

# Fast and Reliable Obstacle Detection and Segmentation for Cross-country Navigation

A. Talukder, R. Manduchi\*, A. Rankin, L. Matthies

Jet Propulsion Laboratory, California Institute of Technology  
Pasadena, CA 91109. Tel. (818)354-1000 – Fax (818)393-3302  
[ashit.talukder,art.rankin,larry.matthies]@jpl.nasa.gov

\*University of California at Santa Cruz  
Santa Cruz, CA 95064. Tel. (831)459-1479 – Fax (818)459-4829  
manduchi@soe.ucsc.edu

## Abstract

*Obstacle detection (OD) is one of the main components of the control system of autonomous vehicles. In the case of indoor/urban navigation, obstacles are typically defined as surface points that are higher than the ground plane. This characterisation, however, cannot be used in cross-country and unstructured environments, where the notion of "ground plane" is often not meaningful. A previous OD technique for cross-country navigation (adopted by the DEMO III experimental unmanned vehicle) computes obstacle by analysing the columns of a range image independently, looking for steps or slopes along the range profile. This procedure, however, is prone to missing obstacles with surface normal pointing away from the line of sight. We introduce a fast, fully 3-D OD technique that overcomes such a problem, reducing the risk of false-negatives while keeping the same rate of false-positives. A simple addition to our algorithm allows one to segment obstacle points into clusters, where each cluster identifies an isolated obstacle in 3-D space. Obstacle segmentation corresponds to finding the connected components of a suitable graph, an operation that can be performed at a minimal additional cost during the computation of obstacle points. Rule-based classification using 3-D geometrical measures derived for each segmented obstacle is then used to reject false-obstacles (for example, objects that are small in volume, or of low height). Results for a number of scenes of natural terrain are presented, and compared with a pre-existing obstacle detection algorithm.*

**Keywords:** Autonomous navigation, obstacle detection, terrain perception, 3-D vision, classification, geometrical reasoning

## 1. Introduction

Path planning for autonomous vehicles requires that the map of all visible obstacles be produced in real time using the available sensing information. The obstacle-free candidate paths leading toward the desired position are then compared in terms of their *hindrance* (measured, for example, by the amount of steering involved [Lacaze98].)

For navigation indoor or in structured environment (roads), obstacle are simply defined as surface elements that are higher than the ground plane. Thus, assuming that elevation information is available (by means of stereo cameras or ladars), the main task of obstacle detection (OD) algorithms for indoor/urban environments is to estimate the ground plane in front of the vehicle. Many papers exist in the literature dealing with such a problem (see for example [Zhang94],[Williamson98],[Broggi00].)

This flat-world assumption is clearly not valid when driving in off-road, cross-country environments. In such cases, the geometry of the terrain in front of the vehicle can hardly be modelled as a planar surface. Figure 1 shows examples of natural scenes where no distinct planar surface can be fit as a ground surface due to inadequate number of visible ground points.

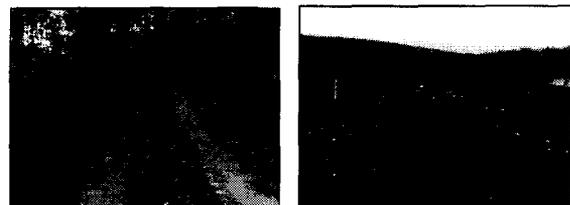


Figure 1: Examples of natural terrain

In principle, one could determine the traversability of a given path by simulating the placement of a 3-D vehicle model over the computed elevation map, and verifying that all wheels are touching the ground while leaving the bottom of the vehicle clear. This procedure, however, besides being computationally heavy, requires the availability of a high-resolution elevation map to work with. Maps are estimated starting from range images (from stereo or ladars). Backprojecting image pixels onto the 3-D world generates a non-uniform point set. Therefore, either the elevation map is represented by a multiresolution structure (which makes the OD task cumbersome), or it is interpolated to an intermediate-density uniform grid, which may imply a loss of resolution in some regions of the map.

On the converse, working directly on the range image domain (pixel-based approach) presents two advantages: first, it is much faster than dealing with elevation maps;

second, it uses range data with the highest resolution available (since the data does not need to be interpolated into fixed-size 3-D cells.) Indeed, the elevation map approach allows one to easily integrate range information as the vehicle moves along and collects more sensing information. This functionality is certainly important for robust path planning, especially when the scene has many visual occlusions, meaning that a single view may not convey enough information. Yet, we argue that for a vehicle moving forward, it is the most recently acquired range image that typically contains the higher resolution range information. Hence, computing obstacles based on the most recent range image makes sense on the grounds of computational efficiency *and* of detection accuracy.

Matthies et al. developed a fast pixel-based algorithm to detect obstacles for cross-country terrain [Matthies94],[Matthies96],[Matthies98]. Their technique (adopted by the DEMO III eXperimental Unmanned Vehicle (XUV) [Bellutta00]), measures slope and height of surface patches (where “slope” is measured by the angle formed by the surface normal and the vertical axis.) Figure 2 shows an example of a 1-D range profile, where slant ( $\theta$ ) and height ( $H$ ) are shown for two different surface patches. Obstacles correspond to ramps with a slope above a certain threshold and spanning a minimum height. The rationale behind this approach is simple: if a surface patch has limited slope, we may assume that it belongs to the ground surface (for example, as part of a path leading to a plateau,) and therefore it is traversable. If a patch is steep but of small size, it corresponds to an obstacle short enough to be negotiable by the vehicle. Thus, the lower patch in Figure 2 would probably be considered traversable, while the higher patch would probably be considered an obstacle.

In fact, the OD technique of [Matthies96] looks exclusively at 1-D range profiles such as in Figure 2, because it analyses each column in the range image separately from the others. This choice, which makes the algorithm very fast, has drawbacks in terms of detection accuracy. It is easy to see that any 1-D range profile corresponding to one column of the range image is equal to the trace left by the visible surface on a slicing plane  $\Pi$  defined by the

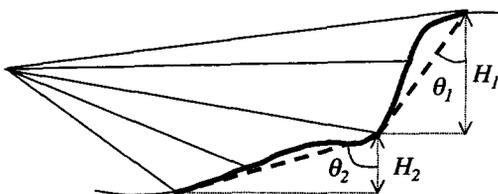


Figure 2: 1-D range profile and obstacle definition

column points in the image plane and the focal point of the camera. The estimated slope of this 1-D range profile is not equal, in general, to the true slope of the visible surface, and can actually be much smaller than that. Thus, an obstacle may be missed by such a technique, if the

obstacle’s surface normal points away from the slicing plane  $\Pi$ .

In this paper, we present an improved version of the column-wise OD algorithm [Matthies96], which computes 3-D slopes and yet retains most of the simplicity and computational efficiency of the original approach. More precisely, this work has three main contributions. First, we provide a simple but rigorous definition of obstacle points that make sense for cross-country environments, formalizing and extending the intuitive notion in [Matthies96]. Second, we derive an efficient algorithm to compute the obstacle points in a range image. Third, we present a technique to correctly *segment* such obstacle points, so that isolated obstacles are identified and labeled. We show that obstacle segmentation (OS) corresponds to finding connected components in a suitable graph built by the OD procedure. Our OS procedure makes full use of 3-D information, and is implemented efficiently in terms of computations and memory.

The paper is organized as follows: The OD algorithm is detailed in Section 2, followed by a discussion of our obstacle segmentation (OS) algorithm in Section 3. In Section 4, we discuss some of the parameters in the OD and OS algorithms, and in Section 5, we detail our 3-D geometrical-based obstacle reasoning and classification method, followed by results of our algorithms and comparison with a pre-existing OD method in Section 6.

## 2. Obstacle definition and algorithms for OD

In this section we give an axiomatic definition of “obstacle” which is amenable for cross-country navigation, and derive a simple and efficient algorithm for obstacle detection (OD). We will show in Section 3 that a simple extension of this algorithm allows us to not only detect obstacle points in an image, but also to identify regions of points belonging to the same obstacle.

In order to introduce our algorithm, we first provide an axiomatic definition of the “obstacles” we want to detect. We will define obstacles in terms of two distinct points in space:

**Definition 1:** Two surface points  $p_1=(p_{1x},p_{1y},p_{1z})$  and  $p_2=(p_{2x},p_{2y},p_{2z})$  belong to the same obstacle (and will be called “compatible” with each other) if they satisfy the following two conditions:

1.  $H_T < |p_{2y}-p_{1y}| < H_{max}$  (i.e., their difference in height is larger than  $H_T$  but smaller than  $H_{max}$ );
2.  $|p_{2y}-p_{1y}| / \|p_2-p_1\| > \cos(\theta_T)$  (i.e., the line joining them forms an angle with the vertical axis smaller than  $\theta_T$ );

where  $H_T$ ,  $H_{max}$  and  $\theta_T$  are suitably chosen constants.

In our definition,  $H_T$  is the minimum height of an object to be considered an obstacle;  $H_{max}$  is a parameter controlling the size of the analysis window in the OD algorithms,

and will be discussed in Section 3;  $\theta_T$  is the smallest value of the slope of the steepest point of an obstacle.

Thus, a point  $p$  is classified as “obstacle” if there exists at least another visible surface which is compatible with  $p$ . Definition 1, however, specifies more than just that: it also formalizes the notion of points belonging to the *same* obstacle. This is rather useful if, beyond determining obstacle points, one wishes to segment the different obstacles visible in an image, as discussed in Section 3.

Figure 3 shows an illustration of the detection of obstacle points (blue) in 3-D space based on slope and height measures relative to ground points (brown).

A naïve strategy for detecting all the obstacle points in an image would thus examine all point pairs, resulting in  $N^2$  tests. Note that testing if two points are compatible requires 5 sums, 4 multiplications and 3 comparisons, all on floating-point numbers. A more efficient algorithm can be designed starting from the following observation. According to Definition 1,  $p$  is an obstacle point if and only if there exists at least one visible surface points located

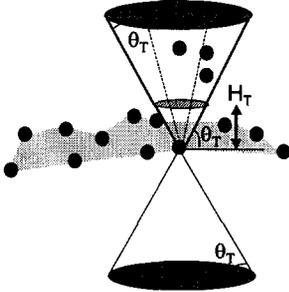


Figure 3: 3-D obstacle search method using double cone that locates ground pixels (brown) & obstacle pixels (blue).

inside the double truncated cone of Figure 3. Searching for such points in 3-D space, however, is an expensive operation. Instead, we observe that the double truncated cone centered in  $p$  projects into a double truncated triangle in the image plane centered in pixel  $\underline{p}$  (the projection of  $p$  on the image plane<sup>1</sup>). Each such triangle has height equal to  $H_{max}f/p_z$ , where  $f$  is the camera’s focal length. Note, however, that an image point in such triangles is not necessarily generated by a 3-D point within the cone. Thus, a strategy for detecting all obstacle points in the image is the following one:

**Obstacle Detection (OD) Algorithm 1.**

- For each pixel  $\underline{p}$ , determine the set  $I_p$  of pixels belonging to the double truncated triangle centered in  $\underline{p}$ . Define a scanning order for the points in  $I_p$ .

<sup>1</sup> As there is a one-to-one correspondence between 3-D points  $p$  in the range image and their projections  $\underline{p}$  in the image plane, we will that two 2-D points  $\underline{p}_1$  and  $\underline{p}_2$  are compatible, meaning that their corresponding 3-D points are.

- Scan the points in  $I_p$  until a pixel  $\underline{p}_2$  compatible with  $\underline{p}$  is found, in which case classify  $p$  as an obstacle point.
- If no such pixel is found,  $p$  is not an obstacle point.

We managed to reduce the complexity of the algorithm from quadratic to linear in  $N$ . Note, however, that there is the possibility that a pair of points  $(p_1, p_2)$  are tested twice. If  $\alpha$  is the proportion of obstacle points in the image, and  $K$  is the average number of points in each projected triangle on the image plane, then this algorithm requires an expected number of  $2N(\alpha+(1-\alpha)K)$  tests.

Let us now introduce a second strategy, which does not require duplicate tests:

**OD Algorithm 2.**

- Initialization: classify all pixels as non-obstacle.
- Scan the pixels from bottom to top and from left to right; For each pixel  $\underline{p}$ :
  - Determine the set  $U_p$  of pixels belonging to only the upper truncated triangle with lower vertex in  $\underline{p}$  (see Figure 4).
  - Examine *all* points in  $U_p$ , and determine the set  $S_p$  of points  $\underline{p}_2 \in U_p$  compatible with  $\underline{p}$ .
  - If  $S_p$  is not empty, classify all points of  $S_p$  and  $p$  as obstacle points.

It is easy to see that each pixel is tested just once against all the other points in the upper and lower truncated triangles in OD 1. With reference to the quantities introduced earlier, now  $NK$  tests must be performed over the image. Thus, if  $\alpha < 0.5$ , the second algorithm results in higher computational efficiency. More importantly, a simple modification of this algorithm allows one to easily segment obstacles in the image, as described in the next section.

Figure 4 shows an illustration of our OD 2 algorithm, where the search area in the image depends on the distance of the point from the image plane.

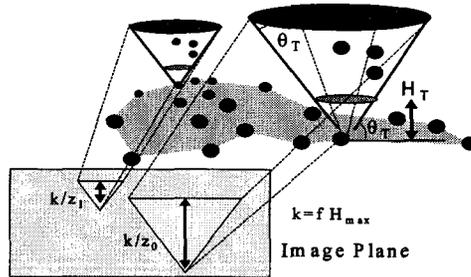


Figure 4: Implementation of OD Algorithm 2 on 2-D image data using triangular projections.

### 3. Obstacle segmentation

Definition 1 specifies only a *sufficient* condition for two points to belong to the same obstacle, not a *necessary* one. Two points may well belong to the same obstacle without being compatible (for example, if the two points are very close to each other.) In fact, the missing “if only” part is implicitly defined by the following transitivity property: if  $p_1$  and  $p_2$  belong to the same obstacle, and  $p_2$  and  $p_3$  belong to the same obstacle, then  $p_1$ ,  $p_2$  and  $p_3$  belong to the same obstacle. We maintain that two points  $p_1$  and  $p_M$  belong to the same obstacle *if and only if* there exists a chain of point pairs  $(p_1, p_2), (p_2, p_3), \dots, (p_{M-1}, p_M)$ , such that all pairs  $(p_j, p_{j+1})$  are compatible. We can represent the set of points as the nodes of an undirected graph (*points graph*); two nodes in the graph are linked if they satisfy the conditions of Definition 1. Thus, two points  $p_1$  and  $p_2$  belong to the same obstacle *if and only if* there exists a path in the graph from  $p_1$  to  $p_2$ . We can extend this notion to define a single obstacle as a maximal connected subgraph (i.e., a connected component) of the point graph.

Classical depth-first or breadth-first search algorithms [Mehlhorn84] can find the connected components of the points graph with complexity linear in  $(N+M)$ , where  $M$  is the number of edges in the graph. Note, however, that our OD technique *does not* yield an explicit graph representation, as required by classical connected component algorithm. In the following, we discuss some possible procedures for computing the connected components of the points graph as it is being built in the loop of OD Algorithm 2.

The first proposed algorithm is based on pixel re-coloring:

#### Obstacle Segmentation (OS) Algorithm 1.

Modify the initialization line of OD Algorithm 2 as follows:

- Initialization: Classify all image points as non-obstacle; no image point is labelled; initialise the label graph to the void set.

The following instructions are added to the loop on the points  $p$  in OD Algorithm 2:

- If no point in  $\{p, S_p\}$  was already labelled, color all points in  $S_p$  and  $p$  with the corresponding label.
- Else, if just one point in  $\{p, S_p\}$  was already colored, color  $p$  and all points in  $S_p$  with such a label.
- Else, there is a label conflict: two or more distinct labels  $\{l_1, \dots, l_L\}$  are used for the same connected component. Choose any such label (say,  $l_1$ ), and find the set of pixels that have been already colored with any label in  $\{l_2, \dots, l_L\}$ ; change the label of such pixels to  $l_1$ .

OS Algorithm 1 always keeps the number of existing labels small, so that the likelihood of label conflicts is

minimized. However, pixel re-coloring is an expensive operation. Of course, one may use a hashing table, but that requires a significant amount of additional memory.

The second proposed algorithm introduces an auxiliary *labels graph*, whose nodes correspond to labels used to color the nodes of the point graph.

#### OS Algorithm 2.

Modify the initialization line of OD Algorithm 2 as follows:

- Initialization: Classify all image points as non-obstacle; no image point is labelled; initialize the label graph to the void set.

The following instructions are added to the loop on the points  $p$  in OD Algorithm 2:

- If no point in  $\{p, S_p\}$  was already labelled, create a new node in the labels graph and color all points in  $S_p$  and  $p$  with the corresponding label.
- Else, if just one point in  $\{p, S_p\}$  was already colored, color  $p$  and all points in  $S_p$  with such a label.
- Else, there is a label conflict: two or more distinct labels  $\{l_1, \dots, l_L\}$  are used for the same connected component. Color all unlabelled points in  $\{p, S_p\}$  using any one of such labels (say,  $l_1$ ), and add edges in the labels graph linking the nodes corresponding to such labels. Re-color all pixels in  $\{p, S_p\}$  to label  $l_1$ .

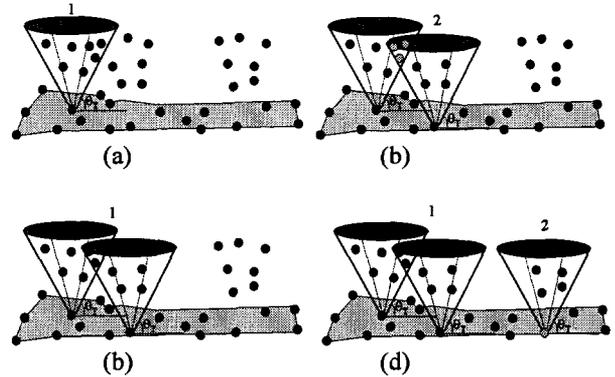


Figure 5: Labelling process during 3-D obstacle detection: (a) Obstacles for one ground pixel; (b) for second ground pixel, (c) Merging of overlapping obstacles, and (d) new obstacle label.

When the procedure terminates, all nodes in the points graph are labelled; the nodes belonging to any connected component of the labels graph represent the set of labels coloring the nodes of one connected component of the points graph. Thus, in order to identify all the obstacles in the scene, one has to compute the connected components in the labels graph. This operation takes a negligible amount of time if there are much fewer labels than points in the image. Note that the operation of pixel re-coloring within  $\{p, S_p\}$  in case of label conflict is not strictly neces-

sary, but it helps reduce the likelihood of label conflict. In fact, we noticed that only a minimum amount of over-segmentation is introduced if one neglects to compute the connected components of the labels graph (i.e., if each node of the labels graph corresponds to one obstacle.)" Figure 5 shows the process of segmentation/labeling of obstacles that occurs implicitly in our 3-D obstacle detec-

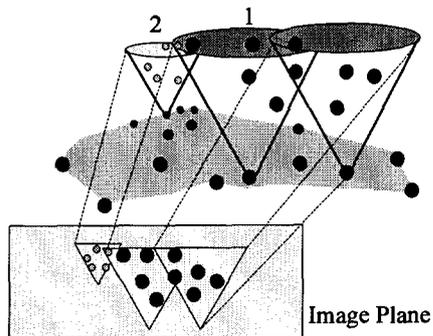


Figure 6: Obstacle labelling in our 3-D obstacle detector where adjacent obstacle points in 2-D image space (green, blue pixels), but distant in 3-D space, are assigned unique labels.

tor. Figures 6 and 7 show examples of a synthetic and real example where disparate objects that touch in 2-D space, and therefore assigned to one obstacle by a 2-D blob coloring procedure [Bellutta00], are labelled with different colors using our OS algorithm.

Having formalized the notion of points belonging to the same obstacle, the role of the parameter  $H_{max}$  should now be clear.  $H_{max}$  enforces separation of two obstacles in those cases where pairs of points exist, one for each obstacle, satisfying the slope condition but located far apart. Typically, such situations arise from missing range measurements (due, for example, to poor stereo matching quality.) Rather than linking two obstacles when there is not enough range information, the first condition in Definition 1 keeps such obstacle separated. Note in passing that larger values of  $H_{max}$  imply larger triangles in Figure 4, and therefore higher computational complexity. On the other side, too small a value for  $H_{max}$  could be liable for missing many obstacle points. An interesting case is represented by obstacles which are not connected to the ground (e.g., concertina wire). For the wire to be detected as an obstacle,  $H_{max}$  should be at least as large as the height of the wire with respect to the ground.

#### 4. Spatial Resolution of 3-D obstacle detector

In order to evaluate the efficacy of our 3-D obstacle detector, it is important to realise the spatial resolution limitations of our algorithm. The spatial resolution determines how close two obstacles can be and still be segmented as two different objects. *This information is critical when the terrain is densely occupied by non-traversable ob-*

*stacles*, in which case the gaps between two obstacles should be accurately located to allow autonomous navigation and effective movement of the vehicle in such densely occupied landscapes.

If the search height of the cone in 3-D space is  $H_{max}$ , as specified earlier, our algorithm classifies any two occupied pixels that are separated by a horizontal distance of  $2H_{max}/\cos(\theta_T)$  with the same obstacle label. Therefore, two different obstacles that are at least  $H_{max}$  in height and separated by a horizontal distance of more than  $2H_{max}/\cos(\theta_T)$  are assigned different labels by our 3-D obstacle detector. This implies that the spatial resolution for obstacles that are at least  $H_{max}$  in height is

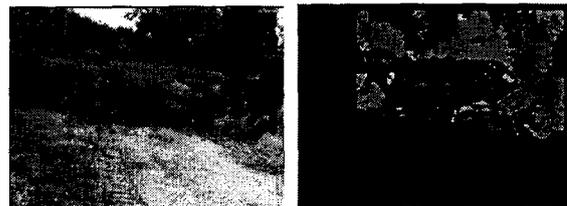


Figure 7: Obstacle labelling and segmentation

$2H_{max}/\cos(\theta_T)$ . For obstacles of height  $H$  lesser than  $H_{max}$ , the spatial resolution of our algorithm is much better; it is  $2H/\cos(\theta_T)$  for obstacles with lower height.

If the width of the vehicle  $W$  is lesser than the spatial resolving power of the algorithm, then our algorithm would safely and accurately locate all visible, traversable paths in the terrain. This is generally true for autonomous cross-country vehicles, such as the HMMWV or the URBIE robot testbeds used at JPL. In practice, the range estimates from stereo or laser-range sensors is corrupted by noise and susceptible to measurement/estimation errors, especially for points far from the sensor. This is further magnified by errors due to incorrect sensor calibration. Therefore, the spatial resolution of our 3-D obstacle detector is typically worse than the theoretical limits discussed here.

#### 4.1. Parameter selection in 3-D obstacle detector

As discussed in Section 2 earlier, our 3-D obstacle detector involves searching a cone region around each point in 3-D space for the presence of an obstacle. If the ground terrain is flat (horizontal), obstacle search at a point  $(x_0, y_0, z_0)$  at a distance  $z_0$  from the image plane would involve searching an area corresponding to the projection of the cone on the 2-D image, which is an inverted triangle of height  $H_i = (H_T) f/z_0$  (whose vertex is  $(x_{oi}, y_{oi})$  in the image  $I$  as  $x_{oi} = x_0 f/z_0$ ,  $y_{oi} = y_0 f/z_0$ ), and with vertex angle  $90 - \theta_T$ , as shown in Figure 4 earlier. This is the region  $U_p$  in OD 2.

However, in reality, the terrain is uneven, and many ground pixels in the terrain do not lie on the camera plane. Additionally, the camera plane may not be horizon-

tal if the vehicle is on a slope. Therefore, the projection of the cone will change with terrain elevation variations.

We analyse the change in projection of the 3-D cone along each spatial dimension  $x, y$  as the terrain configuration changes. When the terrain is flat, the projection (region  $U_p$  in OD 2) is a triangle with a horizontal base of height  $(H_T) f/z_0$ . If the terrain is sloped, or the camera plane tilted, it is possible that the cone generatrix (the slanted line with angle  $\theta_T$ ) becomes parallel to the camera plane. In such a case, the region  $U_p$  is a slanted triangle whose base is at angle  $\theta_T$  with the horizontal. The hypotenuse of the triangle is parallel to the vertical axis of length  $((H_T^2/\cos^2(\theta_T)) + H_T^2)^{1/2} f/z_0$ ; this is larger than  $(H_T) f/z_0$  for a projection of a cone on a ground plane that is horizontal and at the same elevation as the image plane.

Similarly, if the camera/vehicle, or a ground plane segment were tilted from the  $x$ -axis, the hypotenuse of the projected triangle could be parallel to the  $x$ -axis, thereby yielding a projection of size  $(H_T^2\cos^2(\theta_T) + H_T^2)^{1/2} f/z_0$  along the  $x$ -axis.

Therefore, the shape and size of the search region  $U_p$  varies with terrain orientation. *In practice*, we use a square search window at least of size  $(H_T^2\cos^2(\theta_T) + H_T^2)^{1/2} f/z_0$  for our 3-D obstacle detection to accommodate all possible terrain variations. Typically, a window of about 4-5 times the minimum size is employed to ensure that valid obstacles are not discarded due to spatial resolution limitations of the estimated range-from-stereo data.

### 5. 3-D Shape Reasoning for Obstacle Classification using Rule-based Classifier

As discussed earlier, our 3-D obstacle detection algorithm automatically segments the obstacles to yield a unique label for each obstacle in 3-D space. This facilitates the use of 3-D shape and geometrical measures to effectively reject spurious false obstacles that may have been detected.

3-D shape reasoning techniques have been used in robotics in the past. Sutton et. al. [Sutton98] have used 3-D shape reasoning in robotics by building detailed 3-D models of an object from range data, followed by shape-reasoning to label the object's potential functionality. A model-based 3-D geometrical reasoning scheme for land vehicles has been used [Marti96] where prior scene knowledge with a generic 3-D model of the expected scene and the potential objects is compared with the actual scene to do 3-D obstacle classification. Both these techniques require detailed prior knowledge of the objects and the scene which is not expected in real terrain navigation scenarios, and 3-D model-matching which is computationally demanding. In [Crisman98], stereo data is used to locate edges and corner targets for wheelchair navigation in relatively uncluttered, flat urban environments. In another approach [Hoover98], a planar boundary representation space envelope models the empty, unoccupied

volumes in the scene. Reasoning about the scene's content using surface geometry and topology is used to determine the number of visible objects. All these methods require creation of a 3-D model, possibly by converting the 3-D point data into a mesh representation, which is a complex operation and often not suited for real-time applications that have limited computational resources. We compute 3-D geometrical features from the raw point-cloud data, which enables real-time analysis.

In our initial research efforts, we extracted five simple 3-D geometrical measures from each obstacle. This included the perimeter of the 3-D bounding box for an obstacle, the average slope of the obstacle and relative height from surrounding background, the maximum slope, and the maximum relative height of an obstacle from surrounding regions. These geometrical measures are automatically derived during the obstacle segmentation process, without any extra computational overhead. Thresholds are assigned to each of these five 3-D measures. If any of the five variables have a value less than the pre-selected thresholds, it is rejected as a false obstacle. For example, all obstacles with an average slope lesser than 2.5, or maximum slope lesser than 5.0 were classified as false-obstacles. Our rule-based classification therefore rejects obstacles with small bounding volume, average/maximum slopes, or average/maximum relative height.

Our new rule-based 3-D shape reasoning and classification is expected to outperform prior 2-D based obstacle reasoning methods [Bellutta00] that used 2-D area information (not 3-D geometrical measures) to reject small-sized false obstacles. In many cases, an object that occupies a small number of pixels in the 2-D image does not imply the presence of a false obstacle, or conversely a

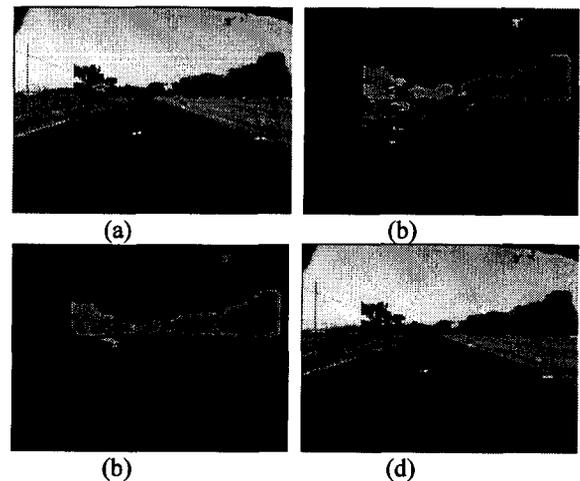


Figure 8: (a), (b) Obstacle regions before and (c), (d) after 3-D rule-based false obstacle removal.

large 2-D pixel area does not signify the presence of a true obstacle. False obstacle regions that are close to the camera could occupy a large number of pixels, and true

obstacles far away from the camera could occupy a significantly small area.

Figure 8a shows a road with surrounding natural terrain and telephone poles. The initial 3-D obstacle detection algorithm locates all true obstacles (telephone poles, trees, etc.), but also locates false obstacles on the flat road on the upper left and upper right parts of the image (due to incorrect range from stereo measurements), as shown by the overlaid blue regions in Figure 8a. The corresponding segmented obstacle label image is shown in Figure 8b. Our rule based classification evaluates the 3-D geometrical measures for each detected obstacle, and correctly rejects the false-obstacles on the road that have low average/maximum height and bounding volumes, as shown by the new labeled data in Figure 8c. The overlaid true obstacle image after our rule-based classification is shown in Figure 8d, where the red regions are the rejected false obstacle regions.

## 6. 3-D Obstacle Detection: Results and Comparisons

We present results on several types of terrain using our new 3-D obstacle detector. These are compared against a prior obstacle detection technique [Bellutta00] that used slope measurements along image columns and 2-D area

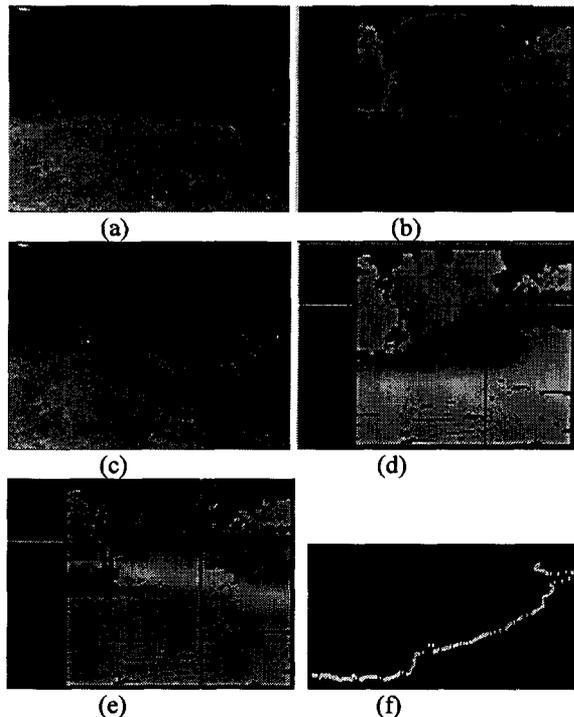


Figure 9: Obstacle regions located using (a), (b) our 3-D obstacle detector and (c) prior 2-D obstacle detector; (d) Range, and (e) elevation maps, and (f) 1-D elevation profile.

measures for obstacle detection. As mentioned earlier, this prior technique is not expected to work well on terrain that contains obstacles that slope along the image plane, rather than vertically downwards along image columns. Figure 9 shows an example of terrain that contains two mounds to the right of the camera with slanted slopes. Figure 9a shows the true detected obstacle regions using our new 3-D obstacle detection algorithm in blue and the corresponding obstacle labels are shown in Figure 9b. The 2-D obstacle detection algorithm [Bellutta00] results are shown as blue regions in Figure 9c, and the rejected obstacle regions are shown in red. As seen, all of the closest mound and much of the larger mound obstacles are not detected due to the columnwise scanning technique used. Additionally, parts of the smaller mound are rejected (red regions) since a 2-D blob area measure is used to reject small obstacles. Note that the previous obstacle detector using 1-D elevation profile fails to detect the two mounds due to the fact that the estimated slope of this 1-D range profile is not equal, in general, to the true slope of the visible surface, as seen in the elevation map in Figure 9(e), and the 1-D elevation profile in Figure 9(f).

Figure 10 shows an image of a road with obstacles (telephone poles, and trees) on the side. Our 3-D obstacle detector locates all obstacles effectively (blue regions in Figure 10a), and the 3-D rule-based geometrical classification rejects false obstacles that are located along the flat road (red regions in Figure 10(b)). In contrast, the prior 2-D obstacle detector is unable to reject false obstacles on the road due to the 2-D pixel area measure used. A large false obstacle region is also incorrectly located on the upper right side of the image.

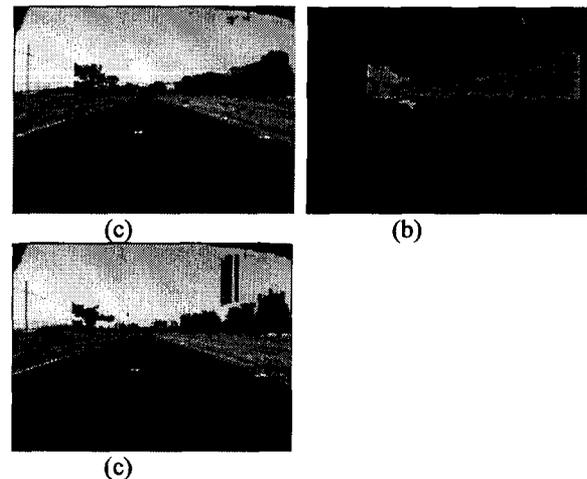


Figure 10: Obstacle regions located using (a), (b) our 3-D obstacle detector and (c) prior 2-D obstacle detector

Figure 11 shows natural terrain with a tall bush on the right, a negative obstacle in front of the camera, and trees in the background. Our 3-D obstacle detector locates all

the obstacles effectively, and rejects false obstacles near the foot of the bush, and in the grassy terrain beyond the negative obstacle. In this case, the 2-D obstacle detector performs comparably, even though parts of the trees in the background are not correctly detected. Note that our 3-D obstacle detector correctly distinguishes between the bush and background trees and assigns unique labels to each (Figure 11(b)), even though they overlap/touch in the 2-D image. This allows for combined color/texture and shape-based classification that could correctly classify low bushes as traversable obstacles. A 2-D obstacle detector on the other hand would label all touching pixels (including the background trees and bush) as one single obstacle, that would result in overall misclassification if color/texture and obstacle shape data were fused.

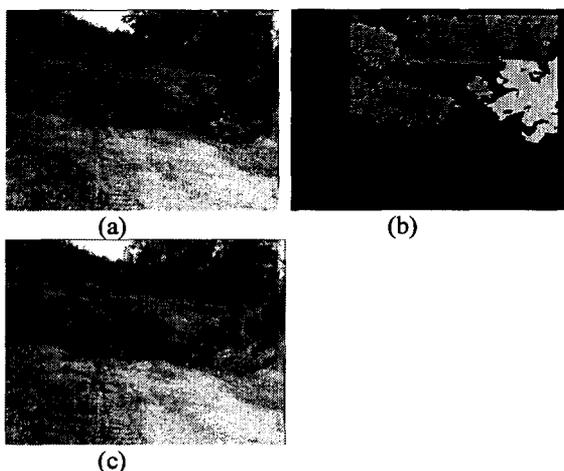


Figure 11: Obstacle regions located using (a), (b) our 3-D obstacle detector and (c) prior 2-D obstacle detector.

Further results on a terrain with a large non-traversable mound is shown in Figure 12b,c. The prior obstacle detector is unable to detect the mound at all (Figure 12d) since it's normal does not intersect the slicing plane  $\Pi$  defined by the column points of the image plane and the focal point of the camera. A few sections that are detected are discarded as false positives in the area-based blob removal stage.

The last results (Figure 13) show a narrow path in a wooded area along a trail lined with bushes. These bushes are correctly located as obstacles along with the trees as seen in Figure 13b,c. The 1-D obstacle detection algorithm (Figure 13d) misses the bushes on the right, which is not critical since these bushes are small. Fusion of color with shape-based reasoning is expected to result in classification of the small bushes on the right as traversable, which will simplify navigation of the vehicle along narrow paths.

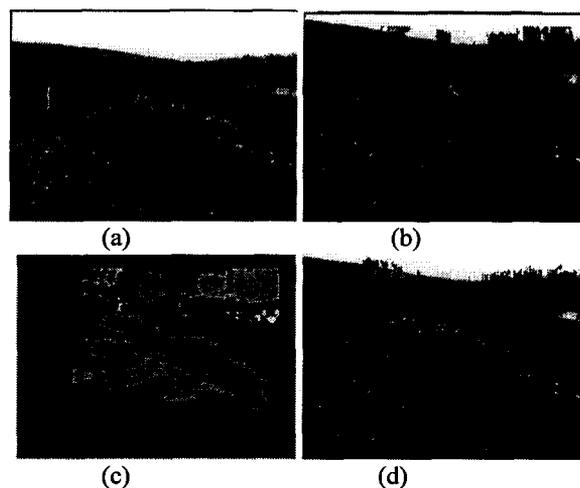


Figure 12: Obstacle regions in image (a) located using (b), (c) our 3-D obstacle detector and (d) prior 2-D obstacle detector.

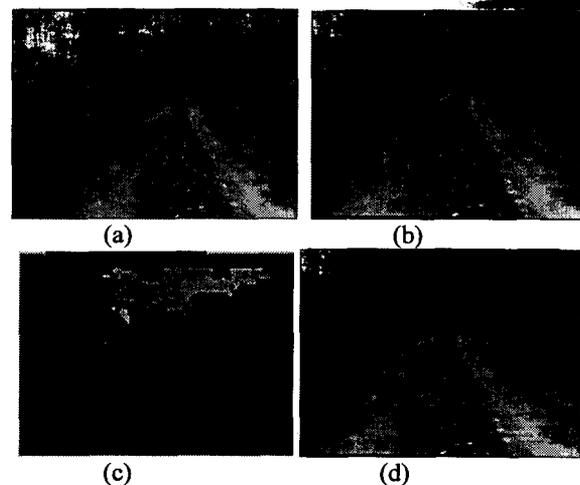


Figure 13: Obstacle regions in image (a) located using (b), (c) our 3-D obstacle detector and (d) prior 2-D obstacle detector.

## 7. Conclusions and Future Work

In this effort, we have detailed a new 3-D obstacle detection algorithm for locating and segmenting obstacles in the scene for autonomous terrain vehicle navigation, and a new 3-D reasoning algorithm to reject false obstacles. The 3-D reasoning technique uses geometrical measures that are automatically derived from the 3-D obstacle detector without any extra computational overhead. Results are presented on scenes of natural terrain that the military autonomous vehicle (HMMWV) is expected to traverse during day and/or night conditions. Our technique is seen

to outperform prior obstacle detection results currently used in real-time JPL autonomous vehicles.

Further improvements to our 3-D obstacle detection and reasoning algorithm will include fusion of shape-based and color or texture information to better classify surrounding terrain into different terrain types (dry/normal vegetation, bush, grass, rocks, telephone poles, fences, etc.). This would enable better classification of traversable objects (small-medium bushes, low-lying grass, tall grass), and non-traversable objects (rocks, poles, trees, tall bushes, steep slopes). Our method can be extended to analysis of multiple frames as the vehicle moves, where incremental 3-D obstacle detection and reasoning could be applied to successive frames to update detected obstacles swiftly.

The 3-D obstacle detector is currently being integrated with a dynamic terrain modeling simulation tool where knowledge of the class and geometrical structure of each obstacle, derived from our obstacle detector, will be used to model the dynamics of the load-bearing surface as the vehicle moves over each traversable object. This is useful in velocity control and prediction of vehicle motion over different terrain types for optimal, safe vehicle performance and navigation.

#### Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the DARPA-ITO Mobile Autonomous Robot Software (MARS) Program through an agreement with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

#### 8. References

- [Bellutta00] P. Bellutta, R. Manduchi, L. Matthies, K. Owens, A. Rankin, "Terrain Perception for DEMO III", *2000 Intelligent Vehicles Conference*, 2000.
- [Matthies94] L. Matthies, P. Grandjean. Stochastic Performance Modeling and Evaluation of Obstacle Detectability with Imaging Range Sensors. *IEEE Transactions on Robotics and Automation, Special Issue on Perception-based Real World Navigation*, 10(6), December 1994.
- [Mehlholm84] K. Mehlholm, *Data Structures and Efficient Algorithms*, Springer Verlag, 1984
- [Matthies96] L. Matthies, A. Kelly, T. Litwin, G. Tharp, "Obstacle Detection for Unmanned Ground Vehicles: A Progress Report", *Robotics research 7*, Springer-Verlag.
- [Matthies98] L. Matthies, T. Litwin, K. Owens, A. Rankin, K. Murphy, D. Coombs, J. Gilsinn, T. Hong, S. Legowik, M. Nashman, B. Yoshimi, "Performance Evaluation of UGV Obstacle Detection with CCD/FLIR Stereo Vision and LADAR", *1998 IEEE ISIC/CIRA/ISAS Joint Conference*.
- [Zhang94] Z. Zhang, R. Weiss, A.R. Hanson, "Qualitative Obstacle Detection", *IEEE CVPR'94*, 554-559.
- [Williamson98] T. Williamson, C. Thorpe, "A Specialized Multibaseline Stereo Technique for Obstacle Detection", *IEEE CVPR'98*, 238-244.
- [Lacaze98] A.Lacaze, Y. Moscovitz, N. DeClaris, K. Murphy, "Path Planning for Autonomous Vehicles Driving Over Rough Terrain", *1998 IEEE ISIC/CIRA/ISAS Joint Conference*.
- [Broggi00] A. Broggi, M. Bertozzi, A. Fascioli, C. Guarino LoBianco, A. Piazzini, "Visual Perception of Obstacles and Vehicles for Platooning", *IEEE Trans. Intell. Transport. Sys.*, 1(3), September 2000.
- [Sutton98] Sutton, M., Stark, L., Bowyer, K., "Function from visual analysis and physical interaction: a methodology for recognition of generic classes of objects", *Image and Vision Computing*, vol.16, no.11, pp. 745-763, 1998.
- [Marti96] Marti, J.; Casals, A., "Model-based objects recognition in man-made environments", *Proceedings 5th IEEE International Workshop on Robot and Human Communication*. 358-363, 1996.
- [Crisman98] Crisman, J.D.; Cleary, M.E.; Rojas, J.C., "The deictically controlled wheelchair", *Image and Vision Computing*, vol.16, no.4, pp. 235-249, 1998.
- [Hoover98] Hoover, A.; Goldgof, D.; Bowyer, K.W., "The space envelope: a representation for 3D scenes", *Computer Vision and Image Understanding*, vol.69, no.3, 310-29, March 1998.