# Spacecraft Navigation in Cluttered, Dynamic Environments Using 3D Lidar

Andrew Bylard[1] and Shreyasha Paudel[2]

***Abstract*— Many new classes of deep-space and on-orbit missions require efficient navigation of environments having untracked dynamic obstacles. Space probes will need to autonomously identify and classify these objects for purposes of localization, collision avoidance, and identification of targets having high value (scientific or otherwise). In addition, lidar is becoming a key sensing tool for close proximity space operations. We implement a lidar simulation for cluttered space environments and explore (i) a k-d tree-based algorithm for clustering and identifying obstacles in 3D lidar point clouds and (ii) Naive Bayes and SVM algorithms for classifying lidar point clusters into obstacle types. The SVM classifier was found to perform better with an overall accuracy of 22-73%, depending on the obstacle type, compared to Naive Bayes which had an accuracy of 15-62%.**

## I. INTRODUCTION

Future operations in space may require autonomous spacecraft to navigate uncertain environments with dynamic obstacles in order to complete their goals. Examples include probes collecting samples of scientific interest in Saturns ice- and rock-filled rings, and manipulator spacecraft navigating and performing tasks in a large-scale on-orbit construction project. In the absence of external sources of tracking data, the spacecraft would need to rely on its own sensors to estimate the size, position, and movement of nearby obstacles for obstacle avoidance. The spacecraft would also need to classify these obstacles for localization and identification of scientific opportunities.

Traditional sensors such as optical, ultrasonic, and radar-based systems have met with mixed success, but 3D lidar is emerging as a low-power, cost-effective solution for onboard sensing of a spacecrafts surrounding environment. Thus, this paper targets the identification and classification aspect of a cluttered space environment sensing using 3D lidar. We explore using machine learning techniques to (i) cluster simulated lidar point cloud data in order to identify obstacles and (ii) classify them into a fundamental set of known obstacle shapes.

## II. RELATED WORK

3D lidar-based perception is becoming increasingly popular and various approaches have been proposed to effectively segment and classify the resulting point clouds. For example, in [1], a regression-based method is used to classify static point clouds into basic geometric shapes, which are then classified into common household objects. A more common use of lidar is in autonomous driving [2], [3], [4]. In [2], a log-based estimator is used to classify objects seen by a car-based lidar system into relevant object classes. Similarly, in [4], the authors use the planar structure of the problem to segment objects in a $2\frac{1}{2}$D occupancy grid and run classification for previously defined object classes. The classification problem is addressed for RGB-generated point clouds in [5], which uses SVMs to classify common household objects based on edge feature data. However these algorithms are either trained to classify commonly available objects for which large training sets are easily available [1], [5] or use the inherent planar structure of the problem to simplify it [2], [3], [4].

Fig. 1. Close observation and sample retrieval from Saturn's rings will require accurate environment sensing for localization, object identification, and obstacle avoidance.

## III. DATASETS AND FEATURES

### A. Lidar Data Generation

The first task of our project was to acquire a large amount of relevant training and test data. However, not much lidar data is freely available from spacecraft proximity operations, particularly for the cluttered scenarios we had in mind. In addition, lidar systems are currently very expensive, and since we did not have access to an existing lidar setup or a testbed that can model a dynamic, cluttered spacecraft environment, collecting data experimentally was out of the question.

Instead, we developed a simulation which could model a variety of obstacles and generate artificial 3D lidar datasets.

*This work was partially supported by NASA Space Technology Research Fellowship Grant NNX15AP67H.

[1]A. Bylard is with Department of Aeronautics and Astronautics, Stanford University, Stanford, CA bylard at stanford dot edu

[2]S. Paudel is with Department of Aeronautics and Astronautics, Stanford University, Stanford, CA spaudel2 at stanford dot edu

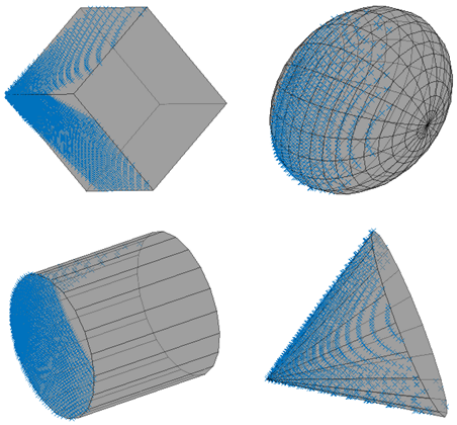


Fig. 2. Sample lidar scans of rectangular, ellipsoidal, cylindrical, and conical obstacles. Each blue point represents a captured lidar data point.

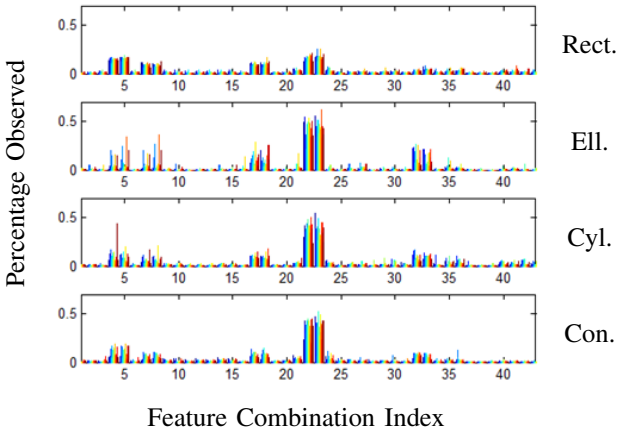



Fig. 3. Comparison of the frequency of the most prominent FPFH feature combinations (particular combinations of $\alpha$, $\phi$, and $\theta$ values) for each obstacle type. Each color represents a single training example.

Our simulation was written in MATLAB, and we made use of a Spacecraft Motion Planning package to generate and keep track of sets of obstacle parameters for different obstacle types. Though the package supports many types of obstacles, including complex shapes such as spacecraft and solar panels, we focused on four fundamental 3D obstacle shapes: rectangular, ellipsoidal, cylindrical, and conical, as our test cases of interest.

The simulation's lidar scanning was modeled after the Velodyne HDL-32E, which has a 360° horizontal field of view and a 40° vertical field of view. The HDL-32E scanner has 32 lasers, each capturing a single point in a given vertical scan, and by its default settings, the HDL-32E will capture 500 vertical scans per horizontal sweep. These parameters were replicated in our simulation.

To generate training data, we randomly generated and scanned one obstacle at a time (example scans are shown in Figure 2. Parameters for these obstacles included the shape, proportions, size, rotation, and distance/angle from the lidar scanner. For our lidar test simulation, we modeled entire cluttered space environments by generating environments with dozens of randomly placed obstacles. To be within the field of view of the lidar scanner, the obstacles were clustered close to a 2D plane with the lidar source at the center.

### B. Feature Extraction

Lidar point clouds for each obstacle observation were distilled into features using Point Feature Histograms (PFHs), a tool widely used in computer vision problems to encode a point or a pixels k-neighborhood geometrical properties. PFHs determine these properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values.

For this project, we implemented a simplified version of PFH called Fast Point Feature Histogram (FPFH). The following subsections briefly give a theoretical primer on PFH and then describe implemented algorithm for normal estimation and feature histogram.

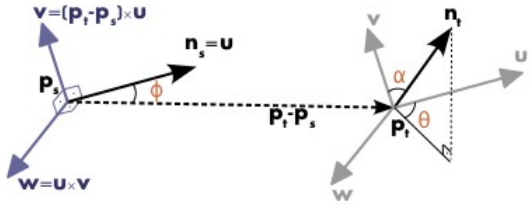



Fig. 4. Geometric representation of PFH descriptors

*1) Theoretical Primer:* The computation of PFH relies on 3D coordinates and estimated surface normals [1]. Let $p_i$ and $p_j$ be two neighboring points, and $n_i$ and $n_j$ be their estimates surface normals respectively. Then, to compute the relative difference, we define a coordinate frame $\mathbf{uvw}$ centered on $p_i$ as shown:

$$\begin{aligned} \mathbf{u} &= n_i \\ \mathbf{v} &= (p_j - p_i) \times \mathbf{u} \\ \mathbf{w} &= \mathbf{u} \times \mathbf{v} \end{aligned} \tag{1}$$

Using this frame, the difference between $n_i$ and $n_j$ can be expressed as a set of angular features by:

$$\begin{aligned} \alpha &= \mathbf{v} \cdot n_j \\ \phi &= \frac{\mathbf{u} \cdot (p_j - p_i)}{||p_j - p_i||} \\ \theta &= \tan^{-1}\left(\frac{\mathbf{w} \cdot n_j}{\mathbf{u} \cdot n_j}\right) \end{aligned} \tag{2}$$

The $\mathbf{uvw}$ frame, the normals, and the angles are shown geometrically in Fig. 4.

FPFH descriptors simplify the computational complexity of PFH from $O(n^2)$ to $O(nk)$ by only computing the $\alpha$, $\phi$, and $\theta$ corresponding to $k$ nearest neighbors of a query point $p_i$ instead of every point in the cloud [6]. These angles are calculated and stored as a Simplified Point Feature Histogram

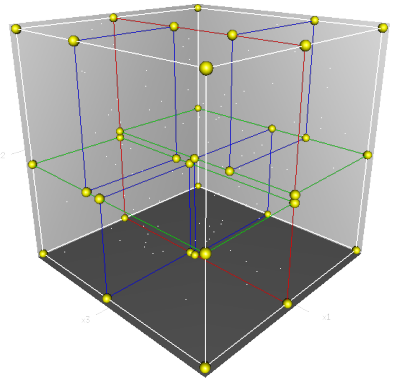


Fig. 5. A 3D k-d tree. The first split (red) cuts the root cell (white) into two subcells, each of which is then split (green) into two subcells. Finally, each of those four is split (blue) into two subcells.

(SPFH). Finally, the FPFH is calculated as shown:

$$FPFH(p_i) = SPFH(p_i) + \frac{1}{k}\sum_{j=1}^{k}\frac{1}{w_k}SPFH(p_k) \quad (3)$$

where, $w_{ik}$ is the distance between $p_i$ and neighbor $p_k$.

*2) Normal Estimation:* The normals were estimated using unconstrained least squares as described in [7]. The closed-form solution for normal vector $n_i$ corresponding to point $p_i$ is

$$n_i = M_i^{-1} b_i \quad (4)$$

where, $M_i = \sum_{j=1}^{k} p_j p_j^T$ and $b_i = \sum_{j=1}^{k} p_j$. The $p_j$'s in the sum correspond to the $k$-nearest neighbors of point $p_i$. For this project, we chose $k = 5$.

*3) FPFH Implementation:* Each feature was discretized into $n_f$ levels, leading to a $N = n_f^3$ possible feature combination for each point. In our implementation, we chose $n_f = 12$ and $N = 1728$. In addition,no features were extracted from point clouds which had fewer than 5 points.

## IV. OBSTACLE IDENTIFICATION

To identify individual obstacles within a lidar scan of a cluttered space, it was necessary to find clusters within the full point cloud. To this end, we implemented a clustering algorithm using k-d trees.

### A. k-d trees

A k-d tree is a generalization of binary search tree to a higher dimensional space. Every non-leaf node of the tree divides the hyperplane into two halfspaces in one of the k-dimensions. For example 3D k-d tree is shown in Fig 5.

Hence, a balanced k-d tree can be constructed by cycling through the axes used to select the splitting planes, and splitting the points by finding the median with respect to the axes selected.

### B. Clustering Implementation

Using an inbuilt MATLAB function, we created a k-d tree representation of the full lidar point cloud. Then the point cloud was divided into clusters by grouping all the points that were within a given range. Algorithm 1 presents the steps used in the clustering.

**Algorithm 1** Pseudocode for clustering

```
Form k-d tree from point cloud
while count ≤ number of points do
  Randomly select p_i from point cloud
  if p_i has not been clustered then
    Add p_i to queue Q
    while Q is not empty do
      p_j = dequeue(Q)
      Add p_j to current cluster
      Find all the neighbors of p_j within range d that
      have not been processed, and add to Q
    end while
    Start a new cluster
  end if
end while
```

For this implementation, we assumed that the space environment being explored is free of unwanted clutter so that all of the lidar readings correspond to objects of interest. In addition, the algorithm relies on a range $d$ as the minimum distance between clusters, based on Euclidean distance. In practice, minimum spacing tends to hold true in space environments, both during on-orbit formation maneuvers and within Saturn's rings, where most of the rocks are separated by meters.

## V. OBSTACLE CLASSIFICATION

Two algorithms were explored for classifying lidar point clusters into obstacle types. In both cases, clusters with too few points to extract features were marked as unclassified.

*1) Naive Bayes Classifier:* For this classifier, we assumed that each feature combination extracted from a given point cluster was an i.i.d. random variable. This leads to the following log-likelihood function:

$$\ell(q) = \log\left(\prod_{k=1}^{N} p(x_i = k|y = q)^{n_k} p(y = q)\right) \quad (5)$$

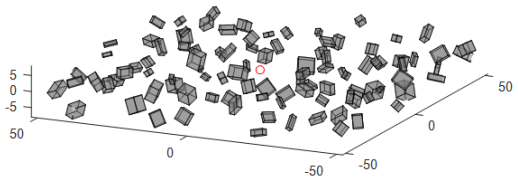
where $q$ represents a particular obstacle type, $N$ is the number of possible feature combinations, and $n_k$ is the number of occurrences of the $k$th feature combination in $x$ (where $x$ is the list of feature combinations extracted from a point cloud, taken from a test obstacle of type $y$). Let $\phi_{k|y=q}$ and $\phi_{y=q}$ be estimates of $p(x_i = k|y = q)$ and $p(y = q)$, respectively. Then $\ell(q)$ is approximately proportional to

$$\sum_{k=1}^{N} n_k \log(\phi_{k|y=q}) + \log(\phi_{y=q}) \quad (6)$$

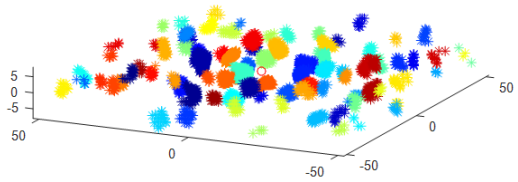
where we assume a uniform prior ($\phi_{y=q} = 1/4$) and

$$\phi_{k|y=q} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i} 1\{x_i^{(j)} = k\} \quad (7)$$

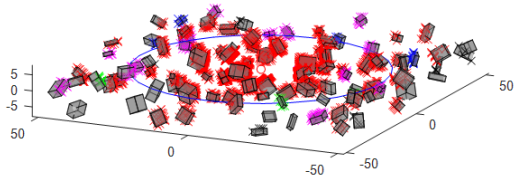
where $m$ is the number of training examples. After training, each test case was classified as the obstacle type resulting in the highest log-likelihood.



(a) Original obstacle set



(b) Clustered lidar cloud



(c) Classified obstacle set

Fig. 6. Full process of identifying and classifying the obstacles. Lidar source is located at the center red circle. (c) Classification output labeled by color (red = Rect; blue = Ell; magenta = Cyl; green = Con). The blue circle represents the maximum distance within which training examples were generated.

*2) SVM Classifier:* The SVM classifier was implemented using the MATLAB wrapper for the libsvm package. This allowed us to easily vary kernel functions, regularization parameters and training size. After 5-fold cross-validation, a Gaussian kernel with $\gamma = 3$ was selected. The results from training and testing are described in more detail in the following section.

## VI. EXPERIMENTS AND RESULTS

The classification algorithms were trained using 2000 examples for each of the obstacle types. Plots of the resulting FPFH data for each type are shown in Fig. 3. For testing, cluttered environments were randomly generated, using one obstacle type per set to permit straightforward accuracy measurements while maintaining the key dynamic of obstacle occlusion during lidar scanning. To check for overfitting, we tested the classifier using increasing numbers of training examples, as shown in Fig. 7. The accuracy leveled off quickly, ruling out overfitting and showing that more training examples were not needed.

The resulting confusion matrices for each algorithm are shown in Tables I and II. SVM clearly outperformed Naive Bayes for each obstacle type. This may be due to Naive Bayes assumption of independence between feature combinations, which is not in fact correct. However, even SVM overall it did not perform as well as might be desired. That being said, as shown in Fig. 6c, classification performed very well on obstacles within the maximum distance from the lidar source in which training examples were generated.

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Rec | Ell | Cyl | Con |
| Actual | Rec | **62%** | 11% | 12% | 15% |
| | Ell | 11% | **46%** | 32% | 11% |
| | Cyl | 25% | 12% | **47%** | 16% |
| | Con | 34% | 15% | 36% | **15%** |

TABLE I
NAIVE BAYES CONFUSION MATRIX

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | Rec | Ell | Cyl | Con |
| Actual | Rec | **73%** | 5% | 18% | 4% |
| | Ell | 12% | **72%** | 7% | 9% |
| | Cyl | 20% | 15% | **51%** | 14% |
| | Con | 44% | 7% | 27% | **22%** |

TABLE II
SVM CONFUSION MATRIX

| | | Class 2 | | |
|---|---|---|---|---|
| | | Rec | Ell | Cyl |
| Class 1 | Con | 88% | 90% | 67% |
| | Cyl | 87% | 86% | |
| | Ell | 98% | | |

TABLE III
SVM TWO-CLASS CLASSIFICATION

To gain further insight into the relatively low accuracy results of SVM, we restricted and trained the classifier on only two obstacles at a time, and ran the tests again. The results in Table III show that SVM mostly performs quite well, and that distinguishing cylinders and cones was the most difficult classification task. This can be explained by the myopic $k$-nearest simplification in the FPFH algorithm, which fails to capture feature information on obstacle surfaces across long distances. Thus, both the curved and flat surfaces of a cone can easily be mistaken for those of a cylinder, and vice versa.

## VII. CONCLUSIONS AND FUTURE WORK

The results of our clustering algorithm for identifying obstacles in lidar point clouds are very promising. In a project developed alongside this for AA 228: Decision-Making Under Uncertainty, we showed that safe navigation is possible in a highly cluttered and dynamic environment using only 8 lasers to detect obstacle surfaces. Thus, we are confident that the obstacle positions identified by the clustering algorithm will be more than sufficient for safe navigation.

For classification, we showed that SVM outperforms Naive Bayes. Future work may be to extract more complex features, e.g. using View-Point Features. Such features could also be used to estimate the size, position, and velocity of obstacles.

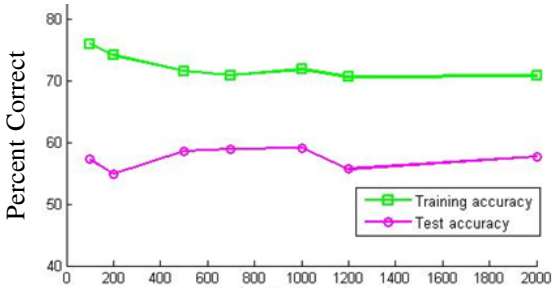



Fig. 7. Convergence of the SVM classifier with increased training

Additional future work to enable safe and efficient navigation in cluttered dynamic environments may be to (i) track obstacles using Kalman filter techniques, and (ii) improve on the related proof of concept from the AA 228 project, extending the work to POMDPs for planning under uncertainty.

## REFERENCES

[1] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D Point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2008.08.005

[2] A. Teichman, J. Levinson, and S. Thrun, "Towards 3D object recognition via classification of arbitrary object tracks," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4034–4041, 2011.

[3] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, a. Quadros, P. Morton, and a. Frenkel, "On the segmentation of 3D lidar point clouds," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2798–2805, 2011.

[4] M. Himmelsbach and T. Luettel, "Real-time object classification in 3D point clouds using point feature histograms," *Iros, 2009*, pp. 994–1000, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5354493

[5] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic Labeling of 3D Point Clouds for Indoor Scenes," *Neural Information Processing Systems*, pp. 1–9, 2011. [Online]. Available: http://pr.cs.cornell.edu/sceneunderstanding/nips_2011.pdf

[6] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[7] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and Accurate Computation of \nSurface Normals from Range Images," no. May, 2011.