Fast Geometric Surface Based Segmentation of Point Cloud from Lidar Data

Aritra Mukherjee[1(✉)], Sourya Dipta Das[2], Jasorsi Ghosh[2], Ananda S. Chowdhury[2], and Sanjoy Kumar Saha[1]

[1] Department of Computer Science and Engineering, Jadavpur University, Kolkata, India
kalpurush1601@gmail.com, sks_ju@yahoo.co.in
[2] Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India
dipta.math@gmail.com, jasorsi13@gmail.com, ananda.chowdhury@gmail.com

Abstract. Mapping the environment has been an important task for robot navigation and Simultaneous Localization And Mapping (SLAM). LIDAR provides a fast and accurate 3D point cloud map of the environment which helps in map building. However, processing millions of points in the point cloud becomes a computationally expensive task. In this paper, a methodology is presented to generate the segmented surfaces in real time and these can be used in modeling the 3D objects. At first an algorithm is proposed for efficient map building from single shot data of spinning Lidar. It is based on fast meshing and subsampling. It exploits the physical design and the working principle of the spinning Lidar sensor. The generated mesh surfaces are then segmented by estimating the normal and considering their homogeneity. The segmented surfaces can be used as proposals for predicting geometrically accurate model of objects in the robots activity environment. The proposed methodology is compared with some popular point cloud segmentation methods to highlight the efficacy in terms of accuracy and speed.

Keywords: Unsupervised surface segmentation · 3D point cloud processing · Lidar data · Meshing

1 Introduction

Mapping of environment in 3D is a sub-task for many robotic applications and is a primary part of Simultaneous Localization And Mapping (SLAM). For 3D structural data sensing, popular sensors are stereo vision, structured light, TOF (Time Of Flight) cameras and Lidar. Stereo vision can extract RGBD data but the accuracy is highly dependent on the presence of textural variance in the scene. Structured light and TOF cameras can extract depth information and

B. Deka et al. (Eds.): PReMI 2019, LNCS 11941, pp. 415–423, 2019.
https://doi.org/10.1007/978-3-030-34869-4_45

RGBD data respectively in an accurate fashion. But these are mostly suitable for indoor uses and in a low range. Lidar is the primary choice for sensing accurate depth in outdoor scenarios over long range. Our focus in this work is on the unsupervised geometric surface segmentation in three dimensions based on Lidar data. We would like to emphasize that models built from such segmentation can be very useful for tasks like autonomous vehicle driving.

According to Nguyen *et al.* [13], the classic approaches in point cloud segmentation can be grouped into edge based methods [3], region based methods [1,9,11,17], attributes based methods [4,6,8,19], model based methods [16], and graph based methods [2,10]. Vo *et al.* [17] proposed a new octree-based region growing method with refinement of segments and Bassier *et al.*. [1] improved it with Conditional Random Field. In [9,11], variants of region growing methods with range image generated from 3D point cloud are reported. Ioannou *et al.* [8] used Difference of Normals (DoN) as a multiscale saliency feature used in a segmentation of unorganized point clouds. Himmelsbach *et al.* [7] treated the point clouds in cylindrical coordinates and used the distribution of the points for line fitting to the point cloud in segments. Ground surface was recognized by thresholding the slope of those line segments. In an attempt to recognize the ground surface, Moosmann *et al.* [12] built an undirected graph and characterized slope changes by mutual comparison of local plane normals. Zermas *et al.* [18] presented a fast instance level LIDAR Point cloud segmentation algorithm which consisting of deterministic iterative multiple plane fitting technique for the fast extraction of the ground points, followed by a point cloud clustering methodology named Scan Line Run (SLR). On the other hand, in supervised methods, PointNet [14] takes the sliding window approach to segment large clouds with the assumption that a small window can express the contextual information of the segment that it belongs to. Landrieu *et al.* [10] introduced superpoint graphs, a novel point cloud representation with rich edge features encoding the contextual relationship between object parts in 3D point clouds. This is followed by deep learning on large-scale point clouds without major sacrifice in fine details.

In most of the works, neighbourhood of a point used to estimate normal is determined by tree based search. This search is time consuming and the resulting accuracy is also limited for sparse point cloud as provided by a Lidar in robotic application. This observation acts as the motivation for us to focus on developing a fast mesh generation procedure that will provide the near accurate normal in sparse point cloud. Moreover, processing Lidar data in real-time requires considerable computational resources limiting its deployability on small outdoor robots. Hence a fast method is also in demand.

2 Proposed Methodology

The overall process consists of three steps as shown in Fig. 1. First, the point cloud is sensed by the Lidar, subsampled and the mesh is created simultaneously. Second, surface normal is calculated for node points using the mesh. Finally, surface segmentation is done by labelling the points on the basis of spatial continuity of normal with a smooth distribution.

Fig. 1. Block diagram of the entire system, the first stage can be merged with Lidar scanning by improvising the Lidar firmware

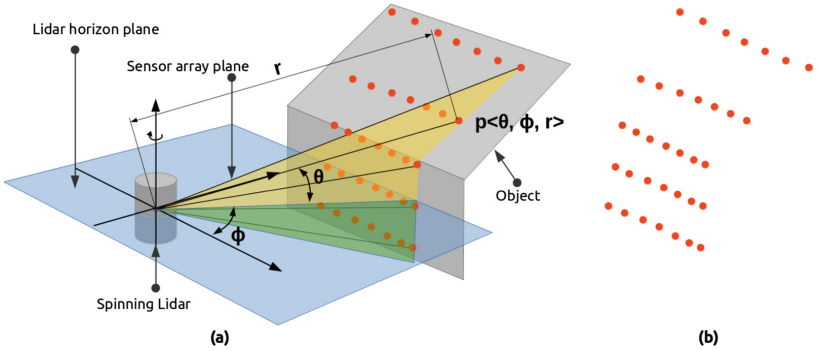

Fig. 2. (a) A schematic showing the formation of point cloud by Lidar and (b) the resultant point cloud

The proposed methodology segments surface from point cloud obtained by spinning Lidars only. Spinning Lidars work on the principle of spinning a vertical array of divergent laser distance sensors and thus extracts point cloud in spherical coordinates. The point cloud consists of a set of coordinates $P = \{p(\theta, \phi, r)\}$ where θ is the fixed vertical angle of a sensor from the plane perpendicular to the spinning axis, ϕ is the variable horizontal angle due to spinning of the array and r is the distance measured by the laser sensor. This form of representation is exploited by our methodology to structure the data in an ordered form the only caveat being running it for a single spin. By varying the factor of sub-sampling of ϕ, the horizontal density of the point cloud can be varied. Figure 2 shows the operational procedure of a spinning Lidar and the resultant point cloud for an object with multiple surfaces. Please note that not every point during the sweep is considered for mesh construction as noisy points too close to each other horizontally produces erroneous normal. Sub-sampling is done to rectify this error by skipping points uniformly during the spin.

Mesh Construction: The significant novelty of this work is the fast mesh generation process that enables quick realization of subsequent steps. The mesh construction is a simultaneous process during the data acquisition stage. The connections are done during sampling in the following manner. Let a point be denoted as $p(\theta, \phi, r)$. Let the range of θ be $[\theta_0, \theta_n]$ which corresponds to $n + 1$ vertical sensors in the array; and the range of ϕ be $[\phi_0, \phi_m]$ where $m + 1$ is the

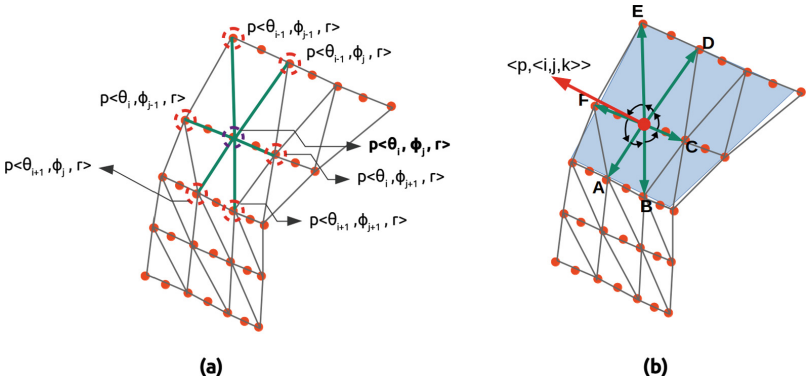

Fig. 3. (a) A schematic showing the formation of mesh on subsampled (factor 2) cloud with the neighbour definition of a point and (b) the normal formation from the neighbours

number of times the sensor array is sampled uniformly during a single spin. The corresponding distance of the point is r from the Lidar sensor. Let the topmost sensor in the array corresponds to angle θ_0 and the count proceeds from top to bottom the vertically spinning array. Further, let first shot during the spin corresponds to ϕ_0. The mesh is constructed by joining neighbouring points in the following manner: $p(\theta_i, \phi_j, r)$ is joined with $p(\theta_{i+1}, \phi_j, r)$, $p(\theta_{i+1}, \phi_{j+1}, r)$ and $p(\theta_i, \phi_{j+1}, r)$ for all points within range of $[\theta_0, \theta_{n-1}]$ and $[\phi_0, \phi_{m-1}]$. The points from the last vertical sensor, *i.e.*, corresponding to θ_n, are joined with the immediate horizontal neighbour. Thus, $p(\theta_n, \phi_j, r)$ is joined with $p(\theta_n, \phi_{j+1}, r)$ where j varies from 0 to $m-1$. For points corresponding to ϕ_m, $p(\theta_i, \phi_m, r)$ is joined with $p(\theta_i, \phi_0, r)$, $p(\theta_{i+1}, \phi_0, r)$ and $p(\theta_{i+1}, \phi_m, r)$. The point $p(\theta_n, \phi_m, r)$ is joined with $p(\theta_n, \phi_0, r)$ to create the whole cylindrical connected mesh. For all pairs the joining is done if both the points have an r that is within range of the Lidar. If all the neighbours of a point is present then six of them are connected by the meshing technique instead of all eight. This is done to ensure there are no overlapping surface triangles. Figure 3(a) shows the connectivity a point which have six valid neighbours on the mesh. The mesh is stored in a map of vectors $M = \{< p, v > | \; p \in P, v = \{q_n \mid q_n \in P, n \leq 6, \quad and \quad q_n \quad is \quad a \quad neighbour \quad of \quad p\}\}$ where each point is mapped to the vector v containing its existent neighbors in an ordered fashion. The computational complexity of the meshing stage in $O(n_{sp})$ where n_{sp} is the number of sub-sampled points. As the meshing is performed on the fly with the sensor spinning the absolute time depends on the angular frequency and sub-sampling factor.

Normal Estimation: The structured mesh created in the previous step helps toward a fast computation of normal at a point. A point forms vectors with its neighbour. Pair of vectors are taken in an ordered fashion. Normal is estimated

for that point by averaging the resultant vectors formed by cross multiplication of those pairs. The ordering is performed during the mesh construction stage only. For a point $p(\theta_i, \phi_j, r)$, vectors are formed with the existing neighbours as stored in M in an anti-clockwise fashion. A normal can be estimated for p if its corresponding v has $|v| \geq 2$. From the neighbour vector v of p obtained from M, let $p(\theta_i, \phi_j, r)$ forms $\boldsymbol{A}$ by joining with $p(\theta_{i+1}, \phi_j, r)$, $\boldsymbol{B}$ by joining with $p(\theta_{i+1}, \phi_{j+1}, r)$, and so on until it forms $\boldsymbol{F}$ by joining with $p(\theta_i, \phi_{j-1}, r)$. Then cross- multiplication of existing consecutive vectors is performed. In general, if every point exists, then $\boldsymbol{A} \times \boldsymbol{B}$, $\boldsymbol{B} \times \boldsymbol{C}$ etc. are computed ending with $\boldsymbol{F} \times \boldsymbol{A}$ to complete the circle. This arrangement is illustrated in Fig. 3(b). The normal is estimated by averaging all the $\hat{i}, \hat{j}, \hat{k}$ components of the resultant vectors individually. The normals are stored in the map $N = \{< p, n > |\ p \in P, n = \{\hat{i_p}, \hat{j_p}, \hat{k_p}\}\}$. Due to the inherent nature of the meshing technique, sometime points from disconnected objects get connected to the mesh. To mitigate this effect of a different surface contributing to the normal estimation, weighted average is used. The weight of a vector formed by cross multiplication of an ordered pair is inversely proportional to the sum of lengths of the vectors in the pair.

Algorithm 1. Surface segmentation on normal distribution

```
L = {< p, l > | p ∈ P, l = 0};
label ← 1, stack S ← {};
for each p ∈ M do
    if l = 0 for p in L then
        L ←< p, l ← label >;
        S.push(p);
        while S ≠ {} do
            p ← S.pop();
            for each q in v corresponding to p ∈ M do
                if l = 0 for q in L then
                    get î_q, ĵ_q, k̂_q for q from N;
                    if |î_p − î_q| < I, |ĵ_p − ĵ_q| < J, |k̂_p − k̂_q| < K then
                        L ←< q, l ← label >;
                        S.push(q);
                    end
                end
            end
        end
    else
        search next p in M
    end
end
Result: L
```

Segmentation by Surface Homogeneity: Based on the normal at a point, as computed in the previous step, we now propagate the surface label. A label map $L = \{< p, l >| \ p \in P, l = 0\}$ is used for this purpose. This label map stores the label of each point p by assigning a label l. If for any point p, its $l = 0$ denotes the point is yet to be labelled. The criteria of assigning the label of p to its neighbour q depends on the absolute difference of their normal components. Three thresholds I, J, K are empirically set depending on the type of environment. Segment labelling is propagated by a depth first search approach as described in algorithm 1. Two neighbouring points will have same label provided the absolute difference of corresponding components of their normals are within component-wise threshold. Computations of normals and mesh, as discussed earlier, generates the normal map N and the mesh M respectively. Subsequently algorithm 1 uses N and M to label the whole sub-sampled point cloud in an inductive fashion. Due to sub-sampling, all points in P will not get a label. This issue is resolved by assigning the label of its nearest labelled point along the horizontal sweep. An optional post-processing may be arranged by eliminating segments with too few points.

3 Experiments Results and Analysis

The proposed methodology is implemented with C++ on a linux based system with DDR4 8 GB RAM, 7th generation i5 Intel Processor. Experiments were performed on a synthetic dataset which simulates Lidar point clouds. The methodology is compared with the standard region growing algorithm used in point cloud library [15] and a region growing algorithm combined with merging for organized point cloud data [19] and it is observed that it excels in terms of both speed and accuracy.

We have used the software "Blensor" [5] to simulate the output of the Velodyne 32E Lidar in a designed environment. Blensor can roughly model the scenery with the desired sensor parameters. BlenSor also provides an exact measurement ground truth annotated over the model rather than the point cloud. We have included primitive 3D model structures such as cylinder, sphere, cube, cone and combined objects placed in different physically possible orientations in the scene to simulate the real environment. Different percentage of occlusion and crowding levels is included in the model environment to test out the property of scene complexity independence, of the proposed solution. We have a total of 32 different environments with increasing order of different types of objects, occlusion, complexity of geometry and pose, and crowding levels. We have used Gaussian noise as our noise model for Lidar with zero mean and variance of 0.01. With Velodyne 32E Lidar the total number of sensors in the spinning array is 32 and a resolution of 0.2 degree is set for horizontal sweep resulting in 1800 sensor firing in one spin. Thus for our dataset the range of θ and ϕ are $[\theta_0, \theta_{31}]$ and $[\phi_0, \phi_{1799}]$. The output at different stages of our methodology is shown in Fig. 4.

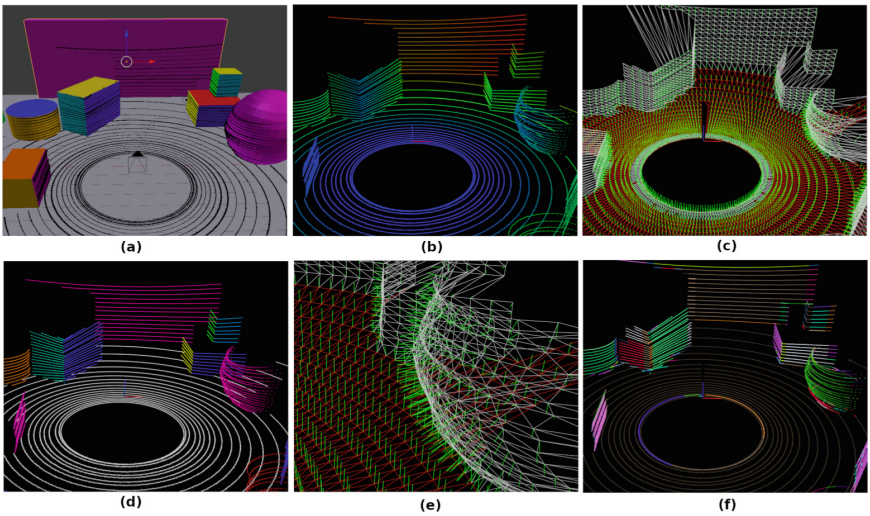

Fig. 4. (a) Synthetic scene with scanned point cloud overlayed (b) Point cloud with distance color coded from blue(least) to red(highest) (c) Mesh and normals with sub-sampling factor of 5 (d) Point cloud segment surface ground truth (e) A detailed look at the mesh and normals (f) Segmented point cloud by proposed methodology (Color figure online)

Table 1. Comparison of execution times (all units in milliseconds) of different competing methods.

Method	Sampling interval	Average time (in ms)	Max time (in ms)	Min time (in ms)
Proposed method	5	54.33	63	41
	10	35.06	48	28
	15	25.53	32	20
Region growing [15]		275.13	1507	134
Region growing with merging [19]		368.46	1691	129

Performance of Proposed Methodology: The input scene and point cloud along with the output at different stages are given in Fig. 4. It should be mentioned that the different colors of the mesh are due to separation of ground plane on the basis of normal and is rendered for better visualization only. Performance is evaluated using the precision-recall and f1 score metric. As the segments may lack semantic labels, edge based comparisons were performed with overlapping of dilated edge points with ground-truths. We vary the sampling interval from 5 to 15 in steps of 5, as a sampling interval of 5 corresponds to 1 degree of sweep of the Lidar. The thresholds for checking normal homogeneity are kept at $< \hat{i}, \hat{j}, \hat{k} > = < 0.2, 0.2, 0.2 >$. These values are determined empirically for scenes

Table 2. Comparison of accuracy of different competing methods

Method	Sampling interval	Average F1 score	Average precision	Average recall
Proposed method	5	0.7406	0.7616	0.7224
	10	0.7147	0.7330	0.6998
	15	0.6910	0.7190	0.6673
Region growing [15]		0.3509	0.4752	0.2804
Region growing with merging [19]		0.3614	0.3849	0.3430

with standard objects on a flat surface. Our methodology is compared with [15] and [19]. The tuning parameters of the methods are kept at default settings. Table 1 shows the execution times for different methods and clearly reveals that the proposed method is much faster. In Table 2, we present the accuracy values for the competing approaches. This table indicates that our solution is much more accurate. Overall, the results demonstrate that we are successful in providing a fast yet accurate solution to this complex problem.

4 Conclusion

In this work we have presented an unsupervised surface segmentation algorithm which is fast, accurate and robust to noise, occlusion and different orientations of the surface with respect to the Lidar. This work serves as the first step for mapping environments with geometric primitive modelling in SLAM applications for unmanned ground vehicles. In future, supervised classifier can be utilized for segment formation on data collected by Lidar on a real environment. Thereafter, the surface segments will enable the model generation of 3D objects.

References

1. Bassier, M., Bonduel, M., Van Genechten, B., Vergauwen, M.: Segmentation of large unstructured point clouds using octree-based region growing and conditional random fields. Int. Arch. Photogram. Rem. Sens. Spat. Inf. Sci. **42**(2W8), 25–30 (2017)
2. Ben-Shabat, Y., Avraham, T., Lindenbaum, M., Fischer, A.: Graph based over-segmentation methods for 3d point clouds. Comput. Vis. Image Underst. **174**, 12–23 (2018)
3. Bhanu, B., Lee, S., Ho, C.C., Henderson, T.: Range data processing: representation of surfaces by edges. In: Proceedings of the Eighth International Conference on Pattern Recognition, pp. 236–238. IEEE Computer Society Press (1986)
4. Feng, C., Taguchi, Y., Kamat, V.R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In: 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 6218–6225. IEEE (2014)

5. Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W.: BlenSor: blender sensor simulation toolbox. In: Bebis, G., et al. (eds.) ISVC 2011. LNCS, vol. 6939, pp. 199–208. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24031-7_20
6. Hackel, T., Wegner, J.D., Schindler, K.: Fast semantic segmentation of 3d point clouds with strongly varying density. ISPRS Ann. Photogram. Rem. Sens. Spat. Inf. Sci. **3**(3), 177–184 (2016)
7. Himmelsbach, M., Hundelshausen, F.V., Wuensche, H.J.: Fast segmentation of 3d point clouds for ground vehicles. In: 2010 IEEE Intelligent Vehicles Symposium, pp. 560–565. IEEE (2010)
8. Ioannou, Y., Taati, B., Harrap, R., Greenspan, M.: Difference of normals as a multi-scale operator in unorganized point clouds. In: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pp. 501–508. IEEE (2012)
9. Jiang, X.Y., Meier, U., Bunke, H.: Fast range image segmentation using high-level segmentation primitives. In: Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV 1996, pp. 83–88. IEEE (1996)
10. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4558–4567 (2018)
11. Li, M., Yin, D.: A fast segmentation method of sparse point clouds. In: 2017 29th Chinese Control and Decision Conference (CCDC), pp. 3561–3565. IEEE (2017)
12. Moosmann, F., Pink, O., Stiller, C.: Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion. In: 2009 IEEE Intelligent Vehicles Symposium, pp. 215–220. IEEE (2009)
13. Nguyen, A., Le, B.: 3d point cloud segmentation: a survey. In: 2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM), pp. 225–230. IEEE (2013)
14. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
15. Rusu, R.B., Cousins, S.: 3d is here: point cloud library (pcl). In: 2011 IEEE International Conference on Robotics and Automation, pp. 1–4. IEEE (2011)
16. Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P.: Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In: ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007, vol. 36, pp. 407–412 (2007)
17. Vo, A.V., Truong-Hong, L., Laefer, D.F., Bertolotto, M.: Octree-based region growing for point cloud segmentation. ISPRS J. Photogram. Rem. Sens. **104**, 88–100 (2015)
18. Zermas, D., Izzat, I., Papanikolopoulos, N.: Fast segmentation of 3d point clouds: a paradigm on lidar data for autonomous vehicle applications. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 5067–5073. IEEE (2017)
19. Zhan, Q., Liang, Y., Xiao, Y.: Color-based segmentation of point clouds. Laser Scan. **38**(3), 155–161 (2009)