Obstacle Detection for High Speed Autonomous Navigation

Sanjiv Singh
Paul Keller

Field Robotics Center
Carnegie Mellon University
Pittsburgh PA 15213

Abstract

We present several schemes for Obstacle Detection for autonomous vehicles traveling at high speeds (above 5m/s). In particular, we discuss schemes that make a globally"flat-world" assumption and ignore vehicle pitch motion. Next, we examine methods that relax the above assumptions. In each case we discuss the strengths and weakness of the solutions proposed. Experimental and simulation results are presented.

1. Introduction

We have previously proposed a paradigm for high speed autonomous navigation called *position based path tracking*[1,2]. In this paradigm, a vehicle tracks an explicitly specified path by continuously making steering corrections using position feedback from accurate inertial sensors. Vehicle speed is determined by choosing the highest speed that meets all of several constraints like maximum lateral acceleration and distance to nearby obstacles. To this end, we propose several schemes for obstacle detection.

More formally, *obstacle detection* is the determination of whether a given space is clear from obstructions for safe travel by an autonomous vehicle. Specifically there are three goals: to detect obstacles in time when they exist, to identify the correct obstacles, and, to not be fooled by objects that are not in the path of the vehicle.

Previous work in the area has been mostly confined to robots moving at slow speeds because the task of perception is usually to determine the cues necessary to navigate in a cluttered world. There are two reasons for this. Active range sensing is a commonly used modality for obstacle detection because the sensors directly supply range from the robot to the world. Active range sensors (laser scanners and sonar) are typically slow, providing in some cases, only two frames (snap shots of the world) per second. Secondly, irrespective of whether active sensing or passive sensing (reflectance images) is used, an enormous amount of data must be processed in real-time to discern obstacles in the world. Hence, successes in this area have been limited to speeds between 2-3m/s even though very sophisticated laser ranging and systolic array processors have been used for sensing and computation[3,4,5]. More recently, some researchers have demonstrated obstacle detection using passsive vision along with special purpose hardware at speeds up to 14m/s on straight stretches of flat highway[6]. Passive vision, however, is limited in scope because it is prone to poor illunination and low contrast conditions.

Our approach has been two fold. Firstly, we use accurate position estimation to determine the location of the autonomous vehicle on the path it is to follow. This enables us to narrow our search for obstacles to that part of the field of view of our sensor where we expect the future path of the vehicle to lie. Instead of finding obejcts in the world so that we can steer around them, we "detect" obstacles when they exist in the path of the vehicle and stop for them. Secondly, we propose the use of laser scanners that produce sparser but more frequent snap shots of the world than have been used by researchers in the past. We have used Cyclone, a laser scanner designed and fabricated inhouse for our initial experiments. Cyclone provides a single scan line (1000 points in a 180 degree field of view) of range data at upto 8 scanlines/sec in our initial experiments.

We chose to begin our experimentation with the assumption that the vehicle travels in a 2-D world with the further simplification that the vehicle does not pitch. These asumptions were relaxed in order to consider a 3-D world in which it is necessary to aim a laser sensor towards the ground in front of the vehicle. We have developed two different methods. One that considers only a few scanlines from a range image, either because the laser is only able to provide sparse data or if a dense range image is available, for the sake of efficiency. The

other method explores the scenario where a dense range map is available but must be processed very rapidly. To this end, we have designed schemes similar to those proposed by Dunlay[3], but, are more efficient, in the amount of computation required.

2. Obstacle Detection in a Flat World

We chose to begin the investigation in a two-dimensional world. Two techniques were designed with the simplification that the vehicle travels in a 2-D world. The first method is based on identifying obstacles by matching range data obtained from a range scanner on the moving vehicle, to a world model. The second method checks a bounded region around the path to be traversed for range data corresponding to objects. We present a condensed version of these techniques. A more complete description can be found in [9]. Figure 1 shows the range scanner mounted on the front of the vehicle.

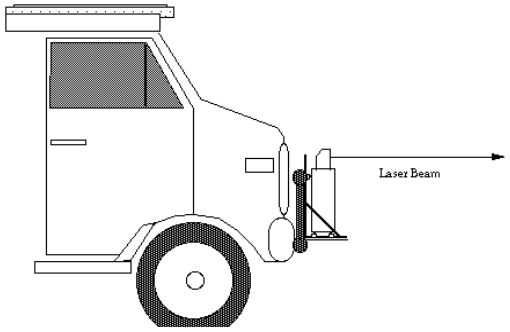

Figure 1: Scanner configuration in a flat world

2.1. Profile Matching.

Profile matching is based on the notion of matching a hypothesized range profile using vehicle position and a world model, with a range profile inferred from range data obtained while in motion. This requires explicit modeling of the world in which the autonomous vehicle travels. Although this scheme was fairly successful in simulation, we noted the following:

- The real world is very hard to model. Building and maintaining a consistent world model is often a weak link in an actual implementation.
- Profile matching can only detect if something has changed in the environment but cannot distinguish between unexpected objects that are on the road ahead and those that are clear of the vehicle path. A further level of processing is necessary to determine if the unexpected obstacle might interfere with the vehicle.
- Errors in position estimation result in errors in the matching process, even with perfect range data.

2.2. Clearance Checking

The above observations led us to focus our attention on the path ahead rather than examining the entire field of view. More precisely, we are interested in the space which the vehicle could possibly sweep out in its future travel. A simple method is to search a section of the path immediately ahead of the vehicle for obstacles. For a vehicle to travel unimpeded, no objects should lie in this area. This method accrues the following advantages:

- There is no need to model the world. There is an implicit model: the road ahead should be clear of obstacles.
- Any objects detected in the path of the vehicle are immediately relevant.
- Errors in position estimation result in the vehicle deviating from the reference path by the amount of the error. However, since the tracking and obstacle detection schemes use the same position feedback, the same deviated path is searched for obstacles.

The space that the vehicle could possibly sweep out in the near future (collision zone) is identified periodically and range data is checked to see if any of the corresponding reflecting points lie within the zone. Figure 2 shows a scenario in which the clearance checking scheme isolates range measurements due to an object within the collision zone.

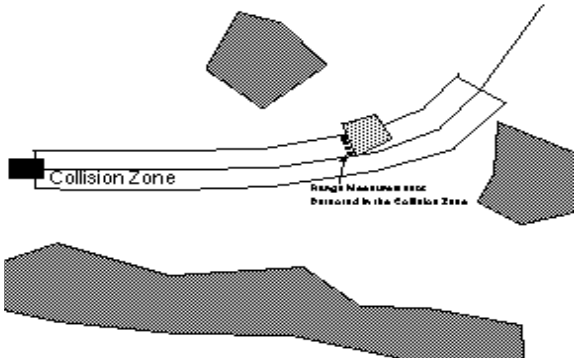

Figure 2: Obstacle detected inside the collision zone

Clearance Checking is given an explicit specification of the path on which the vehicle is to travel in the form of a sequential list of posture- a specification of position, orientation and curvature that the vehicle is to attain. In addition, vehicle position is available by querying an inertial navigation system. The procedure is initiated whenever a scanline of range data becomes

available. Then, the following steps are executed.

1. The current vehicle position is obtained and the current scanner position S_p is computed. P_k, the posture on the reference path closest to S_p is determined as the posture with the smallest lateral distance to S_p.

2. *n* future postures of the vehicle are transformed into vehicle coordinates using a transformation from global to local coordinates. Now, the list of postures local to S_p *i*s given by p_j (p_{j_x}, p_{j_y}, p_{j_θ}) in the following manner:

$$p_{j_X} = (P_{i_X} - S_X)*\cos(S_\theta) + (P_{i_Y} - S_Y)*\sin(S_\theta) \quad (1)$$
$$p_{j_Y} = (P_{i_Y} - S_y)*\cos(S_\theta) + (S_x - P_{i_X})*\sin(S_\theta) \quad (2)$$
$$p_{j_\theta} = S_\theta - P_{i_\theta} \quad (3)$$
$$(i : k+1 .. n; j = i - k)$$

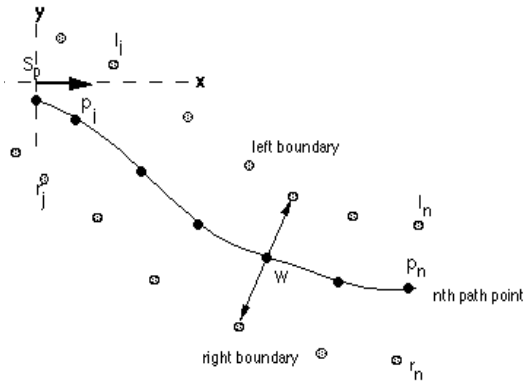

Figure 3: The path and edges in local coordinates

3. The locations of the left and right edges of the collision zone are computed in local coordinates using *W*, the total width of the collision zone, as shown in figure 3. $W = W_V + 2(S_\varepsilon)$ where W_V is the largest dimension of the vehicle and S_ε is the maximum error in position sensing. The coordinates of the left and the right edges of the road (in local coordinates) corresponding to p_j are given by r_j and l_j respectively:

$$r_{j_X} = p_{j_X} + \sin(p_{j_\theta}) * W/2 \quad (4)$$
$$r_{j_Y} = p_{j_Y} - \cos(p_{j_\theta}) * W/2 \quad (5)$$
$$l_{j_X} = p_{j_X} - \sin(p_{j_\theta}) * W/2 \quad (6)$$
$$l_{j_Y} = p_{j_Y} + \cos(p_{j_\theta}) * W/2 \quad (7)$$

4. The length of the collision zone, C_l is determined by taking the minimum of a preset length and the distance along the x axis to a point ahead where the path is oriented more than 90 degrees relative to S_p.

5. A distance C_l ahead of the vehicle is tessellated into *m* intervals (0.5 m long) and the collision zone boundaries are transformed into a hash table indexed by fractions of this distance. For each of the *m* intervals, a left and right edge point of the collision zone is found through interpolation (l'_h,r'_h) as in figure 4.

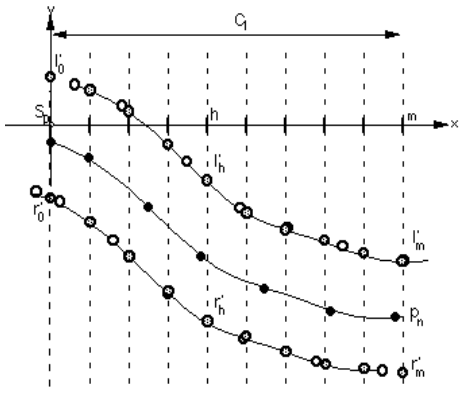

Figure 4: The collision zone stored as a hash table

6. Every range point in the scanline of range data is converted into cartesian coordinates (x_r, y_r) with its reference frame located at *Sp*. x_r is used to obtain the index into the hash table, k_r.

7. A range point lies inside the collision zone if $l'_{k_r} > y_r > r'_{k_r}$. An obstacle is indicated if more than a preset number of range points are found within the collision zone.

Experiments with a target 0.5m wide mounted on a cart moving at 6m/s towards a stationary scanner, demonstrated the ability to detect and bring the cart to a halt in sufficient time. Implementation on the Nav-Lab was partly limited by the ability to decelerate the vehicle; we were able to stop for similar targets at speeds of up to 3m/s.

Some limitations were noted. Since the scan is horizontal to the ground plane, obstacles that are lower than the height of the beam are not detected. Pitch motion of the vehicle as well as graded surfaces pose problems because reflections from the ground are interpreted as obstacles, or conversely the beam completely misses legitimate obstacles.

3. 3-D Obstacle Detection Schemes

As indicated above, range scans parallel to the ground plane are not sufficient to detect obstacles for a vehicle travelling at high speeds. Rather, we found it necessary to angle the scanner down toward the ground in front of the vehicle. This persepctive allows for consideration of a 3D world as well as range scanners that produce multiple lines of range data.

Conceptually, the extension to 3-D is a fusion of the schemes discussed above. Here we have attempted to model the road surface on which the vehicle is to travel. In addition, the collision zone idea is used to

limit treatment of only the relevant part of the world - the area that is swept out by the vehicle as it travels. The selected data is compared to a road model. Obstacles are indicated by deviations of range data from a road model.

Two variations of the 3-D schemes were considered:

- Selected Scan Method - a few scanlines from a complete range image comprised of multiple adjacent scanlines are selected, and an elaborate fit of the road profile (a cross section of the road transverse to the direction of travel) is developed. Roads are assumed to be graded or designed roads, generally smooth and with crowns and/or banks.
- Full Frame Analysis - the entire range image is used and a simpler fit is performed on all the range data corresponding to the road width being verified. Roads are assumed to be graded but neither banked nor crowned. This approach is only tractable when multiple lines of range data and sufficient real-time computing resources are available.

3.1. Scanner Model

Figure 5 shows a side view and a top view of the scanner model used.

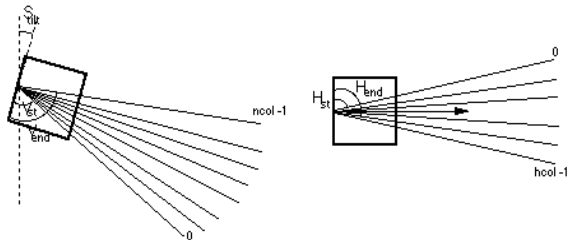

Figure 5: **A generalized Scanner model**

A point in the world *(x,y,z)*, with respect to the scanner coordinate frame can be projected into an image plane to obtain the azimuth and vertical angles (θ,ϕ) in the following manner:

$$\theta = \text{atan2}(y,x) \quad (8)$$

$$\phi = -\text{atan}((z - S_Z/x)) + \pi/2 \quad (9)$$

where S_Z is the height of the scanner from the ground plane.

Given a discrete image plane, as in one that has a finite field of view and resolution, row and column numbers for the point corresponding to the angles (θ,ϕ) are given by:

$$row = [(\phi - V_{st} + S_{tilt})/(V_{end} - V_{st})] * (nrows -1) \quad (10)$$

$$col = [(\theta - H_{st})/(H_{end} - H_{st})] * (ncols -1) \quad (11)$$

where S_{tilt} is the tilt of the scanner, H_{st}, H_{end} are the starting and the ending angles in the azimuth axis, V_{st}, V_{end} are the starting and the ending angles in the vertical axis, and, *nrows* and *ncols* denote the number of pixels in the horizontal axis and the vertical axis, respectively.

3.2. Using Selected Scanlines

This method selects to process only a few but relevant lines, based on vehicle pitch and vehicle speed. The motivation for this method is two fold. Firstly, searching an area in front of the vehicle for obstacles usually reduces to finding obstacles in a small subset of the vertical field of view. Secondly, such a method allows for a wide range of scanner configurations: from a single line scanner to an arbitrary multiple line scanner.

A second order polynomial is fit to the height profile of each useful scan line. That is, an estimate of average height, bank and crown is made for each processed scanline. The algorithm is recursive so that a single estimate is saved for bank and crown to represent all past data. When a new height profile is considered, it is matched against the road model that has been built up from previous scanlines of range data. The amount of weight to be placed on old data as opposed to new data can be adjusted with a trade-off between noise tolerance and sensitivity to terrain changes. That is, in the case that old data is weighted more, the algorithm is more tolerant to noise, but also becomes less "adaptive" and sharp changes in the terrain can be erroneously interpreted as obstacles.

3.2.1. Method

First, as in Clearance Checking, the road edges are determined. However, the hash table constructed before is not useful; we will process the data differently. When a new frame of range data is available the following steps are taken:

Steps 1, 2, and 3 are the same as in the case of Clearance Checking.

4. Based on the current speed and an assumed deceleration of the vehicle, a stopping distance D_S is computed.
5. The distance that the vehicle will travel in the time that it takes to complete one full scan is computed. This distance is divided by the number of scanlines *n* to be examined during this interval, yielding *n* equally spaced road locations.

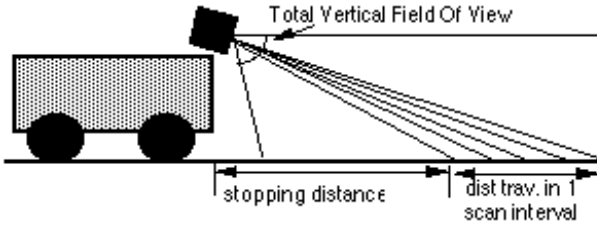

Figure 6: Scan Line Distribution

The magnitude of n is based on the maximum speed of the vehicle, the size of the smallest obstacle to be detected and the computing resources available. The row numbers that correspond to these road locations are then identified as in Eqn. 10 for extraction from the full scan. This process is shown in figure 6.

6. A spline interpolation is performed to define the left and right road edges (in global coordinates) at any arbitrary distance in front of the vehicle. For each of the n lines chosen in step 5, the points of intersection of the scanline with the left and right edges of the road are determined in global coordinates (r_x, r_y) & (l_x, l_y).

7. These edge points are projected onto the image plane, giving the pixel column numbers corresponding to the left and right edges of the road (l,r) as given by Eqn 11.

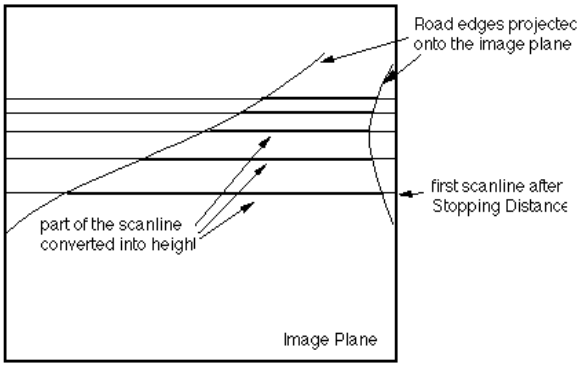

Figure 7: Selected scan lines in the image plane with delimited sections corresponding to the road surface

8. A delimited scanline is created by selecting only those pixels which lie between l and r. Figure 7 shows the view of the scanlines from figure 6 in the image plane.

9. The stopping distance is set as the maximum distance at which both edges of the road are visible within the bounds of the image plane.

10. Each delimited scanline of range values, r_i is converted into a height profile Z_i using a transform from cylindrical to cartesian coordinates:

$$Z_{i-l} = S_z + r_{i-l} * \cos(\theta_{i-l}) * \cos(\phi_{i-l}) \qquad (12)$$

($i = l... r$)

11. This profile is processed to identify unknown objects in two different ways. First, the height profile Z_i is convolved with a 7 point function **U** (first derivative of a gaussian) to obtain Y_i in the following manner:

$$Y_i = \Sigma Z_{k-j} * U_j \qquad (j = -3 .. 3) \qquad (13)$$

$$U_j = [\ -0.5, -1, -2, 0, 2, 1, 0.5] \qquad (14)$$

The resulting convolution Y_k, is very sensitive to rapid changes in the height profile and gives a very high value in the neighborhood of "edges" in the profile. Obstacles are found by looking for values of the convolution greater than a preset threshold (determined by computing the value of the convolution for the maximum expected noise).

The second method filters the height profile to obtain a weighted second order fit to estimate the shape of the profile. Average height is computed from each profile, but bank and crown are computed based on previous data. Sensitivity to new data can be tuned by reducing the weights corresponding to old data. The weighted road profile is checked against the actual road profile and deviations that exceed a preset threshold are counted as belonging to an obstacle.

12. If an obstacle is found, the vehicle is brought to a halt.

After each frame of range data has been processed, a safe distance is sent to the vehicle controller. If this is smaller than the minimum required safe distance, then the vehicle is halted. This allows for the vehicle to start moving as soon as the obstacles are removed.

3.2.2. Results

Figure 8 compares edge detection and thresholding in simulation. Each frame of range data is 256 X256 but only 10 lines of range data are selected for processing. Lines at the bottom of the figure correspond to scanlines that are close to the scanner on the ground (i.e they have the steepest elevation angles). In (a) each height profile is compared to a weighted quadratic fit that uses the recursively built up road model. The bands around each height profile indicate limits of a height threshold used. In (b), the same scanlines are checked for edges. The obstacle is detected by both methods, but the edge operator detects the obstacle about 1m before the threshold is exceeded.

Each of *edge detection* and *road model matching* has strengths and weaknesses. While edge operators are useful, edges are not very distinct in range images for example in the case of a large object lying across the entire road region. Model matching has its problems in that the proper weighting between previous and cur-

rent data must be found through experimentation. One danger of placing too much weight on new data is that an obstacle itself influences the recursive fit making it harder to detect in future height profiles.

Three separate scenarios were tested. The first scenario was a figure-eight path on flat ground (total length 280m, the larger loop with a minimum radius of curvature of 40m, and the other loop with a minimum radius of curvature of 25m). This is shown in figure 9. In this case the vehicle is travelling at 3m/s and the scanner provides 3 scan/s.

Figure 8: Comparison of (a) height threshold and (b) edge detection

In figure 9, the first two objects are off the road while the next 3 are on the road. In each case the correct identification was made. The other two scenarios involved straight but inclined paths with a varying transition from 0% to 10% grade. In one case, the transition occurred over a distance of 10 m, and in the second case over that of 100 m.

Figure 9: Scanlines on a figure 8 path

If only a small number of scan lines are examined from the entire frame, and especially if the extracted scan lines are chosen so that there is no overlap between the frames (as in figure 6), then a dangerous obstacle may appear in a single scanline only. This makes the algorithm sensitive to erroneous data, that is, a single aberrant scanline could cause the vehicle to come to a complete halt. We were able to detect the obstacles, with faces of 0.5m x 0.5 m by allowing separation of scanlines on the ground to reach upto 1m. Computationally, we could process 6 scans/sec but due to NavLab's limited acceleration, we were only able to conduct successfull experiments betweeen 3 to 4 m/s.

3.3. Full Frame Analysis

In this approach to obstacle detection, an entire multiline range image is processed. Obstacles are indicated by pixels in the image which lie on the road and deviate significantly from the expected road height. Groups of these pixels are gathered together into "blobs" and are treated as a unit. This process is called obstacle extraction since the final result is a blob of pixels extracted from a two dimensional array of data.

Obstacle extraction proceeds by first projecting the vehicle path into the image plane. Range data is transformed into height data and a curve is fit to the height at the center of the road. Finally the actual road height is thresholded against the modeled height expectation and obstacles are extracted by grouping points that fall outside the threshold.

3.3.1. Method

In this case also, we start as in the previous two methods with steps 1, 2, 3 from Clearance Checking. The subsequent steps are discussed in detail below:

In order to use all of the available data, the images must be processed at the frame rate. For this reason, most of the computations in the obstacle extraction algorithm are done in the image plane. By projecting the path onto the image, a large portion of the image can be ignored, and many needless computations avoided. Note that this is contrary to standard methods of using range data e.g., as in [4].

Assuming that the vehicle path is specified at regular intervals, the current vehicle position can be used to locate the path segment lying in front of the scanner. This path is transformed from world coordinates into image coordinates by projecting the points corresponding to the road edges into the image plane. A cubic spline is used to interpolate between the edge points. Then, for each row in the image, pixels lying between the road edges are transformed from range data to road heights; outlying pixels are discarded and

not processed any further.

In the Martin Marietta system, the height of the road was determined by the height of the road at the centerline. This method fails when an obstacle lies in the center of the road. In this case the normal road height is assumed to be the obstacle height and the thresholding method suggested by Dunlay would recover "pits" (instead of obstacles) in the space between the obstacle and the road edges. If an obstacle were to span the entire road width, it would not be found at all. Therefore, we suggest a model where the grade is smoothed over a longitudinal section on the road. A third-order least-squares fit is applied to the height data at the center of each image row on the path. This has the effect of modeling the general trend of the road (up and down hills) as well as filtering out the effects of noise and small objects lying in the center of the road.

Obstacles are located by applying a height threshold on each pixel lying between the road edges. This threshold operator is referenced against the *expected* height, as predicted by the third order fit of the road centerline. In this manner, a hill is not considered an obstacle since the height expectation and the measured height should match very closely. A real obstacle does not significantly affect expected road height, due to the least squares fit and therefore is readily found by thresholding. The result of thresholding is a binary image suitable for blob extraction.

A standard blob coloring algorithm groups pixels of similar height. By grouping pixels together into blobs, the obstacles can be treated as whole units amenable to further processing.

3.3.2. Results

For most cases, this algorithm is an effective means of obstacle detection. Figure 10(a) shows the results when an obstacle is present. Figure 10(b) highlights the obstacle that was found in the middle of the road.

Figure 10(a): Fitted road profile in presence of obstacle

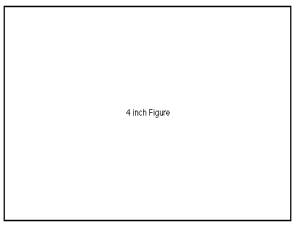

Figure 10(b): Obstacle detected in road region

There are a few cases where this algorithm fails. Road crown and bank cause problems due to lateral changes in the road height. Since the road height expectation is determined by the center of the path, whenever the heights of the edges differ significantly from that of the center then either spurious obstacles are found or valid obstacles missed. Another problem arises when an obstacle lies in the center of the road either at the start of the scanner data or near the horizon. In these cases, the height model tends to follow the obstacle closely since there are no surrounding points to "pull" the fit back down. This results in the road model reflecting the height of the obstacle rather than the height of the road, as can be seen in Figure 11(a).

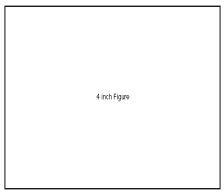

Figure 11(a): Fitted road profile in presence of an obstacle and steep grade

Figure 11(b): Erroneous obstacle found

There are just enough data points collected on the obstacle to prevent the fit from conforming to the steep hill. The result is that the algorithm is fooled into thinking that there is an obstacle at the far end of the road as shown in figure 11(b) . Note that although a bogus obstacle is found, the vehicle would still be slowed down and further processing would show that there is only one real obstacle.

4. Obstacle Detection: Conclusions

Our experiments using the flat world assumption proved to be insufficient because obstacles in realistic outdoor terrain could not be detected reliably. This is mainly because undulating terrain and vehicle pitch can cause either for obstacles to be missed or for false obstacles to be detected. Extending our methods to explicitly consider the road over which the vehicle is to travel made our algorithms less sensitive to the real-life problems mentioned above. Our final experiments with the NavLab and the Cyclone scanner were timed at 4m/s. We were able to detect obstacles in simulation of a scanner mounted on a vehicle traveling at 11m/s, given 256x256 range images.

Performance must be evaluated based on the fastest speed the vehicle is to travel and the smallest object that can be detected. Comparision of the methods presented above reveals that there is a tradeoff between reliability and performance. Full frame analysis is clearly the best technique; however, both the sensing and computation necessary are prohibitvely expensive.

Some caveats should be mentioned. It is possible that no obstacles are found in the collision zone due to occlusions by objects that are outside the collision zone. To guarantee a collision free path, a conservative estimate of the length of the collision zone must be made every time the path turns. Scanner frame rate becomes a factor at high speeds. When traveling around a bend, objects appear shifted in space so that some objects which are not on the path may sometimes appear to be on the path and vice versa.

A few improvements are suggested. In the selected scan approach, the grade of the road profile is calculated for each scan, and the grade on inclined roads is not predicted well. A similar effect is caused by small pitching motions of the vehicle. Both can result in a large difference in the grade being computed in successive scans. For thresholding to be effective, the grade of the road ahead of the vehicle should be fitted using several scan lines, and then each line that is profiled should be checked against this fit.

Fitting a height to just the center of the road can lead to difficulties in extreme circumstances. This could be compensated for by using a weighted history, as in the selected scanline approach. Another method would be to fit a surface to the entire road. To reduce complexity, separate curves could be fit to the left edge, right edge and the center of the road. These lines could then be averaged, filtered or fit laterally to find the height for a given row in the image.

Bends in the path of the vehicle can cause parts of the path to be obscured from the field of view of a scanner if it rigidly mounted to the front of the vehicle. While this condition is detected and the only result is to force the vehicle to slow down, the scanner could be servoed to point towards the road ahead. Also, obstacle detection would be more robust if the scanner could be mechanically stablized to negate the effects of vehicle pitch. Finally, benchmarks have shown that there is a bottleneck in the conversion from cylindrical range data to height data. This could be improved significantly by special hardware.

References

[1] S. Singh, D. H. Shin, "Position Based Path Tracking for Wheeled Mobile Robots, " in *Proceedings IEEE/RSJ International Workshop on Intelligent Robot Systems*, September 1989, Tskuba, Japan.

[2] D. H. Shin, S. Singh, "Vehicle and Path Models for Autonomous Navigation," In *Vision and Navigation: The Carnegie Mellon NavLab*, Editor Chuck Thorpe, Kluwer Press, 1990.

[3] R. T. Dunlay, "Obstacle Avoidance Perception Processing for the Autonomous Land Vehicle," *Proceedings IEEE ICRA*, Philadelphia, 1988.

[4] M. J. Daily, J. G. Harris, K. Reiser, "Detecting Obstacles in Range Imagery", P*roceedings DARPA Image Understanding Workshop*, February, 1987.

[5] M. Hebert, "Building and Navigating maps of Road Scenes Using an Active Sensor," *Proceedings IEEE ICRA*, 1989.

[6] U. Regensburger, V. Graefe, " Object Classification for Obstacle Avoidance," *Proceedings SPIE Symposium on Advances in Intelligent Systems*, 1990.

[7] K. Dowling, R. Guzikowski, H. Pangels, S. Singh, W. Whittaker, "NavLab: An Autonomous Vehicle," Technical Report, CMU-RI-TR-87-24, Robotic Institute, Carnegie Mellon University, 1987.

[8] S. Singh, J. West, "Cyclone: A Laser Scanner for Autonomous Vehicle Navigation," Technical Report to be published, Robotics Institute, Carnegie Mellon University, 1991.

[9] S. Singh, D. Feng, P. Keller, G. Shafer, D. H. Shin, W. Shi, J. West, B. Wu, "FastNav: A System for Fast Navigation," Technical Report to be published, Robotics Institute, Carnegie Mellon University, 1991.