Team Overbot

DARPA Grand Challenge 2005 Technical Paper

(pre-Grand-Challenge version)

Team #A133

Vehicle #11

Team Overbot
2682 Middlefield Road, Unit N
Redwood City, CA 94063

www.overbot.com
info@overbot.com

Introduction

This is a preliminary report on Team Overbot's entry in the DARPA Grand Challenge. We will publish a more detailed paper after the actual event, when we can discuss the results.

The Overbot

Results from the 2004 Grand Challenge

We're now preparing for the second Grand Challenge, so it's appropriate to learn from last year's results.

Last year was an embarrassing debacle. Twenty teams were at the starting line. Only seven vehicles made it out of the starting area. Only three covered more than five miles. Only one (Team DAD) went more than seven miles without crashing.

Other than mechanical failures, the lessons to take away from last years event were this:

- **Most of the course is on dirt roads.**
  It's not about crossing trackless desert or fields of boulders. The biggest single problem is staying on roads. The boundary of the road may be ill-defined.
- **Manual preplanning is not greatly helpful.**
  The CMU team attempted to manually preplan the route in the two hours between release of the waypoints and the start of the event. Despite this, their vehicle crashed three times in eight miles.
- **Precise waypoint following is very important.**
  Team DAD admits privately that all they were really doing was following waypoint centerlines as precisely as possible.
- **P restored terrain or road data is of little value.**
  This is an artifact of DARPA's waypoints, which, at least for 2004, followed roads very closely when on roads.

The vehicle

The base vehicle is the same 6WD Polaris Ranger as last year, and is described in our 2004 paper. Steering, brake, transmission, throttle, and tilt head are driven by servomotors using Galil DMC-1416 programmable controllers. These controllers work quite well, although they are rather bulky for an automotive application. We had previously used MicroMo controllers, with notable lack of success. The MicroMo units, if used to drive larger motors within their rated capacity, will burn out and produce smoke, due to inadequate heat-sinking of the power semiconductors. This was part of why we withdrew from the Grand Challenge last year. The Galil controllers are much larger, but actually work.

This vehicle was picked because it has a good record of traversing roadless ground successfully. It's not really the right vehicle for the more road-oriented event the Grand Challenge turned out to be. We were worried about becoming stuck in sand dunes or mud flats, which wasn't the problem which needed to be solved.

If we were doing this today, we'd use a Jeep Wrangler with controls from Advanced Mobility Controls. Unfortunately, when we built the existing vehicle, AMC only equipped mini-vans, their primary business being manual controls for the disabled.

Computing and networking

We have five Galil motor controllers, an Ethernet to serial interface for the LIDAR, and two industrial Pentium 4 machines. All these devices are networked using 100baseT Ethernet and industrial Ethernet hubs.

The two larger machines run QNX, the real-time operating system, as a distributed system. While any task can potentially run on either CPU, we normally run all the I/O bound programs on one machine and the steering controller on the other. The machines have watchdog timers, and can reboot if necessary. The overall system is stateless, in that, if restarted, it simply finds its position and picks up the mission from its current location. Other than logging information for post-mission analysis, the system writes nothing to disk during operation.

Software system overview

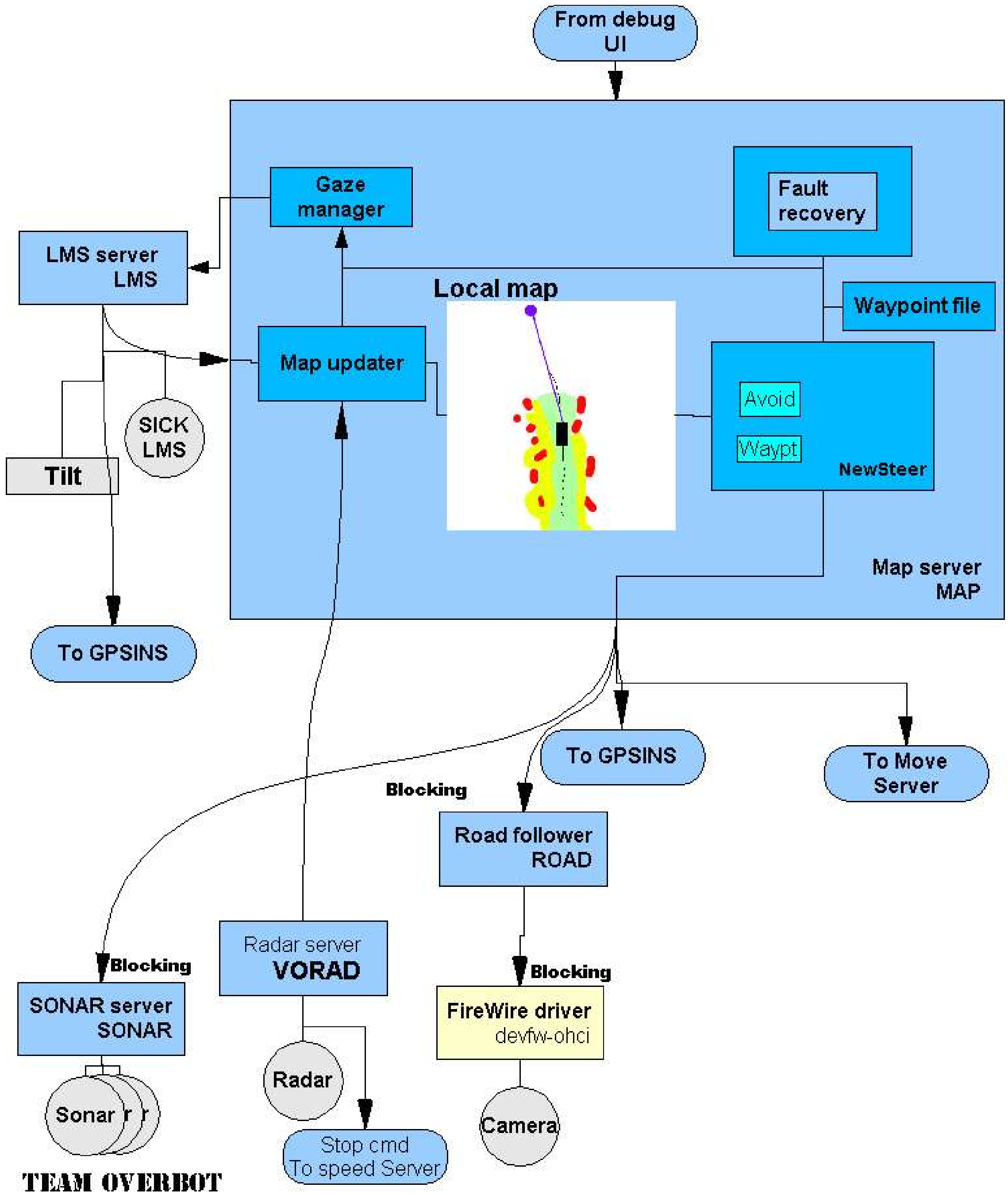
The software is organized as a collection of intercommunicating programs, connected using QNX interprocess messages. Because this interprocess communication mechanism has considerably greater bandwidth than UNIX-type socket-based systems, we are able to divide the system into multiple programs without excessive overhead. Interprocess traffic is heavy; LIDAR and video data are passed around via this mechanism. Each program runs in its own protected address space.

All programs are subordinate to the "watchdog" program, which helps sets up interprocess communication between them and monitors their status. If any program fails, the entire system shuts down and restarts.

The MAP program does the steering control. It runs on a fixed 100ms cycle, producing a new steering plan at the end of each update cycle. The MAP program generates "moves", commands to move a specified distance at a specified curvature at a specified speed. The intent is that generated moves are safe; if run to completion, the move would not produce a collision. But, except in emergencies, moves are not run to completion; the current move is replaced by a new move every 100ms. Only in an emergency stop is a move run to completion.

The GPSINS program combines inertial, compass, driveshaft, and GPS data into a current position estimate. If necessary, it can dead reckon during GPS losses, although this is not valid for more than 50m or so. The real job of the GPSINS program is to smooth out anomalies in the GPS data, so that updates to the local map align properly.

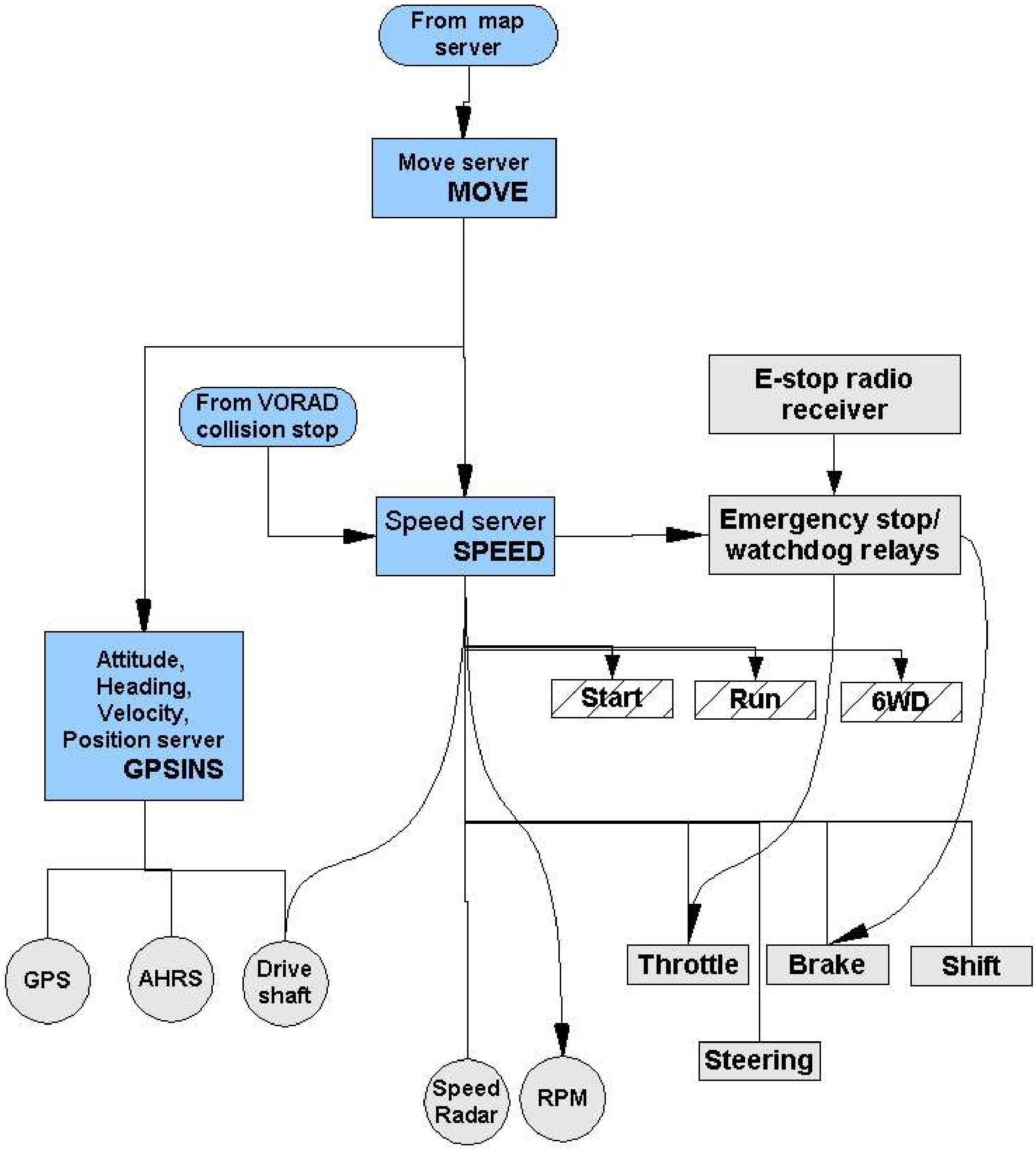
The coordinate system used is a flat earth model in a plane tangent to the earth at the first waypoint.



TEAM OVERBOT

Client/server hierarchy for GC vehicle
Sensors and field map
22 August 2005
John Nagle

Note: Lines enter servers from the top.
Arrows are client -> server
Blocking servers are marked.



TEAM OVERBOT

Client/server hierarchy for GC vehicle

Control level

22 August 2005

John Nagle

Note: Lines enter servers from the top.
Arrows are client -> server
Blocking servers are marked.

Localization

Navigation uses an attitude and heading inertial unit from Crossbow (this includes a magnetic compass), a 15cm accurate GPS from Novatel, and a driveshaft encoder. These are combined in a relatively straightforward manner. A simple dead-reckoner, using the compass and driveshaft encoder, is provided. When high-precision GPS data (15cm) is available, the dead-reckoning data is generally ignored. When 1 meter GPS data is available, dead reckoning is used to clean up jitter in the GPS data. If GPS data is lost, the vehicle runs entirely on dead reckoning. When running without GPS, a circular error probability is computed and subtracted from the waypoint segment width. When the circular error becomes large enough, the vehicle gives up and shuts down. On pavement, dead reckoning results in an error of about 2m to 4m for every 100 meters traveled.

We do not carry any prestored map data on the vehicle other than the waypoint file. We did at one time have a road map database, but after the 2004 Grand Challenge, it was clear that it was unnecessary. Terrain data is almost useless unless of very high resolution, and the teams that obtained such data last year tended to hit obstacles that weren't on their maps. So we avoided that approach. Our general approach is not to overdrive our sensor data, even if it means driving slowly.

Sensors

The sensor set is conventional – a line scanning LIDAR from SICK, and a standard automotive anti-collision radar.

The LIDAR is mounted high, on a tilt head, and normally tracks the ground about 1.5 to 2 stopping distances out from the vehicle. When this does not result in good data, the vehicle stops, and the LIDAR is swept to build a good 3D image. This gets us through tough sections.

We have a weathertight LIDAR unit and a cleaning system for it, so dust should not be a problem.

LIDAR and its discontents

Most teams use SICK LMS laser line scanners in some form or other. These are mechanically scanned line scanners. They're reliable, but bulky and have inadequate range. And, fundamentally, line scanners are simply too limited to be used for driving. Viewing the world through a narrow slit is, at best, a marginal approach.

Better devices have been built, but due to lack of a market, not produced in volume. Multibeam scanners have been built, most notably by General Dynamics Robotics, but only as prototypes. 3D scanners go all the way back to the original ERIM device used on the original CMU Navlab, but no commercial product of that type, other than very slow devices used for surveying, has achieved even modest commercial success. Two-mirror devices with a big enough collecting aperture to be useful are rather large, comparable to television cameras of the 1950s.

True 3D solid-state flash LIDAR devices exist. We've visited Advanced Scientific Concepts in Santa Barbara, CA, and have seen an eye-safe 128 x 128 pixel solid state flash 3D LIDAR suitable for outdoor work in operation on an optical bench. The device consists of two custom chips bonded back to back using ball grid array techniques. The front chip contains the array of detectors, and the rear chip contains the counters, timers, and interface logic. The detector chip typically uses indium arsenide technology. Some versions are front-ended by a photomultiplier cathode, like a night vision device. (The photomultiplier effect is at the atomic level, and has no integration delay, so it can be used to front-end a LIDAR detector.) The two-chip approach is a convenience for prototyping; a volume production unit would probably be a single chip.

Prototypes of this unit now exist, and are quite compact, about 20cm on a side. But under Grand Challenge rule 3.4 (3), we cannot use it, since it is a “patented invention that was developed under government funding” but was not “available to all Grand Challenge teams on June 8, 2004”. Next year, perhaps.

Clearly, this is the right direction. Rotating optical machinery is a dead end, like the mechanically scanned television of the 1930s. In hopes of accelerating commercialization of this flash 3D LIDAR technology, we introduced a venture capitalist who funds photonics companies to the makers of this prototype, but the market did not appear sufficient for serious investment. Flash LIDAR built using this approach should ultimately become reasonably cheap, but only when there's a market measured in tens of thousands of units per month. It's not clear where a market of that size will come from in the near term.

LIDAR data cleanup

Thus forced to work with line scanners, we must struggle with the problems of extracting good terrain profiles using only a line scanner.

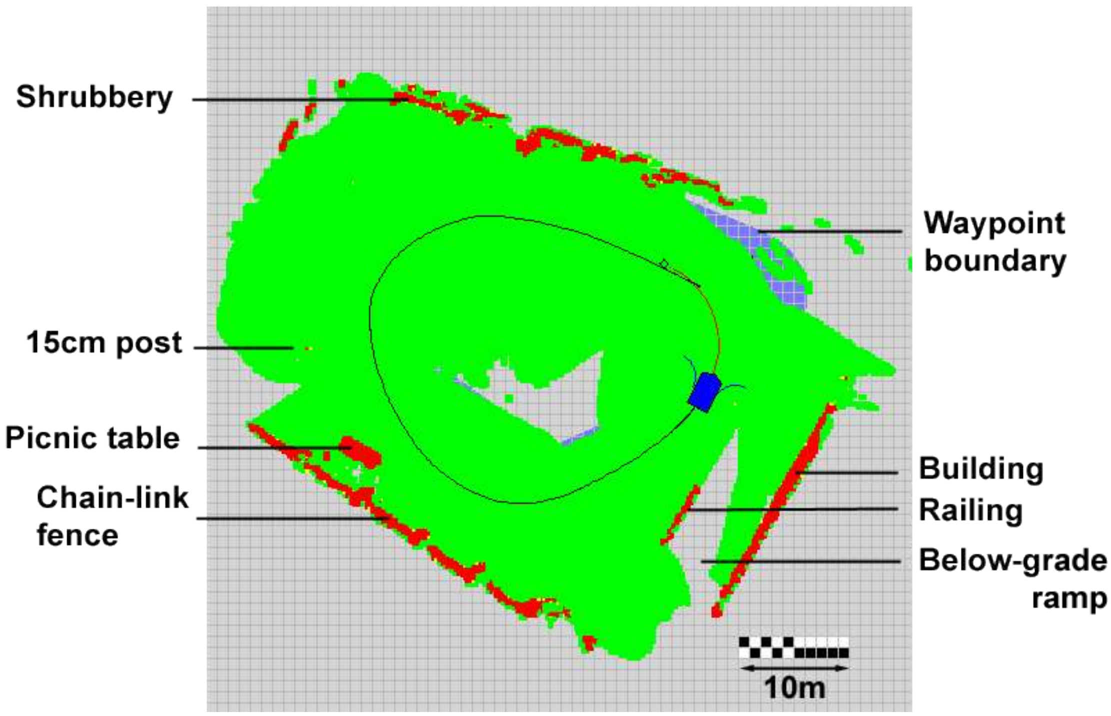
We decided early on that evaluating terrain was essential. Merely detecting “obstacles” was not good enough. Thus, we have a line scanner mounted high, on a tilt head, and aimed downward. It's common in indoor robotics to have one mounted low, at bumper height, and aimed horizontally, as a simple obstacle detector. But that won't keep a vehicle from driving into a ditch or off a cliff.

Working with line scan data from a moving vehicle has obvious problems. One approach is active mechanical stabilization, such as CMU uses. We considered that, but our modest vehicle couldn't support a huge 3-axis gimbal without becoming top-heavy. Even the one-axis mount we use required that we put 90 pounds of steel under the floor of the vehicle to bring the center of gravity down.

We are thus using several techniques to clean up single line LIDAR data.

First,we go to some effort to get the GPS, INS, tilt head, and LIDAR data all synchronized. The GPS outputs data at 20Hz, the INS at 50Hz, the LIDAR at 75Hz, and the tilt head servo at 100Hz. A combination of software-implemented phase locked loops and interpolators synchronizes everything to within about one scan line time.

The tilt head is on shock mounts, and vibrates a few degrees in pitch relative to the INS, which is about two meters from the LIDAR. We'd considered mounting a small INS on the LIDAR, but instead developed a new method for correcting for small pitch errors based on the LIDAR data itself. The average range for a narrow band near the centerline is computed for each scan. Short-term variations in the average range are used, after some filtering and processing, to provide information for pitch correction. This reduces the



positional noise at longer scanner ranges.

The result can be seen above. The map is composed of 20cm cells, filled in by the LIDAR processing system. Cells are marked as “red” (impassable), “yellow” (marginal), and “green” (flat and clear) by comparison with adjacent cells.

Note that the LIDAR is not aimed horizontally. It is actively profiling the ground, and servoed to track the ground. Elevation data is extracted. Negative obstacles, such as the below-grade ramp shown above, are thus detected.

Vision

We have a visual road follower, which looks for longitudinal features and tries to match curves to the view ahead. It's less edge-oriented than most road followers, since it is intended to work on dirt roads. The output of the road follower is used as a hint to steering control, to generally move the vehicle towards the center of the road. The traversability data from the LIDAR has priority over the road follower data.

We considered stereo vision, and built up a prototype system. But dirt roads don't have enough hard edges for stereo lock, and stereo vision was thus unable to detect anything but blatant obstacles. We have a VORAD anti-collision radar to stop the vehicle before it hits something big, like a car, so that's already covered.

Planning and steering control

We have a serious sensor myopia problem Because we insist on profiling terrain, and not outdriving our sensors, we can't look that far ahead. With the sensors we have, steering control has to work as if driving in heavy fog.

Steering control is based primarily on the LIDAR-generated local map, as shown above, with its colored cells. A second level of processing evaluates marginal cells, based on their actual elevation. The elevation data is quite good if we stop and tilt the LIDAR, adequate on straight, flat roads, and of limited use otherwise. So we can drive well on straight, flat roads, and muddle through tough spots, although slowly.

Our first steering controller was arc based. This did not work well in narrow sections. Obstacle avoidance using arcs works in open spaces, but in narrow waypoint segments usually results in placing the vehicle in a situation where it is angled and stuck between the obstacle and the boundary. So we were forced to go to an S-curve planner. Results with S-curves are much improved over the arc based planner. We evaluate about 20 possible S-curves for each steering decision, pick one, and then improve it by slightly adjusting the endpoint to improve clearances. When an S-curve intersects a lateral boundary, its ending direction is forced to become parallel to the boundary, which helps in maneuvering in narrow waypoint sections.

The steering controller outputs “moves”, a command to move a specified distance along a specified, known to be safe, path. Speed is limited so that the vehicle can stop within this safe move distance. Every 100ms, a new “move” command is generated, replacing the previous one, so no “move” is ever actually run to completion, except during an emergency stop. A typical move is 10-25 meters.

We have the ability to back up short distances, which is used mainly to recover from minor problems. We don't have a true maze-type A* planner; the sensor data is seldom complete enough for one to be useful.

All this, of course, is driven overall by waypoint following. We generally try to follow the centerline between waypoints, and in the absence of any other constraints will do so. Departures from the centerline can be forced by obstacles, unfavorable terrain, or the waypoint boundaries themselves in tight turns. We can negotiate 90 degree waypoint turns without difficulty. Tighter turns generally result in stopping and backing up to reposition.

Our basic goal is to not crash. Many other entries will drive faster than we will. We expect that the ones that outdrive their sensors will crash at some point.

Vehicle control

Vehicle control is conservative. We limit the vehicle to 15 degrees of pitch and roll, per the Polaris Ranger specs. Stopping distance is computed using a value of 0.25g deceleration, including the gravity component, which limits downhill speeds.

Driveshaft speed, brake pressure and engine RPM are measured and used with simple control loops to control acceleration and deceleration. Brake control is good, and engine control is sluggish. The Polaris Ranger has a continuously variable transmission front-ended by a centrifugal clutch. This device behaves weirdly enough that an adaptive model-based controller is needed to drive it properly. Unfortunately, we're controlling the throttle with an ordinary PID loop, detuned to avoid overspeed.

If the vehicle slips back while going uphill, “hill holding”mode is activated, and the engine is throttled up with the brakes locked until a specified RPM value is reached. The brakes are then released. This produces rather rough acceleration, but works.

With 6 wheel drive and large 10PSI tires, the vehicle tends not to get stuck as long as the wheels are on dry ground and the vehicle is more or less upright.

Fault recovery

Much of recovery is based on the fact that the system is nearly stateless. The entire system can be restarted at any time, and the vehicle will continue the mission from its current location. The CPUs are equipped with hardware stall timers, and if all else fails, a reboot will bring the system back to life.

But that's for system failures. The usual recovery action is to stop, sweep with the LIDAR, build a good local map, and try going forward again. If that doesn't work, small backwards moves are attempted. Alternating between backwards and forwards motion will then be attempted. Such muddling through is able to deal with many minor problems.

If the vehicle finds itself a very short distance outside the lateral boundary, it will attempt to get back in at very slow speed. If more than a few meters off course, the vehicle will shut down, for safety reasons.

There's no higher level planner. We generally don't have enough global information for one to be useful.

Testing

Most of our testing has been in a fenced yard in an industrial park. We've also tested extensively in a very large parking lot built for an office park that was never finished. Off-road testing has been on dirt roads at a nearby horse ranch. When we need more space, we use an off-road vehicle park. We have not taken the vehicle to a desert location. We have operated it in rain, heat, and cold, without much difficulty. All onboard equipment is rated for industrial temperature ranges, and suitable fans, filters, and shock mounts have been provided.

Component reliability has not been a serious problem with the current components. We did have some overheating on a hot day (39C) in one of the Galil motor controller power amplifiers, but it properly shut down at 65C without damage. Extra fans were added to that unit, and the hood was painted white. We've since gone over the vehicle with an infrared thermometer to check for hot spots.

As a policy, we never drive the vehicle autonomously with a person on board. We've had an industrial E-stop radio system with a continuously maintained link from the beginning, and always have someone with their finger on the red button.

Lessons learned

The price of entry to the Grand Challenge is substantial. It's therefore appropriate to examine the expected return on the investment.

The Grand Challenge rules purport to require that teams not be Government-funded. This rule is not enforced. In practice, several teams have team members or leaders who are funded by the Government for robotics research under other contracts, and are affiliated with institutions which receive Government funding for robotics research. One team is even from a Federally Funded Research and Development Center, a quasi-governmental organization.

The payoff is direct and immediate for such teams, even for teams that lose. As one such team put it in a press release, “Win, lose or draw, the work of the ... Team will not go unrewarded.” Entering the Grand Challenge for such teams is effectively a sales expense for a government contractor. The private sector must wait for an actual market.

While some like to compare the Grand Challenge with the Wright Brothers' first flight, that's a stretch. A realistic analogy is the first automated aircraft landing, on August 23, 1937, by Cpt. Carl J. Crane, USAF. That technology was badly needed during WWII. Thousands of crashes on fog-shrouded runways in England might have been prevented. But it was not developed at the time, and was not available during WWII. Autoland didn't become usable until the late 1960s and did not become pervasive in air transport operations until the 1990s. It was thus half a century from the first successful demo to volume commercial sales.

Automated freeway driving was demonstrated convincingly in August 1997 as part of the National Automated Highway System Consortium's “Demo 97”, a joint venture of Bechtel, CALTRANS, CMU, Delco, GM, Hughes, Lockheed Martin, and the University of California. The demo, using actual freeway lanes in San Diego and over a dozen automated vehicles, including automated maintenance vehicles, was quite successful. In 1998, the program was completely defunded and almost all US work on automated freeway driving has ceased. And this was forty years after GM first demonstrated automated lateral vehicle guidance with their Firebird III prototype.

Thus, commercial payoff in this area may lie decades in the future.

Once we recognized this, we cut back on our effort. We spent twice as much on the 2004 Grand Challenge as we did on the 2005 Grand Challenge.

Conclusion

The Overbot has adequate, but not definitive, solutions to the major problems. With current sensors, the Grand Challenge is just barely possible. Single-line LIDAR is just too limiting. Another year of sensor development will cure that problem. Meanwhile, we can drive the course at modest speeds with some assurance of not hitting anything or falling off a cliff.

The Grand Challenge has resulted in real progress in automatically evaluating terrain while driving. But we're still a long way from situational awareness or image understanding. We'll stop for a tumbleweed, because we can't distinguish it from a rock. Nobody seems to have a good solution to that problem yet.

After the 2005 Grand Challenge, we will revise this paper based on the results obtained.