2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Robust Method for Removing Dynamic Objects from Point Clouds

Shishir Pagad*, Divya Agarwal*, Sathya Narayanan Kasturi Rangan, Hyungjin Kim, Ganesh Yalla

***Abstract*— 3D point cloud maps are an accumulation of laser scans obtained at different positions and times. Since laser scans represent a snapshot of the surrounding at the time of capture, they often contain moving objects which may not be observed at all times. Dynamic objects in point cloud maps decrease the quality of maps and affect localization accuracy, hence it is important to remove the dynamic objects from 3D point cloud maps. In this paper, we present a robust method to remove dynamic objects from 3D point cloud maps. Given a registered set of 3D point clouds, we build an occupancy map in which the voxels represent the occupancy state of the volume of space over an extended time period. After building the occupancy map, we use it as a filter to remove dynamic points in lidar scans before adding the points to the map. Furthermore, we accelerate the process of building occupancy maps using object detection and a novel voxel traversal method. Once the occupancy map is built, dynamic object removal can run in real-time. Our approach works well on wide urban roads with stopped or moving traffic and the occupancy maps get better with the inclusion of more lidar scans from the same scene.**

I. INTRODUCTION

Various autonomous robotic systems rely on maps for precise localization and navigation. Maps serve as a redundant source of information for finding the location of the autonomous robotic systems and also improve the localization accuracy. 3D point cloud maps are one of the common map formats used for this purpose and they represent a snapshot of the static environment around the robotic systems. However, most of the 3D point cloud maps are built from data collected by driving a mobile mapping system on roads filled with dynamic objects like vehicles, pedestrians etc. So, it is important to remove the dynamic objects from maps so that the autonomous robotic systems can use clean point cloud maps for localization.

Furthermore, 3D point cloud maps are not a scalable map representation as they occupy a lot of memory. There have been many popular approaches to model 3D environments [1], [2] or sub-sampled representations [3], [4]. However, full resolution 3D point clouds form a basis for extracting useful information from the map. For example, we can extract static features like road and lane markers from the 3D point cloud maps. But the dynamic object points in point cloud maps are rendered as "ghost" tracks in the point cloud map, often overlapping and occluding the view of road markers, traffic signs and other important static features, making it difficult to extract the static features from road, refer Fig. (1). For this reason, the dynamic object points need to be filtered out from point cloud when building the 3D point cloud maps.

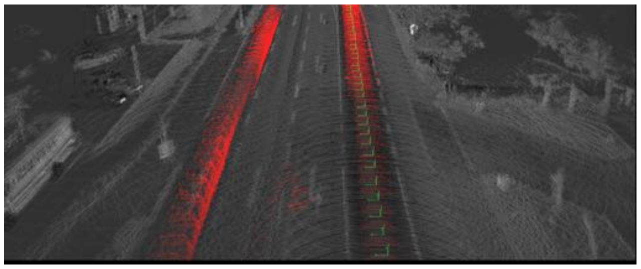

Fig. 1. Dynamic objects, highlighted in red, causing ghost trail effect in a point cloud map

In order to solve the problem discussed above, we borrow valuable ideas from [3] and [5] and build an occupancy map using octree data structure, in which the voxels represents occupancy state of the volume of space for extended period of time. After building the occupancy map, we identify the points which fall in free voxels and remove them from 3D point cloud scans. The contributions of our work are the following:

- We propose a new occupancy probability update strategy which builds persistent occupancy maps by considering the occupancy history of voxels. Unlike our approach, [3] favors quick update of occupancy score of voxels, favoring latest seen occupancy states.
- We provide an optional method to accelerate occupancy map building process by classifying object points using object detection method and a strategy for updating occupancy map using these points.
- Furthermore, we provide a unique way of generating artificial endpoints which are used to update the occupancy score of the voxels.

The input to our algorithm is a set of registered 3D point clouds, typically acquired by 3D laser scanner. First, we classify the points in the point cloud into 3 categories: object points, ground points and unknown points using ground plane detection and object detection algorithms. Next, we perform voxel traversal on the unknown and ground points and decrease occupancy scores of all the voxels which are on the path of the ray from the sensor origin to the endpoints, but increase the occupancy score of the endpoint voxel. Similarly, we perform voxel traversal on the object points, but instead of increasing the occupancy score of the endpoint voxel, we decrease its occupancy score as we already know from object detection that the endpoint falls on a moving object. We also maintain a set of ground voxels which

*Authors have contributed equally. All authors are with Autonomous Driving, Perception Team, NIO USA Inc., San Jose, CA, USA [shishir.pagad, divyaagarwal31, sathya.aeronautics, hjkim0508]@gmail.com, gyalla@us.toyota-itc.com

correspond to the ground points, and prevent them from being marked free during the two voxel traversal steps. We repeat this process for all the point clouds in the registered set and build an occupancy map. Finally, we overlay the fully built occupancy map on a point cloud map and remove points which are in free voxels (refer Fig. 2). It should be noted that the occupancy map grows more stable and accurate with integration of more point clouds over time.

The paper is organized as follows. Section II discusses related work; Section III describes our methodology pipeline and the algorithm; Section IV presents experiments and its analysis; finally, Section V provides concluding thoughts and future work.

II. RELATED WORK

A good amount of work has been done on detecting/removing dynamic objects in laser scans, and different methods have been proposed to solve this problem. These methods can be broadly classified into three categories:

- Model-free, change-detection based approach which relies on comparing current laser scan with previous or a set of previous scans and future scans [6], [7], [8], [9], [10].
- Neural-network model based approach of classifying the points corresponding to dynamic objects [11], and generating bounding boxes around objects in laser scan [12], [13], [14] and classifying the points inside these bounding boxes as dynamic object points.
- Map based approach in which a global occupancy map/voxel grid is built from laser scans using Bayes' rule [3], or by just storing laser scan identifiers in the voxels [5], and the occupancy map is used as a filter to remove points in the free space. The dynamic object removal method proposed in this paper is a hybrid of neural network model based and occupancy map based approaches.

In the model-free, change-detection based approaches, [6] compares query scan against a reference scan and uses either point to point or point to plane error metric to identify the moving objects. The misclassified dynamic points are corrected by comparing them against the free space of another scan and the points not in free space are not dynamic and are changed to static. [7] uses Dempster-Shafer Theory (DST) to extract mobile objects from lidar point cloud and maps them back to images to extract images of moving objects. Current lidar scan is compared against *n* scans before and after the current one. [8] identifies the dynamic points in a dataset by constructing occupancy grid using DST of source and target datasets and finding conflicts between the two occupancy grids. [9], similar to [8], uses DST to find conflicting data between two laser scans. And finally, [10] segments and tracks the moving objects using motion cues and performs point level matching between consecutive scans. The main drawback of model-free approach is that for dynamic objects to be fully detected/removed, it needs to have fully moved outside of the volume it occupies between the two frames being compared. One common scenario where this category of algorithms wont work well is when vehicles have stopped at a traffic signal. Our method is independent of dynamic object speed.

The model based dynamic object removal methods do not require comparison of multiple scans to detect moving objects. The probability scores are generated for individual scans. [15] uses neural network to predict the probability of 3D laser points being reflected by dynamic objects. The computed probabilities are used to build a 3D grid map where each cell represents the probability that a beam is reflected by a static or dynamic object. [11], [12], [13] use neural network to detect static/dynamic objects in laser scan and generate bounding boxes around objects. Once we get the bounding boxes, removing the points which lie inside the bounding boxes is a trivial task. However, model based dynamic object removal methods have a few drawbacks: they can't detect objects which they have not been trained on and they occasionally fail to detect objects.

In map based approaches, both [3] and [5] build occupancy map of the area being mapped. [3] uses octree [16] data structure to store the occupancy information, and each node in the octree stores occupancy probability of the node in the form of log-odds for efficient update. [5] uses voxel grid instead of octree and each voxel in the grid stores identifiers of all the laser rays that end in the voxel. Both [3] and [5] use voxel traversal [17] to update occupancy information of the nodes/voxels, where all the voxels which are in line of sight of the sensor but containing non empty set of laser endpoints are marked as dynamic. The fully built occupancy grid acts as a binary classifier to filter the points from the actual point cloud map. However, the probability update function used in [3] is very sensitive, and it's suitable for mapping free space and allows for quickly adding obstacles to the scene. But, our goal is to map the static objects in scene and to make probability update function less sensitive to dynamic objects.

Our approach is a hybrid of model-based and occupancy map based approaches. We use best of both the approaches to remove dynamic objects more effectively and maintain occupancy maps which represent long-term occupancy states of the area being mapped.

III. METHODOLOGY

In section III-A we describe why we prefer octree data structure, section III-B we describe how we use object detection to speed up the process of inserting point cloud in the map. Section III-D, III-E and III-F discuss use of *free counter* to make occupancy maps favor persistency over easy updatability, and how we prune nodes with *free counter* values. Finally, section III-G describes a unique filtering strategy to improve quality of our occupancy map.

A. Occupancy Octree

Octree is a hierarchical data structure used for storing information about 3D space. Each node in an octree represents a volume of space, called a voxel. We use the log odds form to represent the occupancy information of a space. We also store the number of times a node was measured free as *free*

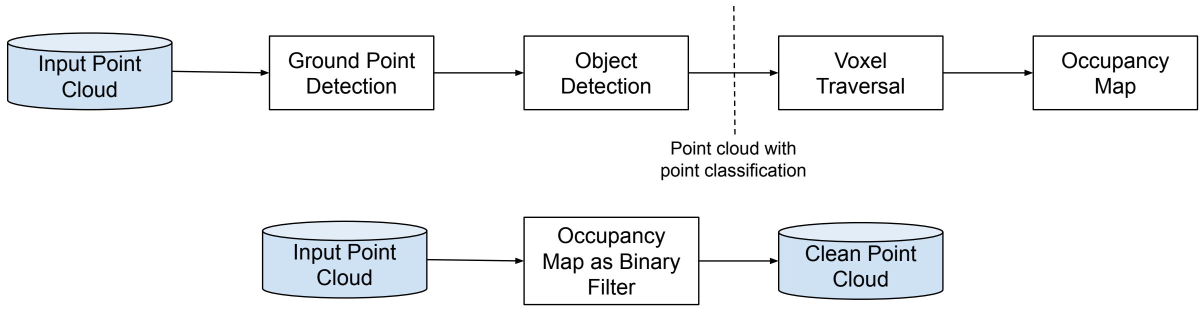

Fig. 2. The overall pipeline of our system. (top) Input Point Cloud goes through ground point detection and object detection. Processed point cloud contains points classified as ground, object and free. Voxel traversal is done and an occupancy map is built. (bottom) Octree occupancy filter is applied to input point cloud map to get clean point cloud map.

counter in each node. We explain more about how we use *free counter* to update the occupancy of voxel in Section III-D.

A 3D space can be represented using several data structures, prominent among them are voxel-grid, k-d tree and octree. We chose octree for occupancy maps because the hierarchical structure of octrees allows for compact and efficient representation of space. In-contrast to voxel grid, we only need to create nodes where occupancy information is measured. In this paper, we are using octrees of fixed maximum depth of 16 and leaf node voxel size of 0.3 meters.

B. Object Detection and Voxel Traversal

Object Detection can be used to accelerate the process of generating occupancy map and improve the precision and recall scores for dynamic object removal. Our approach is not dependent on any specific object detection method, but in our experiments we used AVOD (Aggregate View Object Detection) [13] network to get bounding boxes. Currently, the network is trained to detect small and large vehicles.

However, many of the neural network based object detection methods occasionally fail to detect objects. The models can only detect the objects classes which they have been trained on. Furthermore, the bounding boxes often don't completely encompass the detected objects, shown in Fig. (3). So, the point cloud maps built using the above method will still have some dynamic points.

To remove the dynamic objects which were missed by object detection method, we use the above method to classify the points in a point cloud into two classes: *object points*, points which lie inside bounding boxes of detected objects, and *non-object points*, points outside the bounding boxes. For each of the *non-object points* in the point cloud, we perform voxel traversal to find all the voxels along the laser ray from sensor origin to the endpoint. We decrease the occupancy probability of all these voxels except the endpoint voxel, and increase the occupancy probability of endpoint voxel. Next, we follow the same steps for inserting the *object points*, but instead of increasing the occupancy probability of the endpoint voxel, we decrease its occupancy probability. This is because we already know, from object detection, that the *object points* corresponds to a dynamic object. We explain how *free counter* value of a node/voxel is updated in Section III-D.

As mentioned previously, occasionally, the bounding boxes generated around objects do not completely encompass the object. We partially solve this problem by inserting *object points* after *non-object points* have been inserted into the occupancy map. This ordering ensures that the part of the dynamic object not encompassed by the bounding box will be removed when performing voxel traversal on *object points*.

C. Ground Point Detection

As noted in [3], performing voxel traversal to laser rays sweeping a flat surface at shallow angles lead to undesirable discretization effects. The voxels which have been measured occupied during voxel traversal may be marked free when traversing another nearby voxel. This effect usually happens on flat surfaces like ground and flat walls, and the effect is rendered as holes in the flat surface, shown in Fig. (4). Octomap [3] overcomes this effect by updating a node only once for a given point cloud and by giving preference to occupied nodes over free nodes. However, this method does not work in our case because we insert *object points* after inserting *non-object points*, and we mark the nodes corresponding to endpoints in *object* point cloud, as free. Furthermore, the bounding box generated by object detection may include the ground points under the detected objects and falsely classify the ground points as dynamic object points. As a result of this, some of the ground voxels may still be marked as free.

To solve this problem, we maintain a counter per ground voxel which indicates the number of times the voxel has been classified as ground. We also maintain a set of all the detected ground voxels and update this set for every lidar scan. Only those voxels in the ground voxel set which have the counter value greater than a certain threshold are considered as true ground voxels and are prevented from being marked as free voxels during ray traversal. Thus, the voxels which have been wrongly classified as ground voxels in a few instances will have smaller counter value compared to the true ground voxels which have been consistently detected in most of the lidar scans. We use RANSAC based ground plane detection described in [14], [18].

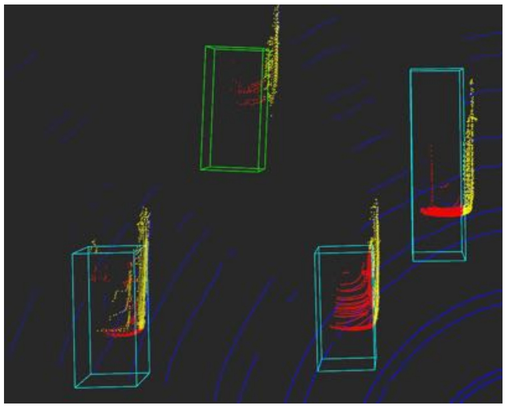
Fig. 3. Dynamic Object points which are outside the bounding box, become part of the point cloud causing false negatives. *Object points* are in red, *non-object points* are in yellow.

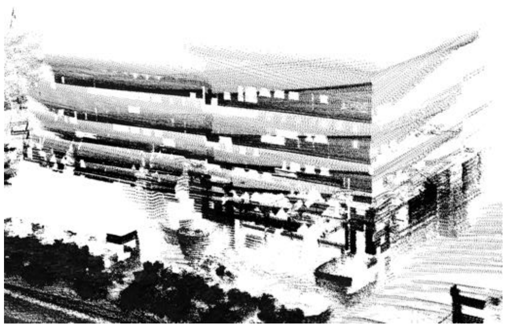
Fig. 4. Holes on flat walls caused by laser rays incident on the surface at shallow angles

D. Weighted Probabilities

Octomap [3] uses clamping policy to allow easy updatability and compressibility of occupancy octree map. The clamping policy ensures that the log-odds value of a node does not fall below the low threshold, l_{min} and does not go beyond the high threshold l_{max}. A node is considered stable when its log-odds value reaches either of the thresholds, and these nodes have been measured free or occupied with high confidence.

The clamping policy ensures that all stable free and occupied nodes have same log-odds values, thus enabling the neighboring nodes with the same log-odds value to be pruned. It also ensures that the occupancy states of the nodes are easily updatable. For example, consider a robot which has mapped a certain area and has measured a few nodes in front of it as free. Now, if a person walks in front of the robot and stands in its path, the robot should be able to quickly update the nodes as occupied.

$$L(posterior) = L(measurements) + L(prior) \quad (1)$$

$$L(posterior) = L(\frac{measurements}{freecounter}) + L(prior) \quad (2)$$

where L represents log odds.

However, the occupancy update policy in [3] favors latest occupancy state of a voxel. In contrast, our goal is to create occupancy map of an area which represents long-term occupancy state of the environment, we want our occupancy update algorithm to be less sensitive to dynamic objects. We achieve this by maintaining *free counter* for each voxel and by using weighted probability. The *free counter* of a voxel is used to count the number of times the voxel has been measured free. It is incremented by one, every time the voxel is measured free during voxel traversal, and decremented by one if its value is greater than one and the voxel is measured occupied during voxel traversal.

To understand how this works, consider two scenarios. First scenario where a voxel has a *free counter* value greater than one, and the voxel has been measured occupied when inserting the current point cloud into occupancy map. Since the *free counter* value is greater than one, it indicates that the voxel was measured free previously, so there is a high possibility that this voxel may have been occupied by a dynamic object in the current point cloud. So, we lighten the probability update by dividing the probability of hit value of the voxel with *free counter* value as seen in equation (2). The intent behind this is to make it hard to increase the occupancy probability of a voxel which has been measured free previously. Higher the *free counter* value of a voxel, harder it is to increase the occupancy probability of the voxel. Second scenario is when the voxel has been wrongly marked as occupied, due to wrong detection by object detection method. In that case, we use the original equation (1). This allows for easy updatability of the voxels belonging to dynamic points.

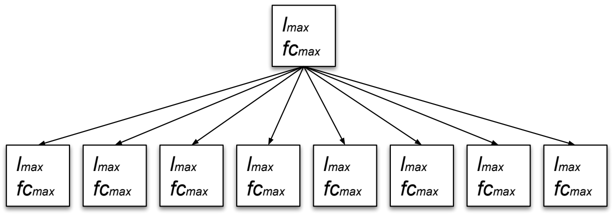

Fig. 5. We use the log odds occupancy probability and *free counter* value for pruning and expanding the nodes.

E. Pruning and Expanding Nodes

The hierarchical structure of octree enables pruning of nodes for efficient representation of space. If all the children of an inner node have same occupancy probability and *free counter* value, then children can be pruned, and their occupancy probability and *free counter* value are stored in the parent node (refer Fig. 5). We also add clamping of the occupancy probability in the same way as done in [3] and clamp the *free counter* value to a maximum value of fc_{max} and a minimum value of 1. After adding sufficient number of point clouds to the occupancy map, the nodes corresponding to the static area will converge to same max occupancy probability score and max *free counter* value of fc_{max}.

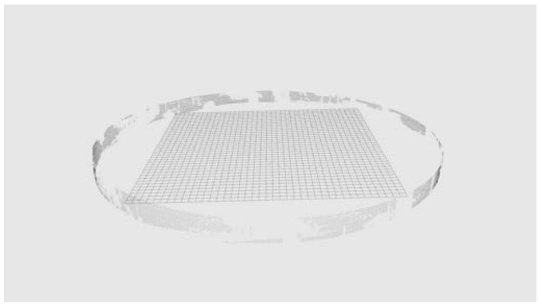

Fig. 6. Virtual sphere used to generate artificial endpoints.

F. Spherical Projection

The endpoints far from laser sensor are often noisy, so when mapping an outdoor space, it's necessary to trim the point cloud to a certain distance from the sensor to avoid noisy data. However, by removing the endpoints which fall beyond a certain range from the laser sensor, we lose valuable information about free space. Even if these endpoints are noisy, we know the area between the laser and these endpoints is free. We use this information to refine our object detection algorithm. The endpoints which lie beyond a radius r from the sensor are projected back onto a virtual sphere of radius *r*, centered at sensor origin, shown in Fig. (6). Then, we perform voxel traversal for each endpoint projected on the virtual sphere and decrease occupancy probability of all the voxels along the rays from sensor origin to the endpoints, including the voxel corresponding to the endpoint. This allows us to artificially generate endpoints, for better occupancy updates.

To project the points onto virtual sphere, we transform the points from cartesian coordinate space to spherical coordinate space (r, θ and ϕ). Keeping θ and ϕ constant, we transform the points back to cartesian space by replacing radius of the points with radius of the sphere r_{sphere}.

G. Occupancy Octree Map as a Binary Filter

The stability and accuracy of occupancy octree map increases with integration of data from multiple data collection drives for the same region. With the assumption that the point cloud registration works reasonably well, the static parts of the map from different data collection drives should be mapped to the same set of voxels in the occupancy octree map, thus increasing the occupancy score and stabilizing these voxels. On the contrary, the voxels corresponding to the area of the occupancy octree map traversed by dynamic objects would see fluctuations in occupancy values. With our approach, only the voxels which are repeatedly measured as occupied are marked as occupied, and once a voxel is measured as free then it makes it hard to mark it as occupied.

After the occupancy octree has stabilized, it can be superimposed on the point cloud map and used as a binary filter to remove all the points which fall in the free space of the octree map.

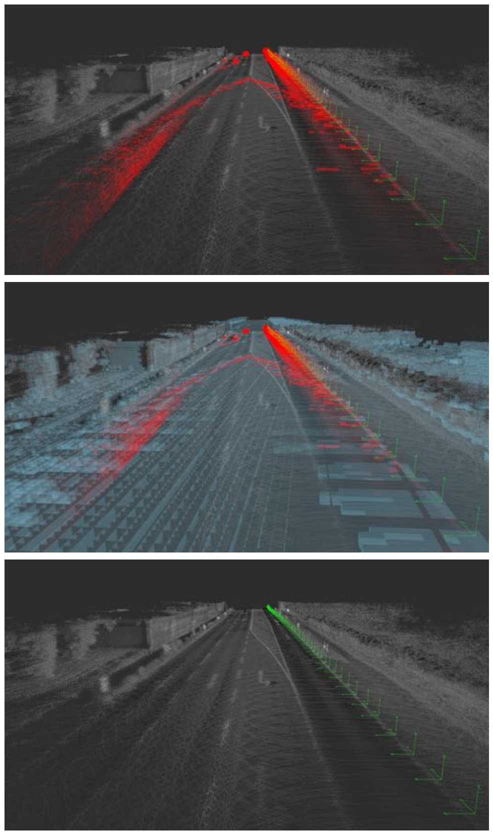

Fig. 7. The pipeline above shows from top to bottom a) input lidar scan with ghosting effect b) octomap generated from the input scan c) output clean lidar scan, generated on KITTI dataset.

IV. Experiments And Results

The goal of our experiments is to show our approach of dynamic object points removal is superior compared to removing the dynamic object points using just object detection. We test our approach on real world data collected from busy streets. We also benchmark our algorithm with KITTI dataset.

A. Dataset

Our approach of removing dynamic object points from point cloud relies on building occupancy maps. It is hard to estimate the time invariant occupancy state of an environment with a single set of scans of the area as scans may contain moving objects. Therefore we need multiple sets of scans collected from the same area, preferably at different times to identify temporarily static objects like parked cars which otherwise get marked as occupied. To the best of our knowledge, the KITTI dataset has only a few loops of driving through same parts of the map, mostly for loop closure. So, we also build our own dataset using the setup described below.

Our setup consists of Velodyne VLP-32C Lidar mounted on car roof and Novatel RTK GPS. Both sensors are synchronized. The point cloud obtained is motion compensated using GPS data. We collected four sets of data, each consisting of a single loop around the urban roads near NIO office. We use NDT [19] to match lidar scans.

B. Evaluation

We assess the performance of our approach statistically by using precision and recall. Precision represents the percentage of removed dynamic object points which actually corresponds to dynamic objects. Recall represents the percentage of total dynamic object points that were actually removed.

Precision scores are negatively affected by false positives i.e, classifying static object points as dynamic object points. Laser rays incident on flat surfaces at shallow angles are one of the main cause of decrease in precision scores. This effect is very clearly explained in [3]. Furthermore, false positives in object detection can also decrease precision score. To solve the first problem, we use ground plane detection method explained in section III. This strategy helps minimize false positives but not eliminate them.

C. Results

To evaluate our algorithm accuracy, we compare our proposed method against KITTI dataset and [9]. Table I is an overview of how our algorithm did on various KITTI sequences. We used KITTI sequences that have wide roads and moving cars, to better demonstrate our algorithm capabilities. We have ground truth for moving objects for KITTI to calculate P&R values. Table I shows better recall than precision values. This is because we do not run our object detection pipeline on KITTI dataset and we only have one loop of the KITTI sequences. The values will significantly improve if we have more loops of the same sequence.

We also ran it on sequences mentioned in [9] as shown in Table II, but we notice bad P&R values, as those sequences have narrow roads surrounded by flat building walls. This causes false positives leading to bad precision values.

Table III shows, proposed method's performance on our dataset. We show differences in P&R values with and without object detection. We also show that spherical projection helps reduce false positives and false negatives. We can see the P&R values are much better compared to Table I, due to a number of reasons. Firstly, we use object detection to remove dynamic objects. Secondly, we process the point cloud in a particular order explained in Section III. Thirdly, we have multiple loops of the data as shown in the table III, which improves precision values. We do not perform manual labelling of the dataset, hence we do not have ground truth. We assume the output of the object detection method as our absolute ground truth and calculate the P&R scores. And we can see our method does significantly better, even with one loop of driving data. The overall pipeline shown in Fig. (7).

Since the method used by [5] uses very dense point clouds generated from terrestrial scanner, we cannot directly compare our results. It also requires expensive surface normal computation, while our method relies on weighted probability described in Section III-D.

Another observation from our dataset, as we collect more data our method gives almost similar P&R numbers, with and without Object Detection, seen for loop4 in Table III. This is why we mentioned object detection as an optional algorithm for using our method.

Our pipeline is robust to scenarios with stop-and-go traffic as opposed to [6]. And performs good with consistent motion also. The pipeline struggles with narrow roads surrounded by tall flat buildings, due to discretization in the flat walls. The results will be better if object detection and ground detection are perfect.

V. CONCLUSIONS

In this paper, we propose a novel and robust method to remove dynamic objects from point cloud maps. We use the occupancy octree map to create clean point cloud. There are limitations with the implementation of occupancy octree in Octomap, it is not scalable for large scale outdoor maps. The area covered by the occupancy octree map is limited by the maximum depth and size of the voxels at maximum depth. In our case with the maximum octree depth of 16 and voxel size of 0.3 meters, it can only cover an area of $(2^{16} * 0.3)^3 m$, which is 7599.82 cubic km of volume. This limitation can be overcome by tiling, which is next topic of our research. We rely on multiple loops of data in scenarios with heavy traffic due to lidar occlusions. Our method is robust and can be considered for building long term good quality 3D point cloud maps.

ACKNOWLEDGMENT

This work was supported and developed at NIO USA Inc. We would like to thank NIO USA Inc. for sponsoring this R&D work.

TABLE I
KITTI DATA CATEGORY: WIDE ROADS / HIGHWAYS

	number of scans	w/o Obj Detection	
		P	R
seq 004	345	0.738	0.570
seq 015	303	0.492	0.579
seq 016	285	0.495	0.738
seq 042	1176	0.460	0.827

TABLE II
KITTI DATA CATEGORY: NARROW CITY ROADS

	number of scans	Proposed Method		Postica. et al [9]	
		P	R	P	R
seq 091	346	0.14	0.51	0.25	0.75
seq 095	274	0.21	0.53	0.19	0.79
seq 104	318	0.11	0.49	0.44	0.87

TABLE III
PRECISION AND RECALL RESULTS ON OUR DATASET
PROBABILITY OCCUPANCY THRESHOLD = 0.8

	w/o Obj Detection		w. Obj Detection		w/o Virtual Sphere	
	P	R	P	R	P	R
loop1	0.163	0.939	0.161	0.917	0.177	0.659
loop2	0.191	0.890	0.192	0.874	0.229	0.790
loop3	0.215	0.878	0.214	0.870	0.268	0.793
loop4	0.226	0.868	0.225	0.860	0.307	0.811

REFERENCES

[1] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2006, pp. 2276–2282.

[2] I.-S. Kweon, M. Hebert, E. Krotkov, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *IEEE International Conference on Robotics and Automation*. IEEE, 1989, pp. 997–1002.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[4] Y. Roth-Tabak and R. Jain, "Building an environment model using depth information," *Computer*, vol. 22, no. 6, pp. 85–90, 1989.

[5] J. Schauer and A. Nüchter, "The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid," *IEEE robotics and automation letters*, vol. 3, no. 3, pp. 1679–1686, 2018.

[6] D. Yoon, T. Tang, and T. Barfoot, "Mapless online detection of dynamic objects in 3d lidar," in *2019 16th Conference on Computer and Robot Vision (CRV)*. IEEE, 2019, pp. 113–120.

[7] B. Vallet, W. Xiao, and M. Brédif, "Extracting mobile objects in images using a velodyne lidar point cloud," *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, vol. 2, no. 3, p. 247, 2015.

[8] W. Xiao, B. Vallet, and N. Paparoditis, "Change detection in 3d point clouds acquired by a mobile mapping system," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 1, no. 2, pp. 331–336, 2013.

[9] G. Postica, A. Romanoni, and M. Matteucci, "Robust moving objects detection in lidar data exploiting visual cues," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1093–1098.

[10] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3d lidar scans," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4508–4513.

[11] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4490–4499.

[12] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 770–779.

[13] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[14] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[15] P. Ruchti and W. Burgard, "Mapping with dynamic-object probabilities calculated from single 3d range scans," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6331–6336.

[16] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[17] J. Amanatides, A. Woo *et al.*, "A fast voxel traversal algorithm for ray tracing," in *Eurographics*, vol. 87, no. 3, 1987, pp. 3–10.

[18] M. Y. Yang and W. Förstner, "Plane detection in point cloud data," in *Proceedings of the 2nd int conf on machine control guidance, Bonn*, vol. 1, 2010, pp. 95–104.

[19] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, vol. 3. IEEE, 2003, pp. 2743–2748.