Moving Object Segmentation via 3D LiDAR Data: A Learning-Free Real-time Online Alternative

Zinuo Yi[1], Felix Neumann[1], Georg von Wichert[1], and Darius Burschka[2]

***Abstract*— Motion detection in 3D LiDAR is crucial for autonomous systems. While deep learning dominates Moving Object Segmentation (MOS), the potential of learning-free approaches remains underexplored. Unlike problems like semantic segmentation, motion can be explicitly modeled, potentially enabling efficient, interpretable, and computationally lightweight solutions. Motivated by this, we introduce a novel real-time, online, learning-free MOS method. We propose the novel Join Count Feature to extract motion cues from a local window of range images, and long-term filtering with efficient two-step association to enhance accuracy. Compared to learning-based models, we achieve superior precision and competitive IoU for saliently moving objects on SemanticKITTI. Further evaluation on HeLiMOS demonstrate stronger generalization by the proposed method across different LiDAR sensors. These results highlight the potential of learning-free methods for motion detection in 3D LiDAR data. Code will be released upon acceptance.**

I. Introduction

With the rapid advancement of deep learning, many researchers have adopted it to enhance perception modules. Deep learning has demonstrated remarkable performance across various tasks, but traditional approaches offer advantages such as interpretability, scalability, lower computational cost, and independence from large-scale annotated data. Although learning-free methods are nearly infeasible for tasks such as semantic segmentation, motion information can be explicitly described and modeled, suggesting the potential for motion detection without deep learning. Additionally, learning-free methods have recently been applied successfully to ground segmentation [1], [2] and clustering [3], [4], which are important tasks for systems such as autonomous mobile robots. These systems furthermore need to reason about the dynamic state of objects in their surroundings to make informed and safe decisions. Inspired by these advancements, we explore the extent to which learning-free approaches can compete with deep learning methods in Moving Object Segmentation (MOS).

Although learning-free methods have been widely used in motion detection, they are often developed for different objectives, such as tracking or static map construction, and are not directly compared to learning-based methods for the same task. For instance, learning-free motion detection has been explored in moving object tracking using RANSAC-based estimation [5], mapless alignment [6], and particle filters [7]. Meanwhile, static map reconstruction methods incorporate motion detection to remove dynamic points [8], [9]. While these methods are effective in their respective applications, there has been no direct comparison with learning-based approaches for online MOS.

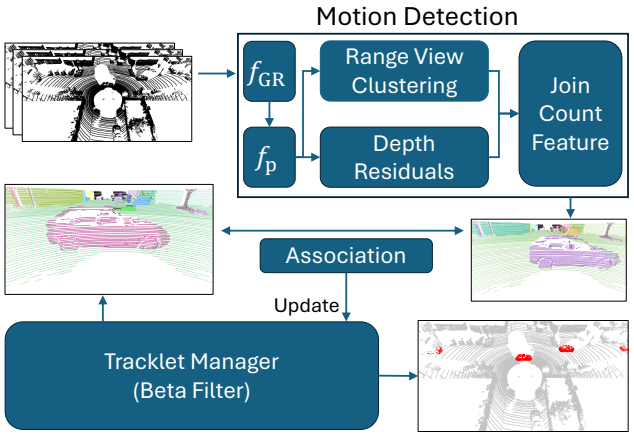

Fig. 1: Overview of our proposed method. Inputs consist of a query LiDAR scan and two reference LiDAR scans, which we project into the range view. We cluster non-ground objects in the query range image and determine their moving state using the proposed Join Count Feature on depth residual images. Objects are tracked over time to accumulate evidence for the moving and static state using the proposed Beta Filter.

To investigate the potential of learning-free methods on this task, we design an approach that efficiently leverages visibility differences in the range view of LiDAR sensors. We propose to estimate the moving or static state of objects using a novel feature descriptor for depth residual images based on Join Count Statistics [10], and to accumulate these estimates over time using a Binary Bayes Filter for tracked moving objects. An overview of this approach is shown in Figure 1. Specifically, our contributions are:

- We propose a novel real-time, online, learning-free approach for point-wise Moving Object Segmentation.
- We evaluate our approach on the SemanticKITTI and HeLiMOS datasets, demonstrating higher precision and notable IoU for saliently moving objects compared to learning-based methods while requiring significantly less computational overhead.
- To the best of our knowledge, this is the first study to explore learning-free methods for online MOS in 3D LiDAR data, providing insights into their limitations as well as potential for motion detection.

[1]Siemens Technology, Friedrich-Ludwig-Bauer-Straße 3, 85748 Garching, Germany, {zinuo.yi, neumann.felix, georg.wichert}@siemens.com

[2]Machine Vision and Perception Group, School of Computation, Information and Technology, Technical University of Munich, Friedrich-Ludwig-Bauer-Straße 3, 85748 Garching, Germany burschka@tum.de

II. Related Work

A. Learning-based models for Moving Object Segmentation

Chen *et al.* [11] introduced the first MOS benchmark on the SemanticKITTI dataset and proposed LMNet, the first deep learning model for MOS. LMNet employs range view and depth residual images to estimate moving object labels in the range view, before re-projecting the estimates into 3D space following Milioto *et al.* [12].

Since LMNet, numerous deep learning models have advanced state-of-the-art MOS performance. Mersch *et al.* [13] proposed 4DMOS, leveraging sparse 4D CNNs with voxelized aggregated scans as input. Neng *et al.* [14] extended 4DMOS with InsMOS, incorporating instance labels for additional supervision and jointly predicting object instances and moving object segmentation. Similarly, Cheng *et al.* [15] proposed MF-MOS, a dual-branch model leveraging semantic labels to predict both moving and movable objects and gain additional supervision signals.

While these models achieve strong performance on SemanticKITTI, their generalization to different domains and sensor types without fine-tuning remains uncertain. To address this, Lim *et al.* [16] introduced HeLiMOS, a dataset featuring 3D LiDAR data from diverse sensors, with point-wise moving object labels, which we also use in our experiments.

B. Learning-free methods for motion detection

Although not specifically designed for MOS, learning-free methods exist for motion detection in 3D LiDAR data. Dewan *et al.* [5] employ RANSAC to iteratively sample points, estimate motion from a scan, and assign each point to a specific motion. Yoon *et al.* [6] proposed aligning two reference scans, one before and one after the query scan, with the query scan itself, using alignment discrepancies as motion cues. Lee *et al.* [7] explicitly model object motion states and track them using a particle filter.

Meanwhile, many learning-free methods for 3D static map reconstruction incorporate motion detection modules. Removert [8] builds a static map by iteratively leveraging visibility differences in multi-resolution range images. ERASOR [9] and ERASOR++ [17] use bin-wise feature descriptors to remove dynamic points by comparing per-scan descriptors with those of the overall map. However, these methods are not designed for online processing, as they require all scans at once. Nevertheless, we take inspiration from their use of range view visibility differences for motion detection.

More recently, Reich *et al.* [18] proposed a learning-free approach for moving object detection and tracking using particles. Their method leverages 1D range images to localize moving objects while employing particles for tracking. Our approach shares similarities in ground removal, clustering, and range view representation. However, their reliance on 1D range images limits detection to the closest objects with planar movements, whereas our use of 2D range images captures a more comprehensive scene representation and can detect motion in all directions.

While these methods were developed in different contexts, none have been directly compared to learning-based approaches. Reich *et al.* [18] evaluated their method against CenterPoint [19], which, however, is not specifically designed for moving object tracking.

C. Range view clustering methods

Range view clustering is a common technique for processing 3D LiDAR data. Bogoslavskyi *et al.* [3] proposed an angle-based range view clustering method, leveraging the intuition that depth values change significantly at object boundaries. Hasecke *et al.* [4] introduced FLIC, which clusters range view pixels multiple times using different connectivity strategies based on Euclidean distances. We draw inspiration from these range view clustering methods and use a similar algorithm in our approach.

III. Methodology

The overview of our approach is shown in Figure 1. Our method comprises three main components: the motion detection module, the tracklet manager, and the association step. The motion detection module processes the input and computes the Join Count Feature, which serves as a moving probability estimate and helps distinguish moving objects from the static ones. However, due to noise and pose estimation errors, these predictions are prone to false positives. To address this, we introduce tracking to smooth the probability estimation.

We track potentially moving and the static objects separately. To associate objects over time, we propose a Tracking-by-Clustering approach for the static objects and employ a computationally more expensive descriptor-based matching only for moving objects. The tracklet manager iteratively updates the moving probability of instance tracklets based on these associations. Finally, point-wise moving object labels are obtained from the point clusters associated with instances with a high moving probability.

A. Problem Formulation

Following the common setting for MOS, we consider a sequence of LiDAR scans $\mathcal{S} = \{\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \dots\}$, along with their estimated and calibrated poses $\mathcal{T} = \{\mathbf{T}^{(1)}, \mathbf{T}^{(2)}, \dots\}$, obtained via odometry or SLAM algorithms. Each scan consists of a point cloud with 3D coordinates $\mathbf{S}^{(j)} = \{\mathbf{p}_i^{(j)} = (x_i^{(j)}, y_i^{(j)}, z_i^{(j)}) \in \mathbb{R}^3\}$. Each pose $\mathbf{T}^{(j)} \in \mathrm{SE}(3)$ represents a transformation from the local to the world coordinate frame. At each time step, three scans are used as input. The most recent scan is denoted as the forward reference scan $\mathbf{S}^{(\mathrm{f})}$, while the query scan is the previous scan, i.e., $\mathbf{S}^{(\mathrm{q})} = \mathbf{S}^{(\mathrm{f}-1)}$. The backward reference scan, taken k steps before, is $\mathbf{S}^{(\mathrm{b})} = \mathbf{S}^{(\mathrm{f}-k)}$, where k is called the span of reference scans. Our goal is to predict point-wise moving labels for the query scan, denoted as $\mathbf{Y}^{(\mathrm{q})}_{\mathrm{moving}}$.

B. Range View Projection, Clustering, and Re-projection

To reduce computational complexity, our approach primarily operates in the range view domain, leveraging LiDAR properties. First, we detect ground points in the query scan $\mathbf{S}^{(\mathrm{q})}$ using Patchwork++ [2], which produces ground labels $\mathbf{Y}^{(\mathrm{q})}_{\text{ground}} \in \{0,1\}^{N_{\text{points}}}$ for each of the N_{points} points in the scan. Next, we transform the reference scans $\mathbf{S}^{(\mathrm{b})}$ and $\mathbf{S}^{(\mathrm{f})}$ into the coordinate system of $\mathbf{S}^{(\mathrm{q})}$ and project all three scans into range view. For each point $\mathbf{p}_i$ in the scan, we calculate its corresponding pixel coordinates (u_i, v_i) in the range view image as follows:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \lfloor \frac{1}{2} \left[1 - \arctan(y_i, x_i)\pi^{-1}\right] W \rfloor \\ \lfloor \left[1 - \left(\arcsin(z_i r_i^{-1}) + \mathrm{f}_{\text{down}}\right)\mathrm{f}^{-1}\right] H \rfloor \end{pmatrix} \quad (1)$$

where W and H are the width and height of the desired range view image, $r_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$ is the depth of the point, and $\mathrm{f} = \mathrm{f}_{\text{up}} + \mathrm{f}_{\text{down}}$ is the vertical field of view of the LiDAR. Each range view pixel stores the 3D coordinate and distance to the closest point that was projected into it. After projection, the resulting three range view images are denoted as $\mathbf{V}^{(\mathrm{q})}$, $\mathbf{V}^{(\mathrm{b})}$, and $\mathbf{V}^{(\mathrm{f})}$. Pixels with no projected points are considered invalid and are excluded unless explicitly mentioned.

Next, we perform range view clustering on the query scan $\mathbf{V}^{(\mathrm{q})}$ to obtain clusters, where pixels with Euclidean distances below a threshold τ_{d} form clusters. Using the ground labels $\mathbf{Y}^{(\mathrm{q})}_{\text{ground}}$, we exclude pixels containing projected ground points, resulting in a ground-free range image $\mathbf{V}^{(\mathrm{q})}_{\text{ng}}$. This ensures objects are no longer connected to each other through the ground.

To efficiently cluster the range image, we first compute neighboring points within a fixed window size in parallel using the 3D coordinates stored in each pixel. We then apply the union-find algorithm, an efficient data structure that maintains disjoint sets and supports two operations: *find* and *union*. The *find* operation retrieves the representative pixel of a set, while *union* merges two sets. Here, we use union-find to merge precomputed neighbors into clusters and assign unique object IDs, which form an object ID image $\mathbf{I}^{(\mathrm{q})}_{\text{obj}}$ for the query scan. To propagate the object IDs back to the point cloud, we employ the re-projection of RangeNet++ [12]. A small adaptation is introduced by not replacing the depth values at the center row of the unfolded range image, which corresponds to the 8-9 lines of the original re-projection algorithm in the RangeNet++ paper. This fixes the issue that every point receives an amplified vote from the object ID of the pixel it is projected to, regardless of the distance between the point and the pixel. We refer readers to the RangeNet++ paper for more details. Using re-projection, we obtain object IDs for the query scan point cloud, denoted as $\mathbf{Y}^{(\mathrm{q})}_{\text{obj}}$.

C. Join Count Feature

With object cluster regions identified in the range view, we introduce the Join Count Feature, a novel descriptor that converts motion cues within each cluster into a motion probability estimate. We use differences between the query and reference scans, i.e., depth residuals, as motion cues. While depth residuals capture variations in depth, not all of them indicate object motion. An illustration of the Join Count Feature calculation is shown in Figure 2. In the following, we detail the computation of depth residuals and the Join Count Feature.

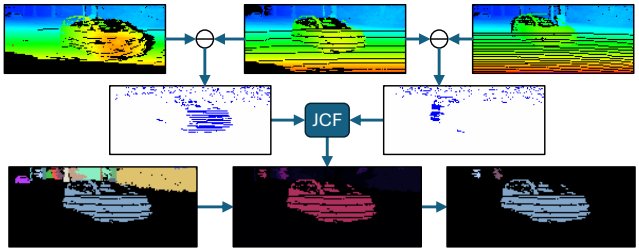

Fig. 2: Illustration of Join Count Feature (JCF) computation. The first row presents depth images from three input scans, which are compared to derive depth residuals (second row). Using the query scan's object ID image (third row, left), JCF is computed within each cluster region (third row, middle). After thresholding, objects with high JCF are considered as potentially moving (third row, right). While the moving car is correctly detected, some static objects are also selected, which is addressed by the proposed temporal filtering.

1) Depth Residuals: Let $\mathbf{D}^{(\mathrm{b})}$, $\mathbf{D}^{(\mathrm{q})}_{\text{ng}}$, and $\mathbf{D}^{(\mathrm{f})}$ denote the depth images, i.e., the depth channel of the range images of the query and reference scans. The depth residuals are then computed as discretized pixel-wise differences. For a depth difference $r^{\text{ref}} = \mathbf{D}^{(\mathrm{q})}_{ng} - \mathbf{D}^{\text{ref}}$, the depth residual $\tilde{\mathbf{D}}$ is calculated as $\tilde{\mathbf{D}}^{\text{ref}} = \text{sign}(r^{\text{ref}})[|r^{\text{ref}}| > \tau_{\text{vis}}]$, where $[\cdot]$ is the iverson bracket and τ_{vis} is a discretization threshold. This results in residual images $\tilde{\mathbf{D}}^{(\mathrm{b})}$ and $\tilde{\mathbf{D}}^{(\mathrm{f})}$ which can assume the values $\{-1, 0, 1\}$. Positive residuals indicate points in the query scan that are occluded by points in the reference scans, as $\mathbf{D}^{(\mathrm{q})}_{ng} > \mathbf{D}^{\text{ref}}$. This can occur either due to a moving object, sensor noise or due to perspective changes from ego motion. Negative residuals indicate points in the query scan that occlude points in the reference scans, such that the points in the query scan lie in the observable free space of the reference scans, i.e. we can trace rays from the sensor to reference scan points that intersect query scan points. This can only occur due to a dynamic object or sensor noise. We therefore only consider negative residuals as motion cues, as they are a more reliable indicator and produce fewer false positive detections due to perspective changes. Since we consider references before and after the query scan, we can detect objects that move radially towards and away from the sensor, as well as objects that move laterally to the sensor. We refer readers to Figure 3 for an illustration of depth residuals.

Only considering negative residuals results in binary depth residual images, as shown in Figure 2, which we still denote as $\tilde{\mathbf{D}}^{(\mathrm{b})}$ and $\tilde{\mathbf{D}}^{(\mathrm{f})}$ for simplicity.

2) Calculation of Join Count Feature: We now have object cluster regions indicated by $\mathbf{I}^{(\mathrm{q})}_{\text{obj}}$ and depth residual images, namely $\tilde{\mathbf{D}}^{(\mathrm{b})}$ and $\tilde{\mathbf{D}}^{(\mathrm{f})}$. Next, we seek to summarize

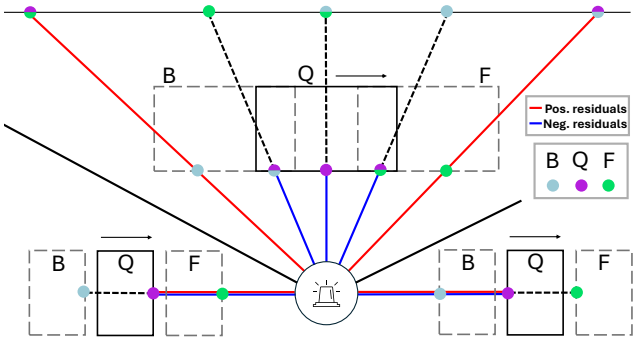

Fig. 3: Illustration of depth residuals caused by radial and lateral object motion. Moving object positions in the backward, query, and forward frames are marked as 'B', 'Q', and 'F'. Solid and dotted lines represent query and reference scan rays, respectively. Light blue, violet, and green dots indicate intersection points in each frame. Negative (blue) residuals are produced when intersection points in the query frame occlude points in reference frames, while positive (red) residuals occur in the opposite case. Radial motion triggers both residuals. Only negative residuals are retained as they correspond to points on moving objects in the query scan.

the depth residuals within each cluster region in the range view. As shown in Figure 2, depth residuals appear grouped together in the region of a moving object, while the reaction caused by noise is distributed sporadically. Inspired by Join Count Statistics [10], we propose a Join Count Feature, which quantitatively turns the spatial aggregation pattern to a normalized value between zero and one. Unlike simple depth difference-based methods, our Join Count Feature captures the spatial consistency of motion cues for each object cluster. The statistical aggregation makes the motion detection more robust to sensor noise. This can be observed in Figure 2, where the depth residuals contain many noisy false positives in the background, which are suppressed by the Join Count Feature computation.

First, we fuse the two depth residual images using a pixel-wise OR operation to produce $\tilde{\mathbf{D}} = \tilde{\mathbf{D}}^{(\mathrm{b})} \vee \tilde{\mathbf{D}}^{(\mathrm{f})}$. We identify if two pixels i and j in a cluster with ID c_i are neighbors as $n_{ij} = w_{ij}[\mathbf{I}^{(q)}_{\mathrm{obj}}[i] == c_i][\mathbf{I}^{(q)}_{\mathrm{obj}}[j] == c_i]$, with

$$w_{ij} = \begin{cases} 1, & \text{if pixel } i \text{ is a direct neighbor of } j \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Next, we compute the one-one join counts within a cluster c_i denoted by BB_{c_i}, and all possible joins A_{c_i} as

$$BB_{c_i} = \sum_{i,j,i\neq j} \tilde{\mathbf{D}}[i]\tilde{\mathbf{D}}[j]n_{ij} \quad \text{and} \quad A_{c_i} = \sum_{i,j,i\neq j} n_{ij}. \quad (3)$$

We define Join Count Feature by normalizing BB_c with the number of all possible joins A_{c_i} of the cluster as follows:

$$J_{c_i} = \frac{BB_{c_i}}{A_{c_i}} \in [0, 1] \quad (4)$$

An example of the Join Count Feature calculation for a cluster is illustrated in Figure 4. Join Count Feature J is

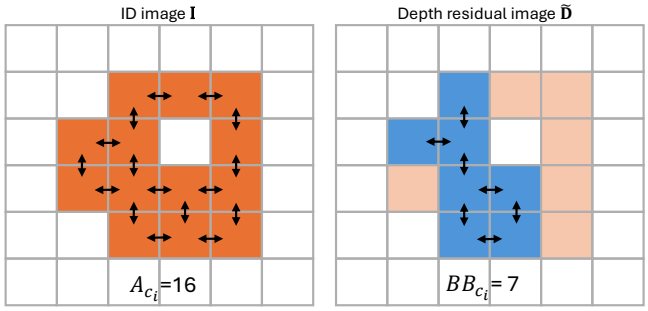

Fig. 4: Example of the calculation of the Join Count Feature for a cluster. Pixels belonging to the cluster are shown in orange, depth residuals in blue. The Join Count Feature is computed only within a cluster.

equal to one if a moving object is fully covered by depth residuals, which happens when the object moves so fast that its image in the range view of two consecutive scans does not overlap at all, or when it is moving radially. J is close to zero when the object is static and the depth residuals are result from random noise. J can thus be interpreted as a probability indicating the likelihood that the object is moving.

However, the Join Count Feature alone is susceptible to issues such as pose estimation errors and small clusters influenced by noise, as shown in Figure 2. To mitigate these issues, we introduce tracking and filtering, which smooths moving probabilities and enhances motion detection accuracy. Nevertheless, the Join Count Feature remains essential for identifying potentially moving objects, forming the foundation for an efficient two-step association.

D. Association and Tracklet Management

To filter moving probabilities over time, we use a tracklet manager that maintains a list of tracklets. Each tracklet stores relevant information about the instance associated with it, including instance IDs d_i and state variables for moving and static states. We refer to the results of instantaneous clustering as objects $\{c_i\}$ and objects tracked over time as instances $\{d_i\}$. At each time step, detected objects are associated to tracked instances. The point cloud and instance IDs of the previous query scan, denoted by $\mathbf{S}^{(\mathrm{q'})}$ and $\mathbf{Y}^{(\mathrm{q'})}_{\mathrm{Ins}}$ respectively, are stored in the tracklet manager. We propose to employ two separate association approaches for potentially moving and static objects to reduce the required computational resources. For potentially moving objects we apply a descriptor-based matching approach, which is computationally expensive as point cloud descriptors are calculated for each object. To track static objects, we propose a Tracking-by-Clustering approach, which leverages the geometric consistency of static objects and is computationally lightweight. We identify potentially moving objects $\{c_{i,\mathrm{m}}\}$ as clusters in the query scan with a Join Count Feature J_{c_i} exceeding a threshold τ_J and match them with potentially moving instances $\{d_{i,\mathrm{m}}\}$ stord in the tracklet manager. We describe how potentially moving instances are identified in Section III-E.

1) Descriptor-based Moving Instance Matching: Using $\mathbf{S}^{(\mathrm{q'})}$ and $\mathbf{Y}_{\mathrm{Ins}}^{(\mathrm{q'})}$, we extract point clouds of potentially moving instances, denoted by $\{\mathbf{P}_{d_i,\mathrm{m}}\}$. Similarly, we extract point clouds of potentially moving objects from $\mathbf{S}^{(\mathrm{q})}$ and $\mathbf{Y}_{\mathrm{obj}}^{(\mathrm{q})}$ (obtained through re-projection in Section III-B), denoted by $\{\mathbf{P}_{c_i,\mathrm{m}}\}$. We match these two sets of point clouds using the Hungarian algorithm [20]. The cost matrix used during matching is comprised of a feature similarity cost and a Euclidean distance cost based on the assumptions that moving instances (1) do not change their shapes significantly, and (2) do not move far in the time between two consecutive scans. We compare the similarity of two 3D point clouds $\mathbf{P}_{n_i}$ and $\mathbf{P}_{m_i}$ using their respective mormalized Globally Aligned Spatial Distribution (GASD) descriptors [21], denoted by $f_{\mathrm{gasd}}(\cdot)$. The cluster similarity is calculated as $s_{\mathrm{gasd}}(n_i, m_i) = f_{\mathrm{gasd}}(\mathbf{P}_{n_i}) \cdot f_{\mathrm{gasd}}(\mathbf{P}_{m_i})$. The Euclidean similarity based on the cluster centroids $\bar{\mathbf{P}}_{n_i}$ and $\bar{\mathbf{P}}_{m_i}$ is calculated as $s_{\mathrm{dist}}(n_i, m_i) = \exp(-\|\bar{\mathbf{P}}_{n_i} - \bar{\mathbf{P}}_{m_i}\|/\sigma_{\mathrm{dist}})$, with scale parameter σ_{dist}. The total similarity is calculated as a weighted sum $s(n_i, m_i) = \lambda\, s_{\mathrm{gasd}}(n_i,\ m_i) + (1-\lambda)\, s_{\mathrm{dist}}(n_i, m_i)$, with weight λ. We set the similarity to zero if the distance exceeds a threshold τ_{dist}, or the GASD similarity is lower than a threshold τ_{gasd}. Additionally, if the ratio between the volumes of the bounding boxes encompassing the two cluster point clouds is smaller than τ_{volume}, the similarity is also set to zero. The cost matrix is then constructed as the negative similarity of each possible pair, and the Hungarian algorithm is used to solve the assignment problem. Only pairs with a positive similarity are considered as valid.

While this association method works for static objects too, we find that the computation of the GASD descriptor is expensive and propose to instead propagate cluster IDs based on overlapping cluster regions, which we call Tracking-by-Clustering.

2) Tracking-by-Clustering and Label Extension: The goal of Tracking-by-Clustering is to propagate the instance IDs of static background from the previous query scan to the current one, using the instance ID image and the range image of the previous query scan, i.e., $\mathbf{I}_{\mathrm{Ins}}^{(\mathrm{q'})}$ and $\mathbf{V}_{\mathrm{ng}}^{(\mathrm{q'})}$, which are obtained by projecting $\mathbf{S}^{(\mathrm{q'})}$ and $\mathbf{Y}_{\mathrm{ID}}^{(\mathrm{q'})}$ into the range view of the query scan.

We only label pixels containing range measurements that do not belong to the ground or potentially moving objects. To label a pixel in $\mathbf{V}_{\mathrm{ng}}^{(\mathrm{q})}$, we consider neighboring points within a fixed-size window centered at the query pixel in $\mathbf{V}_{\mathrm{ng}}^{(\mathrm{q'})}$. Only points within the fixed window that have a Euclidean distance less than τ_{TBC} to the query pixel are considered. The query pixel is then assigned the most common instance ID in the neighborhood. This process is illustrated in Figure 5. Since the algorithm operates on a fixed-size window, it can be efficiently parallelized for improved computational performance.

During Tracking-by-Clustering, some pixels in $\mathbf{I}_{\mathrm{ID}}^{(\mathrm{q})}$ may receive no votes due to viewpoint changes or the appearance of new objects. To address this, we apply the range view clustering approach from Section III-B to the unlabeled pixels $\mathbf{V}_{\mathrm{ng}}^{(\mathrm{q})}$ and label all unlabeled pixels in a cluster with the mode of instance IDs in the cluster. If a small group of unlabeled pixels cannot be assigned an existing ID and forms an independent cluster, a new instance is generated, creating a new tracklet, as described in Section III-E.

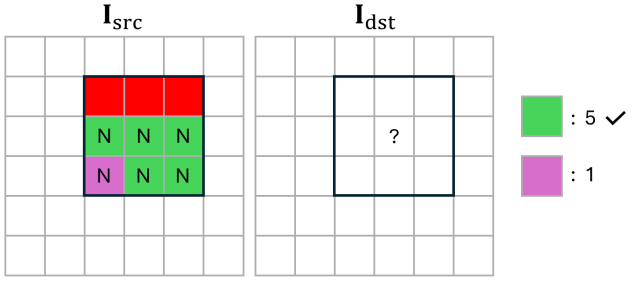

Fig. 5: Illustration of the Tracking-by-Clustering approach for static objects. Static instance IDs of from the previous query scan are propagated into the current one in this step. A query pixel (denoted by '?') is labeled with the mode of the projected IDs in a local Euclidean neighborhood. Pixels in red did not receive valid projections.

The final instance ID image $\mathbf{I}_{\mathrm{Ins}}^{(\mathrm{q})}$ consists of the IDs from the descriptor-based tracking for potentially moving objects and the IDs of Tracking-by-Clustering for static objects, as well as any newly created instances. After association, we re-calculate the Join Count Feature using cluster regions from $\mathbf{I}_{\mathrm{Ins}}^{(\mathrm{q})}$ instead of $\mathbf{I}_{\mathrm{obj}}^{(\mathrm{q})}$, obtaining $\{J_{d_i}\}$, for updating the state of instances.

E. Motion Evidence Accumulation

For more robust estimation of the motion state of an instance, we propose to accumulate evidence for the instance being in motion or static over time. We model this state using a Beta Distribution $\mathcal{X}_{d_i} = \mathrm{Beta}(\alpha_{d_i}, \beta_{d_i})$, where α_{d_i} represents the accumulated evidence for the instance moving and β_{d_i} represents the accumulated evidence for the instance being static. We assume that the motion state is constant in small time intervals. At each time step we update the parameters of the Beta distribution of an instance d_i with the re-calculated Join Count Feature motion probability as $\alpha_{d_i}^{(t+1)} = \alpha_{d_i}^{(t)} + J_{d_i}^{(t)}$ and $\beta_{d_i}^{(t+1)} = \beta_{d_i}^{(t)} + 1 - J_{d_i}^{(t)}$.

Each tracklet also records the number of associations and the time since its last association. A new tracklet is initially treated as a candidate until it is tracked for longer than a birth threshold τ_{birth}. Conversely, a tracklet is removed if the time since its last association exceeds a death threshold τ_{death}.

For a new instance, we create a tracklet with a unique instance ID d_i and initialize its state variable $\mathcal{X}_{d_i}$ as a Beta distribution using the corresponding Join Count Feature J_{d_i} as $\mathcal{X}_{d_i} \sim \mathrm{Beta}(J_{d_i}, 1 - J_{d_i})$. For initialization, tracklets are generated for every object. The latest moving probability estimate for each instance is given by the mean of the Beta distribution:

$$\hat{p}_{\mathrm{moving},d_i}^{(t)} = \mathbb{E}[\mathcal{X}_{d_i}^{(t)}] = \frac{\alpha_{d_i}^{(t)}}{\alpha_{d_i}^{(t)} + \beta_{d_i}^{(t)}} \tag{5}$$

After updating state variables, an instance is classified as potentially moving if its moving probability exceeds τ_J. These potentially moving objects are stored for the descriptor-based association in the next time step. If the moving probability surpasses τ_{m}, it is classified as moving for final point-wise prediction. Point-wise predictions are generated by re-projecting $\mathbf{I}_{\text{Ins}}^{(\text{q})}$ to 3D space (Section III-B) and labeling all points belonging to moving instances.

IV. Experiments and Analysis

All experiments are conducted on the open datasets SemanticKITTI [22] and HeLiMOS [16]. Both provide manually annotated moving labels, with SemanticKITTI additionally offering instance labels. Since our approach relies on reference scans, the first and last scans of each sequence cannot serve as query scans. Therefore, predictions for these scans are set to all static.

A. Parameter settings and evaluation metrics

The parameter values for our approach are set as follows: $k = 2$, $W = 1024$, $H = 64$, $\text{f}_{\text{up}} = 2.0$, $\text{f}_{\text{down}} = -24.8$, $\tau_{\text{d}} = 0.7$, $\tau_{\text{vis}} = 0.5$, $\tau_J = \tau_{\text{m}} = 0.4$, $\tau_{\text{birth}} = 3$, $\tau_{\text{death}} = 2$, $\lambda = 0.4$, $\tau_{\text{dist}} = 8$, $\tau_{\text{gasd}} = 0.8$, $\tau_{\text{volume}} = 0.5$, $\sigma_{\text{dist}} = 2$, $\tau_{\text{TBC}} = 0.5$. The window size is five for re-projection and Tracking-by-Clustering, and nine for range view clustering.

For evaluation, we follow the benchmark by Chen *et al.* [11] and use IoU, precision, and recall of moving points.

B. Moving Object Segmentation in SemanticKITTI

As suggested by many learning-free motion detection methods [6], [5], [18], a condition for detectable moving objects is necessary, as these approaches rely solely on motion cues rather than semantic information. Motion cues, in turn, are influenced by motion saliency. We find that the moving object labels in SemanticKITTI label all objects that move at any point during a sequence as moving in the all frames of the sequence. This is inaccurate, as temporarily static objects, such as a car at a traffic light, are also labeled as moving. Leveraging instance labels from SemanticKITTI, we re-label the dataset, only labeling objects with translations exceeding the minimum translation threshold τ_{vis} between consecutive frames as moving. Since τ_{vis} represents the minimum depth residual difference, objects labeled as moving in the original dataset with smaller translations are ignored during metric computation.

To evaluate our approach, we compare it against three representative learning-based models: LMNet [11], InsMOS [14], and MF-MOS [15]. LMNet pioneered deep learning for MOS, establishing the benchmark. MF-MOS is the current state-of-the-art open-source model, ranking first, followed by InsMOS. For each model, we adopt the best-performing configuration reported in the original papers. The results are summarized in Table I.

Method	Rec.	Prec.	IoU	GPU Mem.	Time
LMNet[11]	0.929	0.706	0.669	0.96 GB	16 ms
InsMOS[14]	<u>0.961</u>	0.631	0.616	2.93 GB	75 ms
MF-MOS[15]	**0.967**	<u>0.784</u>	**0.764**	21.32 GB	77 ms
Ours	0.831	**0.861**	<u>0.733</u>	0.68 GB	78 ms

TABLE I: Performance comparison with three learning-based approaches to MOS on the SemanticKITTI validation set. Our learning-free approach is slightly less sensitive to moving objects but also less prone to predicting false positives, yielding a competitive IoU score.

Despite being a learning-free approach, our method outperforms state-of-the-art deep learning models in precision and achieves the second-best IoU, highlighting its efficiency given the significantly lower computational overhead. However, deep learning models excel in recall, where our approach falls short. We attribute this to only few depth residuals being generated at the boundary of slowly moving objects when considering consecutive LiDAR frames, which leads to a weaker response of the Join Count Feature. Further potential reasons are a loss of peripheral target points caused by conservative ground removal, range-view clustering, and fixed-window-based re-projection. Another factor is the Beta Filter's birth mechanism, which retains moving predictions for tracklet candidates. When association is interrupted, a new birth threshold period must pass before the moving instance is re-established. This suggests a promising direction for improving recall performance.

C. Performance per Min. Translation Analysis

To further examine the impact of motion saliency, we analyze the performance of our approach and the three deep learning models across different minimum translation thresholds τ_{vis} . Since learning-based models may overfit the training set and no instance labels are publicly available for the test set, this analysis is conducted on the SemanticKITTI validation set. A zero translation threshold corresponds to the original labeling scheme. The results are shown in Figure 6.

As expected, the recall of our approach increases significantly with the minimum translation threshold, reaching a level comparable to learning-based methods once the threshold exceeds 0.6 m. Interestingly, recall also initially improves for deep learning models, suggesting that detecting objects with small translations is a universal challenge, not limited to learning-free methods. This highlights a potential improvement direction for all approaches.

Apart from the original scheme, our approach consistently outperforms deep learning models in precision. Consequently, our IoU surpasses deep learning models when the minimum translation threshold exceeds 0.6 m, indicating that our method is more robust to motion saliency and achieves better performance when motion cues are more reliable. An interesting observation is that precision decreases for all methods as the threshold increases, likely due to the rising influence of false positives as more moving objects are filtered out.

For a comprehensive evaluation, we also assess our approach on all sequences with available instance labels, in-

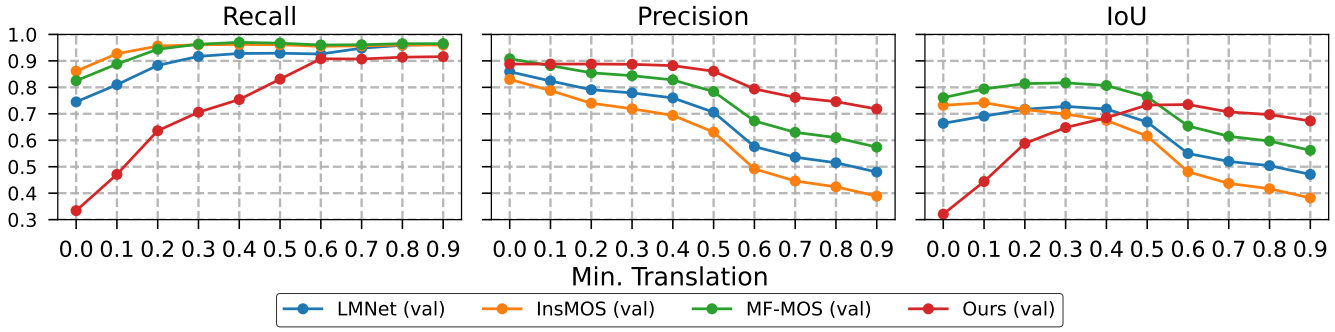

Fig. 6: Performance comparison across different minimum translation thresholds on the SemanticKITTI validation set. Our approach still struggles to detect slow moving objects as they do not cause large enough depth residuals, but we match and exceed the performance of learning-based methods for faster objects. This indicates that learning-free methods may be a viable alternative to deep learning models for the task of MOS, albeit further work to improve the recall is required.

Method	**Aeva**			**Avia**			**Ouster**			**Velodyne**			**Avg.**		
	Rec.	Prec.	IoU	Rec.	Prec.	IoU	Rec.	Prec.	IoU	Rec.	Prec.	IoU	Rec.	Prec.	IoU
LMNet[11]	0.683	0.064	0.063	0.591	0.349	0.281	0.854	0.058	0.057	0.228	0.229	0.129	0.717	0.067	0.065
InsMOS[14]	**0.969**	0.032	0.032	**0.844**	0.059	0.059	**0.951**	0.034	0.034	**0.844**	0.055	0.054	**0.937**	0.036	0.036
MF-MOS[15]	0.719	0.223	0.206	0.754	0.503	**0.432**	0.516	0.601	0.384	0.371	0.262	**0.182**	0.611	0.332	0.274
Ours	0.342	**0.922**	**0.333**	0.275	**0.722**	0.249	0.548	**0.847**	**0.498**	0.134	**0.826**	0.130	0.411	**0.855**	**0.384**

TABLE II: Performance comparison on the HeLiMOS validation containing LiDAR sensors outside the training domain of deep learning methods. LMNet and InsMOS predict large amounts of false positives on most sensors, shown by their high recall but very low precision. MF-MOS achieves a better balance, but is still outperformed by our approach on overall IoU.

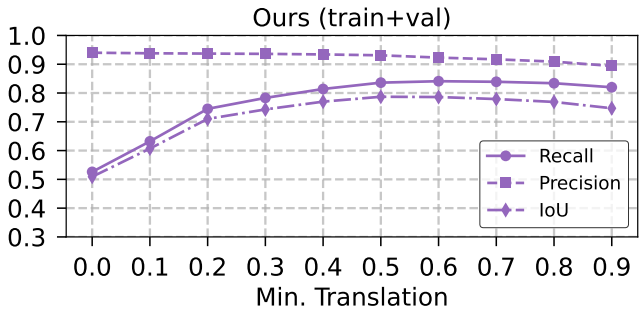

Fig. 7: Performance comparison of our approach across different minimum translation thresholds on the SemanticKITTI training and validation sets. On this more diverse data split we achieve a higher performance than just on the validation split in Figure 6, especially for slow moving objects

cluding the training and the validation sets. As shown in Figure 7, the results follow a similar trend: recall increases with the minimum translation threshold, while precision remains consistently high. The IoU aligns with previous findings, peaking at 0.787 when the threshold is 0.5 m.

D. Moving Object Segmentation in HeLiMOS

We further evaluate our approach on the recently released HeLiMOS dataset [16], which presents greater challenges due to its diverse LiDAR sensors, including omnidirectional (Velodyne VLP-16, Ouster OS2-128) and solid-state (Livox Avia, Aeva Aeries II) types. For evaluation, we extract validation scans and form them into three consecutive subsequences. However, HeLiMOS lacks instance labels and contains numerous pedestrians. To partially compensate, we increase the reference scan span k to nine, allowing query scans to be compared with earlier backward reference scans, enhancing motion cues. All other parameters remain unchanged. For reference, we also evaluate the three deep learning models on HeLiMOS without re-training. The results are shown in Table II.

The results align with previous experiments on SemanticKITTI, where deep learning methods excel in recall. However, all of the learning-based approaches, but especially LMNet and InsMOS, have very low precision, because they produce a large amount of false positive estimations. MF-MOS demonstrates better generalization compared to LMNet and InsMOS. Our approach on the other hand is more conservative and achieves very high precision, albeit a lower recall due to the afforementioned issues of slowly moving objects. Nonetheless, our method attains the highest IoU on two sensor types and on average, demonstrating the universal effectiveness of motion cue aggregation and filtering.

E. Computational Efficiency

Statistics regarding the computational efficiency are collected by running inference on the SemanticKITTI validation set using a single Nvidia RTX 6000 Ada GPU and a single core of an AMD Ryzen Threadripper PRO 5965WX CPU (3.8 GHz). Our approach achieves an average runtime of 78 ms per time step, covering all stages from range view projection to point-wise moving label prediction, ensuring real-time

processing. Additionally, our method requires significantly less GPU memory, as shown in Table I, demonstrating its excellent computational efficiency.

V. Conclusions

In this paper, we propose a novel learning-free, real-time online approach for MOS. Leveraging the proposed Join Count Feature and two-step association for filtering, our method efficiently and accurately detects saliently moving objects. Experiments demonstrate that despite significantly lower computational overhead, our approach outperforms state-of-the-art deep learning models in precision and achieves competitive IoU for saliently moving objects. Evaluation on the HeLiMOS dataset further confirms its universal effectiveness and the sensitivity of learning-based approaches to shifts in the input domain. Our findings provide strong evidence that well-designed learning-free approaches still hold great potential in motion detection. Moving forward, we aim to enhance recall, particularly for slower-moving objects, by incorporating adaptive filters and integrating explicit motion models to further refine motion detection accuracy.

References

[1] H. Lim, M. Oh, and H. Myung, "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6458–6465, 2021.

[2] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 276–13 283.

[3] I. Bogoslavskyi and C. Stachniss, "Fast range image-based segmentation of sparse 3d laser scans for online operation," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 163–169.

[4] F. Hasecke, L. Hahn, and A. Kummert, "Flic: Fast lidar image clustering," *arXiv preprint arXiv:2003.00575*, 2020.

[5] 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.

[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] H. Lee, H. Lee, D. Shin, and K. Yi, "Moving objects tracking based on geometric model-free approach with particle filter using automotive lidar," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 863–17 872, 2022.

[8] 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, Conference Proceedings, pp. 10 758–10 765.

[9] 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.

[10] P. A. Moran, "The interpretation of statistical maps," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10, no. 2, pp. 243–251, 1948.

[11] 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*, vol. 6, no. 4, pp. 6529–6536, 2021.

[12] 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.

[13] B. Mersch, X. Chen, I. Vizzo, L. Nunes, J. Behley, and C. Stachniss, "Receding moving object segmentation in 3d lidar data using sparse 4d convolutions," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7503–7510, 2022.

[14] N. Wang, C. Shi, R. Guo, H. Lu, Z. Zheng, and X. Chen, "Insmos: Instance-aware moving object segmentation in lidar data," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 7598–7605.

[15] J. Cheng, K. Zeng, Z. Huang, X. Tang, J. Wu, C. Zhang, X. Chen, and R. Fan, "Mf-mos: A motion-focused model for moving object segmentation," *arXiv preprint arXiv:2401.17023*, 2024.

[16] H. Lim, S. Jang, B. Mersch, J. Behley, H. Myung, and C. Stachniss, "Helimos: A dataset for moving object segmentation in 3d point clouds from heterogeneous lidar sensors," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 14 087–14 094.

[17] J. Zhang and Y. Zhang, "Erasor++: Height coding plus egocentric ratio based dynamic object removal for static point cloud mapping," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 4067–4073.

[18] A. Reich and H.-J. Wuensche, "Fast detection of moving traffic participants in lidar point clouds by using particles augmented with free space information," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 538–543.

[19] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[20] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[21] J. P. Silva do Monte Lima and V. Teichrieb, "An efficient global point cloud descriptor for object recognition and pose estimation," in *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2016, pp. 56–63.

[22] 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 *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.