High Speed Navigation of Unrehearsed Terrain: Red Team Technology for Grand Challenge 2004

Chris Urmson, Joshua Anhalt, Michael Clark, Tugrul Galatali,
Juan P. Gonzalez, Jay Gowdy, Alexander Gutierrez, Sam Harbaugh,
Matthew Johnson-Roberson, "Yu" Hiroki Kato, Phillip Koon,
Kevin Peterson, Bryon Smith, Spencer Spiker, Erick Tryzelaar
& William "Red" Whittaker

CMU-RI-TR-04-37

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

June 1, 2004

This research was sponsored by generous donations from:

Boeing	Intel	SAIC
Alcoa	Applanix	Seagate
The Robotics Foundary	Advanced Motion Controls	BF Goodrich
Caterpillar	Chip Ganassi Racing	CMLabs
Crouzet	ExelTech	Earlink
TerraSim	ATI	StarBand
Google	HD Systems	Hutchinson
Mechron	Zombie Studios	Kubota Industrial Engines
Lord Corporation	Space Imaging	OmniStar
Acumen Systems	Rod Hall: Team Hummer	Trimble
Vicor	Visual Intelligence	Aliyance Group
Yamaha	BAE Systems	Carnegie Mellon University
Driverless MotorSports.com	Harris	ISI
Motion Picture Marine	NATC	S-B Equipment Service
Ro-Pro Design	J&J Truck Bodies & Trailers	

Abstract

This report presents technologies that have been created and integrated to achieve high speed navigation of unrehearsed terrain. The discussion is organized into three technology areas: mapping & pre-planning, hardware, and navigation software.

The mapping & pre-planning component of the Red Team generated a massive, multi-data format map of the ~50,000 sq. mile region surrounding the race corridor. This data was utilized by an automated pre-planning system and a team of human experts to generate a safe route across the Mojave Desert.

Sandstorm is the most recognizable component of the Red Team race system. It incorporates a highly capable mobility base and high performance computing and sensing. A critical component of the sensing system is a stabilized gimbal, used to ensure useful data can be collected for the on-board software to digest.

The navigation software is capable of driving off-road trails at speeds over 35 mph. This software is described with an explanation of its capabilities and limitations.

This report concludes with a summary of the Red Team's performance at the 2004 Grand Challenge, and outlines several lessons learned during the process of developing the race system.

Table of Contents

Abstract i
Table of Contents ii
List of Figures iv
List of Tables iv
Context 1
Introduction 1
Mapping & Pre-Planning 2
  Infrastructure 4
  Automatic planning 4
  Planning the path 5
  Vectorizing the Path 6
  Route Editing Tool 7
Vehicle Hardware 9
  Electronics Box 10
  Computation 11
  Sensors 12
    LIDAR 13
    Stereo Vision 14
    RADAR 14
    Pose Estimation 14
  The Gimbal 15
    Implementation 15
Navigation Software 18
  Overview 18
  Infrastructure 19
  Gaze Control 20
    Control Architecture 20
    Calculation of stopping distance 21
    Calculation of desired yaw 21
    Calculation of desired pitch 21
  Gimbal Control 22
    Interfaces 22
    Coordinate Systems 22
    Internal States 23
    System and Control 23
  Terrain Evaluation and Path Adjustment 26
    Terrain Evaluation 26
    Path Adjustment 27
  Binary Obstacle Detection 28
  Path Tracking 28
  Vehicle Controller 29
    Gear Selection 29
    Steering Control 29
    Velocity Control 29

Performance 31
Description of Significant Incidents on Race day 31
Impact with Fence Post #1 31
Impact with Fence Post #2 32
Momentary Pause 33
Impact with Fence Post #3 33
Impact with Boulder 34
High Centering in the Hairpin 36
Other Points of Interest 38
Lessons Learned 39
Conclusions 40
References 40

List of Figures

Figure 1. A comparison of USGS and Visual Intelligence DEM data quality. 2
Figure 2. A comparison of USGS and Visual Intelligence image quality. 3
Figure 3. An illustration of the operation of the global path planner. 6
Figure 4. Three views of data available to the route editors. 7
Figure 5. An illustration of how splines are constructed for the route editing tool. 8
Figure 6. The arrangement of shock isolation components that support the electronics box 11
Figure 7. The overlapping fields of view of sensors on Sandstorm 12
Figure 8. A model of Sandstorm showing the placement of sensors. 13
Figure 9. An exploded view of the gimbal roll axis. 16
Figure 10. The fully assembled gimbal 17
Figure 11. The on-board software architecture. 18
Figure 12. Various configurations where a non-pointed sensor performs poorly. 20
Figure 13. The gimbal control system. 24
Figure 14. A model of the gimbal control system. 24
Figure 15. The Root locus and Bode plot of the gimbal control system. 25
Figure 16. The frequency response of the gimbal control system. 25
Figure 18. A plot showing the response of Sandstorm's velocity control loop. 30
Figure 19. A sensor view of the first fence post Sandstorm hit. 32
Figure 20. A Nissan Xterra parked in the opening of the second gate. 32
Figure 21. A sensor view of the second fence post Sandstorm hit. 33
Figure 22. A sensor view of the large boulder. 34
Figure 23. The large boulder Sandstorm hit. 34
Figure 24. A plot showing the path constraints imposed on Sandstorm by the pre-planned route. 35
Figure 25. The large rock that eventually ended Sandstorm's run. 36
Figure 26. A sensor view of the trail in the Dagget Hairpin. 37
Figure 27. The pre-planned route around the hairpin. 37
Figure 28. Sandstorm's final resting place on race-day. 38

List of Tables

Table 1. A summary of HMMWV capabilities. 10
Table 2. A summary of the gimbal performance characteristics. 17

Context

The 2004 Grand Challenge was a 143 mile race across the Mojave Desert from Barstow California to Primm Nevada. The team whose robot completed the course in the least amount of time, and did so within 10 hours would win one million dollars. Prior to competing in the race, each robot was required to demonstrate safety and navigation capabilities in a qualification and demonstration event held at the California Motor Speedway.

After successfully completing qualifications, invited teams transitioned to the race start area near Barstow California. Roughly 3 hours prior to the race start, the organizers provided each team with a CD describing the route. This description consisted of 2586 waypoints that marked the corridor within which the robots were required to travel. With each waypoint, the organizers provided a maximum speed limit and a lateral offset that defined this corridor. Once away from the starting lines, the robots were required to be completely autonomous. The only communication allowed to the robot was through a remote safety kill and telemetry system used by race personnel to ensure the safe operation of the robots.

To maximize performance, the Red Team combined autonomous and human pre-planning to optimize the route in the three hours prior to Sandstorm's departure. Once underway, the robot used onboard sensors to adjust the preplanned route, to account for obstacles that were not visible in the pre-planning database, and to correct for errors in its position estimate.

The Red Team system proved to be the most successful approach to solving this grand challenge and demonstrated a new and unique approach to the cross country navigation of unrehearsed terrain. The Red Team was the only team to qualify based on the original specification for the qualification event and went the farthest and fastest on race day.

Introduction

Completing the Grand Challenge requires a leap in autonomous navigation capability. To date, autonomous robots have enjoyed successes traveling at high speeds on regular, well structured terrain, such as highways [12][16]. In other work, robots have been shown to drive autonomously at slow speeds across a variety of off-road terrains [3][8][13][14]. There has been limited success in driving at high speeds off-road [4]. At this point, the basics of the cross country navigation problem are well understood [11] and ongoing research can be divided roughly into two thrust areas: driving faster, and driving in more complicated terrain.

The technology implemented by the Red Team for the Grand Challenge makes novel use of prior information and real-time planning combined with a good mechanical design and sensing scheme to achieve high speed travel over unrehearsed, off-road terrain. Past systems have used low resolution a priori knowledge to generate high-level plans [17]. These systems generally operate with very course prior data, thus requiring the real-time navigation system to operate in an exploratory manner [18]. By utilizing high resolution

prior data, an onboard navigation system is exhibited to plan over a smaller area without compromising global optimality, and can potentially plan with much higher resolution.

This report describes the component technologies created and integrated to perform high speed navigation over unrehearsed, off-road terrain. Three key technology areas are described: Mapping and pre-planning, hardware and the navigation system.

Mapping & Pre-Planning

Mapping synthesizes a high resolution, high accuracy map of the race region to enable autonomous planning and human refinement of a winning route. In building the "perfect map" for the 2004 Grand Challenge, the team collected roughly 650GB of various forms of cartographic data over the months leading up to the race. The bulk of this data was acquired early on through freely available data sources originating from the United States Geological Survey (USGS). This data was in the form of:

Digital Line Graphs (DLG): Vector representations of various layers such as road networks, rail roads or lakes and rivers.

Digital Elevation Models (DEM): A regular grid of elevation samplings available in a geo-referenced raster format called GeoTIFF.

Digital Orthographic Quarter Quads (DOQQ): Overhead imagery also in GeoTIFF format.

As more data sources became available, the early data set was augmented with very high resolution data acquired from two partner companies: Space Imaging and Visual Intelligence. Space Imaging provided high resolution satellite imagery but because of difficulties with ortho-rectification, it was not properly integrated into the mapping database and thus was not used extensively. Visual Intelligence provided narrow swaths of ultra high resolution/accuracy imagery (approximately 20-40cm pixel size) and digital elevation models with 5m postings. Figure 1 shows a comparison between USGS and Visual Intelligence elevation data, while Figure 2 shows a comparison of USGS and Visual Intelligence image quality.

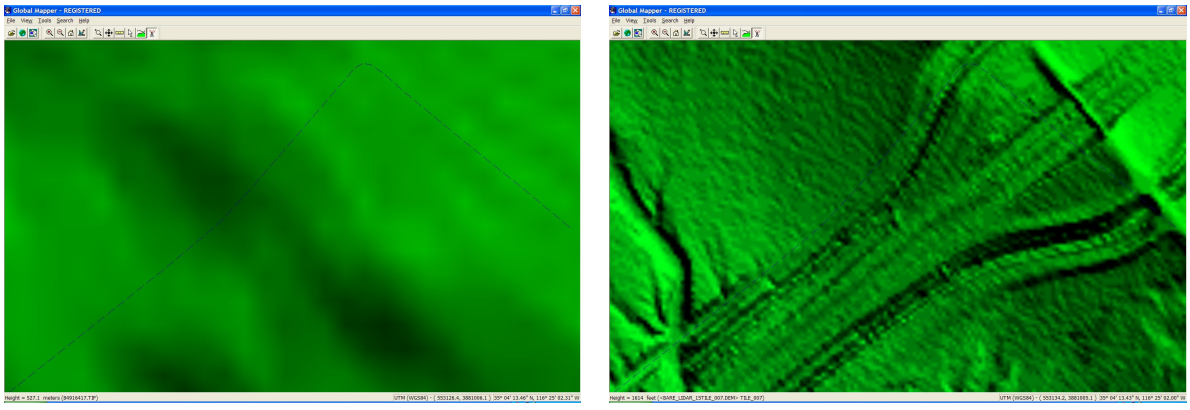

Figure 1. A comparison of USGS and Visual Intelligence DEM data quality.

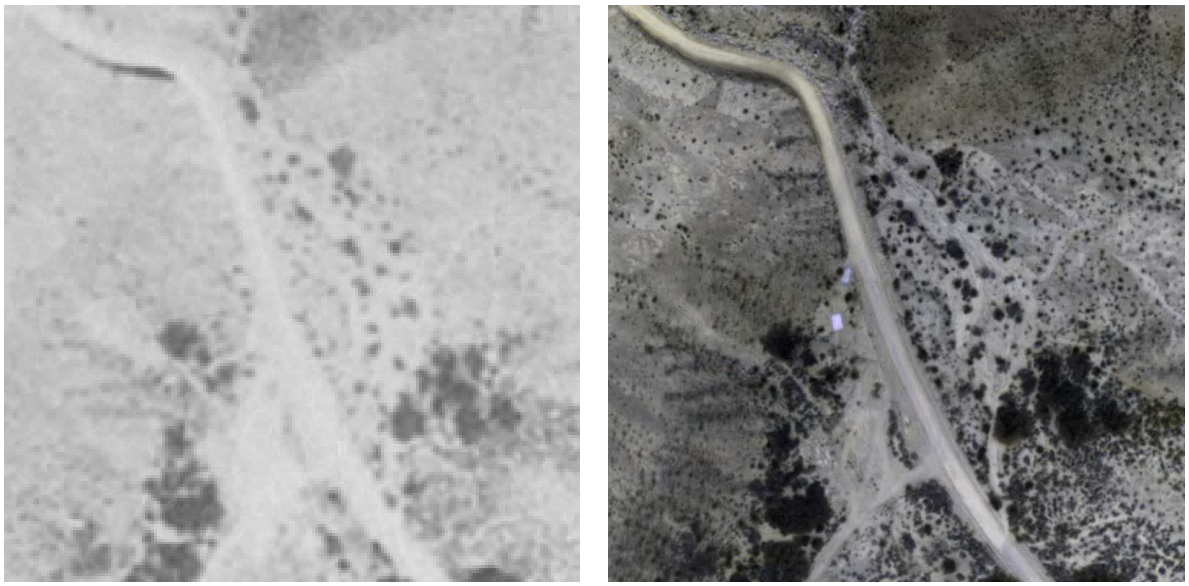

Figure 2. A comparison of USGS and Visual Intelligence image quality.

The final data type utilized in the map database came from ground reconnaissance of the pre-race area. A team drove many of the possible routes, recording GPS traces of the trails and roads with sub-meter accuracy. Since this data set is very accurate, ground truth, location of traversable routes, it is trusted by the planning system more than any of the other data types.

The road network DLGs form the backbone of the map, but are insufficient in themselves as they are not held to specific standards of accuracy. In contrast to this, the DOQQ's are held to national map accuracy standards and are commonly used to correct DLG's. Using the available DOQQ imagery, the team corrected the DLG data throughout the ~50,000 sq. mile region surrounding the Barstow to Vegas corridor. Approximately 1000 man hours were invested in this effort, and although large regions were revised, the coverage of the revisions was not comprehensive enough, nor of sufficient quality to be trusted completely.

Given that it was impractical to improve the mapping database for the entire race region, a new approach was needed. The Grand Challenge rules indicate that the race route would be revealed at least two hours prior race start. In this time, it is possible to correct the narrow ribbon of the data that comprises the race route. This shift from producing a "perfect map" to a producing a "perfect route" required the implementation of a system capable of rapidly delivering the appropriate excerpts of the cartographic data to a cluster of machines running custom route editing software, and synchronizing the revisions of the route from the cluster. At each of the cluster machines, a human editor improves the autonomously generated pre-plan using the full data set available in the database.

In this process the segments, defined by consecutive waypoints in the DARPA specified route, are used as a fundamental work unit. The route revision process begins with an automated planner which processes the route and then uploads way points specifying where Sandstorm should drive within the route corridor. The server then assigns equal length groups of work units to each of the cluster machines. As each of the 14 human route editors progresses through their assigned segment, the route editing software uploads the revised work units to the server which maintains the latest composite route.

Periodically, the server generates an estimate for the race completion time. This estimate is crucial in that it allows the team supervising the pre-planning effort to understand the characteristics of this particular race and to formulate an overall race speed strategy. Given the completion time estimate, the supervisory team is able to scale speeds in the route as appropriate to achieve a target elapsed time for navigating the route. The software that performs the scaling is intelligent enough to avoid breaking speed limits either defined in the route definition file or those imposed internally by the team. This capability allows the team to set a reasonable pace for the race that may be slower than the individual speed recommendations from the human route editors, resulting in a route that is safe and consistent with the team's race strategy.

Infrastructure

To store and transport the vast amounts of data between the central repository and the route editing workstations, it is necessary to utilize a high performance file server in combination with a gigabit network. This system is implemented using a two terabyte RAID5 array attached to a dual Xeon processor based server. The server runs Linux and Samba to allow for easy management and development while providing file serving to the Windows based editing cluster.

The massive volume of data passed between the server and cluster machines is sufficient to saturate a gigabit network, if implemented naïvely. To prevent this, a local image cache is implemented on each client machine. Due to the intensity of pre-race training, on race-day, most of the client machines already had local copies of all of the imagery required for editing, greatly reducing the network burden.

To allow access to the massive data set, a standardized file organization is utilized and the various data formats are aggregated into a Geographic Information Systems (GIS) aware database. An XMLRPC server provides a front end for the client cluster, allowing the route editing software to query the database for the availability of a particular form of cartographic data in a given region. The database then replies with the standard location for the relevant files in the archive. The client can then check its local cache for the file, resorting to the network server only when necessary.

Automatic planning

The automatic planner uses the route definition file provided by the race organizers in conjunction with elevation maps, road vector data, and GPS traces, to output an optimized route with respect to travel time and implicit safety. The various data sources are sampled to a 1m grid which creates a map over which the planner can operate. DLG

data, combined with pre-recorded GPS traces are used to provide six base tiers of cost values. From lower to higher cost, the tiers are:

1- Highway or wide road **with** GPS trace
2- Narrow road **with** GPS trace
3- Off-road **with** GPS trace
4- Highway **without** GPS trace
5- Narrow road **without** GPS trace
6- Off-road **without** GPS trace

This tiering explicitly represents a higher confidence in areas that have been previously traveled by a ground reconnaissance team. Cost values for each tier are set based on the expected time to travel 1 meter of that terrain type.

To keep Sandstorm close to the center of the corridor, cost is increased as a function of the distance from the center of the corridor. Similarly, in order to stay away from unsafe areas of the terrain, cost is increased as a function of the slope of the terrain. The original tiers are used as limits on how much the cost can be increased, such that the final cost still remains within the same cost tier. For example, the planner will never generate a path on "flat" off-road terrain instead of on a sloped, known road.

Planning the path

To generate the routes, a modified wave front planner is used. A wave front planner is utilized since it has low memory requirements and because the search space is frequently restricted to corridors with a large aspect ratio. These two facts essentially eliminate the benefits gained by using a heuristically guided search, such as A*.

The planner incorporates a small modification that breaks ties between neighboring cells of equal cost by preferring those that are closer to the line connecting the start to the goal. In the race relevant planning domain this helps reduce problems associated with 8-connectivity when planning in open areas. Since the cost values used for planning the path represent the time required to travel each unit of area, the resulting path is a path that optimizes the expected travel time.

Because the route is defined by corridors that the planner must stay within, rather than waypoints that need to be reached, the planner generates paths for groups of three waypoints at a time. If there are four consecutive waypoints P_1, P_2, P_3, and P_4 with associated corridor widths, the planner first plans a path from P_1 to P_3. This path must remain within the corridors between P_1 and P_2 and between P_2 and P_3. A new point is now generated, P'_2 which is the point on the path closest to P_2, as illustrated in Figure 3. The segment of the path from P'_2 to P_3 is then discarded and the planner is then run again from P'_2 to P_4, with the constraints that the path must remain within the corridor from P_2 to P_3 and from P_3 to P_4. By iterating this process over all of the waypoints in the route, an efficient path is generated for the entire race route.

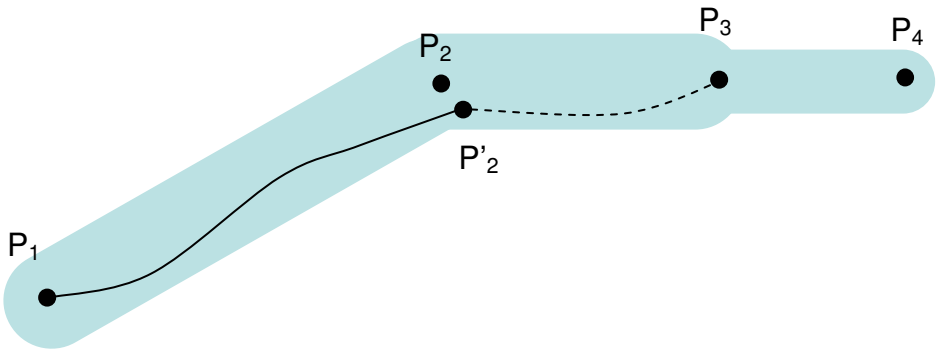

Figure 3. An illustration of the operation of the global path planner.

Vectorizing the Path

The path calculated by the wave front planner is a sequence of points on a regular one meter grid. This dense path representation is not easy for human editors to verify or modify. Additionally, the resolution of the ground truth GPS traces used as input for the planner may be on the order of 10 cm; resolution that would be lost if it were sampled to a one meter grid. To avoid these problems, the path is vectorized, allowing a sparse representation of segments where this is possible and a finer representation where necessary.

To perform this data reduction, consider the points that make up the race route. If a group of consecutive points can be represented by a straight line with a mean square error of less than 2 meters, then the group of points is replaced by the end points of the line. For example, in a one kilometer straight road this change of representation reduces the number of points from 1000 to two. The vectorized representation is more suitable for human editing, since manipulating the endpoints of the resulting vector implicitly changes the location of the thousand points in the original path.

In order to increase the accuracy of the resulting path for segments with GPS traces, the algorithm considers groups of points that lay on or near the GPS traces. These groups of points are then replaced by the original GPS trace as long as they stay within one meter of the planner output. Since the GPS traces can also have a large number of more or less redundant points, they are also vectorized, but with a tighter error tolerance of 20 cm. The resulting path is a vector path, containing few points to describe straight segments of the road, and preserving the resolution of the original GPS traces.

The autonomously planned route may contain errors due the limited representation used. Human editors are generally also able to extract subtle cues regarding the terrain traversability from the various image data formats, allowing the route to be tweaked and improved. To do this the human editors use a custom developed route editing tool.

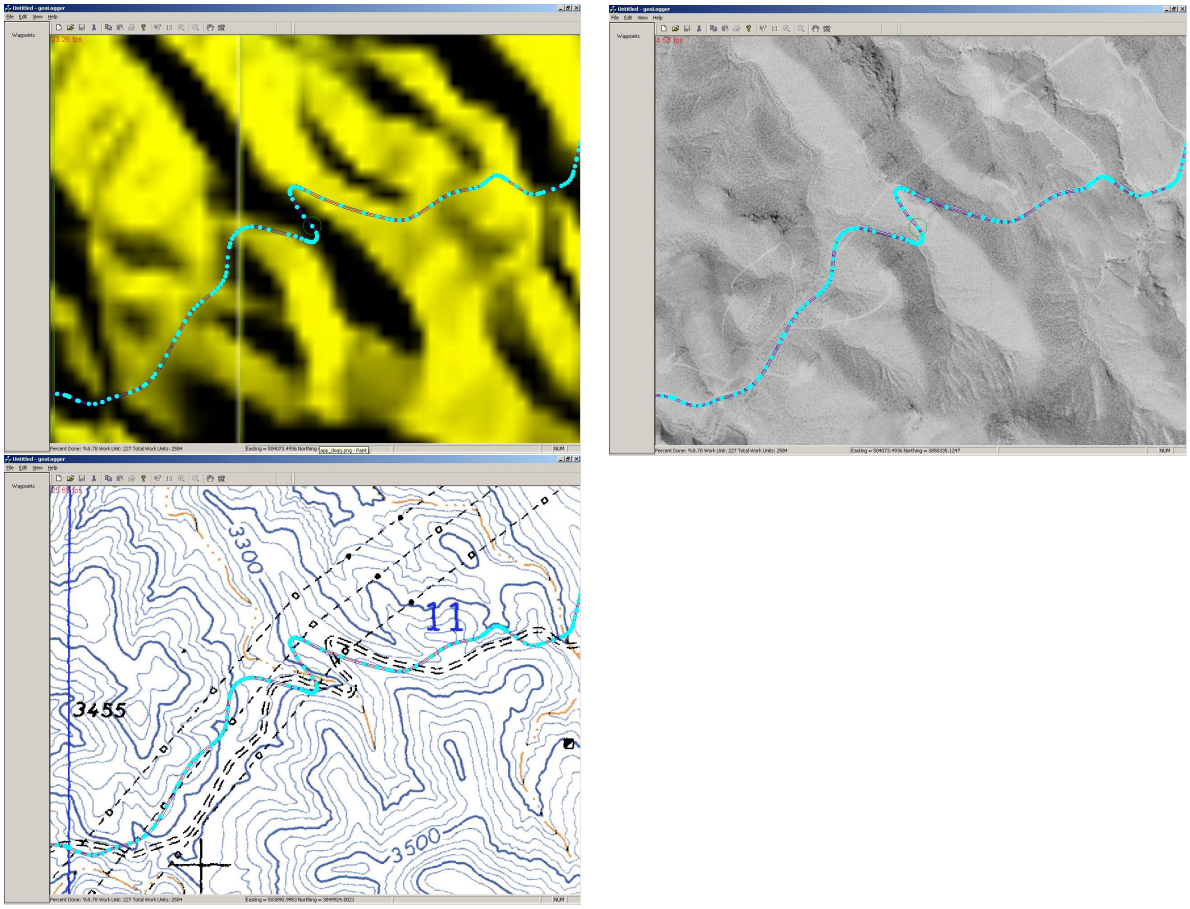

Figure 4. Three views of data available to the route editors.

Route Editing Tool

The interface for the route editing tool was derived from an SAIC tool created to perform a similar role [6]. The original software provided much of the low level capabilities required to interface with GIS data sources. The interface is simple: it provides a mechanism to view the route in the context of all of the cartographic information, and then change the route as necessary. To do this, the software connects to the central database server, and downloads the pre-planned route and imagery associated with the current segment of the route the user is editing. Figure 4 shows three of the cartographic data sources in the route editing tool. The top left image shows an overlay of the route (cyan line) over DEM data. The top right image again shows the route, but overlaid on satellite imagery. The bottom left image shows the route overlaid on a DLG based data product. Note that though the route is intended along the path indicated in the data source, there is a significant offset between the route and the path due to registration error in the data source.

The route is represented to the user in two ways, first as a set of linear segments and secondly as smooth spline. In addition to this information the application shows the corridor as defined by DARPA, a safety corridor (which is 2 meters narrower than the course corridor) and any GPS traces that were collected in the area. Each data layer can be toggled on or off, and each data type is displayed in a customizable color.

To create executable routes for the vehicle, it is desirable to create paths that are smoothly varying. A specialized spline was developed to generate smooth paths with an intuitive user interface, which was important given the small amount of time available to train the route editors. To be "intuitive", the spline should:

- Intersect all of its control points.
- Localize the effect of moving a control point.

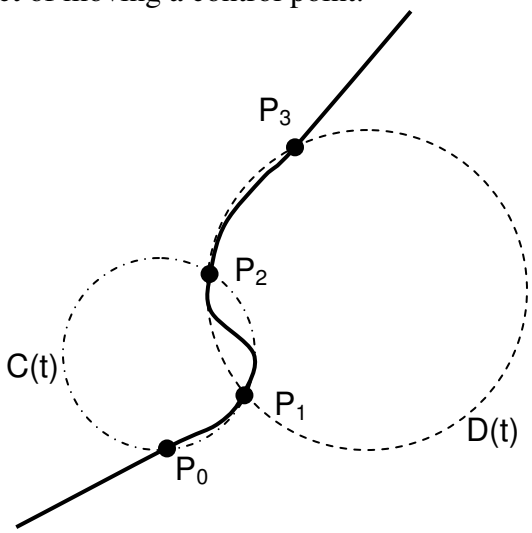

Figure 5. An illustration of how splines are constructed for the route editing tool.

The spline that best met these criteria is a weighted interpolation of arcs. Given an ordered set of 4 points, there are two sets of three consecutive points which can define a pair of circles, as shown Figure 5. The equation of these circles being C(t) and D(t) with the constraints that:

$$C(0) = P_1$$
$$C(T) = P_2$$
$$D(0) = P_1$$
$$D(T) = P_2$$

For the segment where the two sets of points overlap, the spline is calculated as the weighted sum of the two curves:

$$S(t) = \frac{t}{T} C(t) + \frac{T-t}{T} D(t) \quad (0 \le t \le T)$$

In the degenerate case at each end of a list of waypoints, the second circle is replaced by a linear fit between the remaining two points.

To edit the path, the user either selects and moves control points (i.e.: the endpoints of the linear segments), or clicks somewhere along a linear segment to insert a new control point. A feature, called "Snap to GPS" moves the currently selected points onto the closest GPS trace. Associated with each point is an editable speed that represents the

maximum speed the robot should travel at that point. Also displayed for each point is the DARPA mandated speed limit for that segment of the route. The application provides an extensive set of keyboard shortcuts, to allow for efficient user operation.

The "Snap to GPS" implementation calculates the closest path segment to the point being edited. A naïve way to perform this operation would be to find the closest point after performing a search through all 200,000+ segments. Though the calculation is relatively simple, this would not operate in real-time. To achieve real-time performance, an efficient search is needed.

The application utilizes an axis-aligned bounding box (AABB) tree algorithm to implement an efficient search. The AABB approach provides a fast search and is easy to implement. An AABB is simply a box defined by the minimum and maximum extents of the contents it represents. To build the AABB tree, the GPS trace is broken in to sets of 100 adjacent nodes. Each set is a leaf in the AABB tree. The second layer of the tree contains the pairs of adjacent leaf nodes; each additional layer is built by recursively combining pairs of adjacent nodes. Searching this tree is very efficient, a point is recursively compared to the nodes in the tree, if the point falls in the bounding box defined by a node, the search descends that branch, otherwise a comparison is made to the second branch, if the point is in neither box, the search terminates, indicating no node is near by. At the leaves of the tree, the closest node is found by linearly searching through the 100 segments.

The ability to rebalance the route editing work load in real-time is an important capability for a system of this type. To achieve this capability, the route is divided up into several discrete work units. Each work unit is comprised of one or more points. Each work unit has two main pieces of information: an array of all the points in that work unit, and an array of indices specifying the order of the points.

With this framework, insertion and deletion are simple operations to perform. New points are inserted into the rear of the point array. Then, the index of this point is inserted into the proper place in the index list. To delete a point, simply delete the index from the indices array. The overhead of keeping old points around is minimal, as both arrays are refreshed if the application receives any updates to a work unit. To enable efficient insertion of new points, the AABB tree data structure is also used to represent the path. Through this mechanism, a central control station is able to add and remove work units, which are partitions of the pre-planned route, from each of the clients.

This hardware and software combined with an intensively trained team of editors provides the a priori knowledge essential for Sandstorm to perform well during race operations.

Vehicle Hardware

To maximize the probability of success in the Grand Challenge it is important to have great software and a forgiving vehicle. The best mobility platform for a Grand Challenge robot combines reliable, robust terrainability with agility and sufficient payload capacity

to support the large amount of sensing and computing hardware necessary. Toward these goals, Sandstorm is built around a 998 military High Mobility Multi-purpose Wheeled Vehicle (HMMWV). This foundation provides a very capable platform able to traverse a wide variety of off road terrain. Table 1 highlights some of these capabilities.

Table 1. A summary of HMMWV capabilities.

Ground Clearance	40 cm
Approach Angle	72º
Departure Angle	45º
Breakover Angle	32.5º
Sideslope Capability	40%
Gradability	60%
Turning Radius	7.4m
Payload Capability	> 1350 kg

Actuators and enhancements were appended onto this platform to make it amenable to computer control and to improve upon its basic capabilities.

Electronics Box

The electronics box encloses all of the computation and power control components and acts as the common inertial reference frame for the onboard sensors. The box was designed to minimize the connections between the host vehicle and the navigation "brain". By containing all of the intelligence and power distribution in a single unit, a replacement HMMWV could be rapidly automated should the original vehicle become irreparably damaged or should a better vehicle platform become available.

The electronics box uses an open, forced air system for cooling. Investigation of the expected race day temperatures, and temperatures leading up to race day indicated that forced air would be sufficient, and considerably simpler to implement, than a closed air-conditioned system. Initial analysis predicted a 4000W thermal load that would need to be removed from the box. A brief thermal analysis showed that, given a 24ºC ambient air temperature, this heat could be transferred to the air with roughly a 5ºC temperature increase if 1100 cubic feet/minute of air is moved through the box. This specification allowed the computers to operate within their nominal temperature range under the expected race conditions. Additional fans are used to overpressure the electronics enclosure to reduce the intake of dust and other contaminants.

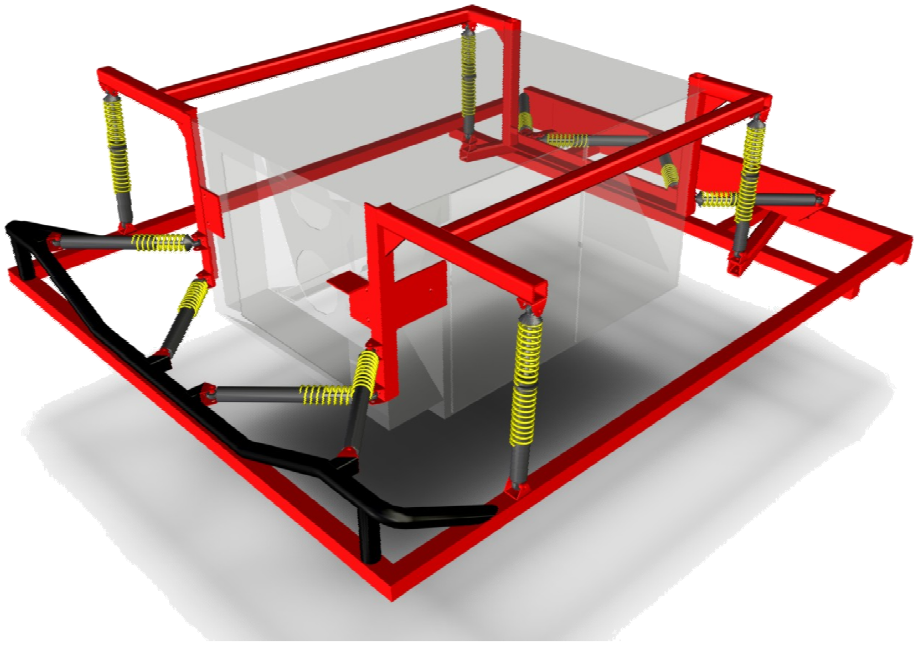

Figure 6. The arrangement of shock isolation components that support the electronics box.

The electronics box is mounted to the vehicle platform through a set of coil over damper shocks. The suspension lowers the natural frequency response of the electronics enclosure so that sensors can be rigidly mounted to it and still provide useful measurements of the environment. It also minimizes the shock dose to the computers and electronics so they are protected from severe accelerations. Figure 6 shows the configuration of shock isolation components that support the electronics box. The orthogonal configuration of the shocks decouples the vertical and horizontal forces, simplifying the tuning of spring and dampening constants. It also allows translations in all directions without inducing any rotations. The suspension configuration leaves the driver cockpit area open, which is important for logistical reasons.

Computation

Sandstorm's computing was designed to ensure that computational power would not limit performance. Sandstorm's onboard computing includes a quad processor Itanium II, three dual processor Xeon rack mount computers, and four Pentium III class PC-104 computers.

The Itanium II processor was intended to be used to perform faster than real-time dynamic simulation of Sandstorm's motion over terrain. The simulation would be used as the feed-forward portion of a randomized dynamics cognizant planning algorithm [19]. Unfortunately, there was insufficient time to implement this planning scheme and so the Itanium II was not used in the race day configuration.

The dual Xeon computers interface with Sandstorm's sensor suite, and provide processed data products to be used by the terrain evaluation and path adjustment process(es). The

various sensor interfacing processes were distributed across these machines to balance I/O and processor bound tasks where possible.

The Pentium III class PC-104 computers are used to perform embedded control tasks throughout Sandstorm and as an interface to the high speed stereo vision system. For each of the embedded control tasks the same set of processor and input/output boards were used. This facilitated the reuse of code and the sharing of software implementation knowledge among systems.

These eight computers and the various Ethernet interfaced sensors are connected using bridged, gigabit Ethernet. By utilizing a gigabit Ethernet in a bridged, star configuration, any point-to-point high bandwidth communication does not effect throughput or latency between other nodes on the network. In this way, the throughput of the network is maximized allowing for the efficient transfer of information between processes in the system.

Sensors

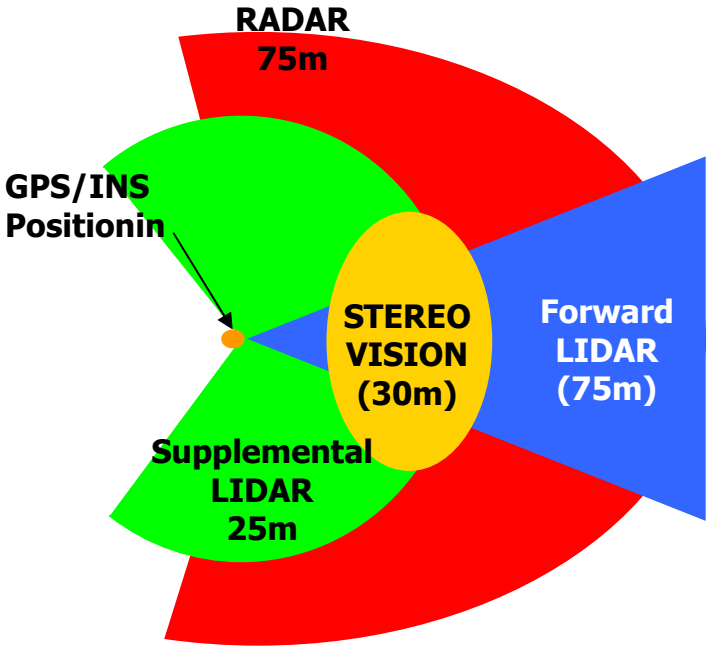

Figure 7. The overlapping fields of view of sensors on Sandstorm.

Sandstorm was designed to provide all weather sensing and driving capabilities. To achieve this goal, it is necessary to incorporate a variety of sensing modes with overlapping ranges and capabilities. Sandstorm incorporates a variety of LIDAR sensors, stereo vision and RADAR and a combined inertial/ and differential GPS sensor for pose estimation. Figure 7 illustrates the overlapping fields of view of each of these sensors. The Riegl LIDAR is used to profile the terrain at long range, while the stereo system was intended to provide short to middle range dense terrain modeling. Short range SICK laser scanners are used to provide secondary obstacle detection, and

potentially to assist in modeling underpasses and nearby walls in constrained operations. A RADAR was included in the sensing package to provide sensing in dusty conditions and to assist in the detection of robots and other moving vehicles. Figure 8 shows the locations where these sensors are mounted on Sandstorm.

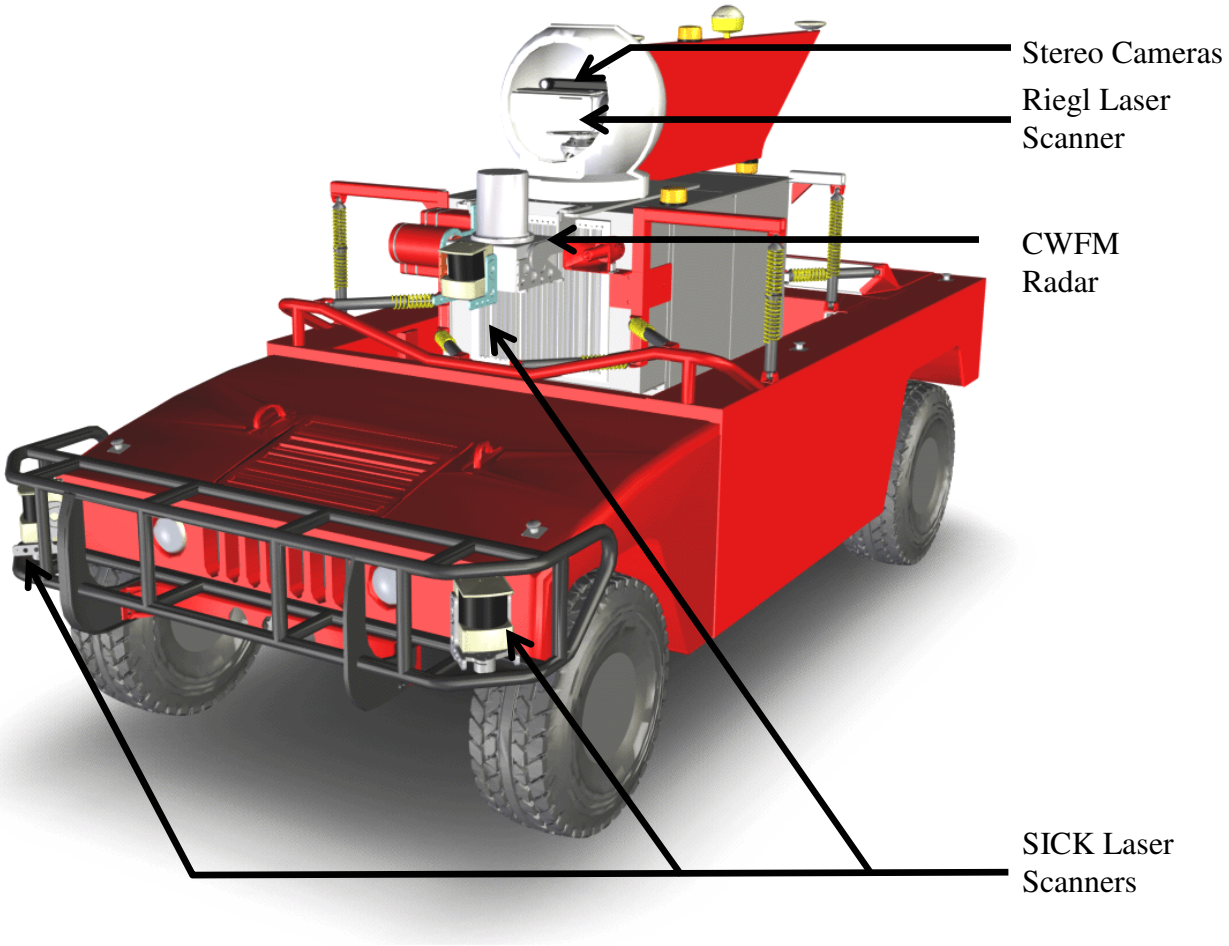

Figure 8. A model of Sandstorm showing the placement of sensors.

LIDAR

A Riegl Q140i scanning laser range finder was selected as the primary sensor, since it can provide a model of the terrain at relatively long ranges, is easy to integrate and has few, well understood failures modes. A limitation of scanned LIDAR is that it is generally only possible to collect dense point data in a single plane. Flash LIDAR does not have this problem, but is still too short range to be useful at speed. Two axis mechanically scanned LIDAR have reasonable range, but cannot scan rapidly in both axes and thus don't provide much benefit over a single scanning plane for this application. The Riegl LMS Q140i Airborne line-scanner that Sandstorm uses, operates with a 60-degree field of view, 30KHz pixel rate, and has a specified line-scan period of 20ms (50Hz).

Each Riegl line-scan paints a straight-line series of pixels along a horizontal path with respect to the robot's forward motion; successive line-scans (starting at –30°, ending at

30°) paint a series of stripes at forward intervals determined by the robot's forward velocity. The scanner is stabilized and can also be pointed by the gimbal.

Three SICK LMS laser scanners are used to provide short range supplemental sensing. Two are mounted in the front bumper, providing low, horizontal scans over a 270° wedge centered in front of the robot. These sensors can be used to detect obvious, large positive obstacles. A third SICK LMS laser scanner is mounted to provide profiling of over hanging obstacles, and to detect airborne dust.

Stereo Vision

To complement the low density, long range stereo vision system, Sandstorm incorporates a high speed stereo vision system. The stereo system, provided by SAIC, utilizes the DeepSea Stereo engine which is a hardware implementation of the CENSUS disparity matching algorithm [20]. The system is capable of achieving frame rates of ~120Hz on 512x512 images. The entire stereo vision processing unit comprising of a Pentium 3 based PC104 stack and the stereo engine are mounted in a ruggedized aluminum casing on the electronics box.

The stereo vision system processes images generated by a stereo camera pair mounted on the stabilized gimbal. The stereo system is therefore pointable and can be directed to look at particular areas of interest.

RADAR

Both Stereo vision and LIDAR can have difficulties sensing in dusty environments. RADAR, on the other hand operates at a wavelength that penetrates through dust and other visual obscurants but provides data that is more difficult to interpret. Objects are detected by finding amplitude peaks in the frequency shifted return signal. The amplitude of the signal can be used to estimate the size, and thus significance, of an object. This process can be confounded due to surface properties of the object, and the orientation of the object relative to the transceiver.

To fill the role of complementary sensor, the NavTech DS2000 Continuous Wave Frequency Modulated (CWFM) radar was selected. The DSC2000 provides 360° scanning, at 2.5 Hz and provides an Ethernet interface to the RADAR range measurements. The DS2000 system utilized on Sandstorm has a specially designed antenna that generates a beam that is tilted down 3.22°. This down angle was selected to cause the 4° vertical beam width to cover a range from roughly 24-100m. The RADAR was not integrated with the primary navigation system due to difficulties extracting noise free data.

Pose Estimation

Reliable and robust position sensing is essential. The implementation of position sensing is a major undertaking that can drain valuable development resources. To avoid this problem, Sandstorm uses an off-the-shelf pose estimation system. The Applanix POS-LV provides position estimates by fusing inertial and differential GPS position estimates through a Kalman filter. The output estimate is specified to have sub meter accuracies,

even during extended periods of GPS dropout. The POS-LV system also provides high accuracy angular information, through carrier differencing of the signal received by a pair of GPS antennas, and the inertial sensors. The POS-LV system outputs a pose estimate over a high speed serial link at a rate of 200 Hz. This constant stream of low-latency pose information simplifies the task of integrating the various terrain sensor data sources.

The Gimbal

Both the Riegl LIDAR and the SAIC stereo vision systems represent the state of the art in high-fidelity terrain characterization sensors. The ability to interpret data from these sensors can be severely hampered by pitching and rolling induced by robot motion over terrain. The passive stabilization system discussed previously is somewhat able to smooth these motions and reduce the principal frequency of terrain inputs, but is unable to completely remove the disturbances.

The performance of a single axis scanning LIDAR is particularly affected by mechanical excitation in the pitch axis. When sensing at reasonably long ranges, even small pointing errors can result in a dramatic change in the location where the sensor's beam intersects the terrain. Because of this, range data associated with small obstacles or terrain details at distance become ambiguous, and the overall perception performance is severely degraded and convoluted with interdependencies on terrain dynamics, vehicle speed, chassis and sensor-mount characteristic responses. Active attenuation of the terrain excitations via a gimbal reduces this effect, yielding interpretable terrain range data.

The pitch, roll and yaw axis definitions are relative to a coordinate system attached to the electronics box, not a fixed reference to the chassis or to world coordinate space. Under these constraints the pitch, roll and yaw axes cannot be viewed as independently stabilized entities and in fact are relationally interdependent. Since stabilization is relative to the world coordinate frame, the axes are only completely decoupled when system coordinate frame is exactly aligned with world coordinate frame.

This condition explicitly requires a stabilization scheme that utilizes all three orthogonal axes in order to realize full stabilization of any single axis. In general, sufficient stabilization can be achieved by actively stabilizing the combination of the roll and pitch axes.

Implementation

The relatively low-mass active LIDAR and passive stereo vision sensors are co-mounted to provide a common field of view and reference frame. The sensor mounting design aligns the LIDAR's optical aperture with the center of the gimbal and balances the mass distribution around the rotational center. Each axis includes minimal-mass components and the simplest possible gimbal support structure with the design goal of minimizing the moment of inertia. By minimizing the moment of inertia, the overall responsiveness of the gimbal is increased, yielding better perception data.

The pitch axis is most critically effected by vehicle excitations due to the large lookout distance associated with downfield targets when the robot is moving at speed. The roll

axis is less critical to perception data continuity and generally effects the 'tilt' at which the LIDAR pixel scan-line illuminates the terrain in front of the robot. The yaw axis represents the least critical influence on the usefulness of data collected, since the field of view of the sensor is wide enough that any reasonable excitation would not move the terrain being characterized out of view. This understanding leads to the priority in which the moments of inertia should be minimized: pitch then roll, then and finally yaw.

Harmonic drive (HD) actuators controlled by PID servo loops are used to actuate each of the axes. Each axis also includes incremental and absolute position encoders, a fiber-optic gyro (FOG), and an optical switch based limit detection sensor. Each gimbal axis assembly is designed for electrical and mechanical simplicity. To achieve this goal common design and components are used on each of the axis. The aluminum bracket components are designed to have the minimal mass and moment of inertia about their respective rotational axis while still maintaining sufficient strength and stiffness.

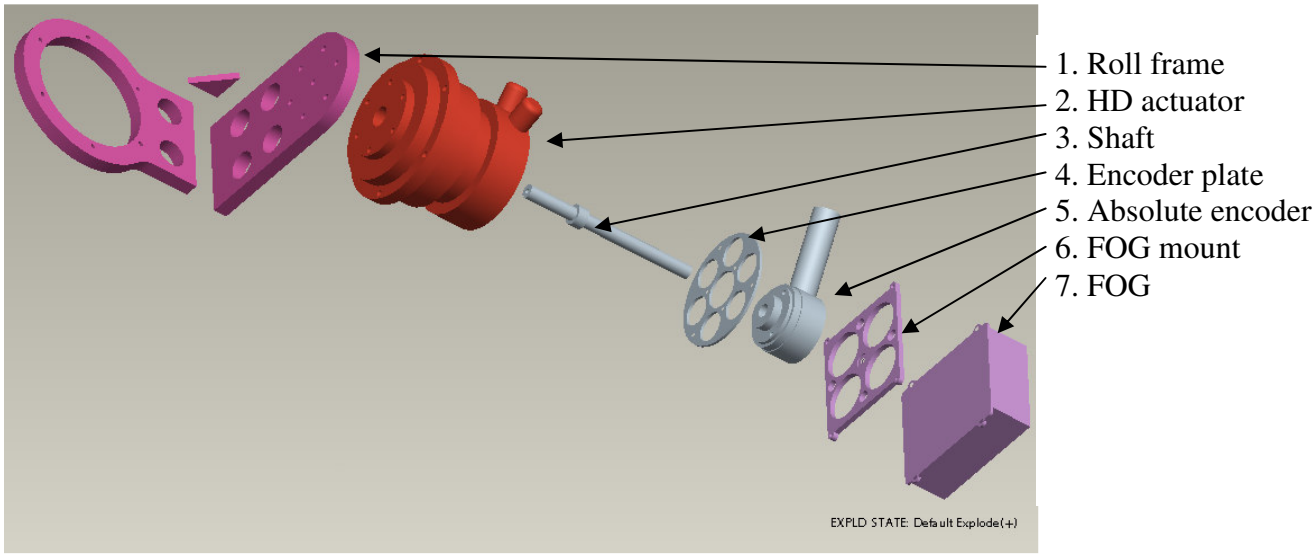

Figure 9. An exploded view of the gimbal roll axis.

Each HD actuator output shaft is attached directly to an aluminum frame using a five-bolt pattern. The frame, shaft and FOG mount are bolted together to form a rigid assembly, thus all of the components rotate in sync about associated axis. The absolute encoder body is mounted and fixed relative to HD actuator housing via an encoder plate using six machine screws. The absolute encoder stator is attached to the rotating axis shaft via a friction-ring and machine screw. This arrangement for the roll axis can be seen in Figure 9.

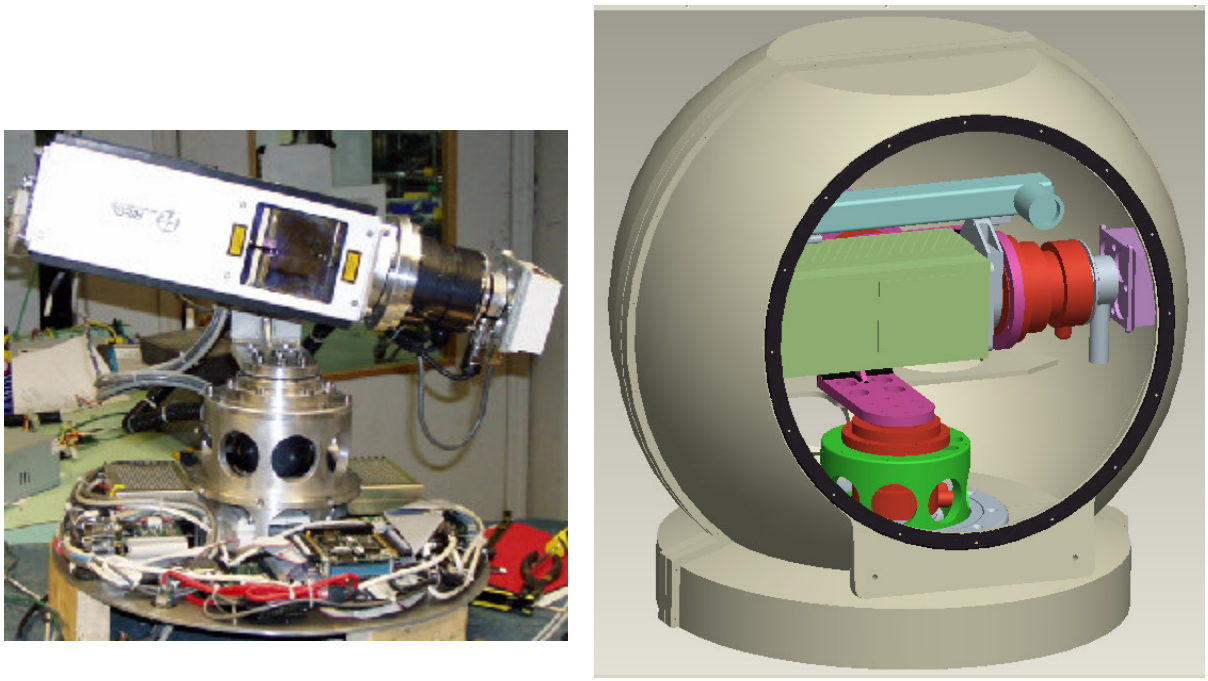

Figure 10. The fully assembled gimbal.

The entire gimbal mechanism is enclosed within a protective shell. The shell prevents water and dust from damaging the mechanism and electronics. The sensors operate through a front window with specially coated optical glass. The fully assembled gimbal is shown in Figure 10. Table 2 outlines the gimbal performance capabilities and summarizes its interfaces.

Table 2. A summary of the gimbal performance characteristics.

Payload Compliment	High Resolution LIDAR line scanner and stereo vision head
Payload Dimensions	24mm x 25mm x 500mm
Payload Mass	12+ Kg
Gimbal Mass	< 25 Kg
Computation	Pentium III PC104 stack
Communication Interface	100Base-T Ethernet
Power Consumption	24VDC @ 50W 85VDC @ 350W
Enclosure	Water resistant lightweight shell with optical window with pass for 300–900nm
Mechanical Mounting	10 bolt pattern in base plate
Pitch	
Range of Motion	± 40°
Angular Velocity	6.28 Rad/s
Angular Acceleration	863.63 Rad/s^2

Roll	
Range of Motion	± 40°
Angular Velocity	6.28 Rad/s
Angular Acceleration	73.89 Rad/s^2
Yaw	
Range of Motion	± 90°
Angular Velocity	6.28 Rad/s
Angular Acceleration	25.59 Rad/s^2

Navigation Software

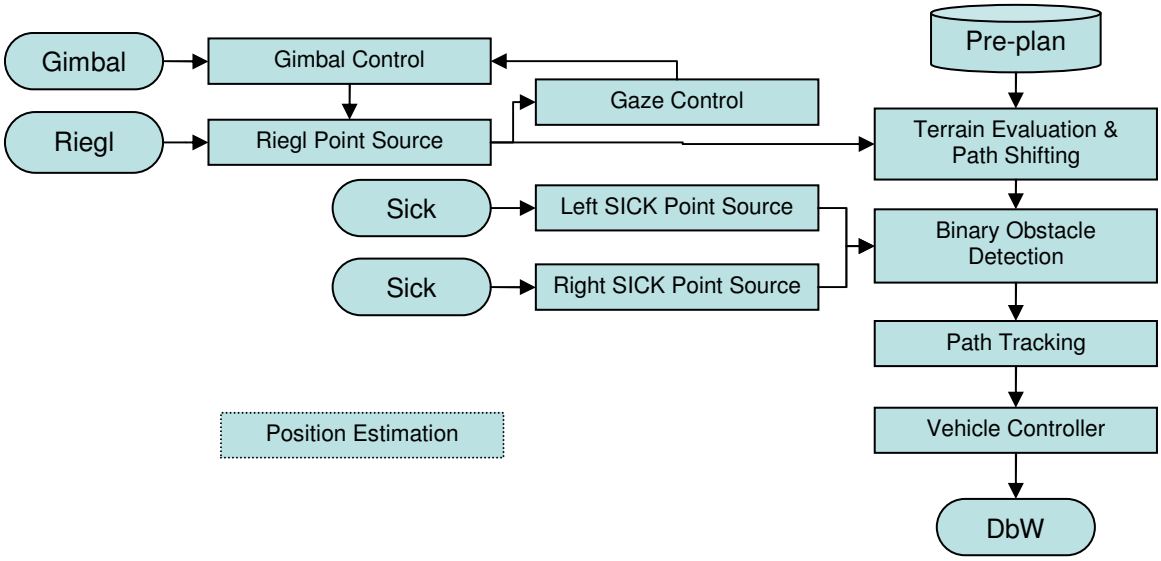

Figure 11. The on-board software architecture.

Overview

The navigation software drives Sandstorm by combining incoming sensor data and the pre-planned path. The pre-planned path is loaded into the terrain evaluation and path shifting program prior to the race start. The path is also passed to the gaze controller which monitors the vehicle's position and independently commands the gimbal controller to point the laser scanner in the direction of anticipated driving. The Riegl point source communicates with the Riegl laser range finder, receiving laser line scans at between 15 and 20 Hz. This range data is then transformed into a vehicle coordinate frame referenced point cloud. This data is passed onto the terrain evaluation and path shifting algorithm which makes adjustments to the path Sandstorm is attempting to follow.

Once the terrain has been evaluated and any necessary adjustments made to the pre-planned path, the modified route is passed to the binary obstacle detection process. This process uses the bumper mounted short range laser scanners to detect if any object in the environment is an immanent threat. If an obstacle is detected, Sandstorm is commanded

to stop. This process is only intended to be run during the qualification phase of the race since the algorithms can potentially produce false positives, and the system is only designed to stop for, not avoid, detected obstacles.

Once obstacles have been taken into account, the resultant path is handed off to a pure pursuit path tracker. The path tracker is tuned to minimize deviation from the route. Curvature and velocity commands are then passed from the path tracker to the vehicle controller which interfaces with the robot hardware. Figure 11 illustrates this data flow. The following sections describe these components in more detail.

Infrastructure

Sandstorm uses architectural and communications tools that were originally developed to support the ongoing robotics research of the NavLab project [15]. The architectural tools allow algorithm developers to view the rest of the system through a set of abstract, reconfigurable interfaces. During initial development and ongoing debugging the interfaces for an algorithm can be configured to read data from time-tagged files using a common set of data access tools. As an algorithm matures, the interfaces are reconfigured to use a common set of interprocess communication tools which integrate the individual algorithm into the larger system.

The vast majority of interprocess communications in the Sandstorm system are analogous to signals in electronics. Examples of such signals include periodic estimates of the vehicles pose or line scanner data. Missing some of this data is unimportant, so long as consumer applications have access to the most recent measurement with low latency. A paradigm for the propagation of signal type information is global shared memory: A producer sets the memory and a consumer reads the most recent value. The Neutral Messaging Library (NML) demonstrates this control-centric method for integrating robotic systems [7]. Sandstorm uses a simpler implementation of global shared memory which has a "single-writer, multiple-reader" model. Processes communicating on the same machine use System V shared memory, whereas processes communicating between machines transparently propagate changing memory values from writer to readers via the UDP socket protocol.

One reason for implementing a simple shared memory communications scheme rather than purely adopting NML is that, while signals make up the bulk of the communications, they are not the only paradigm for interprocess communications in a robotic system. Symbols, i.e., atomic pieces of information, changes in state, or requests for information, are very difficult to communicate via a signal-based communications paradigm. For example, unlike signals, if a symbol value is missed, then information is lost, state changes don't get noticed, and requests are ignored. The guarantee that a symbol has been transported from writer to reader is worth significant additional latency and complexity in the implementation. Symbolic information is typically communicated in robotic systems via TCP/IP message based packages. In order to limit the complexity and size of the software while still providing some abstraction and flexibility, a simple TCP/IP based messaging package called the InterProcess Toolkit (IPT) [9] is used.

A key abstraction built using IPT is the concept of a central black board. Individual algorithms mainly query the black board for their configuration parameters, but can also post information in the black board and watch for changes in values on the black board. Thus, the black board becomes a channel for propagating information through the system that has to be generally available, but for which a certain degree of latency is acceptable. In addition, since the system's information is being funneled through the black board, we have chosen to make the black board manager the system process manager. It initiates, parameterizes, and monitors the system processes. Interestingly, this paradigm of a central black board was one of the earliest used in robotics [10], but it has been often rejected because if the black board is the only means for propagating information through the system, it becomes an intolerable bottleneck for the kind of low-latency, high-bandwidth signal-type information that forms the backbone of information flow for real-time robotic systems.

Gaze Control

The gaze control module points the primary sensors to provide information about the world that a non-pointed sensor would not be able to provide. A non-pointed sensor loses sight of the road when moved over hills and around corners. A pointed sensor can be pitched up, down, left, and right to fill in data that would otherwise be missed.

As shown in Figure 12a, as the vehicle crests a hill, a non-pointed sensor that is aimed to see flat ground at a reasonable distance will be looking out into the sky. Similarly, as shown in Figure 12b, as the vehicle approaches the bottom of a hill a non-pointed sensor will repeatedly sense the bottom of the hill rather than looking ahead up the hill. Immediately before rounding a corner, non-pointed sensors look out beyond the road rather than in the direction the robot is about to travel (Figure 12c).

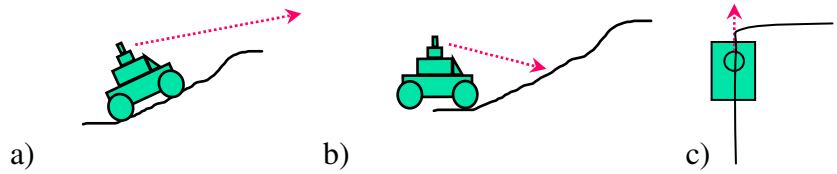

Figure 12. Various configurations where a non-pointed sensor performs poorly.

Control Architecture

Inputs to the gaze control module are the current state of the vehicle, the current pointing of the gimbal, the desired path ahead of the vehicle, and data from the long range LIDAR. Roll and pitch are set to be in stabilized mode, and yaw is set in pointed mode. High frequency pitching and rolling is taken care of by the gimbal control software. High frequency yawing does not occur due to the mass of the vehicle relative to the magnitude of input from the ground.

To make the control of the pointed sensors simple, control of the pitch and yaw are separated. Roll is not pointed. This decentralization generally works well, as it is rare to see hills that are very steep combined with roads that turn sharply enough to prevent this

mode of control from working. The path is fed to the gaze control module as a series of *x,y* pairs in the global coordinate frame. A "look-ahead distance," the distance down the path to look, is chosen based on the current speed of the vehicle. The sensors are pointed so that an obstacle can always be detected in time to stop for that obstacle.

Calculation of stopping distance

Stopping distance estimates of the distance over which the vehicle will stop, based on friction and current velocity. If v is the velocity of the vehicle, μ is the coefficient of friction and g is gravity, the stopping distance, d, is calculated as

$$d = v^2 / (2 \mu g)$$

A point on the path called the "look-ahead point" is chosen by traversing along the path d meters and choosing the path point closest to d meters away. The look-ahead point has coordinates (x_l, y_l) and the vehicle has coordinates (x_v, y_v)

Calculation of desired yaw

Desired yaw of the gimbal relative to the vehicle can be determined through trigonometry. Although it is probably good practice to correct this angle for variation in the terrain ahead, this information is not taken into account. The look-ahead point is first transformed into vehicle relative coordinates (x_l', y_l'). The yaw angle for the gimbal is then

$$\theta_y = atan2(x_l', y_l')$$

Calculation of desired pitch

Desired pitch, θ_p, is calculated as the sum of two factors. First a base pitch angle θ_{pf} is calculated such that on flat ground, the long range LIDAR would hit the top of the minimum sized object at (x_l, y_l). Second, a feedback term, θ_{pc}, based on a window of LIDAR ranges around the path is added to θ_{pf}. Letting minimum obstacle height be h_o, and the height of the LIDAR (which is at the center of rotation of the gimbal) be h_l,

$$\theta_{pf} = -atan2(\ h_l - h_o, \ sqrt((x_l-x_v)^2+(y_l-y_v)^2) \ )$$

Calculation of θ_{pc} is more complicated as it is based on noisy LIDAR data. The set of points within w meters of the center of the most recent LIDAR scan is selected. The mean and standard deviation of the straight-line distance from the center of the vehicle to these points is calculated. Any points more than one standard deviation from the mean distance of the set are discarded and the mean distance of the remaining points, μ_d, is recalculated. This produces an estimate of the distance from the vehicle to where the LIDAR scan intersects the ground. Because this estimate is very noisy, a running average, r_k, is computed.

$$r_k = \mu_d * \alpha + r_{k-1} * (1 - \alpha)$$

The value of r_k serves as an estimate of the actual value of d, the distance to the look-ahead point.

$$e = d - r_k$$

θ_{pc} is updated by adding a term proportional to the error to the current value of θ_{pc}. This is similar to an integral term.

$$\theta_{pc} = \theta_{pc} + K_p * e$$

Finally, the correction θ_{pc} is added to the flat-plane estimation of θ_{pf} to produce θ_p.

$$\theta_p = \theta_{pc} + \theta_{pf}$$

θ_p and θ_y are sent to the gimbal as desired pointing angles. It is important to note that θ_y is relative to the vehicle frame of reference and θ_p is relative to the global frame of reference. This difference is apparent in the calculations for each variable. Calculations for yaw are computed in the vehicle frame of reference while calculations for pitch are computed in the global frame of reference.

Gimbal Control

The three-axis gimbal stabilized platform provides Sandstorm with steady data from the primary perception sensors: the Riegl long range laser scanner and the stereo camera pair. The primary objective of the control loop is to eliminate the effect of terrain inputs that generate vehicle rotations in the global coordinate frame. A secondary goal is to provide a mechanism for sensor gaze control. The complexity of controlling the three axis system was minimized by appending individual inertial sensors to each axis and stabilizing each axis independently. The gimbal control hardware is identical for each axis, simplifying the control loop implementation and allowing significant code reuse.

Interfaces

The gimbal control software accepts vector pointing commands and also accepts commands for how fast the mechanism should servo to this vector. Commands can be issued at up to 10Hz without disrupting operation. By default, the gimbal control software stabilizes all three pointing axes but controlling software can disable this functionality on each of the axes independently. To eliminate gyro drift, the controller incorporates data from the onboard absolute pose estimation system. Periodically, the gimbal control software publishes the current vehicle relative position of the sensor head.

Coordinate Systems

There are 4 coordinate systems to consider in the gimbal control code: local, electronics-box, stabilized, and gyro. In the non-stabilized mode, pointing commands are specified in the local coordinate frame, which is a fixed frame, located at the center of the gimbal. The electronics-box frame is the same as the local frame, modulo translations that move the reference frame to near the IMU in the electronics-box. While stabilizing, pointing commands are issued relative to a stabilized coordinate system, which is coincident with

the local coordinate frame, but the z axis of this frame is always aligned with the gravity vector. Ideally the gyro coordinate frame would be identical to the stabilized coordinate frame, but due to gyro drift, the two are rarely aligned. The gyro frame therefore, is the stabilized coordinate system as measured by the gyros.

Internal States

The gimbal control code has 4 primary modes and 3 transitional modes for each axis. The primary modes are stabilized, unstabilized, marginal and emergency. Stabilized and unstabilized correspond to whether the gimbal is actively stabilizing that axis, or is just providing a pointing interface.

The marginal mode is engaged when the gimbal is about to exceed its safe operating range. The controller enters the marginal run-mode when the pitch angle is outside of the range of -32º to 25º, or the absolute value of the roll angle is greater than 18º or the absolute value of the yaw angle is greater than 80º. When in the marginal mode, control clips the commanded pointing angles to prevent the system from damaging itself. The emergency mode is triggered if the gimbal's inertia, or a software failure, allow it to somehow exceed these soft limits by more than 2º. In this mode, solid state relays are automatically tripped which de-energize the motor amplifiers. At this point a recovery procedure is executed, where each axis is re-zeroed before the system returns to nominal operation.

System and Control

Figure 13 shows a block diagram of the gimbal control system which operates at 1000Hz, limited by the gyro data rate. In stabilized mode, the difficulty of the implementation arises since random shocks can be introduced from the outside world at random timings and in random combination, which can at best be modeled statistically. The passive shock isolation stages soften these shocks, but do not eliminate them. Figure 14 shows a control model of this system, from which the transfer function of the system can be generated.

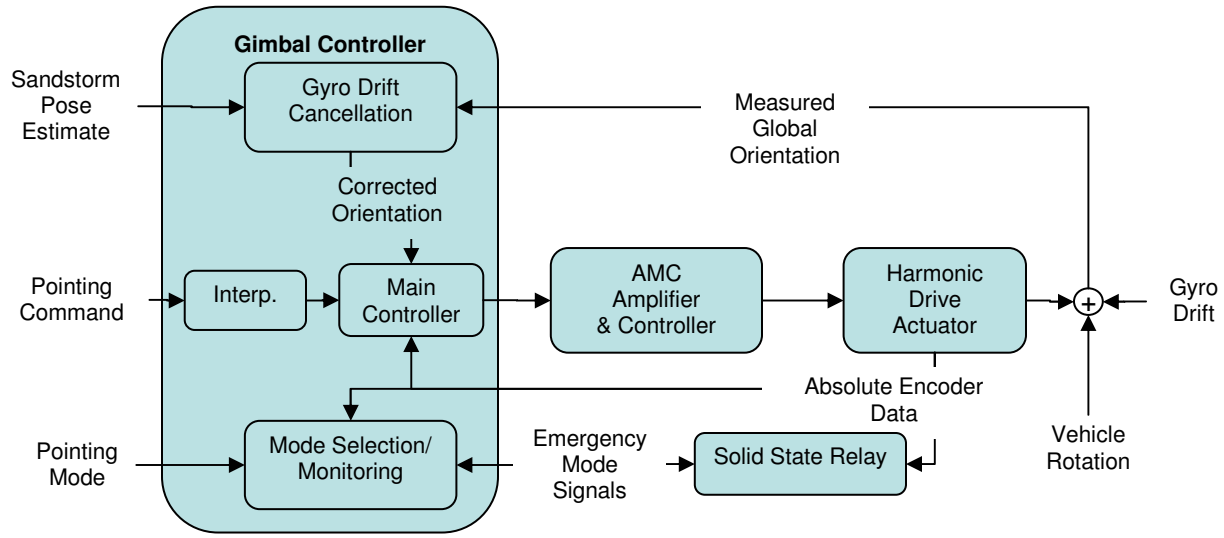

Figure 13. The gimbal control system.

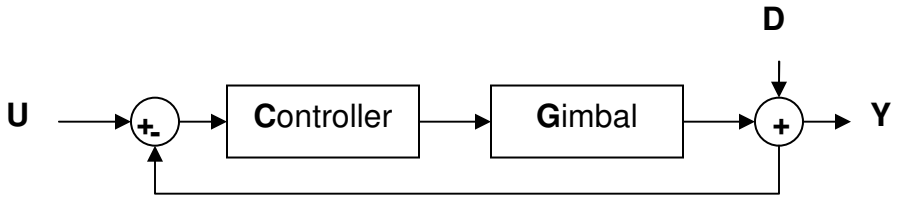

Figure 14. A model of the gimbal control system.

$$Y = \frac{1}{1+CG} D + \frac{CG}{1+CG} U$$

From this equation, it can be seen that to minimize the effect of disturbance, it is necessary to maximize the CG term at frequencies where the disturbance is acting. The frequency response of the gimbal was determined using a series of step inputs. The poles and zeroes of this function are plotted as the blue 'X's' and 'O's' in Figure 15.

The main controller is a lead-lag controller with an integrator. The bode plot and root locus for this controller are shown in Figure 15. In addition to the integrator, the pole-and-zero set for lag are used to increase the magnitude of the open-loop frequency response at low frequencies. The zero-and-pole set for lead increases the phase margin at the crossover frequency.

With this control system, it is important to not saturate the controller command input since this prevents further reaction in that control direction. The controller is optimized to minimize this problem, and in practice behaves well. A second consideration is that pointing the gimbal requires frequent reference command changes. Hence, a reasonable amount of phase margin is required in order to prevent large overshoots. However, the desire for a large phase margin must be balanced against the need to aggressively

counteract any terrain inputs. This tradeoff required a fair amount of careful on-board tuning to achieve. Figure 16 shows the resulting attenuation of input disturbances as the control loop operates nominally.

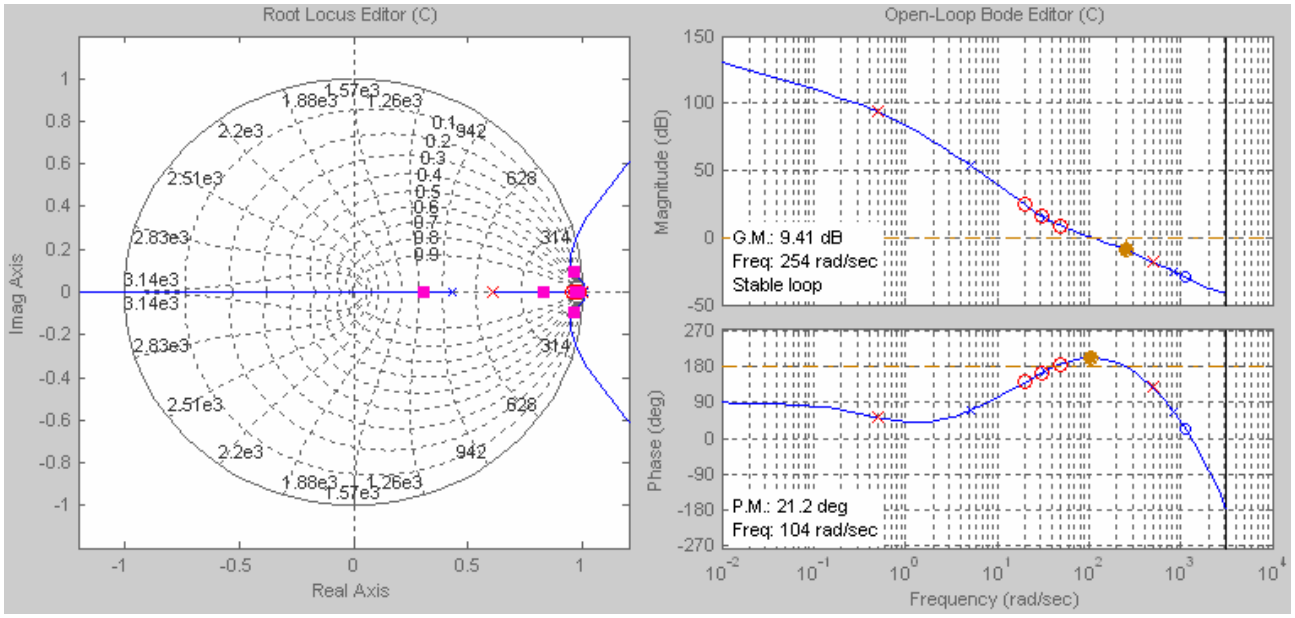

Figure 15. The Root locus and Bode plot of the gimbal control system.

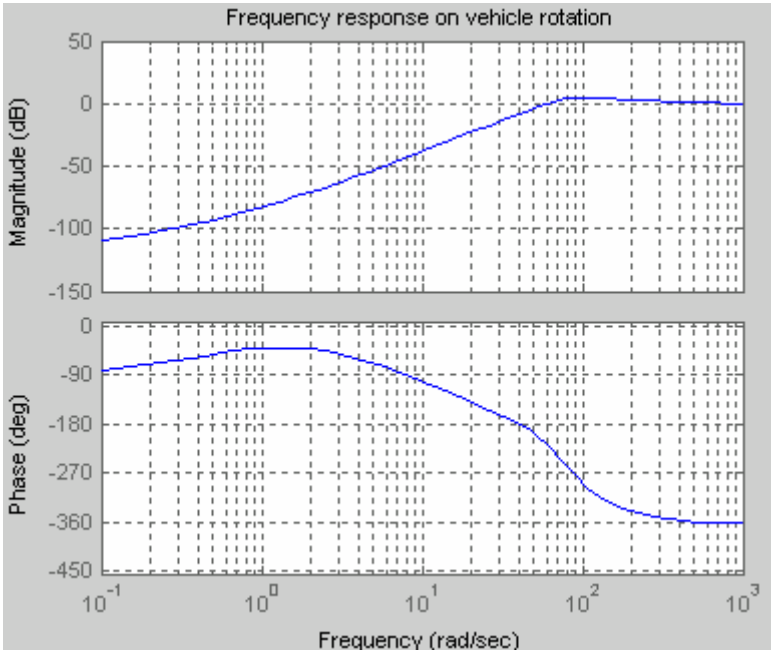

Figure 16. The frequency response of the gimbal control system.

The interpolator module generates a sequence of control inputs between the current pointing angle and the input pointing command. This process generates a smooth trajectory and more importantly, it minimizes the magnitude of the controller command.

This step is vital because the large commands combined with potentially high-magnitude shocks could cause the controller to become unstable due to the small phase margin.

Every time a pointing command is issued, the gyro coordinate system and global coordinate system are compared. The global coordinate frame is calculated in the gyro drift compensation module by using data from the system pose estimation (i.e. the electronics-box coordinate frame) and the absolute encoder data (the local coordinate frame). When the difference between the coordinate frames exceeds the resolution of the absolute encoder, (0.02 degrees) the modified global coordinate data is used to update the gyro coordinate system. The dead-band in the absolute encoder is a potential limiting factor in the pointing accuracy of the gimbal.

The mode selection and monitoring module is used controls which of the appropriate operational modes the gimbal should be in and switches between them as appropriate.

Terrain Evaluation and Path Adjustment

The terrain evaluation and path adjustment software is the heart of the onboard navigation system. It was developed after the disclosure that a majority of the Grand Challenge route would be on trails or poor dirt roads. The module is also designed to transition between sensor based navigation and blind navigation as onboard sensor data is determined to be valid or invalid. The pre-planned path is adjusted by a control-law-like navigation algorithm to ensure a smooth output path.

Terrain Evaluation

The terrain evaluation part of this module uses a statistical evaluation technique akin to that used in the Morphin/Gestalt algorithms [8][18]. For each laser line scan, the evaluation returns a vector of traversability scores, one for each point in the scan, indicating how safe it would be for Sandstorm to drive over that portion of the terrain.

To perform the analysis, for each point in the line scan, the set of points within half a vehicle width are selected. Using linear regression, a line is fit to this set of points. The slope of this line with respect to the gravity vector is used as one part of the cost measure, the steeper the slope, the less desirable the terrain is to drive over. A second measurement of the cost of the terrain comes from the residual between the point set and the line fit, if the residual is large, the terrain is irregular, and likely unsafe for driving. Finally, a measure of the slope of the line relative to a vector in the direction of travel is calculated. If the gimbal is pointed in the direction of travel, this measure can be used to detect vertical relief in the terrain. Unfortunately, this measure will consider terrain seen with the laser pointed in a direction other than along the direction of travel as unsafe. A better approach would be to consider the slopes of various windows of the line scan relative to the slope of the line generated by intersecting the plane of the laser scan with the nominal ground plane; this second approach would avoid the off-pointing problem.

The three cost measures (two slopes and the residual) are scaled to have a value between zero and one, with a cost of zero being equivalent to safe, and a cost of one being unsafe. The worst cost value is returned as the cost for the vehicle to travel with its center over a

particular laser point. Figure 17 shows an example of this analysis. The red and green set of roughly horizontal lines in the figure represents a laser scan intersected with the terrain. Green regions are considered safe for Sandstorm to traverse while the red regions are considered to be unsafe. Yellow circles mark the detected fence posts.

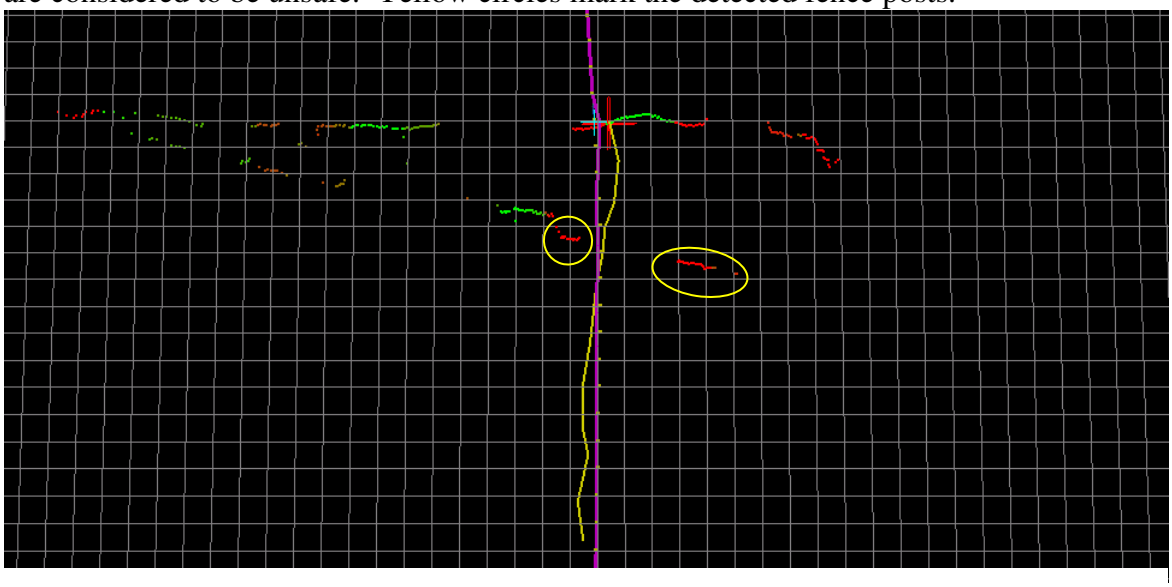

Figure 17. A sensor view of an opening in a fence.

Once the line scan has been evaluated, the costs are clustered into regions. The clustering uses the inherently ordered nature of the line scanner, so that very few distance calculations need to be performed. An initial pass through the line scan finds the lowest cost present in the array. If this cost is below a floor cost of acceptable terrain traversability, the floor cost is used to clip regions; this floor cost is set to be the expected limit of "easily traversable" terrain. If no point in the array has a cost less than the floor cost, the lowest cost in the array is selected, and an amount of cost equivalent to the noise in the terrain cost estimation is added to it to produce the value used to clip the regions. Regions are also clipped if a point in the region falls out of the corridor width specified in the pre-plan for this segment of the route.

Path Adjustment

The sets of connected points within the corridor with cost lower than the clipping value are extracted and the center of each cluster is determined. The signed distance from the pre-planned path to the calculated center point is used as the input to a controller servoing the path. The control law is of the same form as that used to perform the velocity control loop in the vehicle controller:

$$u_{k+1} = u_k + K_p e + K_d \dot{e}$$

In this way, if the pre-planned path has smooth curves, the output of the path adjustment will also be smooth. To simplify the path adjustment code, adjustments to the path are only allowed to be made farther along the route than any previous path adjustments.

During the initial testing of the path adjustment algorithm, the gimbal control code and pointing code were not yet reliable, so a pair of simple filters were used to throw out laser data should the gimbal be pointed incorrectly. The first filter calculates the distance between the center point of the current region and the center of the last region and throws out the new data if it is not within a one to three meter window. The second filter checks to insure that the current region center is within a cone with its apex at the center of the previous region. This second filter insures that the laser is not off-pointed in some odd direction.

Referring to Figure 17 again, the yellow pixels along the vertical purple curve show the preplanned path. The purple curve represents the path that is passed from the path shifter to the path tracker. Finally, the yellow curve represents the time history of the center of the safe region closest to the pre-planned path. Note that after it passes the current laser scan, the route fades back towards the pre-planned route. This stitching of the pre-planned and sensor planned route is required so that in instances where there is sharp acceleration and the pure-pursuit path tracker exceeds the laser look-ahead distance momentarily, there is some path for that software to track.

Binary Obstacle Detection

The binary obstacle detection module was written for the sole purpose of detecting the moving obstacle during the qualification runs. During normal race conditions, this module is removed from the data stream, and the adjusted path is passed directly from the terrain evaluation and path adjustment module to the path tracking module.

The obstacle detection algorithm builds an occupancy grid model of the world using data from a single pair of scans from the forward looking SICK LIDAR scanners. If any cell in the grid receives more than a specified number of hits, it is considered to be occupied, and thus an obstacle. The algorithm then checks to see if any of the obstacle cells are within half a vehicle's width of the adjusted path and are within some threshold distance of the robot. If these conditions are both true, then the output path has all of the speed fields set to zero. This causes the robot to stop, preventing a collision. Since the model is constructed cleanly with each pair of laser scans, when the obstacle is removed, Sandstorm will start in motion again.

This very simple approach has a variety of shortcomings; in particular, it will incorrectly classify gently sloping hills as obstacles. Because of this, it is an inappropriate algorithm to use in race conditions. However, in tightly constrained conditions such as those associated with the qualification course, the algorithm works reliably.

Path Tracking

At the base of the onboard navigation system there is a conventional pure-pursuit waypoint tracking algorithm. As is common, the look ahead distance is adjusted dynamically based on speed. The control gains are configured to provide a balance between good performance at both low speed in tight maneuvering situations, and at high speed on straight-aways and soft corners. Two good references for pure pursuit path tracking are [1] and [5].

Vehicle Controller

All of the software that interfaces with the mobility components of the vehicle is contained within a single PC-104 stack. This stack has four functions, shifting, steering, velocity control and emergency stop.

All control loops operate within a custom written timing loop running on a stock Redhat 9 kernel. This choice was justified since the amount of jitter and scheduling delay is generally insignificant when compared to the mechanical time constants associated with the vehicle. The software is structured so that descendant classes may increase the capabilities of the drive by wire system. Each descendant class overrides the "loop" function, to provide a capability. This software structure allowed for a logical decoupling of each of the control functionalities.

Gear Selection

The HMMWV's automatic transmission greatly simplifies gear selection. In practice the onboard software selects between 3 gears: drive, neutral and reverse. Testing shows that the velocity control loop functions equally well in any of the automatically selected gears so there is no need to actively constrain which gears should be selected by the transmission (via shifting to 1^{st} or 2^{nd} directly).

The electrical interface to the linear actuator controlling the shifter consists of a pair of computer controlled relays. Since the smoothness of the control activity is unimportant, it is possible to avoid adding other control electronics. The position of the actuator is provided via a potentiometer. This interface lends itself to a simple bang-bang control loop. The control loop servos the linear actuator until it is positioned at a set of pre-calibrated locations that represent the detents for each of the gear shifter positions.

Steering Control

The electrical interface for the steering control system uses an off the shelf motor controller (an AMC-DR100EE series) to drive a DC harmonic drive actuator mounted to the steering column. The motor controller provides a voltage controlled velocity loop that is controlled by an analog signal from the drive by wire PC-104 stack. Position control feedback is provided by a rotary variable differential transformer (RVDT) mounted on the output of the power steering box. The measured angular position is then used in a classical PD control loop, operating at 20 Hz, to set the speed and direction of rotation of the steering actuator. An interesting anomaly of the steering configuration of the HMMWV is that the mapping between steering column rotation and vehicle curvature is linear.

Velocity Control

To control vehicle velocity, a single actuator is used to either press the brake pedal or pull on the throttle cable. The actuator is driven by a DC amplifier driven by a differential signal from the drive-by-wire PC-104 stack. The Applanix POS-LV system provides a smoothed measure of the robot's velocity that is used as the control input signal.

The prevailing wisdom regarding HMMWV velocity control is that the engine response is very non-linear and would require a complicated scheme to maintain speed within a reasonable error bound. This wisdom may be drawn from experience operating at very low speeds, and where a +-1 mph error may be unacceptable. The control loop implemented on Sandstorm, utilizes a very simple control law:

$$u_{k+1} = u_k + K_p e + K_d \dot{e}$$

This formulation is similar to a PID controller but forgoes the integral term and instead uses the previous command output. The intuition used to select this formulation is that, independent of the terrain input (i.e uphill or downhill), the desired position for the throttle/brake (expressed as a floating point value in the range of -1 to 1) is likely near the current position. Furthermore this formulation uses the control signal to generate a relative motion, rather than an absolute position, avoiding some of the difficulties associated with a nonlinear controller. With this formulation, the speed controller was able to operate within acceptable steady state bounds (+/- 0.5m/s with an oscillation period of 0.1 Hz) with relatively short rise and drop times, over a range of velocities ranging from 5-22 m/s (11mph- 49mph). Figure 18 shows the controller's response on a cross country run during testing in Nevada. The vertical axis in this figure indicates velocity in meters per second, while the horizontal axis units are controller ticks, which are approximately equal to 0.1 seconds.

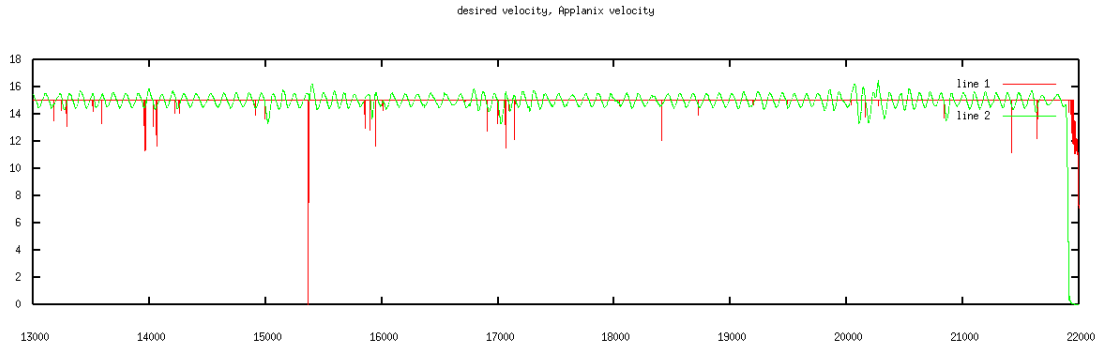

Figure 18. A plot showing the response of Sandstorm's velocity control loop.

The emergency stop system incorporates a variety of safeguards to ensure that should sub-components of Sandstorm fail, the robot can be reliably brought to a stop. The throttle control system has in integral heart beat circuit that causes the vehicle service brakes to be fully applied should the drive-by-wire computer system fail to maintain real time performance or crash. The engine ignition and fuel pump are also wired into the kill system such that should computing and/or actuator power fail, the HMMWV engine is turned off and fuel flow stopped, further more, this system also causes the HMMWV parking brake to engage, causing the robot to stop faster than relying on rolling friction.

Should Sandstorm need to be stopped or paused during operations where the computing system remains operable, any operator requests for a kill are understood by the drive-by-wire system and that system then engages the service brakes. In this state, the onboard

drive by wire computer disregards other drive commands until the e-stop signal is cleared.

Performance

A good measure of a system is performance relative to intentions. Sandstorm was developed to compete in and win the 2004 Grand Challenge. The Red Team and Sandstorm had a strong debut: Sandstorm qualified first, and was the only robot to qualify based on the pre-event rules.

Description of Significant Incidents on Race day

Though Sandstorm did not complete the Grand Challenge course, it traveled 7.4 miles along the route at a pace that would have finished the race within the allotted time. Sandstorm averaged over 15 mph, obeyed lower speed limits where specified, and hit a peak speed of 36 mph while generally demonstrating smooth, stable driving over off road terrain. During the 25 minutes of operation, Sandstorm traveled faster and farther than any of the other competitors. Along the way, there were six incidents of off-nominal behavior:

Impact with fence post #1
Impact with fence post #2
Momentary pause
Impact with fence post #3
Impact with Boulder
High centering in the hairpin

Each of these incidents is described below, with an explanation of why the incident occurred.

Impact with Fence Post #1

Time: 8:43 into run

As Sandstorm approached a road crossing, it was slightly left of a narrow opening (~3.6m wide) between two fence posts. The onboard sensors detected the fence correctly (the yellow circles marked in Figure 19) but the onboard path planning code did not generate a path to avoid these obstacles. This failing was a known weakness of the onboard navigation system.

An assumption implicit to Sandstorm's navigation system is that the trails and paths the robot traverses generally have berms or rough terrain delineating their edges. The path planning algorithm implicitly represents the obstacle at the location where the laser scan intersects the path. In the steady state case, this provides reasonable behavior with a fairly straightforward implementation. Again in Figure 19, this effect can be seen in the yellow control points used to servo the path. Note that it bulges to the right to avoid the obstacle, but the bulge is behind the true location of the post. Furthermore, the purple output path is not responsive to the obstacle since the control loop that shifts the path is over damped, and did not react in a significant way to avoid the obstacle.

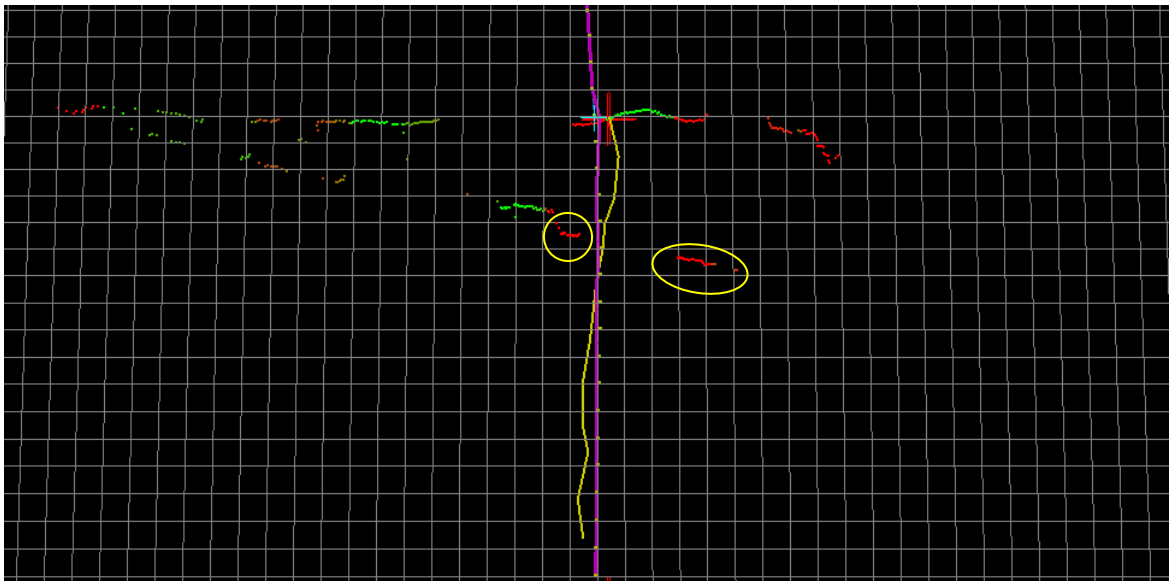

Figure 19. A sensor view of the first fence post Sandstorm hit.

Impact with Fence Post #2

Time: 9:02 into run

Sandstorm also hit a second gate associated with the same road crossing. Again the sensor system appropriately classified the gate as an obstacle but the path adjustment software did not react appropriately. Figure 21 shows the sensor view of this gate. Figure 20 shows the width of the gate, a Nissan Xterra is shown in the gate opening for scale (the Xterra is roughly 35cm narrower than Sandstorm).

Figure 20. A Nissan Xterra parked in the opening of the second gate.

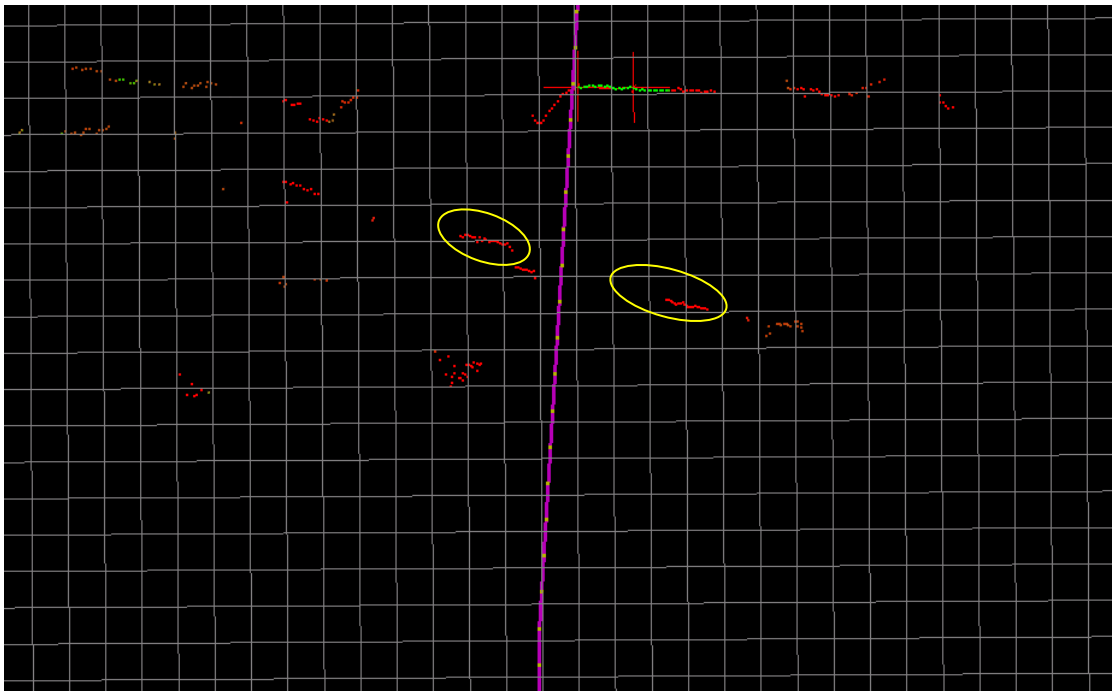

Figure 21. A sensor view of the second fence post Sandstorm hit.

Momentary Pause

Time: 10:36 into run

While traveling towards a trail intersection, Sandstorm decelerated to a stop, pausing for approximately 5 seconds. The cause for this stop is likely a spurious e-stop pause signal. After the pause Sandstorm continued without incident. The onboard navigation algorithm, when running in race configuration, has no way to cause the vehicle to come to a temporary pause. The duration of the stop (roughly 5 seconds) further supports this explanation, as 5 seconds is the minimum amount of time the vehicle controller must stop the robot for upon receipt of a pause signal. During testing, it was noted that the government issued safety system would at times generate spurious signals causing Sandstorm to stop temporarily. Omnitech robotics, supplier of the safety system, believes that the pauses were likely due to external electro-magnetic interference.

Impact with Fence Post #3

Time: 10:57 into run

After the momentary pause, Sandstorm proceeded at low speed for roughly 40 meters before hitting a third fence post. Sandstorm's low speed and the fact that the post was reinforced meant that it took almost 2 minutes of pushing against the post before the robot was able to knock it over and continue.

Once again, the same weakness in the onboard obstacle avoidance software was the principal cause of this incident. Sandstorm "saw" the post, but did not plan the correct avoidance maneuver.

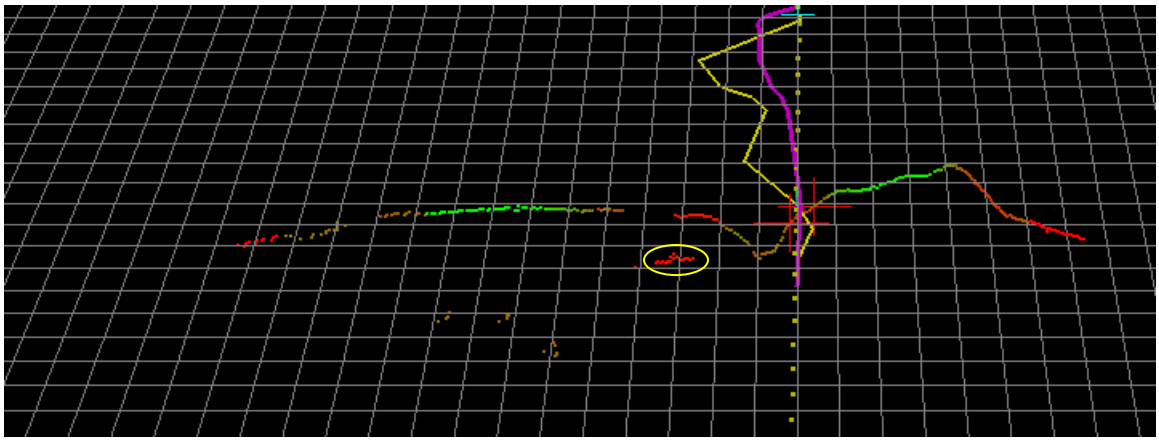

Figure 22. A sensor view of the large boulder.

Impact with Boulder

Time: 23:22 into run

After negotiating another 13 minutes of winding trail, Sandstorm drifted to the left of the road and impacted a large boulder (see Figure 23). This impact caused the steering to be jarred sharply left, causing Sandstorm to deviate from the path and leave the trail. Once off the trail, the onboard recovery software kicked in, reversed Sandstorm, then pointed Sandstorm back down the path and resumed motion. Sandstorm's navigation system overshot the trail once, corrected, and then continued along the course.

Figure 23. The large boulder Sandstorm hit.

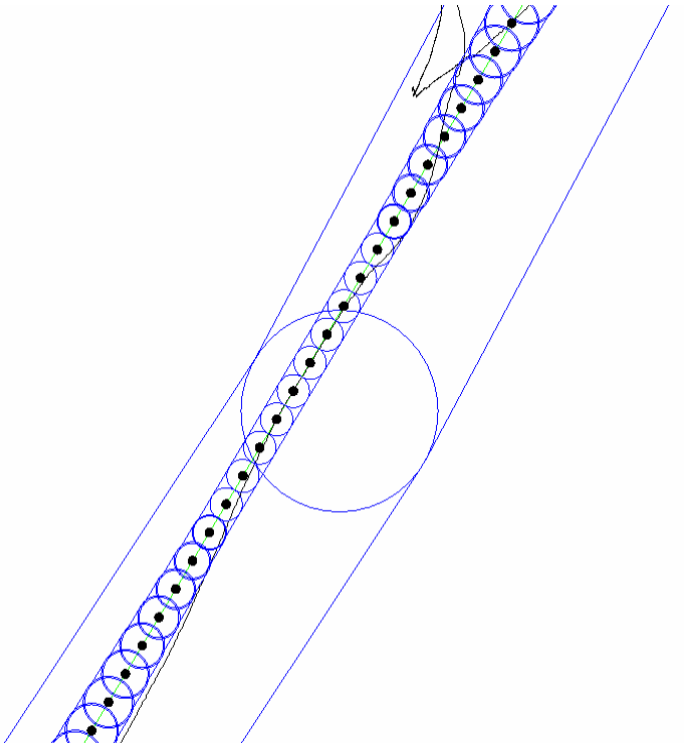

Figure 24. A plot showing the path constraints imposed on Sandstorm by the pre-planned route.

This incident was caused primarily by the preplanned path overly constraining the onboard navigation system. In Figure 24 the inner circles represent the range of freedom that the onboard planning system had to adjust the path to avoid obstacles (the outer circles represent the width of the DARPA route at that location). At its narrowest, this corridor is only one meter wide. Prior to this narrowing, the robots path (the black curve) was at the right edge of this boundary, indicating that the onboard navigation system had detected the road and was tracking it, but was constrained by the corridor. As the corridor narrows, Sandstorm is forced into the undesirable terrain and the onboard control oscillates, possibly seeking marginally better areas to drive in.

The narrow corridor constraints would not have been a significant issue if the preplanned route had been centered on the road. For some reason, at this location, the preplanned route appears to be roughly 1 meter left of road center, when compared to the location surveyed by a post race analysis team. Had the corridor constraints been looser in this area, the onboard sensor based navigation system would have avoided the obstacle. Figure 22 shows the sensor image of this boulder.

From the onboard stabilized stereo cameras, the effect of this boulder on the motion of the robot can be seen, and it is dramatic. Even so, the onboard electronics box and sensors received a peak vertical acceleration of only 16.5 m/s^2, or just under 2 g's. This is a testimonial to Sandstorm's passive shock isolation system. After this impact, Sandstorm appeared to continue down the trail without any obvious difficulty.

High Centering in the Hairpin

Time: 25:11 into run

In the Daggett Switch backs, Sandstorm approached a tight hairpin wide and outside. As it rounded the corner, it cut towards the inside and the left side wheels slipped off of the trail. Sandstorm continued along the berm until it came into contact with a large rock (see Figure 25) buried in the soil. For the next 400 seconds Sandstorm melted the rubber off of its front tires, trying to extract itself from the berm. Eventually the crew in the chasing control vehicle triggered a disable emergency stop which caused the front service brakes to clamp down on the inboard side of the four half-shafts. This sudden stop combined with the momentum carried by the fast spinning front wheels caused both front half-shafts to snap.

Figure 25. The large rock that eventually ended Sandstorm's run.

This failure was a result of a variety of weaknesses acting in concert to end Sandstorm's race. Entering the corner, the onboard navigation system began to filter out laser data. The filtering algorithm was triggered as a result of a sharp angle change in the preplanned path which would not have been present if the path used smooth curves. Figure 26 shows that even though the data was disregarded, the classification of the terrain from the laser scan was still reasonable.

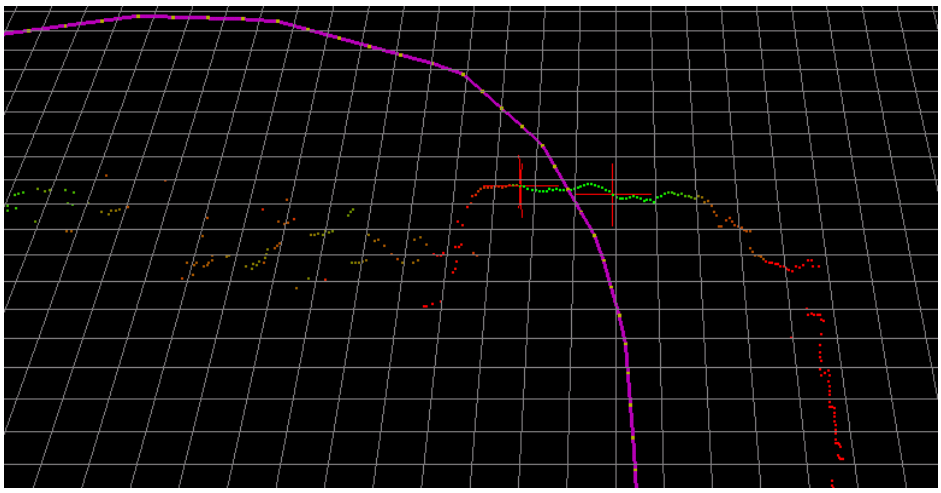

Figure 26. A sensor view of the trail in the Dagget Hairpin.

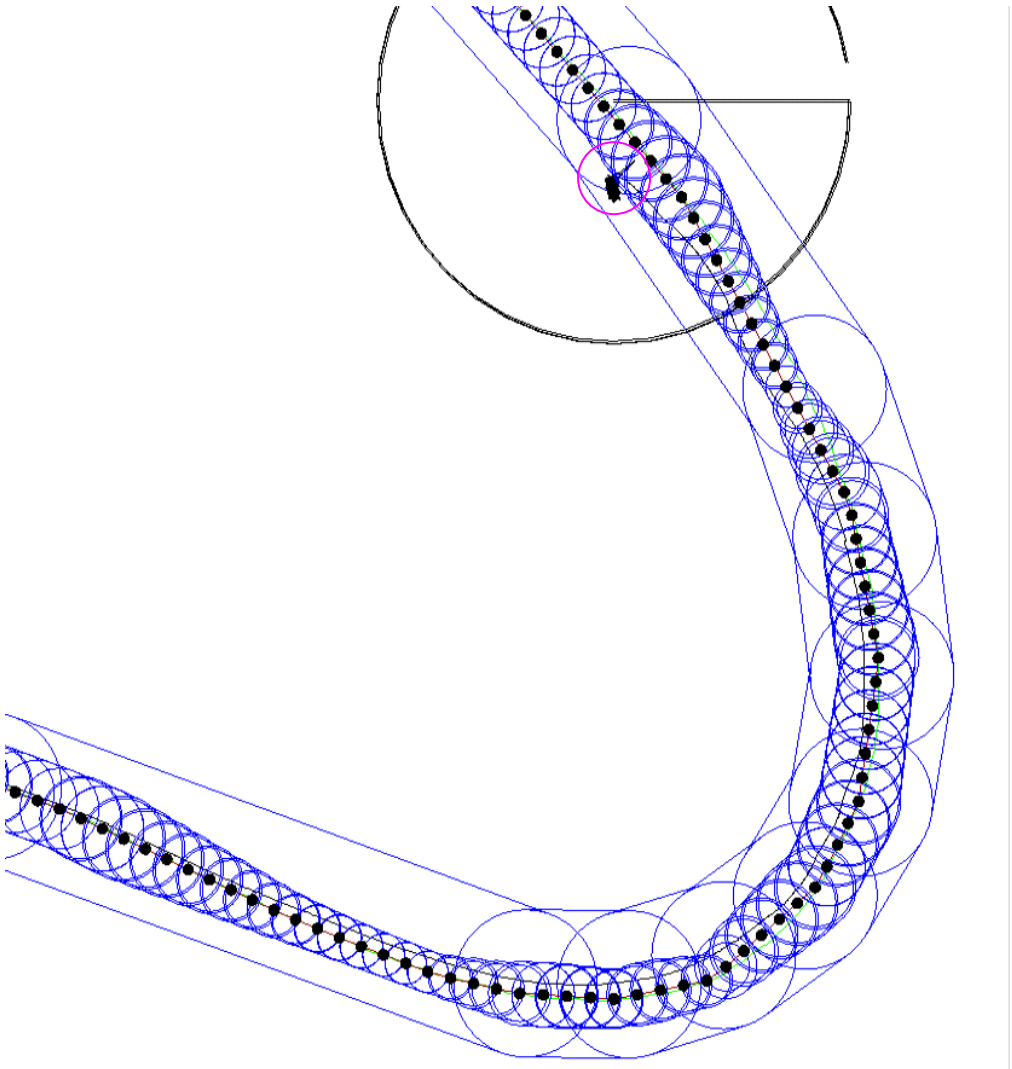

Figure 27. The pre-planned route around the hairpin.

Once the laser data was disregarded, the onboard planning system seamlessly switched to following GPS blindly. At this point, Sandstorm began to cut towards the inside of the curve. Sandstorm's GPS measurement of the preplanned path had errors pushing it towards the inside of the curve. In addition, the faceted nature of the preplanned path caused it to be even farther towards the inside of the corner. Finally, the pure-pursuit path tracking software can cause Sandstorm to cut corners. In this case, these three effects combined to push Sandstorm roughly 1.5 to 2 meters to the left of the road center such that one wheel fell off of the edge. Figure 27 shows a plot of the pre-planned corridor (inner blue circles), pre-race reconnaissance (green) and Sandstorm's ground track (black). From this data, the path error seems to be due equally to the above mentioned sources.

Once one wheel got over the edge, Sandstorm was unable to pull itself back onto the road before it became stuck. Figure 28 shows Sandstorm's resting place, just after the tight hairpin corner, for scale, Sandstorm is roughly 2.25 meters wide.

Figure 28. Sandstorm's final resting place on race-day.

Other Points of Interest

GPS Error: The exact amount of GPS measurement error is unclear. In all incidents described, if the GPS error had been zero, Sandstorm would likely not have failed, but the GPS error appears to have been within the range that the onboard navigation system should have been able to correct for.

It would be useful to better understand the effects of high voltage power lines on the performance of the Applanix POS-LV unit. Though it is unlikely that GPS error was the root cause of any of the incidents, all of the incidents occurred within close proximity of power lines and error in Sandstorm's pose was a contributing factor.

Laser Line Rate: The backup Riegl laser scanner (installed after the pre-race rollover) was used on race day. This operated at only 15 Hz, instead of the specified 50Hz, and only ¾ of the line rate achieved by the original Riegl scanner. The onboard filtering of the laser data was designed to operate with a laser scan rate of 20 Hz. Though it appears to have played no role in any of the incidents during race day, the decreased laser scan rate did cause the onboard system to disregard the laser data several times when Sandstorm accelerated to high speed. Had any of these accelerations occurred in more challenging terrain, this weakness may have led to a failure.

Stereo Data
After reviewing the logged stereo imagery from race day, it is apparent that the onboard stereo system would have been of little benefit during the early portions of the race. For much of the time, Sandstorm was heading easterly into a rising Sun. Sun glare, exposure problems and internal reflections from the dome faceplate rendered much of the imagery unusable for standard stereo vision techniques.

Lessons Learned

Upon reflection on the implementation and testing of the Red Team race system, several important lessons can be extracted:

Prior map knowledge changes the formulation of on-line planning- The availability of high resolution, high accuracy a priori models o the world a robot is operating in greatly reduces the complexity required of the onboard planning system. With sub-meter a priori information, it is unnecessary for the on-board planning system to be concerned with large scale local-minima in the planning space.

The impact of robot dynamics can be significant- Though not discussed in this report, during testing Sandstorm rolled while driving at roughly 50mph. The root cause of the role was an overlap in the route Sandstorm was tracking at the time. The roll occurred because Sandstorm turned very sharply to respond to inconsistent path tracking commands. Had there been a better model of the robots safety margin, the control output could have been limited to prevent the roll over from happening.

Map based on-line planning is essential- The various filtering approaches that incorrectly disregarded the laser data on race day would have been unnecessary if a map based representation had been used instead of a single laser line based planning approach. Furthermore, a map based model would have allowed for an algorithm more capable of representing the true location of obstacles. By maintaining a better model of the world, the online path adjustment software will be able to better react to complicated obstacle configurations.

Understand the nature of measurement errors- The onboard navigation system was constrained to the explicit route boundaries specified in the route definition file provided by the race organizers. Because this error was not taken into account, Sandstorm was constrained into a collision with a boulder at one point. Sandstorm detected the boulder,

and would have avoided the boulder if it were allowed to violate the corridor boundaries. In future systems, this error should be incorporated into the constraint system.

Software development, integration & testing time is critical- The compressed development and testing schedule required to field Sandstorm for the 2004 Grand Challenge forced the cutting of much important scope, and did not provide sufficient time to fully verify the onboard software. Future race robots should benefit from the existence of Sandstorm, allowing for the parallel development of software with any new robotic platform.

Conclusions

Sandstorm and the associated Red Team race system represent the state of the art in terms of the product of speed and distance traversed by an autonomous robot on off-road terrain.

The novel integration of high resolution prior map information combined with a primarily reactive, control-law based navigation approach shows promise for future off-road navigation work. Through a better integration of onboard sensor data and slightly less reactive planning, a significant further improvement in navigation performance is anticipated.

Though the Red Team was not successful in its goal of completing the 2004 Grand Challenge, it generated a significant technological legacy and a basis from which future advances will be generated.

References

[1] O. Amidi, "Integrated Mobile Robot Control", Technical Report CMU-RI-TR-90-17, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1990.

[2] P. Bellutta, R. Manduchi, L. Matthies, K Owens & A. Rankin. "Terrain Perception for DEMO III", Proc. IEEE Intelligent Vehicles Symposium, Dearborn, USA, October 2000.

[3] J. Biesiadecki, M. Maimone & J. Morrison. "The Athena SDM Rover: a Testbed for Mars Rover Mobility", Proc. i-SAIRAS 2001, St-Hubert, Canada, June, 2001.

[4] D. Coombs, K. Murphy, A. Lacaze & S. Legowik. "Driving Autonomously Offroad up to 35km/h", Proc. IEEE Intelligent Vehicles Symposium, Dearborn USA, 2000.

[5] R. Coulter, "Implementation of the Pure Pursuit Path Tracking Algorithm", Technical Report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 1992.

[6] P. Drewes, “Demonstration of a Systems Architecture for Live, Virtual, and Constructive UGV Operation”, AUVSI, Baltimore Md. 2003.

[7] V. Gazi, M. Moore, K. Passino, W. Shackleford, F. Proctor & J. Albus, “The RCS Handbook: Tools for Real-Time Control Systems Software Development”, John Wiley & Sons, NY, 2001.

[8] S. Golberg, M. Maimone & L. Matthies. “Stereo Vision and Rover Navigation Software for Planetary Exploration”, In Proceedings of the IEEE Aerospace Conference, Big Sky, USA, March 2002.

[9] J. Gowdy, “IPT: An object Oriented Toolkit for Interprocess Communication”, Technical Report CMU-RI-TR-96-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1996.

[10] B. Hayes-Roth, “A Blackboard Architecture for Control”, Journal of Artificial Intelligence, Vol. 26, pp. 251-321, 1985.

[11] A. Kelly & A. Stentz. “An Analysis of Requirements for Rough Terrain Autonomous Mobility”, Autonomous Robots, Vol. 4, No. 4, December, 1997.

[12] D. Pomerleau, “RALPH: Rapidly Adapting Lateral Position Handler”, IEEE Symposium on Intelligent Vehicles, September, 1995, pp. 506 - 511.

[13] R. Simmons, E. Krotkov, L. Chrisman, F. Cozman, R. Goodwin, M. Hebert, L. Katragadda, S. Koenig, G. Krishnaswamy, Y. Shinoda, W. Whittaker, & P. Klarer. “Experience with Rover Navigation for Lunar-Like Terrains”, Proc. IEEE IROS, 1995.

[14] A. Stentz & M. Hebert. “A Complete Navigation System for Goal Acquisition in Unknown Environments”, IEEE IROS, 1995.

[15] C. Thorpe. “Vision and Navigation: The Carnegie Mellon Navlab”, Kluwer Academic Publishers, 1990.

[16] C. Thorpe, T. Jochem, and D. Pomerleau. “The 1997 Automated Highway Free Agent Demonstration”, IEEE Conference on Intelligent Transportation Systems, November, 1997, pp. 496 - 501.

[17] P. Tompkins, A. Stentz, and W.L. Whittaker. “Field Experiments in Mission-Level Path Execution and Re-Planning”, Proceedings of the 8th Conference on Intelligent Autonomous Systems (IAS-8), March, 2004.

[18] C. Urmson, M. Dias and R. Simmons, “Stereo Vision Based Navigation for Sun-Synchronous Exploration”, In Proceedings of the Conference on Intelligent Robots and Systems (IROS), Lausanne, Switzerland, Sept. 2002.

[19] C. Urmson, R. Simmons, “Approaches for Heuristically Biasing RRT Growth”, In Proceedings of the Conference on Intelligent Robots and Systems (IROS), 2003.

[20] J. Woodfill, B. Von Herzen, "Real-Time Stereo Vision on the PARTS Reconfigurable Computer," Proceedings IEEE Symposium on Field-Programmable Custom Computing Machines, Napa, pp. 242-250, April 1997.