2022 IEEE International Conference on Robotics and Automation (ICRA)
May 23-27, 2022. Philadelphia, PA, USA

DynamicFilter: an Online Dynamic Objects Removal Framework for Highly Dynamic Environments

Tingxiang Fan*, Bowen Shen*, Hua Chen, Wei Zhang† and Jia Pan†

***Abstract*— Emergence of massive dynamic objects will diversify spatial structures when robots navigate in urban environments. Therefore, the online removal of dynamic objects is critical. In this paper, we introduce a novel online removal framework for highly dynamic urban environments. The framework consists of the scan-to-map front-end and the map-to-map back-end modules. Both the front- and back-ends deeply integrate the visibility-based approach and map-based approach. The experiments validate the framework in highly dynamic simulation scenarios and real-world dataset.**

I. Introduction

Long-range navigation in urban environments is a challenging problem for modern autonomous robotic systems. The High-Definition Map (HD Map) that can provide a highly accurate representation of the urban environment has been widely used for scene understanding and planning. However, constructing and maintaining the HD Map is time-consuming and laborious as it needs to operate a large number of vehicles for data collection to continuously update the map. By online understanding the surrounding scenarios, humans can navigate without the prior assistance of HD Map. Therefore, online scene understanding is a promising direction to eliminate robots' dependence on HD Maps.

By leveraging 3D LiDAR sensors, robots can perform simultaneous localization and mapping (SLAM) to online reconstruct the spatial structure of surrounding environments [1], [2]. However, most existing SLAM techniques assume the perceived objects are static and time-invariant. When a robot navigates in urban environments, the emergence of massive dynamic objects will diversify the spatial structure. Therefore, the online removal of dynamic objects is critical.

In this paper, we focus on the detection and removal of the dynamic points perceived by the 3D LiDAR. It is challenging to detect dynamic points from a single scan. Although learning-based approaches can instantly predict the dynamic segments from a single scan [3], [4], they rely on numerous labeled datasets and may fail when encountering unlabeled classes. By considering the inconsistency between multiple scans, the occupancy map-based and visibility-based approaches can estimate the dynamic points without any semantics labels. However, both approaches have their own shortcomings. The occupancy map-based method [5]–[7], leveraging the ray-tracing, can estimate the occupancy probability in the grid space. Although it has been widely used in 2D mapping, the computational cost of real-time ray-tracing for 3D points is unaffordable. To alleviate the computational burden, the visibility-based method [8]–[10] re-projects the range measurements of the 3D point cloud to the image plane according to the specific field of view (FOV) and resolution. The visibility difference between multiple scans at each pixel is considered as dynamic points. However, it suffers from *incidence angle ambiguity* [8], [11] and *occlusion* [12] issues. These *visibility issues* will decrease its performance.

Fig. 1: *Top*: Collecting data with dense crowds in the mall. *Medium*: The floor plan of the mall. *Bottom*: The point cloud map built by our approach. The red points are the dynamic pedestrians to be removed. The white points are the spatial structure of the environment.

In this paper, we propose an online dynamic object removal framework that integrates both the map-based and the visibility-based methods. Inspired by the modern SLAM paradigm [13], our framework is divided into the scan-to-map front-end and the map-to-map back-end. The front-

* denotes equal contribution, † denotes corresponding author.

T. Fan, J. Pan are with the University of Hong Kong. B. Shen, H. Chen, W. Zhang are with the Southern University of Science and Technology.

This work is supported in part by HKSAR Research Grants Council (RGC) General Research Fund (GRF) HKU 11202119, 11207818, the Innovation and Technology Commission of the HKSAR Government under the InnoHK initiative, the National Natural Science Foundation of China under Grants 62073159 and 62003155, the Shenzhen Science and Technology Program under Grant JCYJ20200109141601708, and the Science, Technology and Innovation Commission of Shenzhen Municipality under grant ZDSYS20200811143601004

2022 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9681-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICRA46639.2022.9812356

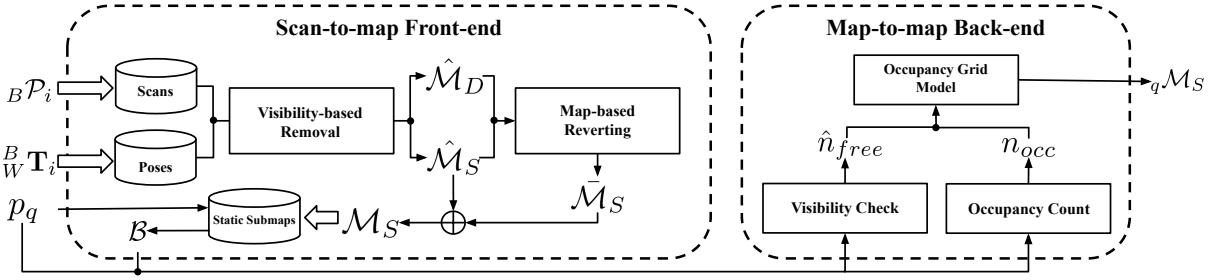


Fig. 2: Our pipeline is composed of front-end and back-end executed in parallel, with the visibility-based and map-based methods integrated.

end is time-critical that needs to instantly compute coarse results, while the back-end could be computation-intensive to provide further optimization results with historical data. Specifically, our framework takes consecutive scans and their poses as input. First, the visibility-based approach is used for dynamic object removal given its superior computational efficiency. After the removal, the raw static submap $\hat{\mathcal{M}}_S$ can be obtained. Note that $\hat{\mathcal{M}}_S$ preserves sparse static points due to the *visibility issues* mentioned above. Then, we propose the map-based reverting approach to recover denser static points $\bar{\mathcal{M}}_S$ from the sparse features. Due to the limited horizon of consecutive frames, the static submap $\mathcal{M}_S$ generated by the front-end may not be satisfactory. Finally, the map-to-map back-end is introduced to further optimize the result. The core idea of the back-end is to compute the occupancy probability in each voxel from multiple submaps. The voxel is marked as free when a ray passes through the voxel, where we use *visibility check* to approximately calculate $\hat{n}_{\text{free}}$, the number of rays passing through each voxel, for accelerating the ray-tracing process. The entire pipeline is demonstrated in Figure 2 and the main contributions are summarized as follows:

- We propose the online dynamic object removal framework that fuses the visibility-based and map-based approaches. To our best knowledge, it is the first work proposing the front-end and back-end structures for dynamic object removal.
- We present the *visibility check* that uses the visibility-based approach to approximate the ray-tracing process and accelerate the occupancy computation.
- We propose the map-based reverting algorithm in the front-end which can mitigate the *visibility issues* caused by the visibility-based removal.
- We quantitatively evaluate the algorithm's performance in complex simulation scenarios and qualitatively demonstrate the online performance in a diverse real-world scenes that are crowded.

II. Related Work

In this section, we review the related work about dynamic object removal from 3D point clouds, which can be roughly divided into the following three categories:

Model-dependent approaches. Dynamic points can be detected based on the prior model. Some methods rely on the ground plane model to separate dynamic objects from the ground. [14], [15] require a pre-processing step of ground removal. [12] is based on the premise that all dynamic objects will be in contact with the ground, so all points except the ground can be filtered out where dynamic objects may exist. Therefore, these methods' performance may degrade in situations where such assumption does not hold.

More generally, with recent advances in deep learning, some dynamic objects such as pedestrians and vehicles can be detected according to their appearance models [3], [4]. However, the detected results only indicate that the objects are potentially dynamic but do not mean that the objects actually are moving. For example, for a car parked in a parking lot, the robot should understand this object as a static obstacle instead of a dynamic object. [16] takes the sequential range images as input to capture the dynamic points based on the inconsistency between these images. However, these learning-based approaches rely on manually labeled high-quality datasets.

Occupancy map-based approaches. By tracking the emitted ray from LiDAR, the robot can consider the ray end point as being occupied and the voxels that the ray traverses as free. In this way, the occupancy probability in the surrounding environment can be computed. However, ray-tracing for 3D LiDAR is computationally expensive. Even with some engineering optimization [5], it is still challenging to process massive 3D data online, though it is indeed an effective method to filter out dynamic points [6]. [7] introduces the occupancy map-based method for offline labeling of dynamic objects and the labeling results can be used as the ground truth for the training step of learning-based approaches.

Visibility-based approaches. Compared to the occupancy map-based methods, the visibility-based approaches only compute the visibility difference [8]–[10]. Specifically, if the observed point is occluded in the line of sight of the previously observed point, this point should be regarded as dynamic. It dramatically reduces the computational cost compared to ray-tracing, though such simplification is also accompanied with performance loss. In the situation of a large incident angle [8], [11], or when occlusion occurs [10], [12], the visibility-based method may not be able to filter out the dynamic points correctly (Fig. 2 in [12]). [10] proposes the *removing-and-reverting* mechanism by iteratively retaining the static points from falsely removed points. However, the reverting process is still based on the visibility process and thus its performance is still limited due to the *visibility issues*. Moreover, [8], [10] only implement their approaches under the offline paradigm.

In this paper, we aim to develop a model-independent

approach that does not utilize any existing model. By fusing occupancy map-based and visibility-based methods, our proposed method is dedicated to the online removal of dynamic points with high efficiency and high accuracy.

III. Methodology

The online dynamic object removal framework consists of two parts: the scan-to-map front-end and the map-to-map back-end, as shown in Figure 2.

A. Problem setup

We focus on the removal of dynamic points from the consecutive 3D LiDAR scans. The point cloud ${}_B\mathcal{P}_i$ of each scan locates in the local sensor frame B. By using state-of-the-art LiDAR-based SLAM approaches such as [1], [2], we can compute the transition ${}^B_W\mathbf{T}_i$ between local sensor frame B and the global frame W.

B. Scan-to-Map front-end

The front-end takes n temporally consecutive frames as input, then transforms these points into a global coordinate, and finally constructs an unprocessed submap $\{{}_W\mathcal{M} \mid \sum_i^n {}^B_W\mathbf{T}_i \times {}_B\mathcal{P}_i\}$.

1) Visibility-based removal: The basic removal process is proposed by [8], [10], which can be briefly summarized as follows. First, we project each frame of point cloud ${}_B\mathcal{P}_i$ and the unprocessed submap ${}_W\mathcal{M}$ to the image plane in the local coordinate system. Then, their range images ${}_BI^{\mathcal{P}}$ and ${}_BI^{\mathcal{M}}$ can be computed respectively. By comparing the difference between these two range images, we can detect the dynamic points as the submap $\hat{\mathcal{M}}_{D_i}$. After processing all frames in the submap, we can roughly divide the processed points into two classes: the static submap $\hat{\mathcal{M}}_S$ that only includes static points, and the dynamic submap $\hat{\mathcal{M}}_D$ that only includes the dynamic points.

2) Map-based reverting: However, the performance of the visibility-based removal will degrade in cases with large incident angles and occlusions. The manifestation of such degradation is that many static points are falsely filtered out as dynamic points. Inspired by [10], our scan-to-map front-end also adopts the reverting phase to recover falsely detected static points. [10] repeatedly executes visibility-based removal but takes the dynamic submap $\hat{\mathcal{M}}_D$ as input, thereby continuously recovering the static points from the dynamic submap $\hat{\mathcal{M}}_D$. However, this reverting method may still suffer from the *visibility issues*.

Unlike [10], we introduce the map-based reverting to alleviate the *visibility issues*. The insight behind is that the static submap $\hat{\mathcal{M}}_S$ is sparse and conservative, which strictly preserves the static points. In terms of the static points that were falsely removed, some work [10], [12] indicate that they often occur on the ground (caused by large incident angle) or at narrow objects, and boundary of objects (caused by occlusion).

Therefore, we revert static points based on the set distance between the removed dynamic points and the static submap in the global coordinate. In particular, for each dynamic point in the dynamic map, we first search for its nearest points in the static map. Then, we employ the principal component analysis (PCA) [17] to compute the eigen vector of the set of nearest points. Similarly, another eigen vector can also be computed for the union of this set of nearest points and the dynamic point itself. Finally, by comparing the normal distance between these two eigen vectors, we can get the distribution change after adding the dynamic point. If the change is small enough, we revert the dynamic point $\hat{\mathcal{M}}_D$ as static $\bar{\mathcal{M}}_S$. The qualitative result is visualized in Figure 3. Note that the reverting process does not rely on any model/plane assumption and thus is model-independent.

After map-based reverting, we obtain a static submap $\mathcal{M}_S$ and add it in the submap buffer along with its relative pose to the global frame averaged over the entire sequence.

C. Map-to-map back-end

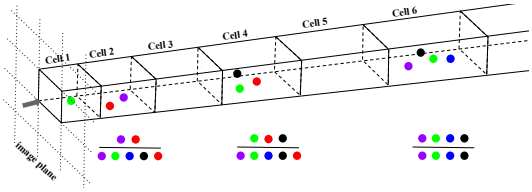
The front-end has the potential to remove all dynamic points in the long sequence [8], [10] under the offline paradigm. However, given the online nature of our work, only short sequences are available, which will limit the front-end performance and thus the front-end may be only effective for some fast-moving objects. Therefore, we propose the map-to-map back-end to continuously improve the removal performance.

1) Occupancy grid model: Given a query pose p_q, we can search for the nearest submaps in the submap buffer within a specific radius. The core idea is to combine multiple nearest static submaps to estimate the occupancy grid map in the environment using the occupancy grid model, which is simple but effective. Beforehand, we discretize the nearby space, i.e., finishing the voxelization step. We then count the number of times each voxel in the space is occupied (i.e., n_{occ}), and the number of times each voxel is passed through (i.e., n_{free}). The occupancy probability is then computed as $\frac{n_{\text{occ}}}{n_{\text{occ}}+n_{\text{free}}}$. n_{occ} can be directly counted given the point cloud, but n_{free} requires the ray-tracing computation. However, the online ray-tracing for each submap is intractable. Thus, we introduce the *visibility check* to approximately estimate the denominator of the occupancy probability (i.e. $n_{\text{check}} = (n_{\text{occ}} + \hat{n}_{\text{free}})$).

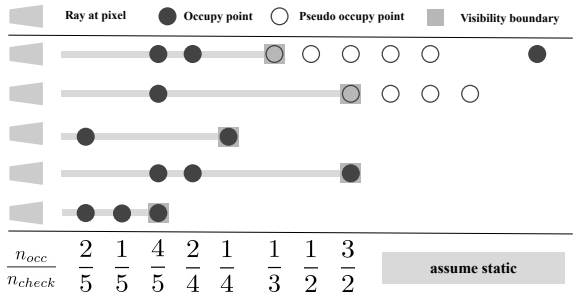
2) Visibility check: To approximate the ray-tracing process, we first discretize the range dimension to multiple depth channels. Specifically, the origin image size (Width $\times$ Height $\times$ 1) is discretized to (Width $\times$ Height $\times$ Depth). As shown in Figure 4a, the depth channel from one pixel is divided into different cells. The farther the cell is, the more voxels will it contain. Then, we can reproject each submap to the image plane at the query pose p_q according to the specific field of view (FOV) and resolution. Next, for each pixel, we can compute the depth that the point in the pixel can reach. The *visibility boundary* of the pixel is defined by the maximum depth. The cell before the *visibility boundary* is approximated to be passed through by ray-tracing.

However, such approximation is still affected by the large angle ambiguity and occlusion (i.e., *visibility issues*). Specifically, since a submap is composed of multiple frames and the occlusion is inevitable, the submap will occupy different

Fig. 3: The qualitative comparison between visibility-based reverting [10] and our map-based reverting. *(a)* is the bird's eye view of the current scene. By stacking the consecutive scans, we can acquire the raw local submap (i.e. *(b)*) in this scene. The point cloud is colored according to its height. After the visibility-based removal, the detected dynamic points $\hat{\mathcal{M}}_D$ are visualized in *(c)*. With visibility-based reverting, some falsely detected dynamic points are reverted to the static map as shown in *(d)*. We present the result of our proposed map-based reverting in *(e)*, which dramatically recovers the falsely detected dynamic points without losing dynamic features.



(a) Different colors represent points from different submaps and the grey arrow denotes the direction of the line of sight. Four submaps can reach the farthest cell (i.e., cell 6) in the line of sight, so the occupied n_{occ} is 4 and the estimated free number$\hat{n}_{\text{free}}$ is 0. Its occupancy probability is $\frac{4}{4}$. The cells before cell 6 tend to be free in cases without occlusion. Therefore, for the cell 4 whose n_{occ} is 3, it may be passed through by five submaps, and the occupancy probability is $\frac{3}{5}$. Similarly, the occupancy probability of cell 2 is $\frac{2}{5}$.



(b) One example of computing the occupancy probability. Different rows represent the rays for different submaps at a same pixel. After distinguishing the occupied point and pseudo occupied point, the *visibility boundary* can be determined. We count the number of times each cell falling in a *visibility boundary* as the n_{check}. Then, the occupancy probability can be computed.

Fig. 4: The ray-tracing approximation by *visibility check*.

depths in the same pixel. Similarly, considering the situation with the large incident angle, the same surface may be detected with different depths. These issues will lead to a situation that multiple different depths are occupied in each pixel, even though it is not caused by dynamic points.

Since the point with a large incident angle can not well approximate the maximum depth range in the submap, we propose the *incident correction*. By computing the incident angle of each point, we can label those points whose incident angle is greater than a threshold as pseudo-occupied points. We only consider the closest pseudo-occupied point as the *visibility boundary*. Any further points beyond the pseudo-occupied point are not taken into account since the approximation of the ray-tracing may not be accurate. It may preserve more static points but miss some dynamic points out of the boundary. In terms of other common occupied points, we combine the precomputed *visibility boundary*

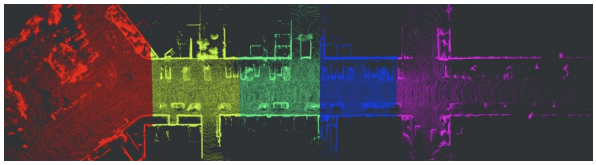


Fig. 5: The result of the submaps merging in SemanticKITTI. Different colors denote the submap that a voxel belongs to.

and compute the farthest *visibility boundary* in each pixel. Finally, the visibility boundaries of all submaps can be obtained. The *visibility check* process has been demonstrated in Algorithm 1. According to the visibility boundaries, the occupancy probability can be approximately computed as shown in Figure 4b.

After computing the occupancy probability, we can obtain the final static local map ${}_q\mathcal{M}_S$ at query poses p_q.

D. Submaps merging

However, a single static submap is only sufficient for local scene understanding. Hence, we need to merge multiple static submaps to obtain the global spatial structure.

The *visibility check* has a better ray-tracing approximation for the points near the submap pose. Therefore, in the merging process, a voxel is marked occupied when it is marked as occupied by the nearest submap, otherwise as free. We visualize the merged result in Figure 5

IV. Experiments and Results

A. Experimental setups

To measure the quality of the preserved static map after removing dynamic points, we use the *preservation rate* (PR) and *rejection rate* (RR) , proposed by [12], as our evaluation metrics. Compared to the precision-recall model used in statistics, it is less sensitive to the size of voxelization.

The metrics are voxel-wise and 0.2 voxel size is used as the resolution for evaluation. Specifically, the metrics are defined as follows:

- PR: $\frac{\#\text{ of preserved static points by the static map}}{\#\text{ of total static points on the raw map}}$
- RR: 1 - $\frac{\#\text{ of preserved dynamic points by the static map}}{\#\text{ of total dynamic points on the raw map}}$

We also compute the F_1 score which is the harmonic mean of the precision and recall.

We choose the SemanticKITTI dataset [18], [19] as our benchmark, which provides the manual label of moving objects and is collected by a vehicle in the urban environment. Although KITTI has been widely used to evaluate SLAM and

Algorithm 1 Visibility check

Input: nearest static submaps buffer $\mathcal{B}$, query pose $\mathbf{p}_q$, range depth $depth$, image resolution res, image FOV fov, and incident angle threshold λ_{thres}.
$\mathcal{D}_I = \{\}$
for $\mathcal{M}_S$ **in** $\mathcal{B}$ **do**
  // Transform submap using the query pose $\mathbf{p}_q$
  ${}_q\mathcal{M}_S$ = transformMap($\mathcal{M}_S$, p_q)
  // Compute normal vectors for each submap
  ${}_q\mathcal{N}$ = computeNormal(${}_q\mathcal{M}_S$)
  // Initialize image with (inf)
  I_{bound} = initializeImage(res, fov, inf)
  $I_{\max}$ = initializeImage(res, fov, 0)
  for $i = 1, ...,$ ${}_q\mathcal{M}_S^i$.size() **do**
    row, col = reprojectToImage(${}_q\mathcal{M}_S[i]$)
    r = computeRange(${}_q\mathcal{M}_S[i]$)
    λ = computeIncidentAngle(${}_q\mathcal{N}[i]$)
    if $\lambda > \lambda_{\text{thres}}$ **then**
      if $r < I_{\text{bound}}[row, col]$ **then**
        $I_{\text{bound}}[row, col] = r$
        $I_{\max}[row, col] = r$
      end if
    end if
  end for
  for $i = 1, ...,$ ${}_q\mathcal{M}_S^i$.size() **do**
    row, col = reprojectToImage(${}_q\mathcal{M}_S[i]$)
    r = computeRange(${}_q\mathcal{M}_S[i]$)
    $r_{\max} = I_{\max}[row, col]$
    if $r > r_{\max}$ **then**
      // set maximum depth
      $I_{\max}[row, col]$ = round(r)
    end if
  end for
  $\mathcal{D}_I$.append($I_{\max}$)
end for
Output: the visibility bounds of all submaps $\mathcal{D}_I$.

perception algorithms, the dynamic objects in the dataset do not appear frequently. Hence, the state-of-the-art approaches such as [12] can achieve more than 90% of PR and RR on the SemanticKITTI dataset. Thus the SemanticKITTI dataset cannot accurately evaluate different algorithms' performance in dynamic object removal.

To better evaluate the performance of dynamic object removal, we introduce a simulation environment as demonstrated in Figure 6 with more dynamic objects, where the Gazebo simulator [20] simulates a mobile robot with a 3D LiDAR and the Menge [21] is implemented with the Gazebo environment to simulate the movement of crowded pedestrians. Three scenes with different number of pedestrians (50, 100, 150) are designed. The robot will run two rounds to collect data in each scene. Among the collected data in three scenes, dynamic points will take 54.8%, 58.5% and 59.3% of the total points, respectively.

We compare our approach with the state-of-the-art visibility-based approach [10] and the model-dependent approach [12]. The results show that our approach achieves the state-of-the-art performance on SemanticKITTI over existing approaches [5], [6]. Note that both approaches are implemented offline.

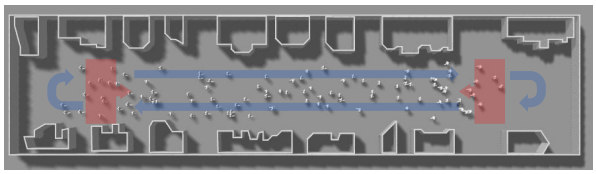


Fig. 6: Highly dynamic simulation scenario with $70\,\text{m} \times 10\,\text{m}$. The red arrow is the moving flow of pedestrians. The blue arrow is the robot trajectory when collecting data.

TABLE I: Comparison with state-of-the-art methods on the SemanticKITTI dataset.

Sequence number	Method	PR[%]	RR[%]	F_1 score
00	Removert [10]	86.83	90.62	0.887
	ERASOR [12]	**93.98**	**97.08**	**0.955**
	Ours	90.07	91.09	0.906
01	Removert [10]	**95.82**	57.08	0.715
	ERASOR [12]	91.49	**95.38**	**0.934**
	Ours	87.95	87.69	0.878
02	Removert [10]	83.29	88.37	0.858
	ERASOR [12]	87.73	**97.01**	**0.921**
	Ours	**88.02**	86.10	0.871
05	Removert [10]	88.17	79.98	0.839
	ERASOR [12]	88.73	**98.26**	**0.921**
	Ours	**90.17**	84.65	0.873
07	Removert [10]	82.04	95.50	0.883
	ERASOR [12]	**90.62**	**99.27**	**0.948**
	Ours	87.94	86.80	0.874

B. Quantitative evaluation

Referring to the setup of [12], we only conduct quantitative experiments on the sequences with dynamic objects. As shown in Table I, our method achieves comparable performance to that of the state-of-the-art visibility-based approach in semanticKITTI. Note that both approaches [10], [12] are offline in which a pre-built map is needed. In addition, the dynamic points account for less than 1% of the total number of point clouds in semanticKITTI, and thus it cannot fully test the performance of dynamic object removal.

As shown in Table II, the performance of baseline approaches dramatically degrade in the highly dynamic scenarios that we built in simulation. [12]'s model-dependent approach, relying on the ground fitting to remove dynamic points above the ground, fails in these highly dynamic scenarios where it is not trivial to determine the ground plane. The performance of [10] dramatically decreases in these scenarios due to the *visibility issues*.

As contrast, our approach achieves more than 90% PR and RR in these three scenarios. Moreover, when increasing pedestrian numbers, our approach can still maintain the robust performance. To further clarify the mechanism of our approach, we next conduct a detailed ablation study.

C. Ablation study

We now investigate four different variants of our approach:

- *front-end only* method only using our front-end;
- *back-end only* method only using our back-end;

TABLE II: Comparison experiments in simulation.

# of pedestrians	Method	PR[%]	RR[%]	F1 score
50	Removert [10]	69.63	64.84	0.671
	ERASOR [12]	72.25	76.26	0.742
	Ours	**95.63**	**94.57**	**0.951**
100	Removert [10]	66.48	57.96	0.619
	ERASOR [12]	69.16	71.87	0.705
	Ours	**95.32**	**94.26**	**0.948**
150	Removert [10]	68.59	51.13	0.586
	ERASOR [12]	71.86	59.13	0.649
	Ours	**95.41**	**90.76**	**0.930**

TABLE III: Ablation study in simulation.

# of pedestrians	Method	PR[%]	RR[%]	F_1 score
50	Removert [10]	69.63	64.84	0.671
	Front-end only	80.69	62.19	0.702
	Back-end only	**95.78**	92.12	0.939
	Non-visibility check	74.48	**99.66**	0.852
	Non-incident correction	90.96	98.92	0.948
	Ours	95.63	94.57	**0.951**
100	Removert [10]	66.48	57.96	0.619
	Front-end only	78.73	57.39	0.664
	Back-end only	95.06	80.85	0.874
	Non-visibility check	71.82	**99.38**	0.834
	Non-incident correction	90.27	99.22	0.945
	Ours	**95.32**	94.26	**0.948**
150	Removert [10]	68.59	51.13	0.586
	Front-end only	76.94	61.42	0.683
	Back-end only	93.83	58.32	0.719
	Non-visibility check	68.12	**99.47**	0.809
	Non-incident correction	89.33	98.97	**0.939**
	Ours	**95.41**	90.76	0.930

- *non-visibility check* method which does not use *visibility check* to approximate $n_{\text{free}}+n_{\text{occ}}$. Instead, we assign the number of the nearest submaps as $n_{\text{free}}+n_{\text{occ}}$, which assumes all voxels are visible in the submap.
- *non-incident correction* approach that does not consider the pseudo occupied points during the *visibility check*.

Similarly, we conduct the ablation study in our simulation environments with different pedestrian numbers. As shown in Table III, the *front-end only* approach outperforms [10], which mainly benefits from our map-based reverting. The qualitative comparison can be found in Figure 3. However, the *front-end only* has a limited efficiency in dynamic point removal but can work as the preprocessing in our framework. The *back-end only* method has superior performance when there are a few dynamic objects, but its performance will drop significantly when the number of people increases. The *non-visibility check* method makes $n_{\text{free}}+n_{\text{occ}}$ larger than the actual value, resulting in a smaller occupancy probability. Although it can remove most of dynamic objects, it also discards some spatial features. The *non-incident correction* and our method have similar performance. Our method with

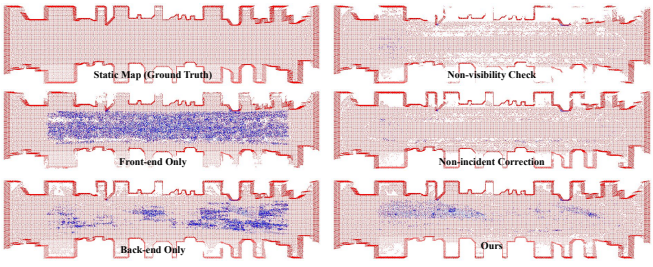


Fig. 7: Qualitative comparisons for the ablation study. The red points denote the preserved static points; the blue points denote the falsely preserved static points.

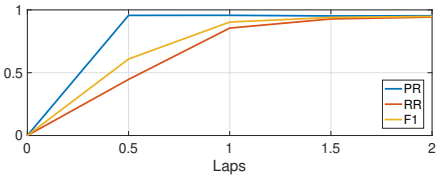


Fig. 8: Online performance for the simulated robot.

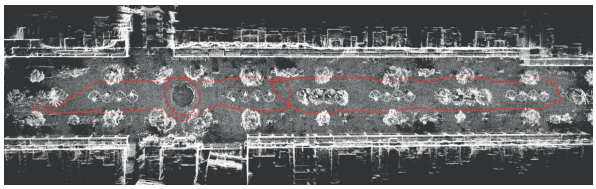


Fig. 9: Our result in the outdoor pedestrian zone. The white points are the preserved static points. The red line are the LiDAR trajectory. The result is clean without any pedestrians, bicycles, etc.

incident correction can preserve more static features, but consequentially some dynamic points are also retained. We found that most of the retained dynamic points are close to the ground with a large incident angle. In general, our proposed framework works as a whole with each module closely related. The qualitative comparisons are shown in Figure 7 .

Additionally, we explore the online performance of the proposed framework. We recorded the online performance when the robot is running in simulation with 100 pedestrians. As shown in Figure 8, after the robot runs around the scene once, our approach can retain the spatial structure in the environment and remove most dynamic points. As the number of observations increases, the performance will increase slightly until saturation after going around the scene twice.

Meanwhile, all the computation can be real-time run on an onboard PC with an Intel i7 8559U, with the processing time of 17.95 frames per second for the front-end and 11.20 submaps per second for the back-end.

D. Real-world experiments

To more realistically demonstrate the online scene understanding of our approach in the urban environment, we recorded two long sequences of real-world dataset in a shopping mall and an outdoor pedestrian zone. The shopping mall has highly dense pedestrians to be removed. For the outdoor pedestrian zone, dynamic objects become more complex, including not only pedestrians but also bicycles, strollers, dogs, etc.

Although both scenarios are challenging for online scene understanding, our approach can still online preserve the dense spatial structure of the urban environment as shown in Figure 1 and Figure 9. All real-world experiments can be found in https://sites.google.com/view/dynamicfilter/.

V. Conclusion

In this paper, we proposed an online dynamic objects removal framework that integrates the map-based and visibility-based approaches. The various experiments validate its efficiency in highly dynamic scenarios.

REFERENCES

[1] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[2] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[3] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving," *arXiv preprint arXiv:2003.03653*, 2020.

[4] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[5] 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.

[6] 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.

[7] P. Pfreundschuh, H. F. C. Hendrikx, V. Reijgwart, R. Dubé, R. Siegwart, and A. Cramariuc, "Dynamic object aware lidar slam based on automatic generation of training data," *arXiv preprint arXiv:2104.03657*, 2021.

[8] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3712–3719.

[9] 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.

[10] G. Kim and A. Kim, "Remove, then revert: Static point cloud map construction using multiresolution range images," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 758–10 765.

[11] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[12] H. Lim, S. Hwang, and H. Myung, "Erasor: Egocentric ratio of pseudo occupancy-based dynamic object removal for static 3d point cloud map building," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2272–2279, 2021.

[13] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[14] 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.

[15] ——, "Rigid scene flow for 3d lidar scans," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1765–1770.

[16] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data," *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[17] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[18] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[20] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IROS*, vol. 3, 2004, pp. 2149–2154.

[21] S. Curtis, A. Best, and D. Manocha, "Menge: A modular framework for simulating crowd movement," *Collective Dynamics*, vol. 1, pp. 1–40, 2016.