Fast ground segmentation of unstructured point clouds using point set maxima

Daniel L. Lu

***Abstract*— We propose a fast $O(n \log n)$ deterministic algorithm to extract the ground plane from a point cloud using classical computational geometry techniques for point set maxima. It needs no training, makes minimal assumptions, is easy to implement, requires no initial guesses, is robust to outliers, and generalizes to different sensor geometries better when compared to recent deep learning methods or methods involving iterative algorithms such as random sample consensus, Gaussian process incremental sample consensus, or agglomerative clustering. We show that a basic 100 line implementation in C++ achieves real time performance on a CPU with accuracy rivalling state-of-the-art methods on hilly streets.**

I. Introduction

Removal of the ground plane is necessary in many lidar-based perception algorithms for autonomous terrestrial vehicles. Once the ground plane is removed, the remaining points may then be identified as obstacles, clustered, and labelled.

Challenges to ground plane removal include sensor characteristics and uneven terrain.

Traditional approaches rely on simple thresholding. For example, by thresholding on the height of points [6]. This fails on sloped ground.

A popular class of non-parametric approaches is region-growing, which initializes a small set of *seed* points which are assumed to be on the ground, and then propagates the label to its neighbors. The propagation criterion may be based on surface normals [5], variance of height from a Gaussian distribution [2], . Propagation may be performed in 3D (computationally expensive) [5] [2], or in the 2D range image space where points are binned into a grid with angular coordinates (only works for certain sensor geometries), such as [9] [3]. Region-growing may not work if the ground appears in multiple disjoint regions, or if the initial seeds are incorrect.

Parametric methods assume that the shape of the ground may be expressed as surface, or model. Certain points are assumed to be inliers, and then the model parameters are fit to the inlier points. The simplest method assumes that the ground is planar. This fails if the ground has an undulating profile. Various methods fit line segments to partial slices of lidar scans [3] [1]. More complicated parametric surfaces are slow to converge, and random sample methods are nondeterministic and do not offer guarantees of convergence.

Recently, deep-learning methods are popular. Convolutional neural networks, originally developed for 2D image classification, may be applied to range images. Such methods achieve state-of-the-art accuracy with fast performance [7] [8]. The drawback of these techniques is the requirement of a large amount of training data, which may involve laborious manual annotation or simulation [8] and large computational resources during training. Moreover, a trained network cannot usually generalize to different sensor geometries, for example, lidars with different orientations, lidars that scan in a pattern that is not easily binned into a range image, or multiple lidars. Deep learning strategies are also difficult to formally verify and validate, and may be affected by adversarial inputs.

We propose a geometric approach to removing the ground plane, which requires only three parameters: maximum ground slope, ground thickness, and number of outliers. The algorithm works using a well-known algorithm from 1975 for finding the maxima, also known as the Pareto front, of a point set [4]. It works in 3D cartesian coordinates and is agnostic with respect to sensor geometry, point set density, sensor orientation, and so forth. It is easy to implement, requiring under 100 lines of readable C++ to achieve real time performance on one CPU core with a 64-beam spinning lidar sensor such as a Velodyne HDL-64E or Ouster OS1.

We demonstrate the accuracy of the algorithm on the hilly streets of San Francisco.

II. Method

Given the maximum ground slope $s \geq 0$, ground thickness $d \geq 0$, and number of outliers $k \geq 0$, the algorithm works like this: For each point, there is a cone with its apex a distance d above each point, and extends upwards to infinity with slope s. A point is considered part of the ground if it is not contained within a stack of k cones where each cone is contained within all cones beneath it.

[1]Daniel L. Lu is a computer vision engineer at Ouster, Inc. daniel.lu@ouster.io

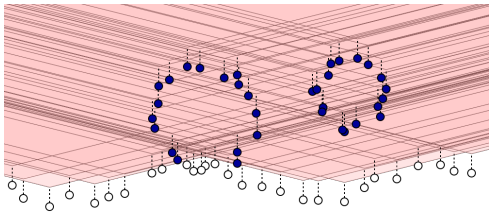

Fig. 1. In this 2D case, blue points are classified as non-ground because they are contained within the pink cones. The 3D case is analogous; we approximate 3D cones by a nonagon composed of the intersection of three triangular pyramids.

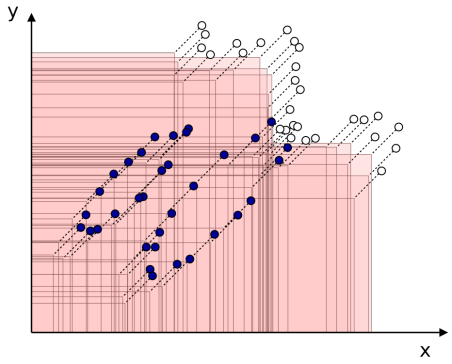

Fig. 2. By scaling and rotating the previous image, we reduce the problem to the point set maxima problem.

For ease of implementation, we approximate the cone with a regular polygonal pyramid with $3m$ sides where m is a positive integer. In practice, $m = 3$ works well, as a nine-sided polygon is very close to a circle.

Suppose that in our original point cloud $\mathcal{P}$, the z-axis is height in the vehicle frame. Then, let the point set $\mathcal{S} = \{\boldsymbol{R}\boldsymbol{p} \forall \boldsymbol{p} \in \mathcal{P}\}$, where $\boldsymbol{R}$ is an arbitary rotation matrix whose third column is all $-\frac{1}{\sqrt{3}}$, such as

$$\boldsymbol{R} = \begin{bmatrix} \frac{\sqrt{2}}{\sqrt{3}} & 0 & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} \end{bmatrix}. \tag{1}$$

A point $\boldsymbol{p}$ is said to be *dominated* by a point $\boldsymbol{q}$ if $q_x > p_x$, $q_y > p_y$, and $q_z > p_z$. A point $\boldsymbol{p} \in \mathcal{S}$ is *maximal* if it is not dominated by any other point $\boldsymbol{q} \in \mathcal{S}$. Due to our transformation $\boldsymbol{R}$, this is equivalent to saying that the point $\boldsymbol{p}$ is maximal if it is not contained within a triangular pyramid extending upwards from a point $\boldsymbol{q}$. In order to obtain the aforementioned $3m$-sided polygon, we repeat the algorithm m times, rotating the point cloud $\mathcal{P}$ about the z-axis by $\frac{2\pi}{3m}$ radians each time, and take the set union of the ground points produced by the m runs.

The 3D maximal set may be calculated using a variety of algorithms, the most well-known of which is Kung's algorithm [4], briefly described here: We first sort the points by z. We iterate over points in decreasing order of z, while using a balanced binary search tree to maintain the set of points whose 2D projections in the xy plane is maximal out of all those seen so far. For each point, we test whether it is maximal amongst the points in the tree, by traversing the tree by comparing on x and then by y. If it is maximal, we remove the points from the tree which are dominated by the current point, and insert the current point into the tree.

The ground slope parameter s is used by scaling the original point cloud $\mathcal{P}$ in its z-axis by a factor $1/s$. In fact, the slope of the nonagon pyramid is a constant factor of s, but we omit the simple calculation for brevity.

To account for the ground thickness parameter d, we duplicate each point $\boldsymbol{p}$ by inserting into $\mathcal{S}$ a new *shifted* point $\boldsymbol{p}' = \boldsymbol{p} + \frac{d}{s\sqrt{3}}\mathbf{1}$, where $\mathbf{1} = [1, 1, 1]^T$. When iterating over points as mentioned previously, if a shifted point is not dominated by points in the binary search tree, we output its unshifted counterpart as a ground point. However, we do not update the binary search tree when processing shifted points. Only unshifted points are used to update the binary search tree.

Occasionally, outliers can cause misclassification — if there is even a single point far beneath the ground (e.g. an erroneous reading due to a specular reflection on the ground), the pyramid extending above it will contain many actual ground points. To combat such outliers, we apply an "onion-peeling" technique, by running our algorithm k times, permanently labelling the maximal points as ground and removing them from consideration each time.

A. Pre- and postprocessing

The algorithm may be refined by the addition of optional pre-processing and post-processing steps.

As with many ground segmentation algorithms, obstacles near the vehicle may occasionally be misclassified as ground. If the lidar's position relative to the vehicle's wheels is known, the point cloud may be pre-processed by inserting extra points on the ground under the wheels. The cones emanating from these points will ensure that above-ground obstacles near the vehicle are classified correctly. Inserting a constant number of points does not change the time complexity of the algorithm.

Since the ground is assumed to have some thickness d to account for sensor noise, the bottom d of an obstacle

may be misclassified as ground. To mitigate this, the point cloud may be postprocessed by projecting all points into the xy plane, and then partitioning into a fine grid. Any ground points with a non-ground point less than a distance d above are relabeled as non-ground. Note that the grid cell spacing introduces an additional parameter to the algorithm, though using d for the spacing produces reasonable results. Using a hash table for the grid, the cost of this postprocessing step is a linear $O(n)$ time.

III. Discussion

The ground segmentation scheme produces ground points which can neither have slopes steeper than s, or steps taller than d, satisfying the basic requirements of traversability. Moreover, the algorithm is guaranteed to detect obstacles taller than d which are more than k points tall. Note that most published methods, particularly ones based on deep learning, offer no such guarantees.

The time complexity is $O(n \log n)$ for sorting. The total number of queries, insertions, and deletions of the binary search tree is $O(n)$, with an $O(n \log n)$ cost for each operation, yielding a total time complexity of $O(n \log n)$. Since the algorithm is run m times for rotation about the z-axis and k times for onion-peeling, the total time complexity may be written as $O(kmn \log n)$, although we note that the constant parameters k and m are both not greater than 3 usually.

A weakness of this approach is that deep holes in the ground or cliffs may cause nearby ground points to appear as not ground points. In practice, deep holes are rarely encountered, most sensor configurations cannot peer into deep holes, and most terrestrial autonomous vehicles should avoid deep holes and cliffs anyway.

IV. Results

V. Conclusion

References

[1] Chen, T., Dai, B., Wang, R., & Liu, D. (2014). Gaussian-process-based real-time ground segmentation for autonomous land vehicles. *Journal of Intelligent & Robotic Systems*, 76(3-4), 563-582.

[2] Douillard, B., Underwood, J., Kuntz, N., Vlaskine, V., Quadros, A., Morton, P., & Frenkel, A. (2011, May). On the segmentation of 3D LIDAR point clouds. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 2798-2805). IEEE.

[3] Himmelsbach, M., Hundelshausen, F. V., & Wuensche, H. J. (2010, June). Fast segmentation of 3d point clouds for ground vehicles. In *Intelligent Vehicles Symposium (IV), 2010 IEEE* (pp. 560-565). IEEE.

[4] Kung, H. T., Luccio, F., & Preparata, F. P. (1975). On finding the maxima of a set of vectors. *Journal of the ACM (JACM)*, 22(4), 469–476.

[5] Moosmann, F., Pink, O., & Stiller, C. (2009, June). Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium*, 2009 IEEE (pp. 215-220). IEEE.

[6] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., ... & Lau, K. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of field Robotics*, 23(9), 661-692.

[7] Velas, M., Spanel, M., Hradis, M., & Herout, A. (2017). CNN for very fast ground segmentation in Velodyne lidar data. *arXiv preprint arXiv:1709.02128*.

[8] Wu, B., Wan, A., Yue, X., & Keutzer, K. (2017). Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. *arXiv preprint arXiv:1710.07368*.

[9] Zermas, D., Izzat, I., & Papanikolopoulos, N. (2017, May). Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on* (pp. 5067-5073). IEEE.