCU interface computer). Urban1 (Dual Core Pentium) is 100% loaded with camera processing, capture, compression, & image stream logging, road following, and proximity detection. Urban2 (also a Dual Core Pentium) is approximately 65% loaded with vehicle control, interfaces, and mission planning. Urban4 (a Pentium-M) is approximately 25% loaded per ladar channel – right now only one is attached, but it is not unlikely that a second channel will be added after the site visit.