MobileClusterNet: Unsupervised Learnable Clustering of Mobile 3D Objects

A. Kulshreshtha

Supervisors: D. M. Gavrila and T. de Vries Lentsch

Tuesday, 22 October 2024

Delft University of Technology

TU Delft

MobileClusterNet: Unsupervised Learnable Clustering of Mobile 3D Objects

by

A. Kulshreshtha

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Tuesday October 29, 2024 at 3:00 PM.

Student number:	5695821	
Project duration:	February 1, 2024 – October 22, 2024	
Thesis committee:	Dr. D. M. Gavrila,	TU Delft, supervisor and committee chair
	Ir. T. de Vries Lentsch,	TU Delft, daily supervisor and committee member
	Dr. H. Caesar,	TU Delft, Cognitive Robotics committee member

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

MobileClusterNet: Unsupervised Learnable Clustering of Mobile 3D Objects

Ayush Kulshreshtha
Delft University of Technology

Abstract

Unsupervised 3D object detection methods can reduce the reliance on human-annotations by leveraging raw sensor data directly for supervision. Recent approaches combine density-based spatial clustering with motion and appearance cues to extract object proposals from the scene, which serve as pseudo-annotations. However, density-based methods struggle with the uneven data densities seen in LiDAR point clouds, and fail to distinguish between foreground and background objects effectively. To address this issue, this thesis introduces MobileClusterNet, a learnable framework designed for 3D spatial clustering. MobileClusterNet incorporates a novel loss module which utilizes appearance embeddings alongside scene flow information, thereby learning to generate high-quality clusters consisting of both static and dynamic mobile objects. Annotations generated by MobileClusterNet can be used for training any existing supervised detector, without the need for extensive self-training. Experimental results on the Waymo Open Dataset demonstrate that MobileClusterNet outperforms traditional density-based methods like HDBSCAN in clustering performance by a large margin, and provides high quality proposals for training supervised detectors.

1. Introduction

Autonomous driving is set to redefine the approach to mobility, safety and urban design, offering benefits such as improved traffic flow, reduction in accidents, lower fuel emissions, and increased accessibility for the disabled and elderly [14]. At the heart of the autonomous driving technology is 3D object detection, which is a core component of the perception pipeline. Current state-of-the-art methods for 3D object detection rely predominantly on supervised learning. Training these sophisticated supervised detection models requires extensive human-annotated datasets, which are not only time consuming and costly to produce but can also be prone to human error [34]. Additionally, these models face challenges adapting across different datasets [53].

In contrast, unsupervised learning offers a promising alternative to reduce the dependency on extensive annotated datasets by directly extracting meaningful patterns and object characteristics from raw, unlabeled sensor data. In the 2D domain, there has been a notable emergence of 'foundation models' [7, 39]—large models pre-trained with self-supervised learning that generate robust visual features applicable to a wide array of downstream tasks, including 2D object discovery, without requiring fine-tuning [46, 47].

Unlike images, 3D point cloud data is typically sparse, and objects within the point cloud are unevenly distributed across its range; that is, objects might not appear salient in the point cloud. Given that a LiDAR sensor captures accurate spatial information of the environment, spatial clustering has become an integral component of 3D unsupervised object discovery. Recent works in this domain [3, 29, 54, 62, 64] employ density-based spatial clustering methods like DBSCAN [13] and HDBSCAN [5] to extract 3D instance proposals from the scene. These methods work on the core assumption that 3D space is characterised by regions of high point density (object clusters) separated by regions of low point density (noise). However, this assumption often falls short for LiDAR point clouds due to factors like data incompleteness (part of the object can be occluded by itself) and low point density in far ranges. Furthermore, density-based spatial clustering does not differentiate between foreground (mobile) and background (non-mobile) object instances. We define mobile instances as objects that have the ability to move, such as vehicles, cyclists, and pedestrians, which may appear either static or dynamic in the scene. Accurately detecting mobile objects and predicting their future trajectories are crucial tasks in autonomous driving, as this allows the vehicle to plan and execute safe maneuvers.

To discover mobile objects in the scene, spatial clustering is often combined with temporal information to generate dynamic object proposals. An existing 3D detector is then trained using these dynamic object proposals, with the hypothesis of inference-time generalization to static instances of mobile classes based on their geometric features. However, this training approach inherently penalizes the detection of static mobile objects, potentially causing the detector to overfit subtle variations in the data distribution between static and dynamic instances. To address

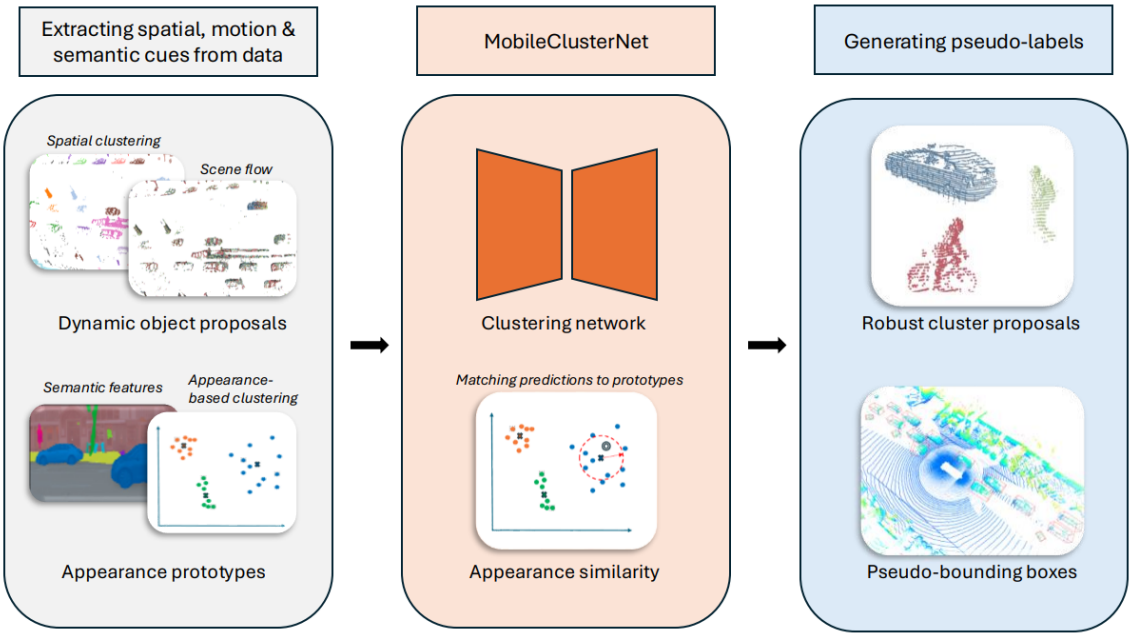


Figure 1. **MobileClusterNet effectively generates robust clustering proposals for both static and dynamic mobile objects.** By leveraging appearance-based prototype embeddings of dynamic objects, it identifies semantically similar static clusters, thereby generating high quality clustering proposals. These can then be utilized to train supervised 3D object detectors.

this, [3, 62, 64] employ self-training, where a detector is iteratively re-trained using labels generated from its own predictions, gradually incorporating static instances into the training set. While self-training can boost performance, it is computationally expensive and risks including static instances of immobile objects in the training set, which can lead to label inconsistencies.

To discover static mobile instances without self-training, UNION [29] employs a vision foundation model to generate a visual appearance embedding for each spatial cluster. This embedding helps in grouping spatial clusters that appear visually similar. Additionally, by estimating the motion of each cluster through self-supervised scene flow, UNION can differentiate between static foreground and static background instances, thereby refining its object proposal selection. However, like other methods based on density-based spatial clustering, UNION's performance is affected by the quality of the initial spatial clusters, particularly in sparse regions of the point cloud where it is prone to under-segmentation/over-segmentation. In contrast, Wang et al. [54] introduced ClusterNet, a learnable point cloud clustering network designed to overcome the limitations of traditional density-based methods. Despite its improvements, ClusterNet is trained solely using dynamic object proposals from LiDAR data, and utilizes a slow, multi-modal joint optimization procedure to discover static mobile instances.

We hypothesize that the quality of spatial clustering proposals can be enhanced by adopting a learning-based approach that integrates semantic features directly during the training stage. This integration enables the network to learn to identify both dynamic and static mobile instances within the scene. The key contributions of this thesis include:

1. The LiDAR-only ClusterNet architecture by Wang et al. [54] is extended with a multi-modal loss module to a new network called **MobileClusterNet**.
2. We propose a novel unsupervised training strategy for MobileClusterNet that incorporates appearance embeddings directly during training, enabling it to learn to cluster all mobile objects in the scene, whether static or dynamic.
3. Validation through experiments on the Waymo Open Dataset [49] demonstrate that MobileClusterNet outperforms traditional spatial clustering methods by a large margin, and it's 3D proposals serve as effective pseudo-labels for training off-the-shelf 3D detectors.

2. Related work

2.1. Unsupervised 2D object discovery

Early works on unsupervised object detection in images focused on generating class-agnostic region proposals as the initial stage of an object detection pipeline. Techniques such as graph-based segmentation leveraged distinct object characteristics, including color information and edge boundaries [1, 17, 50] as well as motion cues [48].

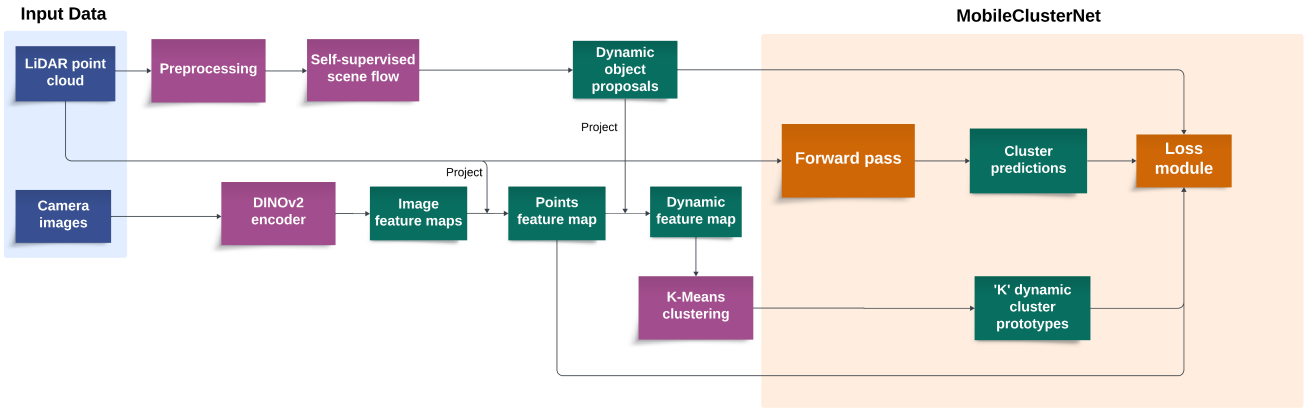


Figure 2. **Overview of MobileClusterNet.** Multi-modal input data (blue) is fed through processing modules (purple) to extract spatial, motion and semantic information (green) before being fed to MobileClusterNet (orange).

More recently, self-supervised learning has emerged as a promising direction, with Vision Transformers (ViTs) trained via self-distillation showing an ability to learn features that inherently encode semantic segmentation information [7, 39]. These learned feature embeddings are effectively employed in methods such as LOST [46], FreeSOLO [51], and cutLER [52] to generate unsupervised object masks, enabling 2D instance segmentation without the need for manual annotations. Self-supervised learning is discussed in greater detail in the section 6.

2.2. Unsupervised 3D object discovery

2.2.1 Spatial clustering

This section only discusses a few relevant methods for spatial clustering. A detailed discussion on additional methods can be found in section 7.

Density-based clustering DBSCAN [13] is a popular method for unsupervised spatial clustering, particularly for identifying arbitrary-shaped clusters. It uses two key parameters which define the minimum number of points required for a cluster and the radius for point reachability. DBSCAN can effectively handle outliers and identify clusters of varying shapes and sizes without needing a predefined number of clusters. However, its performance is sensitive to the chosen parameters, and it struggles with varying point densities or when clusters are not well-separated. HDBSCAN [5] extends DBSCAN to handle clusters with varying densities by building a hierarchical cluster tree and condensing it based on the stability of clusters across different scales. Unlike DBSCAN, HDBSCAN does not require specifying a radius for point reachability, making it more flexible in datasets with uneven densities. Despite these advantages, HDBSCAN still suffers from over-segmentation/under-segmentation issues and cannot differentiate between mobile and non-mobile instances.

Learning-based clustering As opposed to density-based clustering, ClusterNet by Wang et al. can learn to cluster the point cloud into (dynamic) foreground instances. For each point, it assigns a foreground or background label and an instance ID, yield K clusters per LiDAR frame. ClusterNet starts by voxelizing the 3D points and augments them with the features from a transformer-based feature extractor [16]. Inspired by VoteNet [42], its voting module predicts class labels and center offsets, aggregating voted points into spatial clusters via center voting results. Predicted cluster centers form graph vertices which are connected if their distance is below a threshold, with each connected component representing a cluster instance.

ClusterNet is trained only using dynamic object proposals, and penalizes static instances of mobile objects during training. In contrast, our method directly teaches the network to discover all mobile instances during training.

2.2.2 Object proposal refinement

Since density-based clustering methods cannot distinguish between foreground and background, various methods incorporate motion cues from consecutive LiDAR frames to discover moving objects. MODEST [62] utilizes ephemerality as a heuristic to detect objects by requiring multiple traversals over the same area to identify transient objects. This dependency on specific data collection protocols limits its scalability. Alternatively, methods like Najibi et al. [37] and LISO [3] enhance spatial 3D instance proposals with self-supervised scene flow [4, 30, 31] and temporal tracking, extracting dynamic object proposals from spatial clusters. As stated in [3], [37] focuses only on discovering moving objects within the scene and suffers from a large performance gap between discovering dynamic mobile and static mobile objects. LISO [3] undergoes multiple training iterations using self-training, gradually incorporating

static class instances. However, this iterative retraining introduces inconsistencies, with the network being penalized for detecting static class instances, and the propagation of false positive training targets. OYSTER [64] takes a different approach and relies on the notion that spatial clustering methods can perform relatively well on the near range of a LiDAR point cloud, due to higher density. They only extract proposals from the near-range and rely on data augmentations for generalization to far range. Temporal consistency along frames is used to further refine object proposals. However, like other methods, it struggles to segregate foreground from background, affecting its overall effectiveness.

We also use self-supervised scene flow [31] to train MobileClusterNet, but it can learn to directly output high quality object proposals, without relying on self-training.

2.2.3 Static object discovery with semantic cues

In [54], a 3D instance segmentation network (ClusterNet) and a 2D localization network (Faster R-CNN [43]) are consecutively optimized, exploiting spatio-temporal consistency across 2D video frames and 3D point cloud sequences as a supervisory signal. This work does not make use of multi-modal features jointly for generating pseudo-bounding boxes. UNION [29] on the other hand uses appearance embeddings generated by encoding camera images with a DINOv2 vision transformer (ViT) to group spatial clusters into appearance-based pseudo classes. Appearance classes with a high proportion of dynamic components, determined via scene flow, are classified as mobile.

Our method builds on this by computing appearance embeddings similar to those in UNION. Instead of using these embeddings to segregate mobile clusters from the set of all spatial clusters, we use them to establish dynamic object prototype embeddings that serve as direct training signals for our clustering network.

3. Methodology

3.1. MobileClusterNet overview

An overview of MobileClusterNet is shown in Figure 2. The input to the framework consists of raw LiDAR point clouds and multi-camera images captured by M cameras. At each time step t, let $P_t \in \mathbb{R}^{N\times 3}$ denote a single point cloud and $I_{m,t} \in \mathbb{R}^{H\times W\times 3}$ denote an image captured by camera $m \in M$, with height H and width W. We assume that the LiDAR sensor and the camera suite have overlapping fields of view (FoV), and that projection matrices, along with transformation matrices, are available for projecting and transforming data across sensor frames and time steps.

The output of MobileClusterNet includes class-agnostic bounding box annotations β_t for all detected mobile objects in frame t. Each bounding box $b_t = [x, y, z, l, w, h, \theta] \in \beta_t$ consists of the geometric center (x, y, z), dimensions (l, w, h), and orientation θ of the mobile object. These pseudo-bounding boxes can be utilized as supervisory signals for training any off-the-shelf 3D object detector.

The two input modalities undergo distinct processing steps. LiDAR point clouds are preprocessed to extract dynamic object proposals (as detailed in 3.2), while camera images are encoded using the pre-trained DINOv2 encoder [39], yielding appearance feature maps (as detailed in 3.3). Subsequently, LiDAR points are projected onto the image plane and each point is augmented with its corresponding appearance feature vector based on the appearance feature map. These augmented point features are used to generate dynamic appearance prototypes (as detailed in 3.4), which help in discovering static mobile instances during training.

3.2. Dynamic object proposals generation

To extract the dynamic object proposals from LiDAR point clouds, the following processing steps are followed:

Ground-removal. We remove ground plane points from the LiDAR point clouds since they do not represent mobile objects. Following the method outlined in [29], we employ RANSAC [18] for plane fitting. All LiDAR points that are more than 30cm above this plane are classified as non-ground.

Ego-motion compensation. The ego-motion of the vehicle is compensated using the pose information provided for each frame in the Waymo Open Dataset [49].

Scene flow estimation. In the next step, we adopt the self-supervised Fast Neural Scene Flow [31] method to estimate the flow field between consecutive LiDAR frames. This method was chosen primarily due to its fast inference time. All LiDAR points having a flow value larger than $1m/s$ are classified as dynamic points, while others are classified as static points.

Spatial clustering. HDBSCAN [5] is used to form spatial clusters from the non-ground points, yielding O_t clusters for frame t. A cluster is deemed to be a dynamic cluster if at least 80% of its points are classified as dynamic points, resulting in O'_t dynamic spatial clusters.

Bounding-box fitting. Since training MobileClusterNet requires both, point classification labels as well as center offset values, we fit a 3D bounding box b_t to each spatial cluster in O'_t following the same procedure as in [29, 62]. Bounding boxes are filtered based on prior geometric assumptions to exclude implausible detections (further details provided in section 9). This yields a set of dynamic object proposals, $\beta_{dyn,t}$, for each LiDAR frame at time t.

3.3. Visual appearance embeddings

To extract semantic cues from the scene, we utilize image features generated by a DINOv2 ViT encoder [39]. Each camera image $I_{m,t}$ is encoded into a feature map

$F_{m,t} \in \mathbb{R}^{H_f \times W_f \times C_f}$, where H_f, W_f, and C_f represent the dimensions and number of channels in the feature map, respectively (further details provided in section 10).

LiDAR points from P_t are then projected onto the camera image planes using projection matrices, with each point being assigned a corresponding appearance vector $a_f \in \mathbb{R}^{C_f}$ from the image feature map. This results in a **points feature map** $F_{p,t} \in \mathbb{R}^{N \times C_f}$ for each point cloud.

For each pseudo bounding box $b'_{i,t}$ within the set of dynamic object proposals $\beta_{dyn,t}$, we extract the corresponding subset of the points feature map $F_{p',t} \in \mathbb{R}^{N_{dyn} \times C_f}$, where N_{dyn} denotes the dynamic points belonging to $b'_{i,t}$. By taking the mean of all appearance vectors in $F_{p',t}$, we get the visual appearance embedding $A_{i,t} \in \mathbb{R}^{C_f}$ for the pseudo bounding box.

3.4. Calculating appearance prototypes

We aggregate the visual appearance embeddings from all dynamic proposals β_{dyn} across the dataset to form a **dynamic feature map** $F_{dyn} \in \mathbb{R}^{T \times C_f}$, where T represents the total number of dynamic object proposals in the dataset. This consolidated feature map captures crucial information about the consistency and variability of appearance embeddings among dynamic clusters.

In the next step, we want to group the dynamic appearance embeddings into appearance-based pseudo-classes, where each pseudo-class belongs to the category of mobile objects. Similar to UNION, we use the K-Means clustering algorithm [35]. However, while UNION performs K-Means on appearance embeddings from all spatial clusters, we only consider dynamic clusters in calculating mobile object prototypes. Since the number of clusters K is not known beforehand, a silhouette score analysis is done to determine the optimal number of clusters, with an upper limit (max_num_K) set at 50. The centroids of the resulting K clusters serve as the **prototype appearance embeddings**.

To classify any new appearance embedding $A'_{i,t} \in \mathbb{R}^{C_f}$ as 'similar' to a prototype embedding, we define a scalar distance threshold d_k for each prototype $\rho_k \in \mathbb{R}^{C_f}$ using the Euclidean distance metric. Given that the K-Means algorithm is not designed to handle any outliers, we set each prototype's distance threshold as the mean distance of all cluster inlier embeddings from the cluster centroid (here, cluster refers to a K-Means cluster). To further prevent non-mobile instances from influencing the prototypes, we assign a distance threshold of zero to prototypes with fewer than 1% inliers (of all dynamic appearance embeddings). $A'_{i,t}$ is classified as 'similar to a prototype ρ_k if $\left\|\rho_k - A'_{i,t}\right\|_2 \leq d_k$. This process is summarized in Algorithm 1.

As a result, all LiDAR points belonging to a spatial cluster are classified as **mobile points** if the cluster's appearance embedding is close to one of the K appearance prototypes.

Algorithm 1 Calculate Prototypes and Distance Thresholds

Input: $F_{dyn} \in \mathbb{R}^{T \times C_f}$, max_num_K
Output prototypes, distance_thresholds

function CALCULATE_PROTOTYPES(F_{dyn}, max_num_K)
  scores ← []
  distance_thresholds ← []
  all_possible_Ks ← $\{2, \ldots,$ max_num_K$\}$

  for k in all_possible_Ks **do**
    clusterer ← KMEANS(k, F_{dyn}) ▷ *Fit KMeans*
    cluster_labels ← GET_LABELS(clusterer)
    score ← SILHOUETTE_SCORE(F_{dyn}, cluster_labels)
    append **score** to **scores**
  end for

  best_K ← all_possible_Ks$_i$: $i = argmax$(scores)
  best_clusterer ← KMEANS(best_K, F_{dyn}) ▷ *Fit Kmeans to data with n_clusters = best_K*

  cluster_labels_K ← GET_LABELS(best_clusterer)
  prototypes ← GET_CENTROIDS(best_clusterer)

  for this_label in unique(cluster_labels_K) **do**
    centroid ← prototypes[this_label]
    inliers ← GET_INLIERS(cluster_labels_K, F_{dyn})
    if length(inliers) $< 0.01 \cdot$ length(F_{dyn}) **then**
      append **0** to **distance_thresholds** ▷ *Discard clusters with less than 1% inliers*
    else
      dists ← EUCLIDEAN_DIST(centroid, inliers)
      mean_dist ← AVERAGE(dists)
      append **mean_dist** to **distance_thresholds**
    end if
  end for

  return prototypes, distance thresholds
end function

3.5. Training MobileClusterNet

Architecture. The architecture of MobileClusterNet, based on ClusterNet [54], features a transformer-based sparse voxel encoder [16] and dual heads for point classification and center offset voting. First, the encoder voxelizes the 3D LiDAR points to extract voxelized features from them. These features are then projected back onto the points, resulting in point features $f_i \in \mathbb{R}^{3+D}$, where D is the feature dimension. The classification head outputs a class label ($y_i \in 0, 1$) for each point, where 0 represents a non-mobile point and 1 represents a mobile point. The offset head outputs an offset distance vector $\Delta x_i \in \mathbb{R}^3$, indicating distance to its corresponding cluster center. Based on the cen-

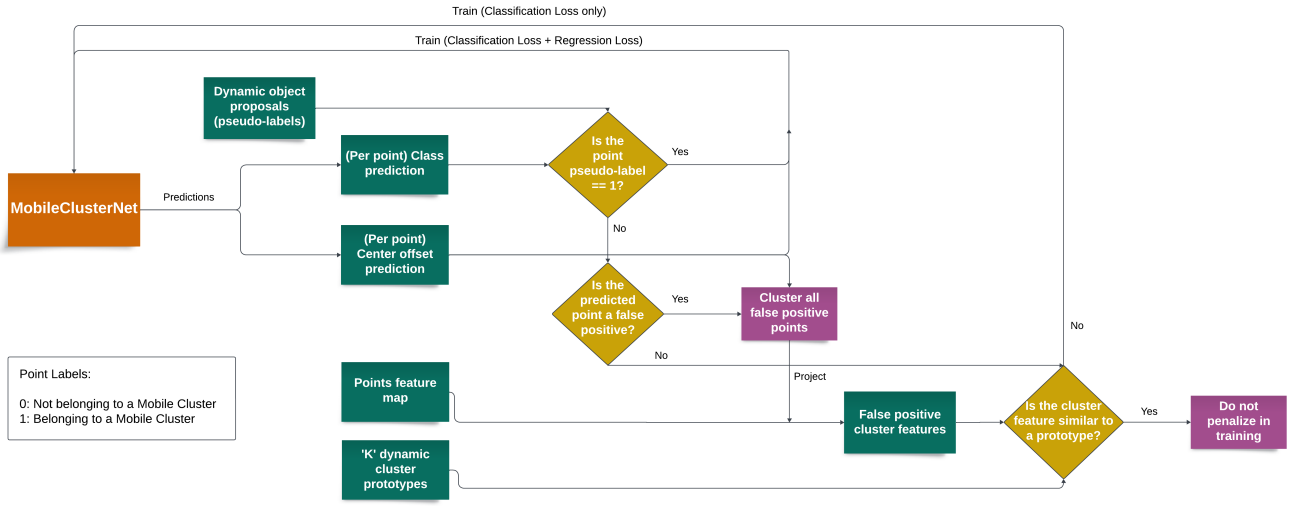


Figure 3. Deep-dive into the loss module of MobileClusterNet.

ter voting results, the predicted mobile points are grouped into spatial clusters using connected-components labeling [16] (further details are provided in section 8).

The pseudo-labels for training MobileClusterNet are derived from the dynamic object proposals β_{dyn}. The mobile classification targets are the inlier points in β_{dyn} and the regression targets are based on the centers of the bounding boxes in β_{dyn}. The loss function combines Focal loss [32] for point classification and L1 loss for offset predictions. The total loss is expressed as a weighted sum $L = L_{offset} + \lambda L_{class}$. The L1 loss term is calculated only for those points which are classified as mobile points in the pseudo-labels. In the following subsection, we discuss MobileClusterNet's training strategy which allows it to discover static instances of mobile objects during training.

Training strategy. An overview of the loss module of MobileClusterNet is shown in Figure 3. During the forward pass, the network predicts class labels and center-offset distances for each point. Points predicted as mobile ($y_i = 1$) but labeled as non-mobile in the pseudo-annotations ($y_{gt} = 0$) are termed as false positives. Points labelled as mobile in the pseudo-annotations contribute to both the offset loss (L_{offset}) and the classification loss (L_{class}).

Next, we segregate all false positive points from the point cloud $P_{fp,t}$ and cluster them based on their predicted center-offsets using connected-components labeling, resulting in a set of false-positive spatial clusters $\zeta_{fp,t}$. For each cluster $\zeta_i \in \zeta_{fp,t}$, we calculate its visual appearance embedding by averaging the appearance features of all points within the cluster, using the previously calculated points feature map (see 3.3). A cluster ζ_i is classified as a mobile cluster if its appearance embedding closely matches any of the K prototype appearance embeddings (refer to 3.4); otherwise, it is classified as non-mobile. All points within mobile clusters in $\zeta_{fp,t}$ are not penalized during training i.e the loss component from these points is set to zero. Conversely, points in non-mobile clusters influence only the classification loss (L_{class}).

This approach ensures that false positives which resemble dynamic mobile instances are treated as static mobile instances and excluded from contributing to the training loss, aiding in static object discovery.

4. Experiments

In this section, we specify our implementation setup, the used dataset, baselines and metrics as well as the our main results.

4.1. Dataset & metrics

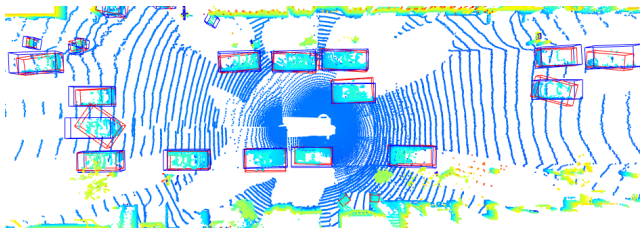
Dataset. Our experiments are conducted on the large and geographically diverse Waymo Open Dataset (WOD) [49] which comprises approximately 158k training and 40k validation frames annotated at 10Hz. To keep the compute resources manageable during the thesis, we subsample the frames at 1Hz, resulting in 10% of WOD which is used as our training set. All experiments are done class-agnostic (ie the semantic labels are mobile and non-mobile objects) and the provided annotations are not used anywhere except during evaluation.

Evaluation: We evaluate using the same protocol as in [3, 37], considering only the predictions and ground-truth within a $100m \times 40m$ BEV grid centered around the ego-vehicle.

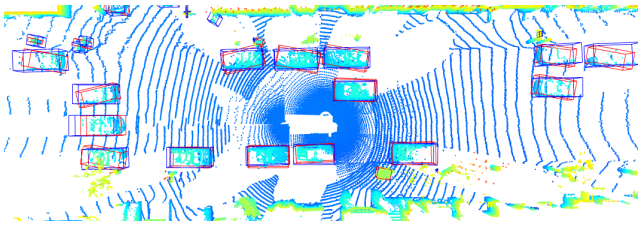
We train a CenterPoint detector [61] on the set of all bounding box annotations β_T, generated by MobileClus-

Table 1. Evaluation results on WOD Validation set. All experiments are conducted class-agnostic on a BEV grid $100m \times 40m$ around the vehicle. †: Results taken from [3]

Method	Annotations	Self-training	Temporal tracking	3D AP@0.4
Supervised				
MobileClusterNet (LiDAR only)	ground-truth	✗	✗	43.75
CenterPoint [61]	ground-truth	✗	✗	67.5
Unsupervised				
HDBSCAN [5]	-	✗	✗	3.25
MobileClusterNet (Ours)	Dynamic object proposals (3.2)	✗	✗	21.73
CenterPoint	HDBSCAN	✗	✗	3.08
CenterPoint	OYSTER [64]	✓	✓	8.4†
CenterPoint	LISO [3]	✗	✓	21.1†
CenterPoint	LISO [3]	✓	✓	30.8†
CenterPoint	MobileClusterNet (Ours)	✗	✗	26.16



(a) MobileClusterNet predictions.



(b) CenterPoint (trained with MobileClusterNet annotations) predictions.

Figure 4. Visualization of 3D object detection on WOD validation set. Blue boxes represent the ground truth annotations while red boxes represent the discovered ones.

terNet on the 10% WOD training set. The main metric for assessing performance is Average precision (AP), calculated on CenterPoint predictions on the WOD validation set. We use 3D AP since that is the default evaluation metric in WOD.

Additionally, we also directly evaluate the raw clustering performance of MobileClusterNet by calculating the AP on all bounding box annotations generated by MobileClusterNet on the WOD validation set.

Baselines: We compare the performance of CenterPoint when trained on MobileClusterNet pseudo-labels against pseudo-labels generated using the following unsupervised, class-agnostic methods: 1) HDBSCAN [5], 2) OYSTER [64], 3) LISO [3]. Additionally, we report the raw clustering performance of MobileClusterNet and HDBSCAN on WOD validation set. The supervised CenterPoint performance is stated as an upper limit to our method's potential.

We do not include MODEST [62], Najibi et al. [37], ClusterNet [54] and UNION [29] in our baselines. MODEST requires multiple traversals of the same location and they do not report performance of their method on WOD. Najibi et al. and ClusterNet follow a different evaluation criteria and did not make their code publicly available. UNION does not provide evaluation results or code compatibility on WOD.

4.2. Implementation details

We utilize the open-source framework MMDetection3D [10] for implementing MobileClusterNet and conducting all experiments involving the CenterPoint detector. All experiments are conducted on 3 NVIDIA V100 32 GiB GPUs. Further implementation details are provided in section 11.

4.3. Main results

The main evaluation results from our experiments are summarized in Table 1. Utilizing only 10% of WOD for training, MobileClusterNet demonstrates strong performance. It significantly outperforms HDBSCAN in raw clustering ability, giving higher quality spatial clusters which can be utilised in various downstream tasks, in an autonomous driving perception pipeline.

When comparing the performance of a CenterPoint detector trained with annotations from different methods, MobileClusterNet surpasses both OYSTER and LISO (without self-training). It is noteworthy that these methods train

on the full WOD training set and refine their object proposals with temporal tracking. While integrating temporal tracking was beyond the scope of this thesis' timeline, it presents a future opportunity for boosting performance. Additionally, MobileClusterNet closes the gap to the performance of LISO with self-training, while avoiding the computationally-expensive iterative re-training procedure.

Qualitative comparisons between annotations from MobileClusterNet and predictions by CenterPoint trained on these annotations (see Figure 4) show improvements in the shape and orientation of predicted bounding boxes by CenterPoint. This improvement is anticipated since MobileClusterNet is primarily designed for spatial clustering rather than (direct) 3D object detection.

4.4. Ablation study

We analyze the contribution of different sources of supervisory information—spatial cues, motion cues, and semantic cues—in the performance of MobileClusterNet. This analysis is performed considering both the direct annotations from MobileClusterNet and the predictions from CenterPoint trained on these annotations. The results of this ablation study are presented in Table 2.

The most significant improvement in MobileClusterNet's performance stems from the incorporation of visual appearance features during its training. This is in-line with the findings of UNION and supports our hypothesis that MobileClusterNet can effectively learn to identify static mobile objects during training, by incorporating visual features alongside motion information.

Table 2. Impact of different supervisory cues on MobileClusterNet performance. Best performance is highlighted in **bold**.

Annotations	3D AP@0.4
MobileClusterNet	
Spatial clusters (HDBSCAN [5])	5.66
+ Motion cues (3.2)	8.46
+ Appearance cues (3.3)	**21.73**
CenterPoint trained on MobileClusterNet annotations	
Spatial clusters	3.53
+ Motion cues	13.51
+ Appearance cues	**26.16**

5. Conclusion

In this thesis, we addressed the problem of unsupervised 3D object detection by introducing MobileClusterNet, a learnable framework for 3D spatial clustering. MobileClusterNet leverages a novel training strategy that integrates spatial cues, motion cues, and semantic cues, enabling it to identify all mobile objects in the scene. Our approach begins with extracting dynamic proposals using density-based clustering and scene flow estimation. Subsequently, camera images are encoded into feature maps to calculate visual appearance prototypes for dynamic objects. We supervise MobileClusterNet using these dynamic object proposals and appearance prototypes, by not penalizing false positive predictions that visually resemble the dynamic prototypes. This allows MobileClusterNet to discover static instances of mobile objects in the scene. Experimental results on the Waymo Open Dataset demonstrate the effectiveness of our learning-based approach to clustering; it outperforms some of the existing baselines using only 10% of the data and avoids the computationally expensive temporal tracking and self-training. We achieve an AP of 26.16 on the WOD validation set.

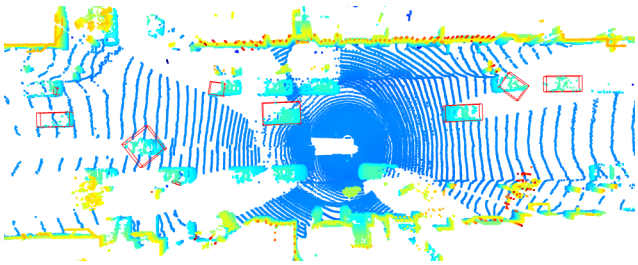


Figure 5. Visualization of the bounding boxes generated by MobileClusterNet when trained using dynamic object proposals only, showing multiple false negatives.

Limitations and Future work: One limitation to MobileClusterNet is that it currently does not distinguish between different object classes. A future extension could involve assigning pseudo-classes to MobileClusterNet's predictions based on the closest appearance prototype.

The calculation of prototypes within MobileClusterNet is done using K-Means, which struggles with noise and outliers. Therefore if non-mobile objects are present in the set of dynamic object proposals, they can influence prototype calculations. To mitigate this, we currently discard all appearance clusters comprising less than 1% of inliers, which risks overlooking rare but genuine mobile objects. Future work can explore using alternative methods for clustering apperance embeddings that can better handle outliers. Furthermore, the current implementation uses the K-Means centroids as prototypes, and there is potential to directly learn robust prototypes from the data, using self-supervised techniques like contrastive learning.

Additionally, the use of scene flow for dynamic object proposal extraction is computationally intensive, and the per-point flow information is not utilized. Future versions of MobileClusterNet might benefit from alternative methods to detect motion, such as integrating radar data, which directly provides the per-point radial velocity, or employing newer unsupervised techniques like M-Detector [56], which can rapidly classify LiDAR points as static or moving based on occlusion principles.

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012. 2

[2] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999. 2

[3] Stefan Baur, Frank Moosmann, and Andreas Geiger. Liso: Lidar-only self-supervised 3d object detection. *arXiv preprint arXiv:2403.07071*, 2024. 1, 2, 3, 6, 7

[4] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13126–13136, 2021. 3

[5] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 160–172. Springer, 2013. 1, 3, 4, 7, 8, 2

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 1

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1, 3

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[9] Nathaniel Chodosh, Deva Ramanan, and Simon Lucey. Re-evaluating lidar scene flow for autonomous driving. *arXiv preprint arXiv:2304.02150*, 2023. 8

[10] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 7

[11] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers, 2023. 5

[12] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[13] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, pages 226–231, 1996. 1, 3, 2

[14] Daniel J Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015. 1

[15] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8458–8468, 2022. 3, 6

[16] Lue Fan, Feng Wang, Naiyan Wang, and Zhao-Xiang Zhang. Fully sparse 3d object detection. *Advances in Neural Information Processing Systems*, 35:351–363, 2022. 3, 5, 6

[17] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59:167–181, 2004. 2

[18] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 4

[19] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901. 5

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 1

[21] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. *ACM Sigmod record*, 27(2):73–84, 1998. 1

[22] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Rock: A robust clustering algorithm for categorical attributes. *Information systems*, 25(5):345–366, 2000. 1

[23] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022. 1

[24] Alexander Hinneburg, Daniel A Keim, et al. *An efficient approach to clustering in large multimedia databases with noise*. Bibliothek der Universität Konstanz Konstanz, Germany, 1998. 2

[25] George Karypis, Eui-Hong Han, and Vipin Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *computer*, 32(8):68–75, 1999. 1

[26] L Kaufman and PJ Rousseeuw. Partitioning around medoids (program pam). finding groups in data: an introduction to cluster analysis. 1990; 344: 68–125. 2

[27] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009. 2

[28] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019. 7

[29] Ted Lentsch, Holger Caesar, and Dariu M Gavrila. Union: Unsupervised 3d object detection using object appearance-

based pseudo-classes. *Advances in Neural Information Processing Systems*, 2024. 1, 2, 4, 7

[30] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021. 3

[31] Xueqian Li, Jianqiao Zheng, Francesco Ferroni, Jhony Kaesemodel Pontes, and Simon Lucey. Fast neural scene flow. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9878–9890, 2023. 3, 4

[32] Tsung-Yi Lin, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2980–2988, 2017. 6

[33] Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017. 7

[34] Xinzhu Ma, Wanli Ouyang, Andrea Simonelli, and Elisa Ricci. 3d object detection from images for autonomous driving: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 1

[35] J Macqueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*, 1967. 5, 2

[36] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205, 2017. 4

[37] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion inspired unsupervised perception and prediction in autonomous driving. In *European Conference on Computer Vision*, pages 424–443. Springer, 2022. 3, 6, 7

[38] Raymond T. Ng and Jiawei Han. Clarans: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering*, 14(5):1003–1016, 2002. 2

[39] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1, 3, 4

[40] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016. 1

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4

[42] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9277–9286, 2019. 3

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016. 4

[44] Jenny Seidenschwarz, Aljosa Osep, Francesco Ferroni, Simon Lucey, and Laura Leal-Taixé. Semoli: What moves together belongs together. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14685–14694, 2024. 3

[45] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 3

[46] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. *arXiv preprint arXiv:2109.14279*, 2021. 1, 3

[47] Oriane Siméoni, Chloé Sekkat, Gilles Puy, Antonín Vobeckỳ, Éloi Zablocki, and Patrick Pérez. Unsupervised object localization: Observing the background to discover objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3176–3186, 2023. 1

[48] Andrew Stein, Derek Hoiem, and Martial Hebert. Learning to find object boundaries using motion cues. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 2

[49] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 2, 4, 6, 1

[50] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104: 154–171, 2013. 2

[51] Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. Freesolo: Learning to segment objects without annotations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14176–14186, 2022. 3

[52] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3124–3134, 2023. 3

[53] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020. 1

[54] Yuqi Wang, Yuntao Chen, and Zhao-Xiang Zhang. 4d unsupervised object discovery. *Advances in Neural Information Processing Systems*, 35:35563–35575, 2022. 1, 2, 4, 5, 7

[55] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963. 1

[56] Huajie Wu, Yihang Li, Wei Xu, Fanze Kong, and Fu Zhang. Moving event detection from lidar point streams. *nature communications*, 15(1):345, 2024. 8

[57] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487. PMLR, 2016. 3

[58] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, pages 574–591. Springer, 2020. 1

[59] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 6

[60] Junbo Yin, Dingfu Zhou, Liangjun Zhang, Jin Fang, Cheng-Zhong Xu, Jianbing Shen, and Wenguan Wang. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In *European conference on computer vision*, pages 17–33. Springer, 2022. 1

[61] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 6, 7

[62] Yurong You, Katie Luo, Cheng Perng Phoo, Wei-Lun Chao, Wen Sun, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Learning to detect mobile objects from lidar scans without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1130–1140, 2022. 1, 2, 3, 4, 7

[63] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pages 12310–12320. PMLR, 2021. 1

[64] Lunjun Zhang, Anqi Joyce Yang, Yuwen Xiong, Sergio Casas, Bin Yang, Mengye Ren, and Raquel Urtasun. Towards unsupervised object detection from lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9317–9328, 2023. 1, 2, 4, 7

[65] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 649–666. Springer, 2016. 1

[66] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996. 1

MobileClusterNet: Unsupervised Learnable Clustering of Mobile 3D Objects

Supplementary Material

6. Self-supervised methods in literature

Overview. Self-supervised learning aims to learn good representations of the data by training on a pretext task that doesn't require annotations, with subsequent fine-tuning on actual tasks to enhance generalization. There are 2 main categories of self-supervised architectures: generative architectures and joint embedding architectures. Generative architectures generate reconstructions from latent representations or corrupted versions of the input, using pretext tasks like image colorization [65], image inpainting [40], and masked image modeling [23]. Conversely, joint embedding architectures learn to produce consistent embeddings across different views of the same input, learning features robust to distortions and invariant to data augmentations. A challenge involved with using joint embedding networks is preventing informational collapse, where all inputs are mapped to a constant embedding. This is achieved via similarity maximization techniques (e.g. contrastive learning [8], clustering [6], distillation [20]) and redundancy reduction techniques (information maximization [63]).

Self-supervised vision transformers. Vision transformers (ViTs) [12] have been a breakthrough architecture in 2D computer vision tasks due to their ability to scale effectively with large input datasets. ViTs operate by dividing images into patches and processing them through self-attention mechanisms. In DINO [7] the authors use a self-distillation approach to train a ViT. This results in learnt features that contain explicit information about image semantic segmentation. These learned representations also exhibit inherent properties such as robustness to image perturbations and adaptability to various downstream tasks. DINOv2 [39] further builds upon DINO by introducing refinements that improve model performance and feature representation. These include optimized training strategies and architectural adjustments, which lead to more distinct and interpretable visual features (see Figure 6).

Challenges in the 3D domain. In contrast to 2D images, applying self-supervised learning techniques to the 3D domain poses some unique challenges. Point cloud sequences are typically sparse and not object centric; that is, objects are of much smaller sizes and unevenly distributed across a wide range. While this research area is comparatively unexplored, [58], [60] look at self-supervised pre-training in LiDAR point clouds.

Figure 6. Visualization of semantic segmentation results from using a pre-trained DINOv2 ViT [39] on a sample image from WOD [49] shows that the features from DINOv2 contain information about the semantic classes within an image.

7. Spatial clustering methods in literature

7.1. Traditional clustering methods

Hierarchical clustering. Hierarchical clustering methods organize data by creating clusters either through a bottom-up or a top-down approach. These methods can be categorized into agglomerative and divisive hierarchical clustering [55]. Agglomerative clustering, a bottom-up approach, begins by treating each object as its own cluster and then progressively merges these atomic clusters into larger ones until all objects are in a single cluster or until pre-specified criteria are met. Conversely, divisive clustering, a top-down approach, starts with all objects in one cluster and divides this cluster into smaller ones until each object is in its own cluster or until pre-determined conditions are fulfilled. The linkage between 2 clusters can be based on the minimum nearest neighbour distance between the cluster members (single-linkage clustering), the maximum furthest neighbour distance between them (complete-linkage clustering) or the average distance between all cluster members (average-linkage clustering). Some examples of hierarchical clustering methods include BIRCH [66], CURE [21], ROCK [22] and chameleon [25].

The downside of hierarchical clustering methods is that they are not robust to noise and outliers, and once two members of the clusters are linked, they cannot move to other clusters in the hierarchy. The computational complexity of hierarchical clustering methods is $O(N^2)$ and they cannot be used efficiently in large datasets.

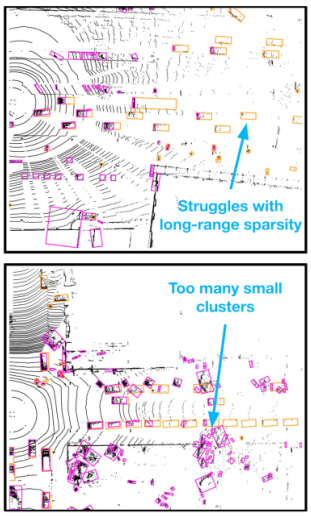
Partitional clustering. Partitional clustering, in contrast to hierarchical clustering, involves grouping data into a predefined number of clusters, K, without building a hierarchical structure. This method optimizes a criterion, typi-

cally the Euclidean distance, to minimize the distance between data points and their designated cluster centers. K-Means [35] is a prominent example of partitional clustering, with some other examples being PAM [26], CLARA [27], CLARANS [38]. In K-Means, the process begins by initializing K centroids, each representing the center of a cluster. The algorithm then iteratively assigns each data point to the nearest centroid, thereby forming clusters. After all points have been assigned, the centroids are recalculated based on the mean of the points in each cluster. This process repeats until the centroids stabilize, indicating that the clusters are as compact and distinct from each other as possible. This method is particularly effective for large datasets where forming a hierarchy is computationally expensive. However, one of the major downsides of K-Means clustering is that the number of clusters needs to be defined beforehand, which is not practical when prior knowledge of the dataset's structure is unknown, which is true in general for point cloud data. Additionally, K-Means cannot handle noise or outliers in the dataset, and every point is assigned to a cluster center.

Density-based clustering. Density-based clustering methods were previously discussed in 2.2.1, and are particularly suitable for 3D point cloud data due to their ability to identify clusters based on the density of points, which aligns well with how objects appear in spatial (LiDAR) datasets. Some additional commonly used density-based algorithms include OPTICS [2] and DENCLUE [24]. The OPTICS (Ordering Points To Identify the Clustering Structure) clustering algorithm is an extension of DBSCAN [13] designed to overcome its sensitivity to the radius parameter that defines the neighborhood size. OPTICS does not produce a clustering of a data set explicitly; instead, it produces an augmented ordering of the data which sorts the data points such that spatially closest points become neighbors in the ordering. Each point is also assigned a reachability distance, which is an indication of how far away it is from the nearest cluster. Points within a cluster have smaller reachability distances compared to those that are lying between clusters. OPTICS can identify clusters of different densities and is particularly useful in scenarios where the contrast between different cluster densities is significant. DENCLUE (DENsity-based CLUstEring) on the other hand uses a density distribution function to model the overall point density of the data space. It defines clusters based on the areas of higher density than the threshold and locates cluster centers at the maximum density function values. DENCLUE is effective in dealing with large data sets and can handle outliers well. However, it requires a prior understanding of the underlying data density distribution, which is a challenge in the autonomous driving LiDAR datasets.

While density-based clustering algorithms have been frequently used for clustering 3D point cloud data, they do have certain limitations. They can struggle with datasets that have varying densities, which are common in point clouds due to sensor noise, varying object distances, or (self) occlusions. This variability can lead to fragmented clusters or the merging of distinct objects into a single cluster if not tuned properly. Additionally, these methods often have several hyperparameters, such as the neighborhood search radius and the minimum number of points required to form a dense region, which can be challenging to set without domain knowledge or a trial-and-error process. This complexity can make them less intuitive and harder to optimize. Some of these limitations are visualized in Figure 7.

(a) Under-segmentation of a scene in WOD [49] when clustered using HDBSCAN [5].



(b) Qualitative results with DBSCAN [13] clustering. Image taken from [64].

Figure 7. Visualization of some challenges associated with density-based clustering on 3D point cloud data.

Graph-based clustering. Graph-based clustering methods represent a group of clustering techniques that leverage graph-theory principles to detect the intrinsic structures and patterns within the data. These methods construct a graph where nodes represent data points and edges represent the proximity or similarity between them. The goal is

to partition this graph into clusters based on the connectivity or the edge weights of the graph. One prominent example of graph-based clustering is the spectral clustering method. Spectral clustering transforms clustering into a graph partitioning problem, focusing on cutting the graph into pieces such that the connections (edges) between different groups (clusters) are weak while connections within a group are strong. Notably, works like normalized cuts (NCut) [45] have derived many fundamental ideas from the spectral graph theory, which in turn has inspired the development of more recent methods like CutLER [52]. The main disadvantage of graph-based clustering methods is that their time complexity increases dramatically with the increasing of graph complexity.

7.2. Learning-based clustering methods

Unlike traditional clustering techniques which operate on predefined distance metrics, learning-based clustering involves training a machine learning model to learn an embedding space where clustering objectives can be optimized directly. One common approach is the use of autoencoders, where the network learns to compress data into a lower-dimensional space and then reconstruct it. The learned compressed features, which capture the essential characteristics of the data, are then used for clustering. Methods like Deep Embedded Clustering (DEC) [57] further refine this approach by jointly optimizing the feature space and clustering objective, leading to improved clustering performance.

SeMoLi [44] is a semi-supervised method for 3D object detection that utilizes a message parsing network (MPN) trained with a small fraction of dataset annotations, to generate spatial cluster proposals on the entire dataset. These clusters are then used to fit bounding boxes, creating pseudo-labels suitable for training any standard 3D object detector. At the core of SeMoLi is a graph-based clustering algorithm that processes node and edge features to segment the graph into connected components, each representing a cluster instance. This graph is constructed using only moving points, with node connectivity being constrained to its k-nearest neighbours. The node and edge features are initialised based on spatial proximity and motion cues. Training the message parsing network involves iteratively updating the node and edge embeddings, and calculating an edge score based on the sigmoid function. During inference, all edges with a score lower than a threshold are ignored, and the remaining graph represents a set of connected components instances. SeMoLi is not a fully unsupervised approach, and requires some annotations to be able to train the message parsing network. In contrast, training MobileClusterNet requires no labels.

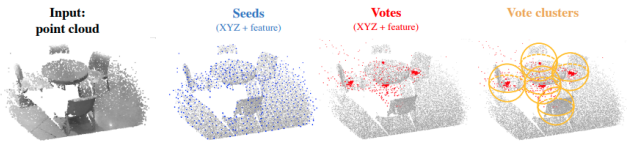


Figure 8. Illustration of the voting module in VoteNet [42]. The input point cloud is passed through a backbone network which subsamples the point cloud into seed points, and extracts point features for these points. Each seed then generates a vote using the voting module, and all votes are grouped together to form vote clusters. Image taken from [42].

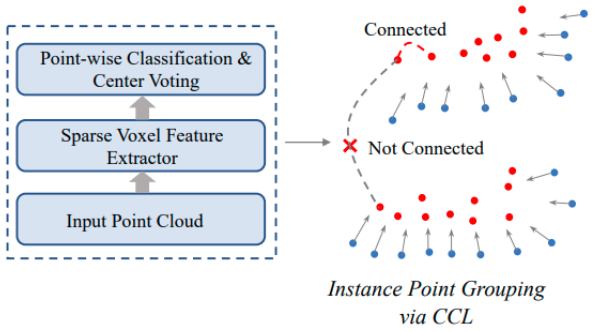


Figure 9. Illustration of connected components clustering. Image taken from [16].

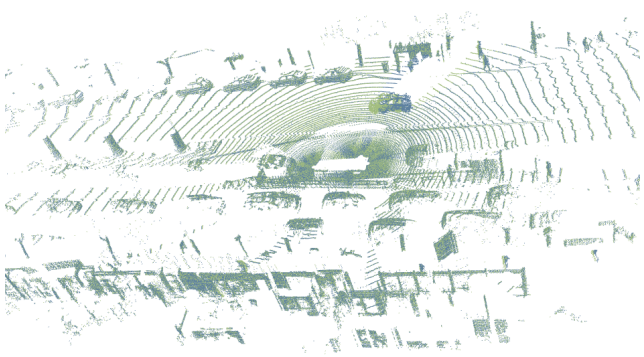
8. MobileClusterNet architecture

The core architecture of MobileClusterNet is discussed in more detail below.

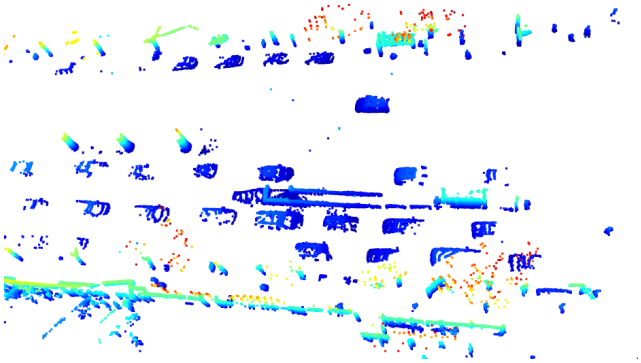
Encoder backbone. MobileClusterNet first voxelizes the input point cloud and then utilises the Single-stride Sparse Transformer (SST) [15] to extract voxel features. These encoded voxel features are then scattered back to the original point locations, and are augmented with the offset values (x, y, z) of the points from their corresponding voxel centers, resulting in point features.

Classification and voting heads. After the point features have been extracted from the input point cloud, they are passed onto 2 heads: One for classifying the semantic label of the LiDAR point (mobile or non-mobile) and the other for center voting. The center voting head is inspired from the voting module in VoteNet (shown in Figure 8) and it predicts, for each mobile classified point, an offset distance (x, y, z) from the point to its corresponding cluster center.

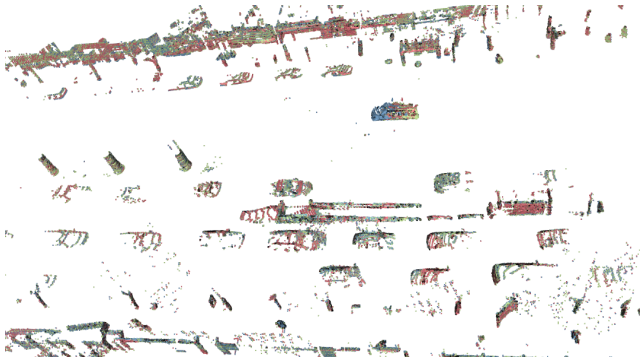
Connected components clustering. Based on the classification results, we first filter out the non-mobile (background) points from the point cloud. For the remaining mobile (foreground) points, we use the predicted offset distances to compute the voted cluster centers. Next, we build a graph using the voted cluster centers as graph vertices. If



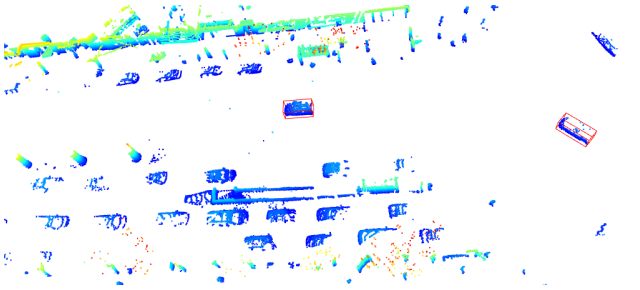
(a) Visualisation of point clouds after ego-motion compensation.



(b) Visualisation of point cloud after removing ground points.



(c) Results from scene flow estimation. Blue indicates the ground truth point cloud at time t, green indicates the ground truth point cloud at time $t-1$, red is the predicted point cloud at time $t-1$ (Reverse flow).



(d) Dynamic object proposals extracted from the given point cloud. Annotations are shown in red.

Figure 10. Visualising different steps involved in dynamic object proposal extraction.

the spatial (euclidean) distance between 2 vertices is less than a pre-determined threshold, then they are connected with an edge. All connected components in this graph (shown in Figure 9) are then viewed as cluster instances, and they share a common cluster ID.

9. Dynamic proposals extraction

A visualisation of some of the intermediate results is shown in Figure 10. The implementation details of the dynamic proposals extraction workflow are discussed as follows:

Ground plane fitting. For estimating the ground plane, we only consider the points within the a Z-axis range of $[-1m, 1m]$ and no more than $75m$ from the ego-vehicle in the XY-plane, termed as ground candidate points. We use RANSAC [18] for plane fitting, and set minimum number of chosen samples as 250, the inlier threshold as $10cm$ and the maximum number of trials as 20. We call this estimated plane as the global plane. In the next step, the global plane area is divided into 8 sub-regions, each with equal field of view (FoV). A more refined plane fitting is then performed within each sub-region using RANSAC, but with a tightened inlier threshold of $5cm$, while maintaining other parameters. This approach allows for an initial coarse estimation of the ground plane followed by more precise fitting in the sub-regions, thereby capturing the local variations across them. Our implementation uses the RANSACRegressor method from sklearn [41].

Scene flow estimation. We estimate the scene flow on the full, ego-motion compensated point clouds. We run the optimization in FNSF [31] for 5000 iterations with a batch size of 1 and a learning rate of 0.001. We use a grid cell size of $0.1m$, and employ early stopping with a patience of 125 iterations and a minimum delta of 0.00025 for termination. Our model is the neural prior, with 8 layers and 128 hidden units per layer.

Spatial clustering. We use the hdbscan python library [36] for extracting initial object proposals, with a minimum cluster size threshold of 16 points and a cluster selection epsilon value of $0.5m$.

Bounding box filtering Similar to UNION [29], we discard all the fitted bounding-boxes with $l > 20m$, $w > 6m$, $h < 0.25m$, $l/w > 8$, $lw < 0.25m^2$ as well as any box lying more than $0.75m$ above the ground plane.

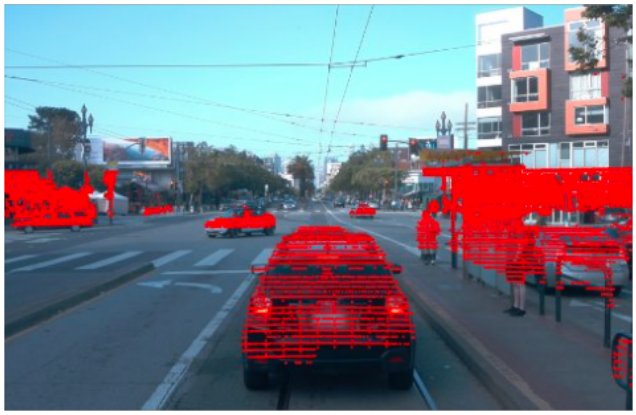


Figure 11. LiDAR point cloud projection to the front camera image in WOD. Ground points have been removed for visualisation.

10. Appearance prototypes

10.1. Point feature map calculation

Encoding camera features. Since the camera images in Waymo have different resolutions—1920 $\times$ 1040 for side left and side right cameras versus 1920 $\times$ 1280 for front, front left, and front right cameras—we first pad all camera images to the same resolution (1920 $\times$ 1280) and then encode them with the DINOv2 vision foundation model. We use the ViT-L backbone with registers [11] and a stride of 14. The resulting feature map is a 3D tensor of size 91 $\times$ 137 $\times$ 1024.

Dimensionality reduction. To enhance computational efficiency, we reduce the feature channel dimensionality from 1024 to 64 using Principal Component Analysis (PCA) [19]. A PCA model is trained on 79 diverse frames from the Waymo Open Dataset (WOD) training set to extract 64 principal components. This PCA model is then used to transform our appearance feature map to a size 91 $\times$ 137 $\times$ 64. We use the PCA implementation in sklearn with the default parameters.

Point features. After projecting the LiDAR point cloud to the 5 camera image planes, each point is associated with the closest camera plane based on its distance to the image plane centers. We then assign each LiDAR point to the corresponding appearance feature map from that camera. Bilinear interpolation is utilized to compute a feature vector for each point, resulting in a points feature map of dimensions $N \times 64$, where N is the number of points.

10.2. Prototypes calculation

Silhouette analysis. The silhouette analysis is a method used to assess the quality of clustering in a dataset, when the ground truth labels are not known. This metric calculates the silhouette score for each point in the dataset, which measures how similar that point is to points in its own cluster compared to points in other clusters. The score ranges from -1 to 1, where a high value indicates that the point is well matched to its own cluster and poorly matched to neighboring clusters, thus suggesting good clustering.

The silhouette score is calculated based on two distances: the average distance between a sample and all other members in the same cluster (a) and the average distance between a sample and all members in the next nearest cluster (b). The silhouette score is then defined as $\frac{(b-a)}{max(a,b)}$. Averaging these scores across across the entire dataset provides a single score that describes the overall clustering effectiveness.

Typically, well-separated clusters with dense cores yield higher average silhouette scores. Figure 12 shows the plot of silhouette scores against the number of clusters for our appearance clustering. Looking at the graph, we can tell that the silhouette score significantly drops when we try to cluster the data into 8 or more clusters.

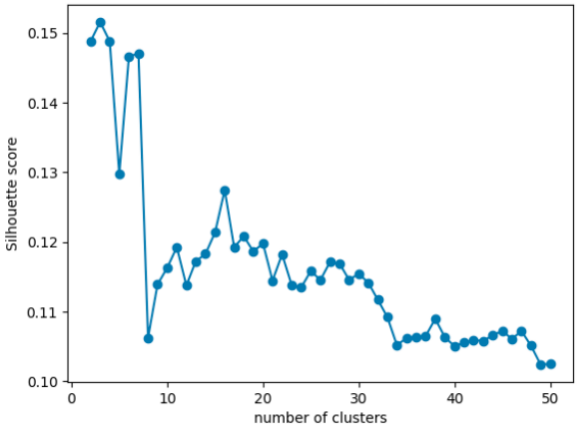


Figure 12. Silhouette score analysis.

Prototypes. For calculating prototypes, we employ the K-Means implementation from sklearn. Based on the silhouette score analysis, we determined the optimal number of appearance clusters to be 3 (refer to Figure 12), and all other K-Means parameters were left at their default settings. These include a maximum of 300 iterations and an initialization method of 'k-means++' which enhances the likelihood of convergence.

To better visualize and understand the semantic information captured in the prototype embeddings, we perform a qualitative analysis of object proposals that were classified as 'similar' to each prototype. We extracted spatial clustering proposals from selected scenes using HDBSCAN and computed an appearance embedding $\in \mathbb{R}^{64}$ for each spatial cluster. We then mapped these clusters to their closest prototypes based on Euclidean distance in the embedding space. Our findings, shown in Figure 13, revealed

(a) Prototype 1 - scene 1. (b) Prototype 1 - scene 2. (c) Prototype 1 - scene 3.

(d) Prototype 2 - scene 1. (e) Prototype 2 - scene 2. (f) Prototype 2 - scene 3.

(g) Prototype 3 - scene 1. (h) Prototype 3 - scene 2. (i) Prototype 3 - scene 3.

Figure 13. Qualitative visualisation of the semantic features captured by the prototype embeddings. The projected cluster points (as shown in red) within the images correspond to a spatial cluster that was semantically similar to one of the prototype embeddings. The rows in the figure correspond to the 3 prototype embeddings while the columns correspond to 3 different scenes chosen from WOD.

distinct semantic characteristics associated with each prototype. The first prototype predominantly corresponded to pedestrian-related features, while the second and third prototypes captured aspects related to vehicles, with each playing a complementary role.

10.3. Training strategy intuition

The key distinction in the training strategy of MobileClusterNet, as compared to the LiDAR-only ClusterNet, lies in the treatment of false positive detections during training. While both frameworks use dynamic object proposals as training targets, all false positive predictions by ClusterNet (with respect to the training targets) contribute to the focal loss term. This approach penalizes the detections of static mobile objects, such as parked vehicles during training, that are not included in the set of dynamic object proposals but are otherwise valid detections.

In contrast, MobileClusterNet incorporates appearance prototypes to evaluate false positive predictions, identifying those that are semantically similar to the calculated appearance prototypes. If a detection, such as a parked car, closely matches an appearance prototype, it is not penalized; that is, this prediction is ignored in the focal loss term. Since localization information is only limited to dynamic object proposals, static mobile objects also do not contribute to the center offset regression loss. By not penalizing the detection of static mobile objects during training, MobileClusterNet enhances its ability to recognize these instances during inference.

11. Training details

11.1. MobileClusterNet

The network is trained on three NVIDIA V100 32 GiB GPUs, utilizing a batch size of two data samples per GPU. Each data sample includes one LiDAR point cloud covering a range of $[-74.88, 74.88]$ in the X and Y axes and $[-2, 4]$ in the Z-axis, along with five camera images of size $1280 \times 1920 \times 3$. The voxel size used is $(0.32m, 0.32m, 6m)$.

The model uses the Dynamic Voxel Feature extractor (DynamicVFE) from SECOND [59] for feature extraction coupled with an SST [15] backbone. In the loss module, both the focal loss and the L1 loss are equally weighted to balance classification accuracy with bounding box precision. The classification head uses a segmentation score threshold of 0.4 to classify points as foreground, while the connected components labeling module uses a distance threshold of $0.4m$ between vertices to connect them with an

edge.

We train MobileClusterNet for 12 epochs, with the first 3 epochs conducted without appearance features to allow for an initial 'warm-up' phase, accelerating the training process by reducing computational demand early on. This warm-up phase incorporates data augmentations like random flipping, rotation, scaling and translation of the bounding box targets. The training utilizes the AdamW optimizer [33] with a cosine learning rate scheduler. The learning rate is initialized at 10^{-5} and the maximum learning rate is 0.001 while the weight decay is 0.05.

11.2. CenterPoint

CenterPoint is trained using the same GPU setup with a batch size of 8 LiDAR point clouds per GPU. The point cloud dimensions and voxel size remain consistent with MobileClusterNet's settings. We use the Pillar Feature Net [28] as the voxel encoder, and to prune the predictions, we integrate non-maximum suppression (NMS) with a 2D BEV IoU threshold of 0.1. CenterPoint is trained over 20 epochs utilizing the Adam optimizer and a cyclic learning rate schedule with a peak rate of 0.003. The training process also includes the same data augmentation strategies as those applied to MobileClusterNet.

12. Qualitative results and Discussion

12.1. Training with spatial clustering annotations

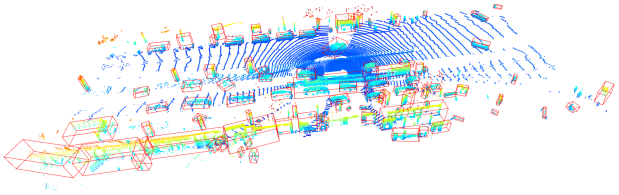
As hypothesized, MobileClusterNet underperforms when trained only with spatial clustering proposals from HDBSCAN, leading to a significant number of false positive predictions involving both mobile and non-mobile objects (as shown in Figure 14a).

Surprisingly, when these training set annotations were used to train the CenterPoint detector, there was a drop in performance. This can be visualised with Figure 14b. In addition to too many false positive detections, the orientation estimation of CenterPoint has collapsed due to many random targets, and the network predicts a constant orientation for all detected objects.

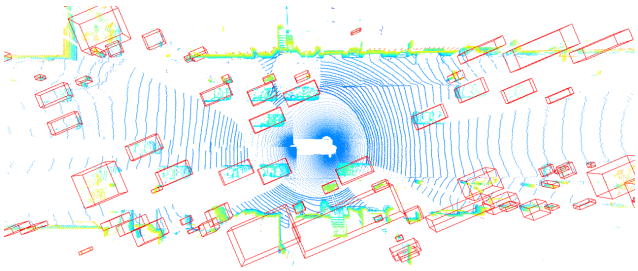
12.2. Training with dynamic object proposals

When MobileClusterNet is trained with only dynamic object proposals, it struggles with many missed detections; that is, there is a large number of false negatives. This is due to the detector generalization being hampered as it is penalized for discovering static mobile objects during training.

We visualize the annotations generated by MobileClusterNet trained with dynamic object proposals on a frame in WOD training set (shown in Figure 15a). While these annotations contain more training targets than the initial dynamic object proposals, the performance of CenterPoint still suffers from the unlabelled static mobile objects. It can be seen in Figure 15b that CenterPoint even generates some false predictions in the scene.

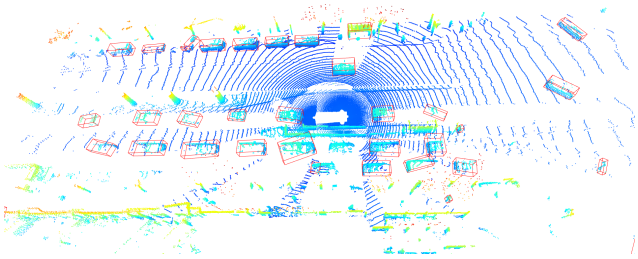


(a) Predictions from MobileClusterNet.

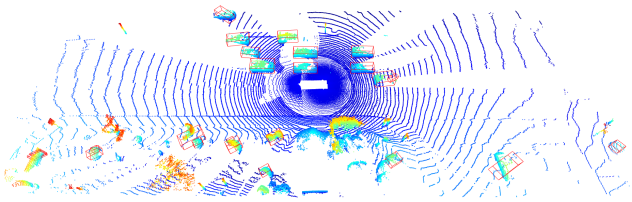


(b) Predictions from CenterPoint.

Figure 14. Visualization of predictions from MobileClusterNet trained using spatial clusters (HDBSCAN) only and CenterPoint when trained using annotations from MobileClusterNet trained using spatial clusters only.



(a) Predictions from MobileClusterNet.



(b) Predictions from CenterPoint.

Figure 15. Visualization of predictions from MobileClusterNet trained using spatial clusters (HDBSCAN) only and CenterPoint when trained using annotations from MobileClusterNet trained using spatial clusters only.

12.3. Training with appearance prototypes

As already highlighted in subsection 4.4, MobileClusterNet, when trained with appearance features, shows a significant performance boost—approximately 2.5 times better than the baseline ClusterNet implementation, which only utilizes dynamic object proposals. As demonstrated by Figure 13, the calculated appearance prototypes capture information related to the semantics of mobile objects, which helps MobileClusterNet in learning to accurately cluster static objects that are visually similar to dynamic mobile objects. Furthermore, Figure 16 demonstrates the improved quality of annotations produced by MobileClusterNet when trained with these appearance prototypes, showcasing its robust generalization across all types of mobile objects within the scene.

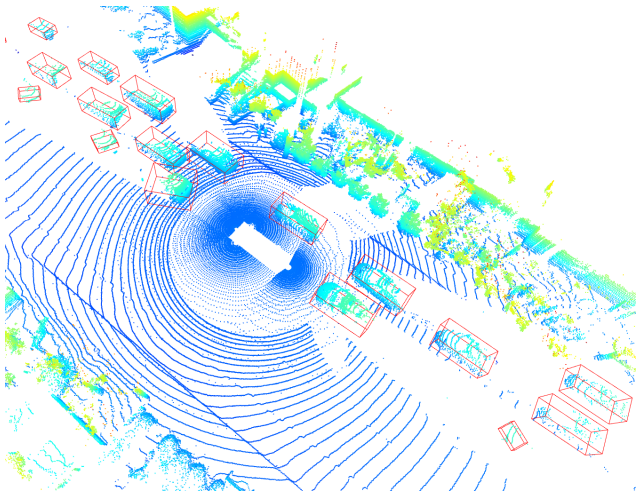


Figure 16. Visualisation of annotations generated by MobileClusterNet on a frame from the WOD training set.

Scope for improvement. One way to further boost the performance of MobileClusterNet is by improving the quality of its initial training targets—the dynamic object proposals. The quality of spatial clustering can be improved by aggregating multiple LiDAR sweeps across a temporal window, which could densify the point cloud. Additionally, our current scene flow estimation method was chosen for speed rather than accuracy; adopting a more robust technique, such as the one suggested by Chodosh et al. [9], could further refine the quality of dynamic object proposals. Finally, leveraging a larger proportion of the WOD training set could also contribute to performance improvements by providing a richer diversity of training data.