Terrain Modeling and Obstacle Detection for Unmanned Autonomous Ground Robots

Amit Puntambekar

October 18, 2006

Acknowledgments

“Thank You” would be a very small word for the support I got from my adviser Dr. Arun Lakhotia, who has analytical abilities and technical expertise of a genius; he consistently conveyed a sense of adventure in regard to research. This thesis would never have been conceptualized without the ideas and motivation that he provided me. Right from my initial days in this country he was there to support me, both morally and technically. One paragraph, by no means whatsoever, does justification for my gratitude towards him.

Dr.Guna Seetharaman’s algorithmic inputs served the seeds from where this thesis took its present shape. I greatly appreciate the trust he showed in me throughout my thesis.

I am grateful to Pablo Mejia for the expert guidance he provided me through out the thesis. Special thanks to Suresh Golconda for the excellent support. Thank you Suresh for always being there and for those countless rides. Thank you Joshua Bridevaux for always keeping the hardware running, Santhosh Padmanabhan and Vidhya Venkat for reading the thesis draft and providing constructive comments. Also, I thank other members of Team CajunBot for their feedback.

I thank my parents for the never-ending trust and confidence they had in me. Now I understand what they told me when I was a kid “it is not machines that made us, the humans; but vice-versa”.

Finally, it would be injustice if I did not mention my dear friends Prajakta, Renuka, Sameera, Soumya and Hema for their constant encouragement during these past two years.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

An Autonomous Ground Vehicle (AGV) is a fully automated vehicle that can travel, unmanned, on a specific predefined route without any human intervention. NASA's Mars Rover [1] is an AGV used for space exploration. AGV's are currently being used extensively for military surveillance, for example, the PackBot Scoutt and Military R-Gator developed by iRobot [2]. AGV's are also used in farming for automated harvesting and treating crops with fertilizers [3].

The research and development effort building the AGVs so far has been on developing vehicles that operate on limited terrain conditions and on applications where speed is not an important issue. These vehicles are not conductive for use in battlefields as they are not suitable for offroad navigation. The US Congress has chartered the Defense Advanced Research Project Agency (DARPA) [4] to bridge this gap and produce Autonomous Robots that can be used in the front by 2015. To accelerate the research and development of AGVs, DARPA conducted the DARPA Grand Challenge 2004 and 2005 competitions [5]. To win, the participating AGVs had to travel about 150 miles in off road conditions within 10 hours. In the 2004 event no team completed the course. Sandstorm from CMU went the farthest traveling 7.2 miles. The 2005 event was won by an AGV Stanley from Stanford University. CajunBot, our teams vehicle, was ranked fifteenth in the race traveling 17.5 miles [6].

The ability to detect and avoid obstacles is a prerequisite for success in building autonomous robots. In view of the DARPA Grand Challenge, the near term goal was to develop an autonomous ground robot which could travel about 150 miles on off-road conditions within 10 hours with ability to:

1. Detect and avoid natural and man made obstacles.

2. Detect rough terrain conditions to decelerate the robot to a safe speed.

3. Distinguish between a moving and stationary obstacles.

In the off-road context, the definition of an obstacle could be anything that "obstructs" the motion of the vehicle. It could be a rock, a un-traversable slope, a tree, or a cliff. Having a single approach to detect all types of obstacles is a major challenge.

Timely processing of sensor data is an important aspect of obstacle detection, as the vehicle is expected to travel at high speeds. A delay in detecting obstacles could be fatal for the vehicle. Highly precise obstacle information could be provided if the system had no time constraints. An ideal system should have a proper balance between the speed and accuracy of the results.

Proposed approach uses LIDAR's for obstacle detection. High precision and low cost were the deciding factors in favor of LIDARs as opposed to other perception sensors like RADARs and camera. The LIDAR, LMS 221 from SICK, scans at 75 hertz with 180 degree field of view and a quarter degree offset.

1.2 Research Contribution

This thesis presents Team CajunBot's insights and innovations in the field of terrain mapping for autonomous ground vehicles. This document deals with the following aspects of the Terrain Mapping Algorithm:

1. An obstacle detection system that does not require stabilization of the sensors, rather it takes advantage of bumps in the terrain to see farther.

2. A robust system to deal with sensor errors like GPS drift, GPS spike.

3. A scalable system that has a provision for adding additional sensors in future.

1.3 Impact of Research

The proposed terrain mapping and obstacle detection technique would, by enlarge, cut down the cost of robots as no sensor stabilizers are required. The proposed technique could be used by any range measuring sensor, not just the LIDAR's. Commercially, a part of the algorithm could be used in LIDAR based surveying applications.

However, it is not claimed that the current implementation would be able to differentiate between the nature of obstacles, like the difference between a bush and a rock.

1.4 Organization of Thesis

Chapter 2 provides a brief history of the work already done in the field of terrain mapping and obstacle detection using LIDAR sensors.

Chapter 3 describes the terrain modeling and obstacle detection algorithms.

Chapter 4 discusses the sensor errors and other sensor related issues affecting the algorithm.

Chapter 5 details various methods used to test and evaluate the algorithm.

Chapter 6 provides conclusion with a discussion on the future work that would aid in improving the current work.

2 Background and Related Work

2.1 Vehicle Information and Sensors

CajunBot was the finalist in the 2004 and the 2005 events of the DARPA Grand Challenge. It is built on a MAX IV six wheeled all terrain vehicle with a 25 hp twin cylinder engine. The vehicle weighs about 1200 lbs and can reach a top speed of 25 miles/hr. The vehicle uses "skid steering" mechanism to steer, very similar to that used in the battle tanks [7].

CajunBot uses two LIDAR sensors, the SICK LMS 291's for terrain mapping and obstacle detection [8]. LIDARs are range measuring devices. The LIDARs operate on the principle of "Time of Flight" to measure the range between the source of laser and the target. The LMS 291's scan 180 degrees in a single pass, called a scan. A scan comprises of 180 beams. LIDARs operate at 75 Hz scan rate with a quarter degree offset between two consecutive scans. The output of the LIDAR is set of (angle, range) value for every return. Cost and accuracy make LIDARs a better choice than other perception sensors like cameras, radars, etc. Radars are less accurate than LIDARs [9] and their field of view is less than that of the LIDARs, the SICK LIDAR has a 180 degree filed of view but on an average a RADAR has about 12 degree field of view [10]. Cameras are more expensive than LIDARs and are also more sensitive to the light.

Two front facing LIDAR's are mounted which scan the terrain for obstacles. The LIDAR's are mounted to look at 16m and 16.3m in front of the vehicle. CajunBot also performs well, albeit at a reduced speed, with only one LIDAR. Figure 1 depicts the top view of the LIDAR sensors.

CajunBot uses an INS (Oxford Technology Solutions RT3102) for autonomous operation. The INS provides precise instantaneous location and orientation of the vehicle. The accuracy of the INS is enhanced by Starfire differential GPS correction signals provided

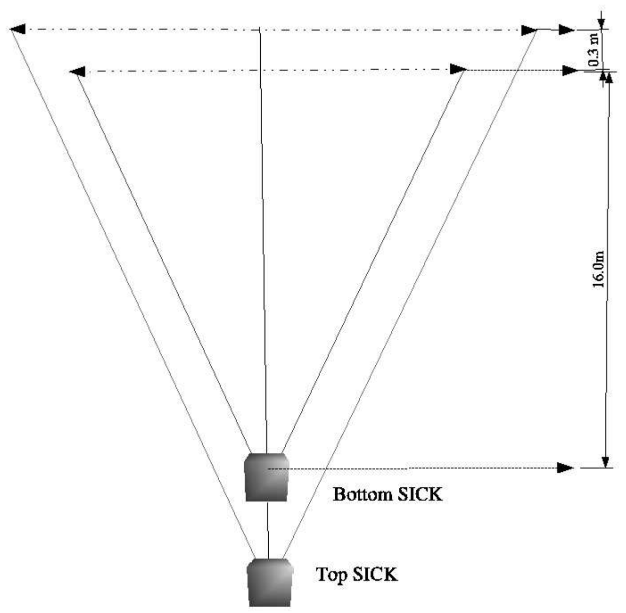

Figure 1: Top view of the top and bottom LIDARs

by a C&C Technologies C-Nav receiver. Figure 2 details the mounting of sensors on CajunBot.

Figure 2 shows the actual picture of the sensors mounted on the metal frame. It is to be noted that the LIDAR's, GPS and the INS is mounted on the same metal frame, the reason for which is explained in the Chapter 3.

Post 2005 DARPA Grand Challenge CajunBot was retired. Focus is on CajunBot-2 - a 2004 Jeep Rubicon. Ability to navigate at faster speed and having better shocks were the favorable features in selecting CajunBot-2 over CajunBot. The sensor placement of CajunBot-2 was similar to that of CajunBot, the difference being all the sensors were not on the same metal frame. This is not a major issue in CajunBot-2 as it is equipped with better suspensions than CajunBot.

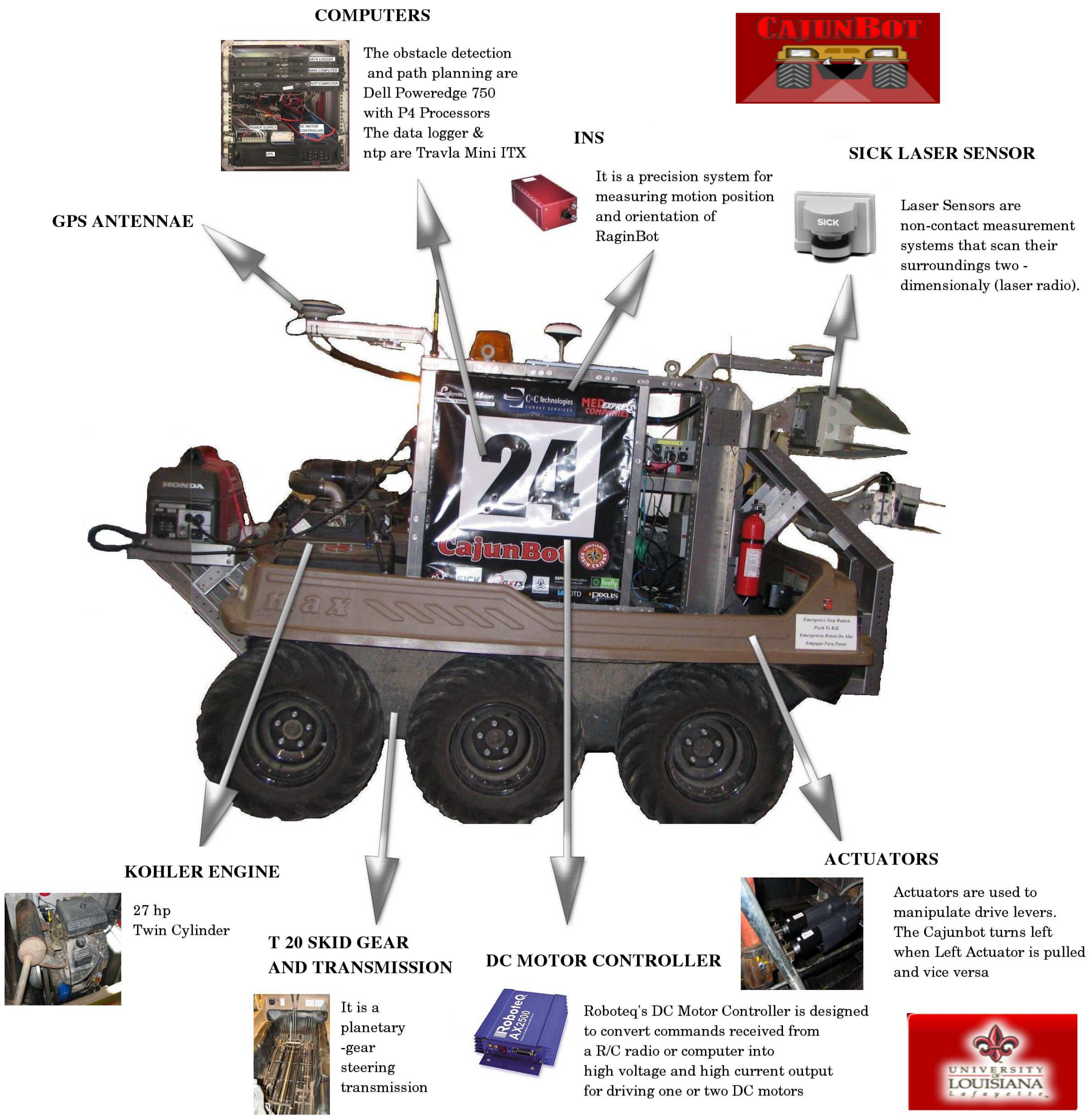
COMPUTERS
The obstacle detection and path planning are Dell Poweredge 750 with P4 Processors
The data logger & ntp are Travla Mini ITX
CAJUNBOT
INS
It is a precision system for measuring motion position and orientation of RaginBot
SICK LASER SENSOR
Laser Sensors are non-contact measurement systems that scan their surroundings two - dimensionaly (laser radio).
GPS ANTENNAE
24
CajunBot
KOHLER ENGINE
27 hp
Twin Cylinder
T 20 SKID GEAR AND TRANSMISSION
It is a planetary -gear steering transmission
DC MOTOR CONTROLLER
Roboteq's DC Motor Controller is designed to convert commands received from a R/C radio or computer into high voltage and high current output for driving one or two DC motors
ACTUATORS
Actuators are used to manipulate drive levers. The Cajunbot turns left when Left Actuator is pulled and vice versa
UNIVERSITY of LOUISIANA Lafayette

Figure 2: Schematic view of CajunBot

2.2 Related Work

There are several approaches proposed for obstacle detection for outdoor robots using LIDAR sensors. Most of these approaches work well when the vehicle is moving slow on a relatively flat terrain, as the vehicle (and the sensors) experience less bumps. Bumps have two effects on the system - 1) Consecutive LIDAR scans do not fall in close geographical proximity, and 2) Fusion of INS and LIDAR data may not be accurate, as, due to sudden bumps, the INS data may not give the orientation of the LIDAR at the time a scan is read. Due to Step 1 mentioned above, analyzing consecutive LIDAR scans to determine the change in their geometry would be ruled out, as the scans are geographically apart. Step 2 would corrupt the transformations used to translate the obstacle information from the sensor coordinate space to the global coordinate space, the detected obstacles would be placed in incorrect locations. Also, Step 2 may introduce non-existent obstacles (false obstacles) as incorrect INS orientation angle is associated to LIDAR scans. In case of slow moving robots, the affect of bumps is less, the consecutive LIDAR scans fall in a close spatio-temporal proximity. Moreover, the consecutive scans define a region which is geographically close. The change in the geometry of these scans closely relate to change in the geometry of terrain. The region in which the change in geometry of the terrain is untraversable by the robot is termed as an "obstacle" region. Most of the related work presented below is based on the above mentioned approach, the difference being the way the "change" in geometry of scans is computed. Some approaches use "discontinuity" of the range data form multiple scans whereas some others use the change in slope of the scans to compute the nature of the terrain.

Many approaches used a customized version of plane fitting algorithm. A plane is fit through a certain number of scans, called the best fitting plane, and then, classify the points that fall outside the threshold of the plane as obstacles [11]. The problem with this approach is it depends heavily of spatial closeness of the consecutive LIDAR scans. Only when the scans are spatially close, would the best fitting plane closely approximate the actual terrain. As the scans disperse, due to bumps experienced by the sensors, the best fitting plane would not represent the actual terrain. Another approach would be to group the spatially close LIDAR scans and then do the plane fitting. Later, in Chapter 3, we notice that the GPS/INS data is locally accurate. Meaning, we could only compare the scans that are temporally close. Therefore, for this approach to work the LIDAR scans should be in spatio-temporal proximity. In the Grand Challenge the vehicles are expected to travel at an average speed of about 15 mi/hr on off-road terrain. The bumps experienced by the vehicle would certainly disperse the LIDAR scans, thereby negating the spatial closeness.

Henrinksen and Krotov [12] categorized mainly three types of hazards detected by an LIDAR - the positive elevation hazard, the negative elevation hazard and the belly hazard. They further suggested a slope based approach to classify each of this type of hazard. Their algorithm, however, is unable to detect steep continuous slopes. They further suggest having a three dimensional approach can be beneficial for analyzing the terrain data.

A "differential depth and slope" approach for fast obstacle detection is suggested in [13] . Though they claim this approach to be "fast" and "accurate", it suffers from a major set back - all the computations are made based on vehicle centric co-ordinate system. The problem arises as the vehicle cannot have any history of previously detected obstacles, as the detected obstacles are relative to the instantaneous position of the vehicle. Progression would be to use GPS and INS to compute the global frame of reference. In Chapter 3 we discuss the problems associated with this approach.

A way of combining stereo camera and LIDARs is suggested in [14]. The color data from the camera is mapped to the range discontinuity data from the LIDARs for obstacle detection and terrain classification. The experimental results show that this approach works well on rough terrains albetit at low speeds which is unsatisfactory for success in DARPA Grand Challenge. At higher speeds this approach would require good sensor stabilizers to stabilize the sensors against bumps, otherwise, the camera and LIDARs might not point at the same geographical location.

Roberts and Corke [15] suggested a moving window based slope computation for obstacle detection for mining vehicles. This approach works well if the vehicle is traveling at low speeds thereby causing the LIDAR scans to incrementally sweep the region. In the Grand Challenge (GC) context, the vehicles are expected to travel with an average speed of 15 mi/hr on considerably bumpy surfaces. This would cause the scans to be dispersed. Having sensor stabilizers is a solution but this increases the cost of production of the vehicle.

Most of the approaches work well with positive obstacles and when the robot is traveling at relatively low speeds. Negative obstacles, like ditches and steep downward slopes, are not detected easily by these approaches due to the low LIDAR data density in those regions. Mounting the LIDARS high and pointing them close to the vehicle would ensure a better data density, but this could decrease the distance at which obstacles are detected. Matthies and Rankin [16] suggest using thermal signatures for detecting negative obstacles and night navigation. The cost of such hardware is a major cause of concern.

From the prior research experience we could safely conclude the following.

Undefined geometric nature of the off road terrain makes it difficult for the traditional obstacle detection algorithms to detect obstacles. The ambiguity in the definition of the obstacle, especially in off road conditions, makes it difficult for the algorithms to classify obstacles. The dispersed LIDAR scans due to the speed and bumps experienced by the robot further complicate the algorithms. Detection of negative obstacles, like ditches, is a challenge because of low data density.

3 Terrain Modeling and Obstacle Detection

Interpreting the sensor data such that it represents the geometry of the terrain is termed as "terrain modeling". The "modeled" terrain is further analyzed to determine if a particular region is an "obstacle" or not. The process of analyzing the terrain to generate a set of obstacles is termed as "Obstacle Detection".

This chapter summarizes CajunBot's terrain modeling and obstacle detection algorithm and highlights the specific features that enables it to take advantage of vibrations along the height axis, i.e., bumps, to improve its ability to detect obstacles.

3.1 Core Algorithm

The data flow diagram in Figure 3 enumerates the major steps of the terrain modeling and obstacle detection algorithm. The algorithm takes as input the vehicle's state and LIDAR scans. The vehicle state data is filtered to attend to spikes in data due to sensor errors (Step 3.1), and then used to compute the global coordinates for the locations from which the beams in a LIDAR scan were reflected (Step 3.2). The global coordinates form a 3-D space with the X and Y axes corresponding to the Easting and Northing axes of UTM Coordinates, and the Z axis giving the height above sea level. Virtual triangular surfaces with sides of length 0.20m to 0.40m are created with the global points as the vertices. The slope of each such surface is computed and associated with the centroid of the triangle (Step 3.3). A vector product of the sides of the triangle yields the slope. The height and slope information is maintained in a digital terrain map, which is an infinite grid of 0.32m $\times$ 0.32m cells. A small part of this grid within the vicinity of the vehicle is analyzed to determine whether each cell contains obstacles (Step 3.4). This data is then extracted as a Terrain Obstacle Map [17].

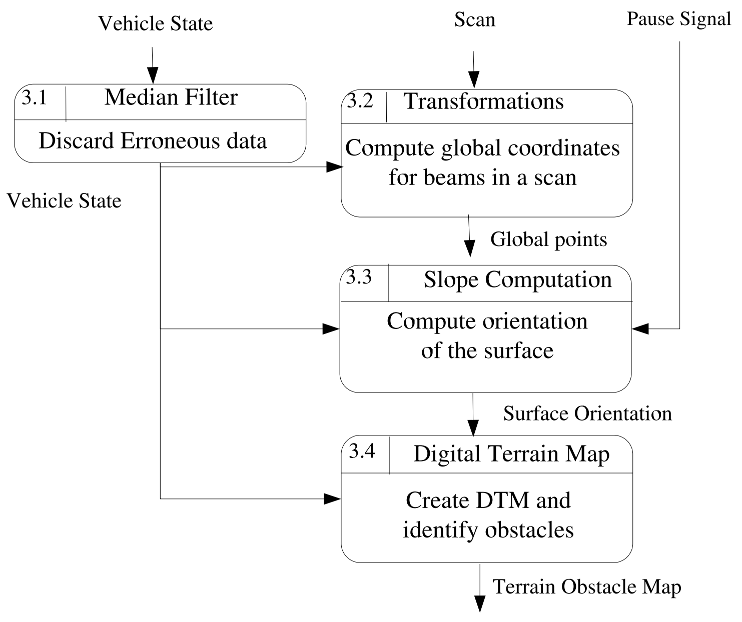

Figure 3: Data Flow Diagram (DFD) for the algorithm

Figure 4 graphically depicts data from the steps discussed above. The figure presents pertinent data at a particular instant of time. The grey region represents the path between two waypoints. The radial lines emanating from the lower part of the figure show the LIDAR beams. There are two sets of LIDAR beams, one for each LIDAR. Only beams that are reflected from some object or surface are shown. The scattering of black dots represent the global points, the points where LIDAR beams from some previous iteration had reflected. The figure is scattered with triangles created from the global points. Only global points that satisfy the spatio-temporal constraints, discussed later, are part of triangles. There is a lag in the data being displayed. The triangles shown, the global points, and the LIDAR beam are not from the same instant. Hence, some points that can make suitable triangles are not shown to form triangles. The shade of the triangles in Figure 4 represents the magnitude of slopes. The black triangles have high slope, +/- 90 degrees, and the ones with lighter shades have much smaller slopes. In the figure, a trash can is detected as an obstacle, as shown by

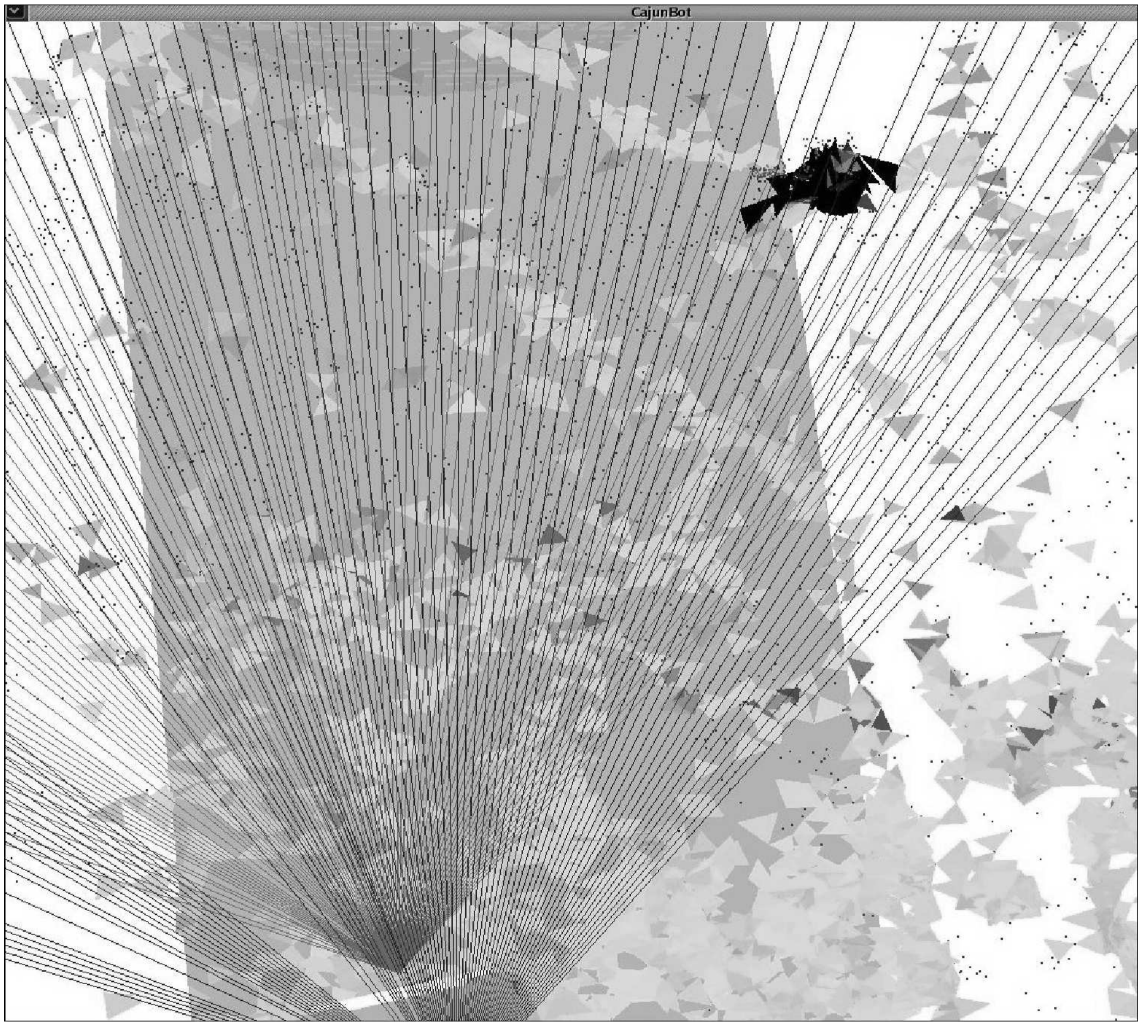

Figure 4: Virtual triangle visualization

the heap of black triangles. The data was collected in UL's Horse Farm, a farm with ungraded surface. The scattering of dark triangles is a result of the uneven surface.

3.2 Obstacle Detection

A cell is classified as an obstacle using the following steps. First, a cell is tagged as a 'potential' obstacle if it satisfies one of three criteria. The number of times a cell is categorized as a potential obstacle by a criterion is counted. If this count exceeds a threshold–a separate threshold for each criterion–it is deemed an obstacle. The criteria used to determine the classification of a cell as a potential obstacle are as follows:

High absolute slope. A cell is deemed as a potential obstacle if the absolute maximum slope is greater than 40 degrees. Large objects, such as, cars, fences, and walls, for which all three vertices of a triangle can fall on the object, are identified as potential obstacles by this criterion. The threshold angle of 40 degrees is chosen because CajunBot cannot physically climb such a slope. Thus, this criterion also helps in keeping CajunBot away from unnavigable surfaces.

High relative slope. A cell is deemed as a potential obstacle if (1) the maximum difference between the slope of a cell and a neighbor is greater than 40 degrees, and (2) if the maximum difference between the heights of the cell and that of its neighbor is greater than 0.23m. This criterion helps in detecting rocks as obstacles, when the rock is not large enough to register three LIDAR beams that would form a triangle satisfying the spatio-temporal constraint. The criterion also helps in detecting large obstacles when traveling on a slope, for the relative slope of the obstacle may be 90 degrees, but the absolute slope may be less than 40 degrees. The test for height difference ensures that small rocks and bushes are not deemed as a potential obstacle. The height 0.23m is 0.02m more than the ground clearance of CajunBot.

High relative height. A cell is deemed as a potential obstacle if the difference between its height and the height of any of its neighbor is greater than 0.23m. This criterion aids in detecting narrow obstacles, such as poles, that may register very few LIDAR hits.

The threshold counts of 5, 5, and 12, respectively, are used for the three criteria to confirm a potential obstacle as an obstacle.

As a matter of caution, Step 3.3 disables any processing when the Pause Signal is activated. This prevents the system from being corrupted if someone walks in front of the vehicle when the vehicle is paused, as may be expected since the Pause Signal is activated during startup and in an emergency.

3.3 Issues with Bumps and Sensor Stabilization

Bumps along the road have impact on two steps of the algorithm, Step 3.2, where data from the INS and LIDAR is fused and, Step 3.3, when data from beams from multiple LIDAR scans are collected to create a triangular surface. The issues and solutions for each of these steps are elaborated below.

In order to meaningfully fuse INS and LIDAR data it is important that the INS data give orientation of the LIDARs at the time a scan is read. Since it is not feasible to mount an INS on top of a LIDAR, due to the bulk and cost of an INS, the next logical solution is to mount the two such that they are mutually rigid, that is, the two units experience the same movements. There are three general strategies to ensure mutual rigidity between sensors: (1) Using a vehicle with a very good suspension so as to dampen sudden rotational movements of the whole body and mounting the sensors anywhere in the body. (2) Mounting the sensors on a platform stabilized by a Gimbal or other stabilizers. (3) Mounting all sensors on a single platform and ensuring that the entire platform is rigid (i.e., does not have tuning fork effects). Of course, it is also possible to combine the three methods.

CajunBot uses the third strategy. The sensor mounting areas of the metal frame is rigid, strengthened by trusses and beams. In contrast, most other GC teams used the first strategy and the two Red Teams used a combination of the first two strategies.

Strategy 3 in itself does not completely ensure that mutually consistent INS and LIDAR data will be used for fusion. The problem still remains that the sensors generate data at different frequencies. Oxford RT 3102 generates data at 100Hz, producing data at 10ms intervals, whereas a SICK LMS 291 LIDAR operates at 75Hz, producing scans separated by 13ms intervals. Thus, the most recent INS reading available when a LIDAR scan is read may be up to 9ms old. Since a rigid sensor mount does not dampen rotational movements, it is also possible the INS may record a very different orientation than the time when the LMS data is recorded. Fusing these readings can give erroneous results, more so because an angular difference of a fraction of a degree can result in a LIDAR beam being mapped to a global point several feet away from the correct location.

The temporally ordered queues of the middleware, CBWare [17], and its support for interpolating data help in addressing the issue resulting from differences in the throughput of the sensors. Instead of fusing the most recent data from the two sensors, Step 3.2 computes global points by using the vehicle state generated by interpolating the state immediately before and immediately after the time when a LIDAR scan was read. Robots with some mechanism for stabilizing sensors can fuse a LIDAR scan with the most recent INS data because the stabilizing mechanism dampens rotatonal movements, thus ensuring that the sensors will not experience significantly different orientations in any 10ms period.

Absence of a sensor stabilizer also influences Step 3.3, wherein triangular surfaces are created by collecting global points corresponding to LIDAR beams. Since CajunBot's sensors are not stabilized, its successive scans do not incrementally sweep the surface. Instead, the scans are scattered over the surface as shown in Figure 5. This makes it impossible to create a sufficient number of triangular surfaces of sides 0.20m to 0.40m using points from successive scans (or even ten successive scans). It is always possible to create very large triangles, but then the slope of such a triangle is not always a good approximation for the actual slope of its centroid.

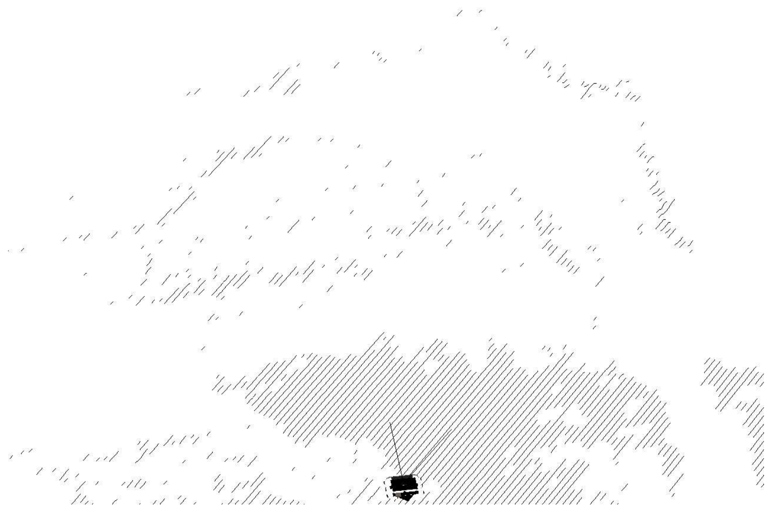

Figure 5: LIDAR beams scattered due to bumps

3.4 Dynamic Obstacles

The confidence factors associated with the algorithm cannot guarantee zero false positives while detecting all “real” obstacles due to the GPS drift and related sensor errors (discussed in Chapter 4). A solution would be to minimize the false obstacles, identify them and to eliminate the false positives while not eliminating the true ones. Also, the algorithm needs a strategy to overcome the issue of “dynamic obstacles”, for example, other vehicles moving in the path.

If the dynamic (moving) obstacles, like a competing robots moving in front of the vehicle, are not flushed from the “obstacle memory” then they would be treated as an obstacle at every location, thereby creating lots of “false” obstacles in the path. “Changing obstacles”, like the gate which is initially closed and opens when the vehicle is very close to it, would be detected as an obstacle too.

Grid Refreshing strategy is used by the algorithm to classify and eliminate the false positives. This approach also helps in dealing with dynamic obstacles.

TOM grid is a spatio-temporal grid. Only a limited, temporally close, data is kept in the TOM cells. Apart from putting new data into the TOM cells, the process of binning involves refreshing the cells if the existing data and the new data are not temporally close. This ensures the false obstacles, if any, gets refreshed periodically. If in a particular iteration, due to corrupted data, the algorithm detects false obstacles, the good data in the consecutive iterations would invalidate the earlier result.

Further, an access time stamp is associated to each cell when new data is put into it. Cells are aged based on its last access time stamp. Time difference of 15 seconds since the last access is used to age the cells. Especially for the gate type of scenario, when the gate is closed it would be detected as an obstacle. Once it is opened, there would be no LIDAR hits in the "obstacle" TOM cells, as the gate which had obstructed the beams is now open. After the 15 seconds time these TOM cells are refreshed as there are no LIDAR hits. Even if the bot is stationary or in motion, the previously detected obstacles are preserved as long as the LIDAR beams are hitting them.

Essentially this approach does not detect a moving obstacle, but Grid Refreshing handles the case when the vehicle encounters dynamic obstacles. Using this approach the trajectory and speed of the moving obstacles cannot be determined.

4 Sensor Errors

The types of sensor errors could be broadly classified as

Static Errors. The errors due to improper calibrations or mounting of the sensors.

Dynamic Errors. The errors induced in the sensors due to corrupt input signals.

In regard to terrain mapping and obstacle detection the source of sensor errors could be in 1) LIDARS and 2) GPS/INS.

4.1 LIDAR Related Errors

1. LIDAR Mounting Angles. The mounting angles, also known as Bore-sight angles, is orientation of the LIDAR with respect to the INS. Typically, they are referred to as roll, pitch and yaw angles representing the orientation in 3 axes.
2. LIDAR Offsets. The offsets are the distances from the INS to the LIDAR. These values are manually measured.

Computing the LIDAR mounting angles and the offset wrt the INS is a potential source of Static Errors [18]. The offset and angles are used in Step 3.2 of Figure 3 to compute the global coordinates of the points where the LIDAR beams hit. Even a small error of about 0.5 degrees in measuring the mounting angles of the LIDAR would offset the global position of the target by 0.4m at a range of 20 meters.

Three approaches are used to compute and verify the mounting angles:

1. The Manual Triangulation Approach, wherein the center and extreme end beams are manually traced and triangulation is used to compute the mounting angles of the sensor with respect to the INS. This approach takes a considerable amount of time and effort.

2. The Ground Line Approach, wherein the the global points on the ground are computed assuming a particular 'hypothetical' mounting angle of the sensor. A line joining these points is called the Ground Line. Line obtained from the actual points of the LIDAR, referred as Actual Line, is superimposed on the Ground Line. The mounting angles of the Actual Line are tuned manually till it overlaps with the Ground Line. The set of mounting angles for which the Ground Line and the Actual Line overlap are the mounting angles of the actual sensor wrt the 'hypothetical' one. This approach requires less time and effort than the previous one, but still involves the process of manual tuning of the mounting angles. Also, this process has to be repeated for each sensor since all sensors cannot be tuned at the same time.

3. The Hypothesis Voting Approach, wherein the bot is run around a set of pre-surveyed objects of known dimensions. The error function computes the difference between the observed location and the surveyed location of the objects. Also, the error function computes the difference in the observed dimension to the actual dimension of the object. The mounting angles are tuned till the error function reports a near zero error. This approach involves minimum human intervention and all these sensors can be tuned at the same time.

4.2 GPS Related Errors

The GPS spike and GPS drift are the typical GPS related errors [19] that fall under the Dynamic Errors category. If not handled correctly, these errors would induce false positives in the system.

1. GPS Spike

   GPS spike is the sudden change in the GPS data in a fraction of a second. Typically,

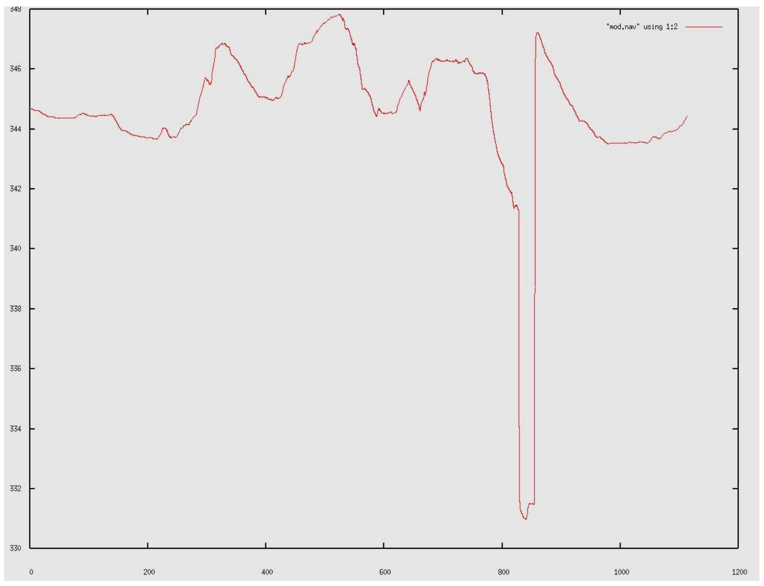

Figure 6: Spike in the GPS elevation data

the height measurement from the GPS is more prone to the spike than the location reading. This error usually occurs when the GPS recieves low or no signals and the INS uses 'dead-reckoning' to estimate the change in the position. When the GPS re-acquires the signal it corrects itself resulting in the data spike. The graph in Figure 6 depicts the spike in the elevation data experienced by the CajunBot in one of the NQE runs in the 2005 DARPA Grand Challenge. The X-axis of the graph represents the time axis and the Y-axis is the height (Z) as reported by the GPS. The graph shows that the height value (Z) changed by 15m in a fraction of a second. The sudden shange in the Z value causes the Obstacle Detection Algorithm to detect a wall like obstacle in the path. This error had caused the CajunBot to stop in the second National Qualifying Event (NQE) Run in the 2005 DARPA Grand Challenge.

Median Filter in Step 3.2 of Figure 3 continuously monitors the data from the INS and GPS for spikes. If a spike greater than a threshold is observed the data is discarded. Also, data in the spatio-temporal grid is refreshed to avoid interaction of the good and

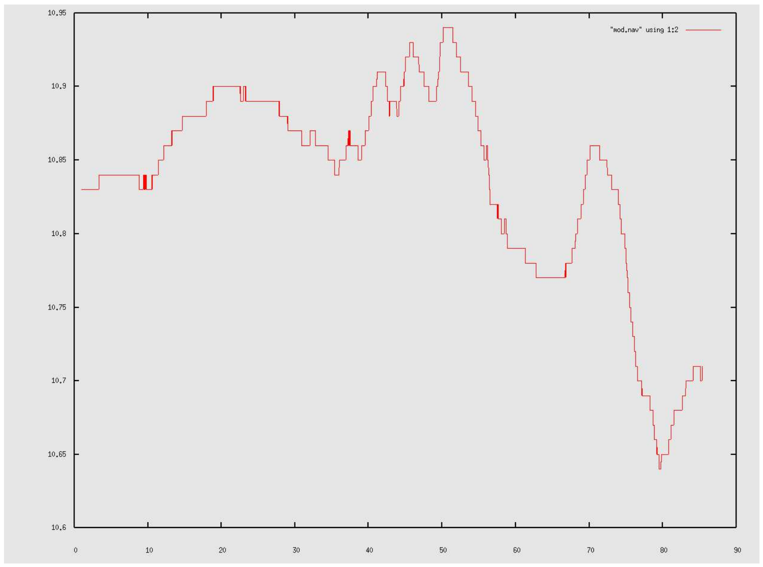

Figure 7: GPS drift

corrupted data.

2. GPS Drift

   GPS drift is the gradual drift in the GPS position and height data. The drift in the location is negligible as compared to that in height. Graph in Figure 7 depicts the GPS drift in the height data. The X-axis of the graph is the time axis and the Y-axis is the height as reported by the GPS. First 60 seconds worth of data in the graph in Figure 7 is when the vehicle is stationary. For the remaining time the vehicle is running on a flat parking lot at a speed of about 5 m/s.

   Graph in Figure 7 shows that even when the vehicle is stationary there is a constant change in the Z value, as reported by the GPS. When the vehicle is stationary the maximum difference in Z value for the same location is 0.18 m.

If the GPS/INS data were very precise then triangles of desired dimensions could be created by saving the global points from Step 3.2 in a terrain matrix, and finding groups of three points at a desired spatial distance. This is not practical because of Z-drift, the drift in

Z values reported by a GPS (and therefore by the INS) over time. When stationary, a drift of 10 cm - 25 cm in Z values can make even a flat surface appear uneven.

The elevation(Z)-drift issue can be addressed by taking into account the time when a particular global point was observed. In other words, a global point is a 4-D value (x, y, z, and time-of-measurement). Besides requiring that the spatial distance between the points of a triangular surface be within 0.20m and 0.40m, Step 3.3 also requires that their temporal distance be under three seconds.

5 Algorithm Testing and Evaluation

5.1 Testing and Tuning

On-field testing is the best way to test the algorithm, but it involves two major overheads, firstly the entire vehicle should be in a workable condition, right from the mechanical components to the electronics, and the sensors to the software. Secondly, requirement of multiple team members, the cost involved in the logistics and time spent in the process. Also, uncontrollable issues like weather conditions may cause unexpected changes in the testing plans.

The above issues can be handled by having an offline, virtual testing environment. CajunBot's Simulator, CBSim, is a physics-based simulator developed using the Open Dynamics Engine (ODE) physics engine. Along with simulating the vehicle dynamics and terrain, CBSim also simulates all the onboard sensors. It populates the same CBWare queues with data in the same format as the sensor drivers. It also reads vehicle control commands from CBWare queues and interprets them to have the desired effect on the simulated vehicle.

Offline-testing and debugging is further aided by the Playback module. This module reads data logged from the disk and populates CBWare queues associated with the data. The order in which data is placed in different queues is determined by the time stamp of the data. This ensures that the queues are populated in the same relative order. In addition, the Playback module, like the Simulator module, generates the simulator time queue representing the system-wide clock. This simple act of playing back the logged data has several benefits. In the simplest use, the data can be visualized (using the Visualizer module) over and over again, to replay a scenario that may have occurred in the field or the simulator. It offers the ability to replay a run after a certain milestone, such as a certain amount of elapsed time or a waypoint is crossed. In a more significant use, the playback

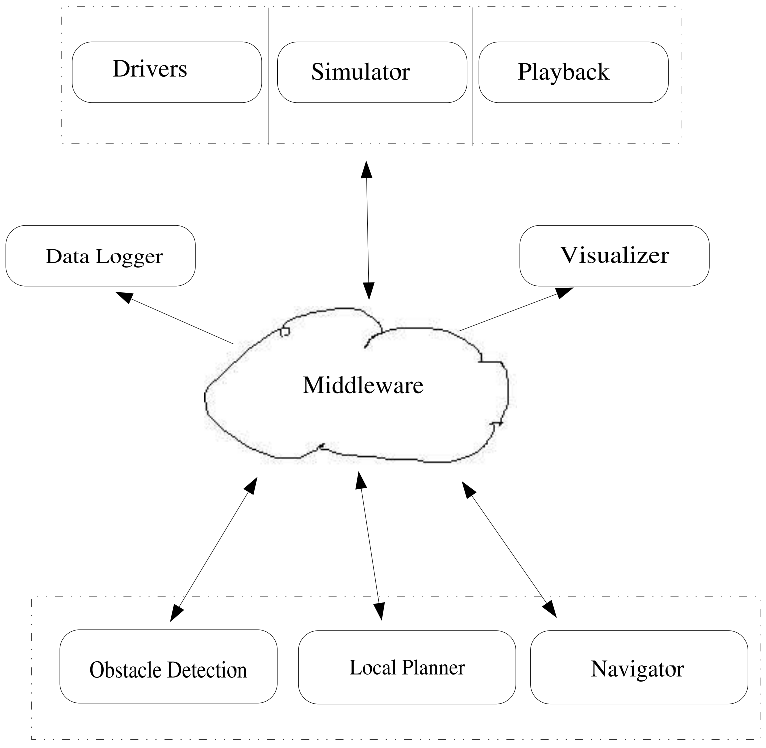

Figure 8: Components involved in the system

module can also be used to test the core algorithms with archived data. This capability has been instrumental in helping us refine and tune the algorithm. It is the common operating procedure to drive the vehicle over some terrain (such as during the DARPA National Qualifying Event), playback the INS and LIDAR data, apply the obstacle detection algorithm on the data, and then tune the parameters to improve the obstacle detection accuracy. Figure 8 graphically illustrates the various components of the CajunBot system.

Tuning and debugging is a time consuming process considering the volume of data that needs to be analyzed. Just by looking at the numbers output by the software and tuning/debugging data is an error prone process. Its humanly impossible to manually dig into the huge data and decipher the cause of the error. Having an interface which would convert this data into graphical form would be the best tool to monitor and analyze this sort of data.

Real-time and off-line debugging is supported by CBViz, the Visualizer module. CBViz is an OpenGL graphical program that presents visual/graphical views of the world as seen by the system. It accesses the data to be viewed from the CBWare queues. Thus, CBViz may be used to visualize data live during field tests and simulated tests, as well as visualizing logged data using the Playback module.

5.2 Algorithm Evaluation

The proposed algorithm is evaluated on the following parameters:

1. Ability to utilize bumps to detect farther obstacles
2. Percentage of false positives or true negative results
3. Space and Time complexities
4. Scalability
5. Results on different types of obstacles
6. Sensor orientation independence

The algorithm is evaluated on three data sets. Feasibility to test and create appropriate test cases were the primary reasons to select the following test environments.

1. CajunBot logged data from 2005 GC final run.
2. Testing in controlled environment on CajunBot-2.
3. Testing in software simulated environment - CBSim.

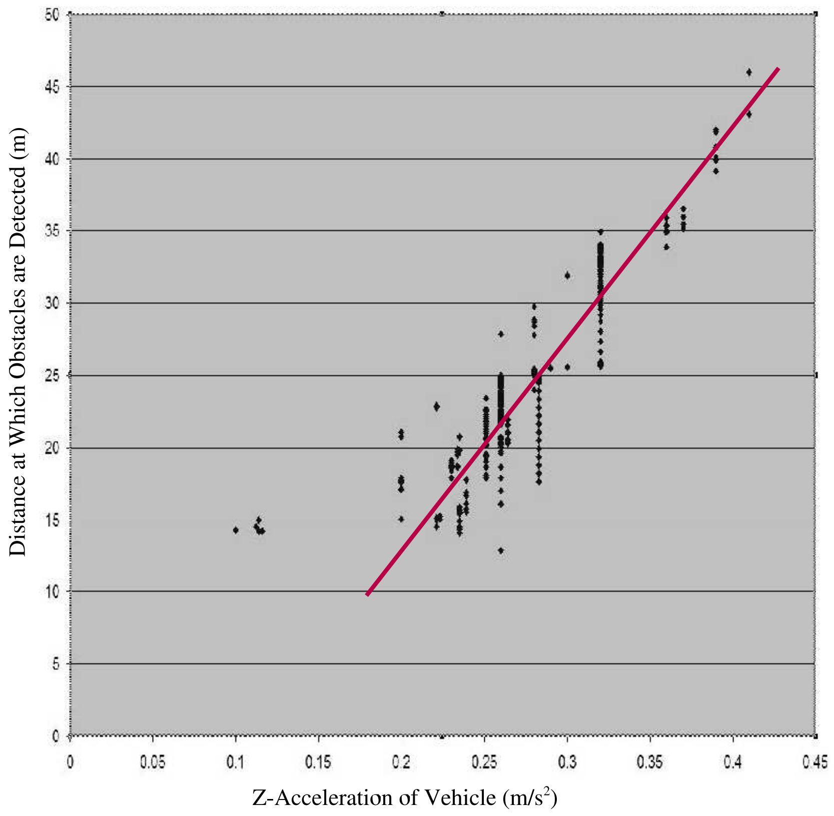

Figure 9: Graph showing distance to detected obstacles Vs Z-acceleration

5.3 Evaluation on 2005 GC Final Run

These are results of post processing the logged data from the 2005 GC Final Run. The vehicle traveled about 17.6 miles before it was stopped due to a mechanical failure.

1. Bumps Utilization

   Figure 9 presents evidence that the algorithm's obstacle detection distance improves with roughness of the terrain (bumps). The figure plots data logged by CajunBot traveling at 7m/s through a distance of about 640m of a bumpy section during the 2005 GC Final. The X-axis of the plot represents the absolute acceleration along the height (Z) axis at a particular time. Greater acceleration implies greater bumps. The Y-axis represents the largest distance from the vehicle at which an obstacle is recorded in the Terrain Obstacle Map. The plot is the result of pairing, at a particular instance, the vehicle's Z acceleration with the furthest recorded obstacle in the Terrain Obstacle Map (which need not always be the furthest point where the LIDAR beams hit). The plot shows that the obstacle detection distance increases almost linearly with the severity of bumps experienced by the vehicle. The absolute vertical acceleration was never less than 0.1 m/s^2 because the vehicle traveled at a high speed of 10 m/s on a rough terrain. That the onboard video did not show any obstacles on the track and that the obstacle detector also did not place any obstacles on the track leads us to believe that the method did not detect any false obstacles.

2. Scalability

CPU Utilization. The average 'percentage CPU utilization', as reported by the Linux utility `top`, sampled every second.

Increase in CPU. The percentage increase in CPU utilization going from one LIDAR configuration to a configuration of two LIDARs.

Table 1 gives the data when the terrain was not very bumpy, whereas Table 2 presents data for bumpy terrain in the actual Grand Challenge Final Run. In both the situations, adding another LIDAR reduces the obstacle detection time at a higher rate (38-48%) than the increase in the CPU utilization (22-28%). This implies our algorithm scales well with additional LIDARs, since the benefits of adding a LIDAR exceeds the costs.

Comparing data across the Table 1 and Table 2 further substantiates that our algorithm takes advantage of bumps. Compare the data for the single LIDAR configurations in the two tables. The CPU utilization is lower when the terrain is

Table 1: Effect of number of LIDARs, with low average bumps: $0.11 m/s^2$

# LIDARs	1	2
CPU Utilization	12.4%	15.9%
Increase in CPU		28.23%

Table 2: Effect of number of LIDARs, with high average bumps: $0.24 m/s^2$

# LIDARs	1	2
CPU Utilization	11.2%	13.7%
Increase in CPU		22.32%

bumpy. The same is true for the dual LIDAR configuration. The more interesting point is that adding another LIDAR does not lead to the same increase in CPU utilization for the two forms of terrain. For the bumpy terrain the CPU utilization increased by 22.32%, which is significantly less than the 28.23% increase for the smoother terrain.

3. Accuracy of Results

In context of the GC Final Run, number of false obstacles on the track is a parameter to analyze the accuracy of the results. In this case we would confine our analysis to the track as the outside region had many bushes, trees, etc, which would be potential obstacles. It is difficult, in this case in particular, to differentiate between the real obstacles and the false ones.

Just about that time CajunBot was started, weather took a turn. The winds picked up, blowing through the dry lake bed and causing a big sand storm. Analysis of the logged data revealed how CajunBot weathered the sand storm. The on-board video show CajunBot completely engulfed in the sand. That led it to see 'false obstacles', forcing it to go out of the track to avoid them. However, after the sand storm cleared,

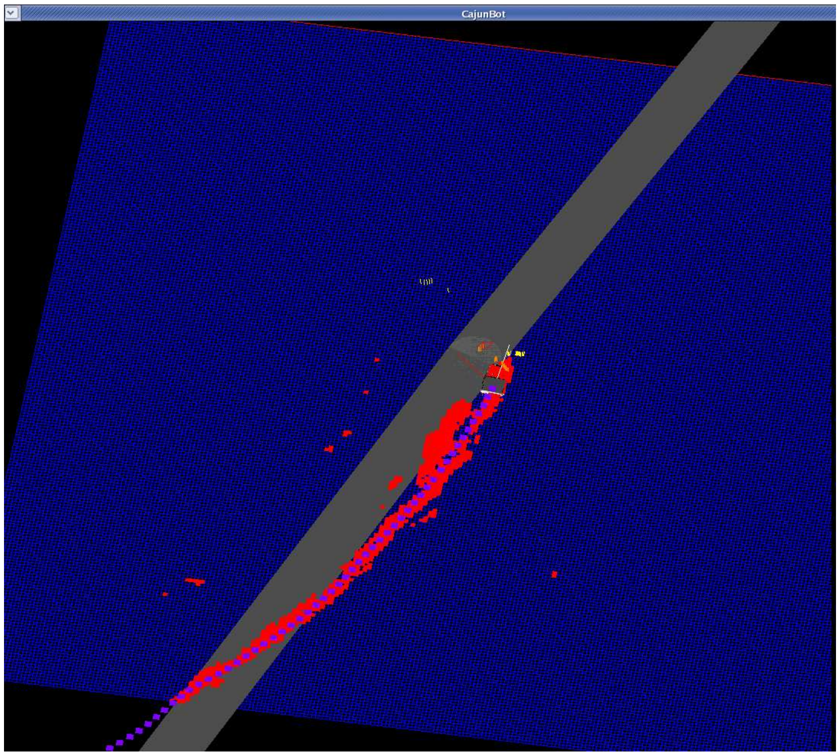

Figure 10: False obstacles due to the sandstorm

the video shows CajunBot running very much along the middle of the track, and passing stalled or stopped vehicles. The logged data shows absolutely zero false obstacles throughout the run after the storm, even in areas where the vehicle experienced severe bumps.

Figure 10 shows the screen shot of CajunBot visualizer for the 2005 GC Final Run between waypoints 41 and 44. The gray patch is the route and the numbered dots are the rddf waypoints provided by the GC officials. The green dots are the trail marks of the vehicle. This is the actual path traveled by the vehicle. The yellow/orange grids are the LIDAR beams. The blue grid is the TOM and the red dots are obstacles, in this case the dust particles.

As seen in Figure 10 the vehicle went about 30m from the center of the track. This

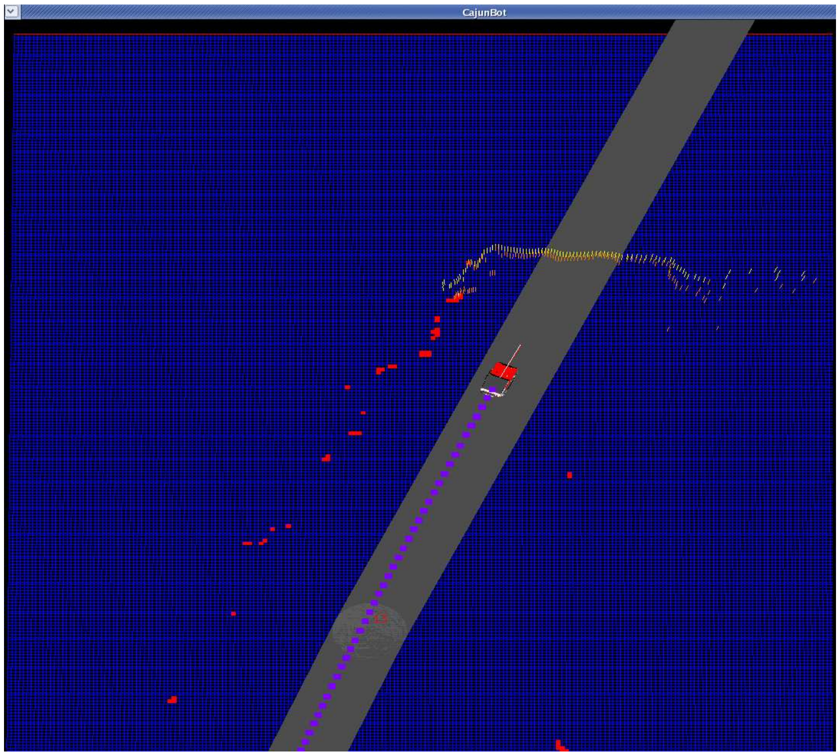

Figure 11: No false obstacles

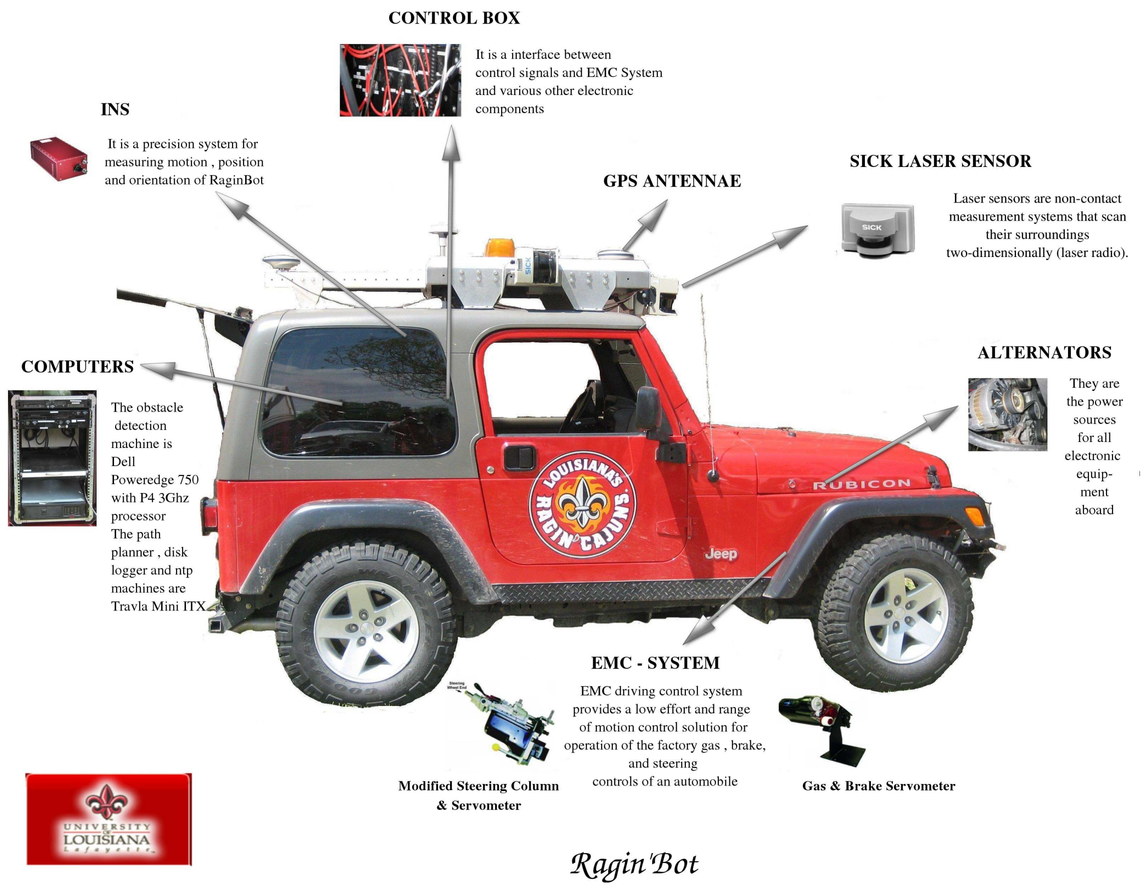
CONTROL BOX
It is a interface between control signals and EMC System and various other electronic components
INS
It is a precision system for measuring motion , position and orientation of RaginBot
GPS ANTENNAE
SICK LASER SENSOR
Laser sensors are non-contact measurement systems that scan their surroundings two-dimensionally (laser radio).
COMPUTERS
The obstacle detection machine is Dell Poweredge 750 with P4 3Ghz processor The path planner , disk logger and ntp machines are Travla Mini ITX
ALTERNATORS
They are the power sources for all electronic equip-ment aboard
EMC - SYSTEM
EMC driving control system provides a low effort and range of motion control solution for operation of the factory gas , brake, and steering controls of an automobile
Modified Steering Column & Servometer
Gas & Brake Servometer
UNIVERSITY OF LOUISIANA Lafayette
Ragin'Bot

Figure 12: Schematic view of CajunBot-2

was due to the 'false obstacles' it as seen on account of the sandstorm. Figure 11 depicts the path taken by the vehicle when there were no false obstacles on the track.

5.4 Testing in Controlled Environment

Post 2005 GC, CajunBot was retired. The following test was done on our proposed entry in the next Grand Challenge, the CajunBot-2 - successor for CajunBot. Focusing on the future we are concentrating our efforts on RaginBot which is a 2004 Jeep Rubicon with a much better suspensions and the ability to drive faster than CajunBot. Figure 12 shows schematic of CajunBot-2 with the sensors mounted.

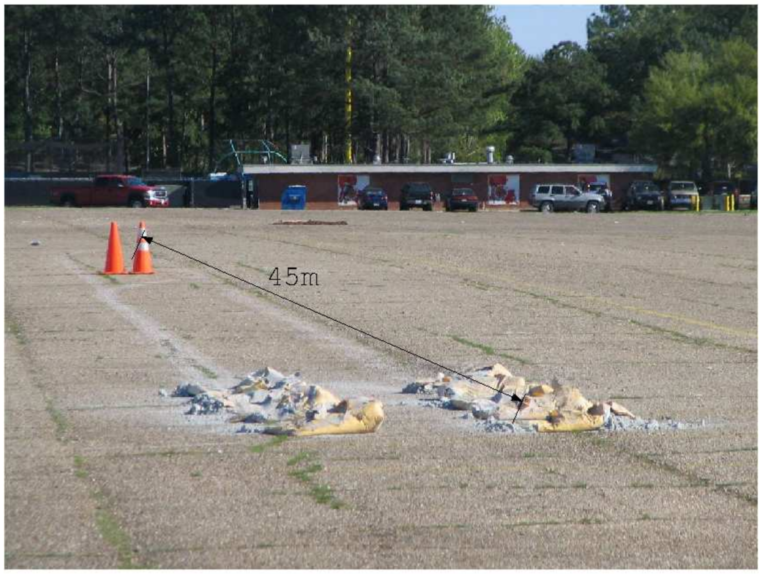

Figure 13: Experimental set up to study the effects of bumps

Experimental Setup. The Figure 13 details the experimental set up created to study the effects on bumps on the accuracy of obstacle detection. Bumps were created artificially using cement bags and obstacles (two cones) were placed at 45 meters from the first cement bag. Four runs were made on the same test track at 7m/s two with bumps and two without bumps.

Results. The Figure 14 and Figure 15 are the screen-shots of CajunBot visualizer when the obstacles are first recorded by the Terrain Obstacle Map, with and without bumps. The blue grid is the Terrain Obstacle Map Grid (TOM) and the red blocks are the recorded obstacles.

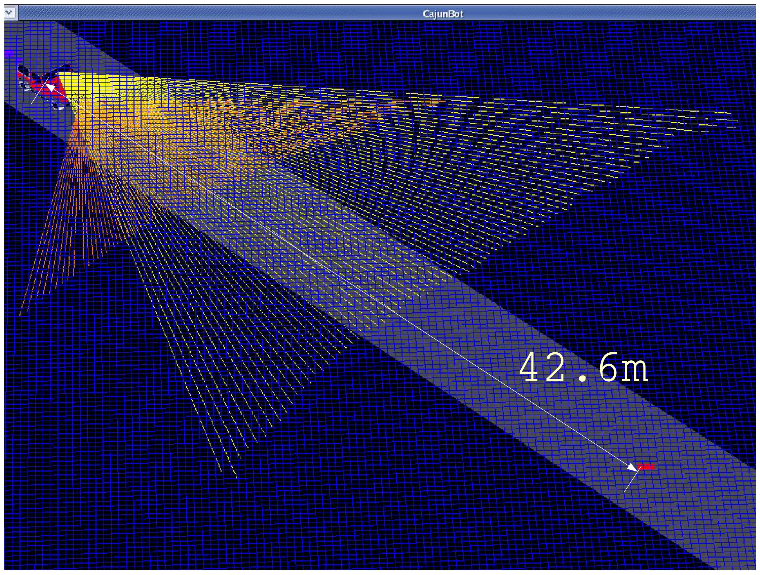

Figure 14: Visualizer showing obstacle detection with bumps

The orangle/yellow lines are the laser beams.

1. Bumps Utilization

   The Figure 14 and Figure 15 substantiate the observation that the algorithm detects further obstacles in the presence of bumps. In the presence of bumps the obstacle is first recorded in the TOM at 42.6m from the vehicle whereas the distance reduces to 28.5m in the absence of bumps. On comparing the values in Table 3 we conclude that bumps at average speeds do not result in false obstacles, nor do they hamper the accuracy of location of the obstacles as in either of the cases the obstacle is detected in

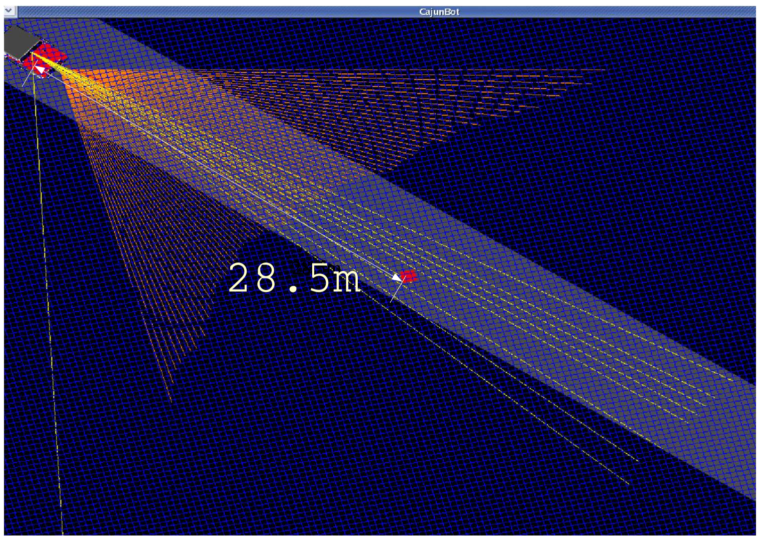

Figure 15: Visualizer showing obstacle detection without bumps

Table 3: Comparing runs with and without bumps

	Bumps	No Bumps
Distance to the Detected Obstacle	42.6m	28.5m
False Obstacles	Nil	Nil

Table 4: Scalability of the algorithm

# LIDARs	CPU for step 3.2	CPU for step 3.3, 3.4	Total CPU
1	$1 \times 2.6\%$	11.9%	14.5%
2	$2 \times 2.6\%$	11.6%	16.8%

the same TOM cell. The granuality of a TOM cell is 0.32m, hence, the maximum possible error in accuracy could be 0.32m, which is acceptable as it would point to the same TOM cell.

2. Scalability. To analyze the scalability, the algorithm can be logically decomposed into two steps based on the computations involved: 1) The sensor-specific computation, and, 2) The data-specific computation. Step 3.2 in Figure 3 is the sensor-specific computation. This step involves the transformation computation required to convert every beam to a corresponding global point. The cost of this step increases with every additional sensor. Steps 3.3 and 3.4 in Figure 3 are data specific. The computation involved Steps 3.3 and 3.4 are inversely proportional to the data density. With higher data density it requires lesser computations to form the necessary triangles. Also, Steps 3.3 and 3.4 are independent on the source of the data. For every additional sensor the computational cost of Step 3.2 increases and the computation cost of Steps 3.3, 3.4 decreases. Table 4 represents the data for one of the test runs. It substantiates the claim that the CPU utilization for Step 3.2 increases, whereas that for Steps 3.3, 3.4 decreases with every additional LIDAR. This feature of the algorithm helps in

scalability as with every additional sensor there is a linear increase in CPU cost. Looking at the data in Table 4, the approximate CPU Utilization for Obstacle Detection Module using 'n' sensors would be:

$$CPU\_Utilization = (\ (n\ *\ 2.6)\ +\ 11.9\ )\%$$

3. Accuracy of Results

   The logged data analysis revealed absolutely no false obstacles. The experiment was repeated with the speed varying from 5m/s to 15m/s with bumps, and there were no false obstacles. Also, the cone was detected in the same TOM cell every time.

CajunBot Vs. RaginBot. The primary difference between CajunBot and RaginBot in the context of obstacle detection is due to the following three factors:

1. CajunBot-2 is equippped with standard shock absorbers while CajunBot does not have any.

2. The top speed of CajunBot is significantly lesser than that of CajunBot-2.

3. Sensors are mounted on a same rigid frame on the CajunBot. In CajunBot, the sensors are physically close and also in close proximity to the INS. In RaginBot, currently, one sensor is mounted on the roof, close to the INS, and the second one on the front bumper.

Issue 1 mentioned above dampens the sudden shocks experienced by the sensors and hence, would be a reason for lesser false obstacles in CajunBot-2. The high speed, mentioned in Issue 2, would be a potential reason for induced error in Step 3.2 of the Figure 3, as the timestamping is done at every scan level as opposed to every beam level. At a speed of 15m/s the vehicle could have travelled 0.2m between the time the first and last beams of a single scan are produced. This error in recording the timestamp at every scan would hamper the accuracy of global points computations (Step 3.2). The solution for this would be to timestamp every beam as opposed to a scan. The mounting of sensors on CajunBot-2, mentioned in Issue 3, might lead to a 'tuning fork' sort of vibrations on the sensor mounted on the bumper. The INS reading might not correspond to the actual state of the sensor, which inturn might be a possible reason for the error induced in Step 3.2.

5.5 Testing in Simulated Environment - CBSim

The current version of simulator does not support creating bumpy terrain. So the effects of bumps is best studied in real field testing and by analysis of the log data. The simulators highlight, with respect to obstacle detection module, is its ability to the study the effects of different sensor orientations on the algorithm.

1. Sensor Orientation Independence

   When multiple sensors are used, the proposed algorithm does not require any particular mounting of the sensors. Though it is ideal to have the sensors at 0.3m separation on flat ground so that the in Step 3.3 more triangles are formed due to better temporal data density, it is not a requirement.

   Many teams in the 2005 DARPA Grand Challenge required a particular mounting configuration of sensors. Team GRAY [20] required moving sensors that were mounted vertically. Team GRAY's algorithm was tightly coupled to the mounting of their sensors, they used the discontinuities in a single scan in the direction of vehicle's motion to detect obstacles. This approach may not work if the sensors are mounted horizontally as there would no data to detect discontinuities in the heading direction. Red Team [21] required two of their LIDAR's stacked up parallely near the from bumper to detect the change in slope.

   The following test was performed to prove the sensor orientation independence of the

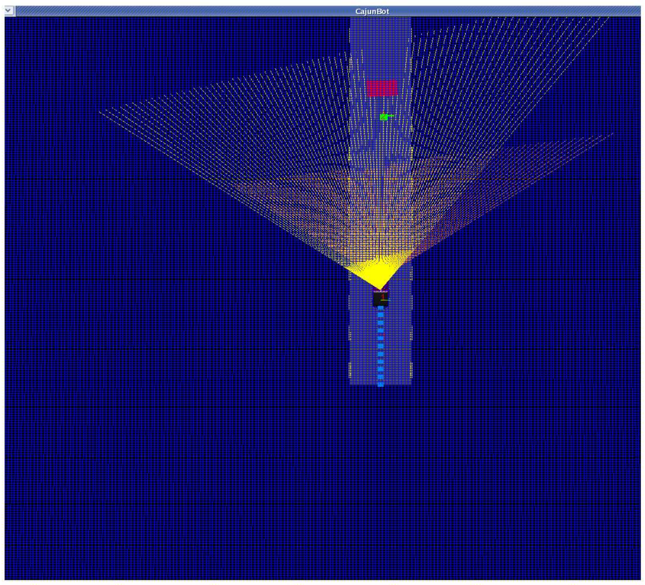

Figure 16: Obstacle detection with sensor configuration-1

proposed algorithm. In CBSim, the algorithm was run multiple times changing nothing but the LIDAR sensor's orientation. The terrain, path, position and type of obstacles were same in between multiple runs. The results were observed on CBViz, the graphical interface to CBSim.

Figure 16 and Figure 17 depicts the screen shots of the visualizer, CBViz, for the experiment. It can be seen that the orientation and position of the LIDARs is different in both the runs.

Figure 16 and Figure 17 point to the fact that the same algorithm can work with any orientation of the sensors. The figures depict that the obstacles are detected at the same correct location irrespective of the orientation and mounting of the sensors. As the sensors are pointing far, in Figure 16, the obstacle is detected at 18m in front of the vehicle where as in Figure 17 the detection distance reduces to 7m as the sensors are pointing more closer to the vehicle.

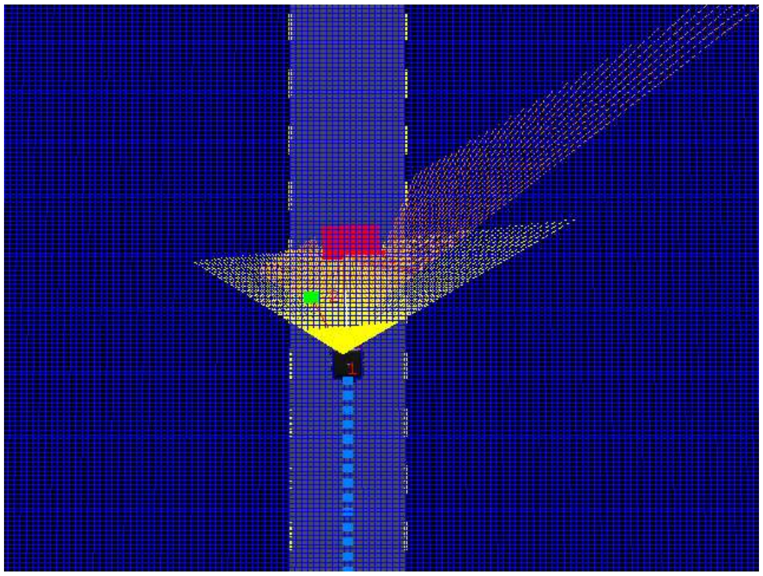
Figure 17: Obstacle detection with sensor configuration-2

The sensor orientation independence is achieved because the algorithm does not need to know the source of the sensor data, nor is the data bound to a particular sensor. Data from all the sensors is pooled into a single repository, using these points the triangles are formed in the Step 3.3.

2. Evaluation on Obstacles of Different Shapes

   Obstacles of different shapes, viz cone, cylinder (pole) and cuboid were used to evaluate the affect of shape of the obstacles on the algorithm. Also, the effect of speed of the vehicle on the detection rate of these obstacles is studied. Multiple runs were made in the simulator on the same route changing nothing but the shape of the obstacle. In every run the obstacle was placed at the same location, 35 meters from the starting point of the track. The top and bottom sensors were pointing at 16.3 and 16 meters. The simulated environment was on a flat terrain and had no bumps, also, there were no GPS related errors like GPS shift and GPS spike. In every run, the distance from the vehicle at which the obstacle was first marked was used to compare the

Table 5: Comparing runs with different obstacle shapes

Obstacle shape	Dimensions (m)	Speed (m/s)	Distance to obstacle (m)	Sub-module
Pole	(r,l) = (0.1, 3)	4	15.2	HD
Pole	(r,l) = (0.1, 3)	10	14.6	HD
Cuboid	(l,b,h) = (1, 0.3, 1)	4	15.8	AS, RS
Cuboid	(l,b,h) = (1, 0.3, 1)	10	15.3	AS, RS
Cone	(r,l) = (0.4, 0.75)	4	15.6	AS, RS
Cone	(r,l) = (0.4, 0.75)	10	15.0	AS, RS

results. Also, the corresponding sub-module of the obstacle detection algorithm (high absolute slope (AS), high relative slope (RS) or height discontinuity(HD)) which was responsible for the particular obstacles to be detected was also recorded in table 5.

Table 5 depicts that, among the three shapes, there is 4,3.2 and 3.9 percent decrease in the distance of the detected obstacle respectively when the speed of the vehicle increases by 150 percent. As more triangles could be formed on the surface of cuboid than on a narrow cylinder, the AS and RS sub-modules were responsible to detect a cuboid as opposed to HD sub-module being the primary one for detecting the cylinder (pole) type of obstacle. As more triangles could be formed on the surface of a cuboid as compared to a cone or a cylinder, the cuboid type obstacles get detected at 3.9 percent earlier than the cylinder and 1.2 percent earlier than the cone type obstacle at 4m/s. At 10 m/s the distance to detect cuboid type obstacles is 4.7 percent lesser than the cylinder and 2 percent lesser than the cone type obstacle.

The efficient and scalable implementation is due to two factors. First, in Step 3.3 it is not necessary that the triangles be created using global points observed by the same LIDAR. The data may be from multiple LIDARs. The only requirement is that the triangles created satisfy the spatio-temporal constraints. The second factor is that we utilize an efficient data structure for maintaining the 4-D space. Though the 4-D space is infinite, an efficient representation is achieved from the observation that only the most recent three seconds of the space need to be represented. This follows from the temporal constraint and that one point of each triangle created in Step 3.3 is always from the most recent scan.

6 Conclusion and Future Work

6.1 Conclusion

This thesis developed an algorithm for terrain mapping and obstacle detection in off road environment. A scalable, robust and accurate technique for obstacle detection using LIDAR sensor is described. To recap, the following features of the Terrain Mapping Module enables it to utilize bumps to improve obstactle detection distance.

- A rigid frame for mounting all sensors.
- Fusing mutually consistent LIDAR scan with INS data based on the time of production of data.
- Using 4-D space and spatio-temporal constraints for creating triangles to compute the slope of locations in the 3-D world.

The algorithm was tested on CajunBot, the finalist in the 2005 DARPA Grand Challenge, on CajunBot-2 - a 2004 Jeep Rubicon and in simulated environment, the CBSim. The algorithm was evaluated as the vehicle was at various speeds from 3 m/s to 20 m/s in a step of 1 m/s. Also, the effects of the bumps on the algorithm was studied by post processing the Grand Challenge data and by creating specific test cases. The sensor orientation independence was studied in CBSim. The algorithm was also evaluated on different obstacle shapes and a comparison was drawn.

The results showed that the algorithm detects obstacles faster in the presence of bumps. The bumps did not cause any false obstacles either. The results also indicated the robustness, scalability and sensor-orientation independence of the algorithm.

In the current implementation every scan is time stamped with a single time as opposed to every beam. This could be a possible source of error when the vehicle is moving at high speeds. It would be worth verifying the results based on beam level time stamping. In future it would be interesting to use moving LIDAR's to get a better 3-D point cloud and test the algorithm. Further analysis of the 3-D point cloud to determine the nature of obstacle like vegetation, rocks, bushes, etc, would be worth investigating.

6.2 Future Work

In the current implementation every LIDAR scan is time stamped with a single time as opposed to every beam. This implies all the 180 beams of a scan would have the same time stamp. This is not accurate as the scans are generated at 75 hertz frequency. The actual time difference between the first and the last beam of a scan is 0.013 seconds. If the vehicle is traveling at 25 m/s then it would have already moved by 0.33 m between the time of first and last beams of a scan. Also, the INS provides vehicle orientation data at 100 hertz. In case of a bumpy terrain, the first beam and the last one may not be experiencing same bump. This would cause an error in the global points computation (Step 3.2 in Figure 3). It would be worth verifying the results based on beam level time stamping.

Currently the LIDARs are statically mounted, they do not move. The static LIDARs only scan a single line at a time limiting the area that is scanned. If they were moving, the LIDARs could potentially generate a better point cloud in terms of the area scanned. In future it would be interesting to use moving LIDAR's to get a better 3-D point cloud and test the algorithm.

Right now the algorithm can only detect if the a given surface is an obstacle or not. It does not give any clue about the type of obstacle, like vegetation, rock etc. A foot long grass could be traversable by the vehicle but not the same sized rock. Further analysis of the 3-D point cloud to determine the nature of obstacle like vegetation, rocks, bushes, etc, would be worth investigating.

References

[1] "Mars rover from nasa," http://marsrovers.nasa.gov/home/.

[2] "irobot company website," http://www.irobot.com/.

[3] B. Southhall, T. Hague, A. Marchant, and B. Buxton, "Autonomous crop treatment robot: Part i. a kalman filter model for localization and crop/weed classification." January 2002, International Journal of Robotics Research.

[4] "Darpa grand challenge website," http://www.darpa.mil/grandchallenge/index.asp.

[5] "Darpa website," http://www.darpa.mil.

[6] "Cajunbot website," http://www.cajunbot.com.

[7] S. Golconda, "Steering control for a skid steered autonomous ground vehicle at varying speed," Master's thesis, University of Louisiana at Lafayette, University of Louisiana at Lafayette, Lafayatte, LA, February 2005.

[8] "Sick website, lms 221 manual," http://www.sick.com.

[9] S. Spry, A. Girard, M. Drew, and J. K. Hedrick, "Sensing and control issues in intelligent cruise control," University of California at Berkeley, Tech. Rep., October 2002.

[10] W. D. Jones, "Keeping cars from crashing," September 2001, IEEE Sprectrum.

[11] A. Talukder, R. Manduchi, R. Castano, K. Owens, L. Matthies, A. Castano, and R. Begg, "Autonomous terrian charecterisation and modelling for dynamic control of unmanned vahicles," October 2002, Proceedings of IEEE/RSJ, Intl. Conference on Intelligent Robots and Systems.

[12] L. Henriksen and E. Krotov, “Natural terrain harard detection with a laser rangefinder,” 1997, IEEE Conference on RObotics and Automation.

[13] T. Chang, Tsai-Hong, S. Legowik, and M. N. Abrams, “Concealment and obstacle detection for autonomous driving,” NIST,” International Association of Science and Technology for Development - Robotics and Applications, October 1999.

[14] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, “Obstacle detection and terrain classification for autonomous off-road navigation,” University of California at Santa Cruz and California Institute of Technology, Tech. Rep., 2005.

[15] J. Roberts and P. Corke, “Obstacle detection for a mining vehicle using 2d laser,” CSIRO Manufacturing Science and Technology, Tech. Rep.

[16] L. Matthies and A. Rankin, “Negative obstacle detection by thermal signature,” California Institute of Technology, Tech. Rep.

[17] A. Lakhotia, S. Golconda, A. Maida, P. Mejia, A. Puntambekar, and G. Seetharaman, “Cajunbot: Architecture and algorithms,” *Journal of Field Robotics*, January 2006.

[18] R. Katzenbeisser, “About the calibration of lidar sensors, ISPRS Workshop ”3-D Reconstruction from Airborne Laser-Scanner and InSAR data”,” October 2003.

[19] E. Abbott and D. Powell, “Land-vehicle navigation using gps,” January 1999, Proceedings of the IEEE.

[20] P. G. Trepagnier, P. M. Kinney, J. E. Nagel, M. T. Dooner, and J. S. Pearce, “Team gray darpa grand challenge 2005 technical paper,” Gray Insurance Company, Inc.,” CSC Technical Report, August 2005.

[21] W. R. Whittaker, "Red team darpa grand challenge 2005 technical paper," Carnegie Mellon University," CSC Technical Report, August 2005.

Abstract

Unmanned autonomous navigation in off-road conditions unfolds many interesting challenges in the field of obstacle detection and terrain mapping due to the unstructured and unpredictable nature of the terrain. Simple issues like the bumps experienced by the vehicle and sensors complicate the algorithms. Most of the algorithms depend on sensor stabilization hardware like the gimbal and vehicle suspensions to dampen the vibration experienced by the sensors. This not only increases the cost of production of these robots, but is also prone to mechanical failures.

This thesis presents an algorithm for terrain mapping and detecting obstacles in off-road environment using LIDAR sensors, without the need for any sensor stabilization and vehicle suspension. The algorithm's highlight is its ability to use the bumps experienced by the vehicle to its advantage to detect farther obstacles.

The proposed algorithm was implemented on an Unmanned Autonomous Robot, CajunBot, finalist in the DARPA Grand Challenge 2005.

Biographical Sketch

Amit Puntambekar was born in Pune, India on November 14, 1981. He graduated with a bachelor's degree in Computer Science in June 2003 from Pune University, Pune, India. He entered the master's program in Computer Science at the University of Louisiana at Lafayette in Fall 2004. Upon completion of this degree, Amit Puntambekar is planning to pursue a PhD in the area of Robotics.