Fast Object Segmentation Pipeline for Point Clouds Using Robot Operating System

Anjani Josyula
Department of Electrical Engineering
Indian Institute of Technology
Hyderabad, India
anjanij@iith.ac.in

Bhaskar Anand
Department of Electrical Engineering
Indian Institute of Technology
Hyderabad, India
ee18resch11006@iith.ac.in

P. Rajalakshmi
Department of Electrical Engineering
Indian Institute of Technology
Hyderabad, India
raji@iith.ac.in

***Abstract*—This paper presents a method to pipeline the segmentation process for point clouds using the Robot Operating System (ROS) and the Point Cloud Library (PCL). The pipeline's objective is to optimize the run time of a conventional segmentation algorithm by working within the Robot Operating System framework. It can be implemented using any system and in conjunction with a GPU. It shows the greatest reduction in run time for the least downsampled clouds. Therefore, it can be used for real-time safety-critical applications especially in scenarios where the point cloud is sparse or of highly uneven spatial density and should thus not be downsampled. It was developed for Obstacle Avoidance for Autonomous Vehicles and Drones where segmentation is only the first step of a larger pipeline involving obstacle detection and tracking. It was observed to reduce run time up to 31.3% on the KITTI data set and up to 44.4% on data collected from a 16 channel Ouster lidar at the Indian Institute of Technology, Hyderabad.**

***Index Terms*—Robot Operating System, Point Clouds, Autonomous Vehicles, Lidar, Intelligent Transportation Systems**

I. INTRODUCTION

Point cloud data is one of the most exhaustive sensor information that can be used for object detection and avoidance. With the advent of cheaper and more accurate Lidar sensors with many channels, this sensor is fast becoming ubiquitous in autonomous vehicles and drones. For autonomous vehicles, obstacle segmentation and detection algorithms must be both fast and accurate. The challenge posed, is to optimize both run time and accuracy while recognizing that to ameliorate one is often to deteriorate the other. This paper presents a method by which the run time of point cloud segmentation can be reduced without compromising on accuracy.

Segmentation of point clouds into various objects is the first step to object detection and tracking. Object segmentation means classifying points with similar characteristics (geometric, radiometric etc) into clusters. Most point cloud segmentation algorithms comprise of many sequential time-consuming steps which can be broadly classified as belonging to (i) ground removal and (ii) object segmentation.

In the proposed pipeline we first divide each point cloud into parts, called slices, based on radial distance from the Lidar sensor. Each of the algorithm's steps becomes a parallel stage in the pipeline and slices are passed from one stage to the next. Like any pipeline, this maximizes throughput and minimizes run time.

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

Point cloud data, by its nature, is prone to highly varying spatial density, particularly in urban environments. In such a scenario, or in cases where point cloud data is very sparse, downsampling the point clouds could lead to loss of critical information. This loss could be the finer details of an object, like the side mirrors of a car. It could also be in the loss of an object altogether if it is sparsely represented or small - like a small pedestrian. Therefore, downsampling results in deterioration of accuracy which is disastrous in safety-critical applications like autonomous vehicles. However, it is often necessary to reduce run time since point cloud data is large. The proposed pipeline offers an alternative to downsampling in terms of reducing the run time of segmentation algorithms. The percentage reduction in run time increases with decreasing downsampling rates making it highly suitable for large point cloud data.

This paper is divided into five sections. The first section gives an introduction to object segmentation and the proposed pipeline. In the next section, we present a brief survey of fast segmentation techniques proposed by various authors. The pipeline stages are described in the third section. In the fourth section, we share our results while the fifth section presents both our conclusions and future work.

II. PREVIOUS WORK

Since segmentation is a necessary first step to many critical tasks for autonomous vehicles, much work has been done on fast segmentation methods. We divide these segmentation methods into two types.

The first type consists of those methods which reduce run time via the algorithm itself. Some examples of these are [3] [4] which use scan line based methods to cluster objects and are both fast and accurate. Using a height threshold [3] [2] and random selection of seed points for ground plane fitting [4] are both used effectively for ground removal. Another approach is to use a method in which the point cloud is conceptually treated as a graph. Two neighbouring points may be connected (depending upon the criteria in the algorithm) for the formation of the graph. Golovinskiy [8] suggested a method in which the k-nearest neighbours (KNN) graph is used for foreground-background segmentation. [5] [6] use geometric properties and Euclidean distance to segment objects. In [7], the authors have suggested point cloud segmentation based on spatial and echo characteristics. Here, the intensity of the echo reflected by objects is observed and used in segmentation.

The second type consists of those methods which rely on implementation or architecture to speed up an already existing algorithm. Much of this research is directed towards using NVIDIA's parallel computing architecture (CUDA) to improve run time. [10] proposes a GPU accelerated registration method which is useful for those segmentation models (like [2]) which use multiple scans over time. [9] proposes using a GPU to make a scan line based segmentation method faster. In [12], which is closest to our work, fast ground segmentation is done using multiple Robot Operating System nodes and a GPU. Najdataei et al. [11] propose a parallel clustering algorithm over the standard Point Cloud Library's Euclidean distance based clustering.

Since the underlying principle for the proposed pipeline is the division of point clouds into parts, it is worthwhile to discuss methods of point cloud division. Himmelsbach et al. [1] propose the division of point clouds by considering the *XY* plane to be a circle of infinite radius. The points are first divided into angular segments or sectors and then further divided into bins based on distance from the center or radius. Points belonging to the ground are removed via line fitting and object segmentation is done using 3D Voxel Grid Fitting. Asvadi Alireza et al. [2] propose the division of point clouds using circles of increasing radii centered at the lidar's position and ground removal using RANSAC ground plane fitting. In this method, temporal information is also considered to classify objects as dynamic or static by combining scans and object segmentation is done using voxels.

III. Segmentation Pipeline

The proposed pipeline was implemented using a modification of [2]. It uses a parallel architecture to reduce the run time of conventional segmentation algorithms within the Robot Operating System (ROS) framework as opposed to much of the previous work in this area which focuses on using the CUDA framework to reduce execution time. This pipeline can be adopted on any system having ROS and is independent of architecture.

The pipeline's principle is to divide the sequential steps of the algorithm into stages which run in parallel. These stages can be classified into two types - those required for ground removal and those required for object segmentation. Point clouds are divided into pieces called slices - each of which must undergo all the stages of the pipeline for segmentation. These slices are passed along from one stage to another after being processed. This allows multiple slices to be processed in parallel as opposed to all slices being sequentially processed in each step of the segmentation algorithm and reduces run time significantly.

The stages were implemented using the Robot Operating System (ROS) *nodelets*. Each stage subscribes to a ROS topic containing messages which are the outputs of the previous stage. Each message represents one slice. It then processes this slice and outputs or publishes it to another topic which is subscribed to by the next slice. This process is outlined in Fig. 1.

ROS *nodelets* were used over *nodes* to pipeline the algorithm because of the disadvantage of data overhead while transferring data between *nodes*. Data overhead refers to the time lost in the transfer of messages between the stages of the pipeline. Data overhead is considerable when working with *nodes* because when transferring data between two nodes, data is serialized at the publisher's end, transferred and de-serialized at the subscriber's end. Thus, it is possible that whatever time is gained in terms of processing is lost in terms of data overhead. However, *nodelets* allow classes to be dynamically loaded to the same node, but behave as independent processes themselves. This allows zero copy transport between publisher and subscriber *nodelets* belonging to the same node. It means that pointers to the data ,instead of the data itself, are passed between the *nodelets*.

In this method, the pipeline's run time is determined by the run times of individual stages. The pipeline is optimized by making the run times of all the stages as similar as possible. This prevents bottlenecks and potential ROS *topic* buffer overflow which leads to data loss dangerous to safety-critical applications.

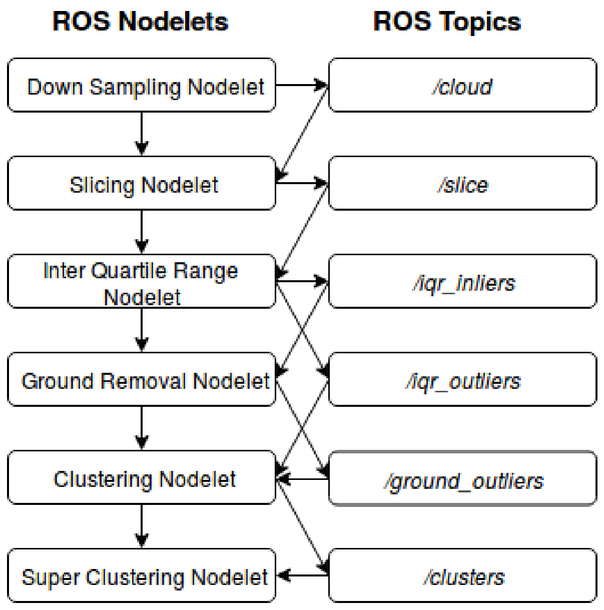

Fig. 1: Pipeline

A. Down Sampling

The pipeline's first step is to downsample the Point Cloud using a Voxel or Box Grid Filter. This step represents the sole bottleneck in the pipeline since it is the only step performed on the entire point cloud. This filter superimposes a Voxel Grid of a given leaf size over the point cloud. The points in each leaf of the grid are replaced by a single point which is their centroid. Most practical implementations of segmentation algorithms downsample the point clouds to reduce computation time or data size. However, this can be disastrous in safety-critical applications such as autonomous driving as point clouds taken in urban scenarios can have drastic changes in spatial density and it's possible that a small pedestrian, for example, could be very sparsely represented. Downsampling in this scenario could lead to losing the object, classifying it as noise or some other wrong classification. One of this pipeline's primary purposes is to provide an alternative to downsampling to reduce run time. As the downsampling leaf size decreases, the percentage improvement in run time increases for the proposed pipeline.

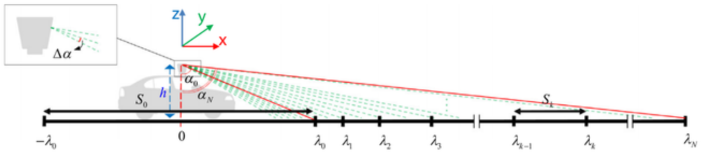

Fig. 2: Slicing [2]

B. Slicing

Point clouds are divided into parts , called slices, as in [2]. Slices are defined by inner and outer radii of circles centered at the Lidar sensor's position. These radii are obtained using the tan function and set values for the sensor's height (h), vertical field of view in degrees (α_0), vertical resolution ($\Delta\alpha$) and range (λ_N).The maximum angle (α_N) as shown in Fig. 2 is given by:

$$\alpha_N = \arctan(\lambda_N/h) \tag{1}$$

The inner bounding radii (r_i) of each slice is given by the function:

$$\lambda_i = h \cdot tan(\alpha_0 + i \cdot \Delta\alpha) \tag{2}$$

where $0 \leq i \leq N$. N denotes the total number of slices and is given by:

$$N = \lfloor \alpha_N - \alpha_0/\Delta\alpha \rfloor \tag{3}$$

where $\lfloor\cdot\rfloor$ denotes the floor or step function.

A point belongs to a slice if its Euclidean distance from the origin is greater than that slice's inner bounding radius and less than the inner bounding radius of the next slice.

C. Inter Quartile Range

The next step is the Interquartile Range (*IQR*) method. This step's purpose is to reduce the number of points which must be tested to see whether they belong to the ground in the next stage. In this method, the heights of the points are arranged into ascending order and divided into two halves, removing the median value to do so if there are an odd number of values. The median of the lower half is called the lower quartile value (*Q25*) while the median of the upper half is called the upper quartile value (*Q75*). The Inter Quartile Range (*IQR*) is the difference between these two values. The upper (*Qmax*) and lower (*Qmin*) gate limits are obtained using these values.

$$Qmin = Q25 - 1.5 \cdot IQR \tag{4}$$

$$Qmax = Q75 + 1.5 \cdot IQR \tag{5}$$

D. Ground Removal

The next step is to remove ground - again by slice. The method selected for ground removal is the RANSAC method as described in [11]. The RANSAC algorithm is a model fitting algorithm. It attempts to fit a particular model to a group of points. It classifies points as inliers of that model, provided they lie within some error threshold.

In an urban scenario, the ground is typically comprised of many planes. Thus applying the RANSAC to the entire point cloud is unlikely to yield good results. However, its performance is improved when applied by slice. It is used to fit a plane to the inliers obtained from the *IQR* gating strategy of the previous stage. The plane equation is:

$$ax+by+cz=d \tag{6}$$

The outliers of the RANSAC algorithm combined with the outliers obtained from the *IQR* method together comprise the non-ground points of the slice while the inliers obtained from the RANSAC algorithm comprise the slice's ground points.

E. Clustering

This stage of the pipeline begins object segmentation. This penultimate stage in the pipeline clusters all non-ground points of each slice into various objects. This clustering is done using the standard Euclidean Clustering Algorithm, as implemented by the Point Cloud Library. In this algorithm, each point is first added to a queue. Then, points are searched for throughout a sphere of radius *dthresh* centered at the point in question. If they lie within this sphere and haven't already been classified, they are added to the queue. Once, all the queue points have been processed they are added to a list of clusters. The proposed pipeline clusters objects by slice. However, if an object lies across multiple slices as is highly possible, it will be over-segmented.

F. Super Clustering

The final stage in the pipeline combines clusters based on the Euclidean distance between their centroids. If this distance between the centroids of two clusters is less than a threshold *superthresh*, they are considered to be a part of the same object and are recombined.

IV. Result and Analysis

The pipeline was run on both the KITTI data set and data collected from an Ouster Lidar on a system having an Intel Core i5 8th Gen processor with 2.8 GHz speed and 8GB RAM. The data sets differ primarily in terms of point cloud size. The KITTI vision benchmark suite [14] is a 3D object detection benchmark consists of 7481 training images and

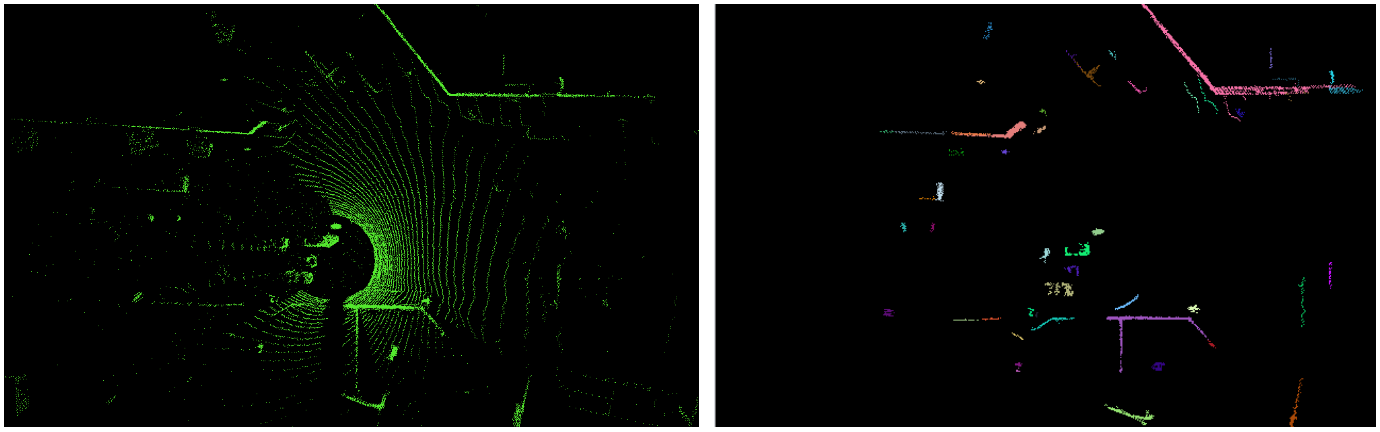
(a) Point Cloud Before Segmentation (b) Clusters After Segmentation

Fig. 3: Segmentation of Point Cloud From KITTI Data Set

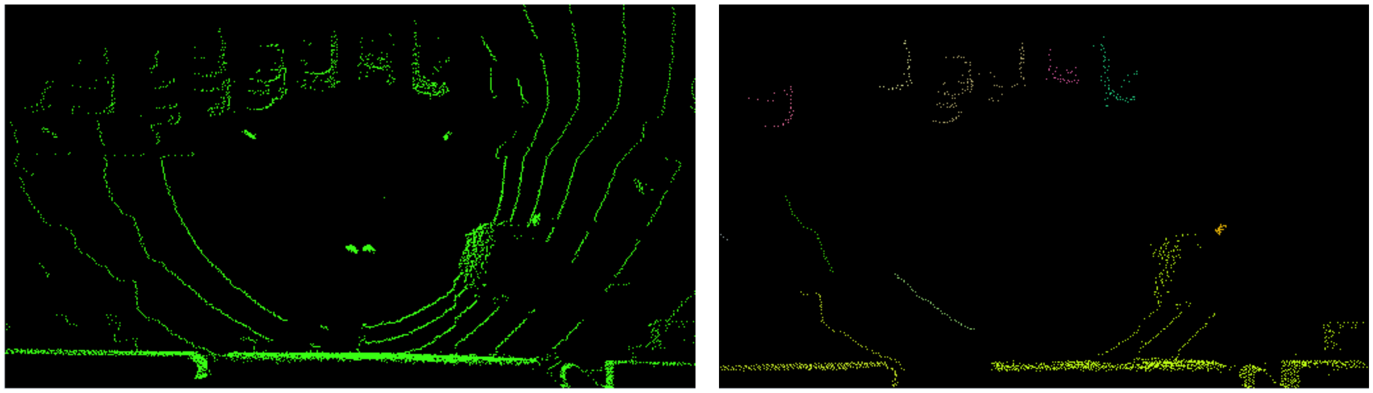
(a) Point Cloud Before Segmentation (b) Clusters After Segmentation

Fig. 4: Segmentation of Point Cloud From Ouster Data Collected at IIT-H

Voxel Grid Leaf Size	Avg Decrease in Run Time (ms)	Avg Decrease in Run Time (%)	STDEV of Decrease in Run Time (ms)
0.21	50.46	13.36	3.80
0.19	58.11	13.87	3.66
0.17	65.57	13.94	2.12
0.15	76.83	14.44	9.60
0.13	91.17	15.03	3.78
0.11	110.90	15.83	2.99
0.09	135.88	16.66	4.37
0.07	186.58	19.07	6.10
0.05	379.56	26.29	138.78
0.03	529.07	31.39	20.74

TABLE I: Run Time Reduction on KITTI Data Set

Voxel Grid Leaf Size	Avg Decrease in Run Time (ms)	Avg Decrease in Run Time (%)	STDEV of Decrease in Run Time (ms)
0.17	38.99	42.98	3.93
0.15	42.73	43.05	2.91
0.13	46.8	43.77	4.88
0.11	52.63	44.45	4.94

TABLE II: Run Time Reduction on Ouster Data Taken at IITH

7518 test images as well as the corresponding point clouds. It has more than 200k 3D object annotations captured in cluttered scenarios (up to 15 cars and 30 pedestrians are visible per image). The data set was collected using a Velodyne HDL-64E with 64 channels and an average point cloud size of over 100,000 points. The point cloud data at IIT-H was collected using Ouster-OS1 lidar. It has 16 channels and an average point cloud size of 65536 points.

Table I shows the results of the pipeline on the KITTI dataset and Table II shows the same for the data set captured at Indian Institute of Technology - Hyderabad (IIT-H) with the 16-channel Ouster Lidar (OS-1). They show the variation in the reduction of run time with the variation of leaf size of the downsampling voxel grid. It can be seen that the greatest percentage reduction in run time is for the smallest leaf size - meaning the least down sampled point clouds.

V. Conclusion

As advances are made in Intelligent Transportation Systems, there is a need for fast and efficient on-board computing techniques for autonomous vehicles. Lidar is becoming ubiquitous in obstacle detection and avoidance for autonomous vehicles and fast segmentation of point clouds is an essential precursor step. Downsampling point clouds is dangerous in this scenario and can have disastrous consequences, despite often being used to reduce point cloud data size and decrease run times for processing algorithms. This paper proposes a pipeline for segmentation that reduces run time significantly for large point clouds within the ROS framework and is independent of GPU usage and architecture.

This pipeline is prone to over-segmentation, single objects being represented by multiple clusters, as opposed to under-segmentation, in which multiple segments are lumped together. Future work involves the rectification of this over-segmentation as in [13], the use of temporal data and the modification of object classification algorithms to use multiple super points or clusters.

VI. Acknowledgment

This work was supported by the project M2Smart: Smart Cities for Emerging Countries based on Sensing, Network and Big Data Analysis of Multimodal Regional Transport System, JST/JICA SATREPS, Japan.

References

[1] Himmelsbach, Michael, Felix V. Hundelshausen, and H-J. Wuensche. ”Fast segmentation of 3d point clouds for ground vehicles.” In Intelligent Vehicles Symposium (IV), 2010 IEEE, pp. 560-565. IEEE, 2010.

[2] Asvadi, Alireza, et al. ”3D Lidar-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes.” Robotics and Autonomous Systems 83 (2016): 299-311.

[3] Chu, Phuong Minh, et al. ”Fast point cloud segmentation based on flood-fill algorithm.” Multisensor Fusion and Integration for Intelligent Systems (MFI), 2017 IEEE International Conference on. IEEE, 2017

[4] Zermas, Dimitris, Izzat Izzat, and Nikolaos Papanikolopoulos. ”Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications.” Robotics and Automation (ICRA), 2017 IEEE International Conference on. IEEE, 2017.

[5] Bogoslavskyi, Igor, and Cyrill Stachniss. ”Fast range image-based segmentation of sparse 3D laser scans for online operation.” Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on. IEEE, 2016.

[6] F. Moosmann, O. Pink, and Ch. Stiller. Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In Proc. of the Intelligent Vehicles Symposium, pages 215220, 2009

[7] J. Hongbo and J. Qisong, ”TLS point cloud segmentation based on points features,” 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, 2017.

[8] Golovinskiy, A., Funkhouser, T. (2009, September). Min-cut based segmentation of point clouds. In Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on (pp. 39-46).

[9] Hu, Xiangyun, Xiaokai Li, and Yongjun Zhang. ”Fast filtering of LiDAR point cloud in urban areas based on scan line segmentation and GPU acceleration.” IEEE Geoscience and Remote Sensing Letters 10.2 (2013): 308-312.

[10] Eckart, Benjamin, and Alonzo Kelly. ”REM-Seg: A robust EM algorithm for parallel segmentation and registration of point clouds.” Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, 2013.

[11] Najdataei, Hannaneh, et al. ”Continuous and Parallel LiDAR Point-Cloud Clustering.” 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2018.

[12] Baker, Stuart P., and Robert W. Sadowski. ”GPU assisted processing of point cloud data sets for ground segmentation in autonomous vehicles.” Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on. IEEE, 2013.

[13] Held, David, Devin Guillory, Brice Rebsamen, Sebastian Thrun, and Silvio Savarese. ”A Probabilistic Framework for Real-time 3D Segmentation using Spatial, Temporal, and Semantic Cues.” In Robotics: Science and Systems. 2016.

[14] Fritsch, Jannik, Tobias Kuehnl, and Andreas Geiger. ”A new performance measure and evaluation benchmark for road detection algorithms.” 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). IEEE, 2013.