4D Unsupervised Object Discovery

Anonymous Author(s)
Affiliation
Address
email

Abstract

Object discovery is a core task in computer vision. While tremendous success has progressed in supervised object detection with vast annotated data, its unsupervised counterpart remains largely unexplored. With the growth of data volume, the expensive cost of annotations is the major limitation hindering further study. Therefore, discovering the objects without annotations has great significance. However, the task seems impractical on still-image or point cloud alone due to the lack of discriminative information. Previous studies overlook the crucial *temporal* information and constraints naturally behind multi-modality. In this paper, we propose *4D unsupervised object discovery*, jointly discovering objects from 4D data——3D point clouds and 2D RGB images with temporal information. We present the first practical approach for this task by proposing a *ClusterNet* on 3D point clouds, which is *joint iterative optimizing* with a 2D localization network. Extensive experiments on the large-scale Waymo Open Dataset suggest that the localization network and ClusterNet achieve competitive performance on class-agnostic 2D object detection and 3D instance segmentation, bridging the gap between unsupervised methods and full supervision. Code will be released.

1 Introduction

Computer vision researchers have been trying to locate objects in complex scenes without human annotations for a long time. Current supervised methods achieve remarkable performance on 2D detection [25, 14, 24, 32, 5] and 3D detection [41, 22, 28, 27, 39], benefiting from high-capacity models and massive annotated data, but tend to fail for scenarios that lack training data. Therefore, unsupervised discovery is critical for relieving the demand for training labels in deep networks, where raw data are infinite and cheap, but annotations are limited and expensive.

However, unsupervised object discovery in complex scenes used to believe impractical. Only a few studies pay attention to this field and achieve limited performance in simple scenarios, far inferior to the supervised model. Recent methods [29, 34] discover objects on 2D still-image utilizing the self-supervised learning [6, 35] to distinguish primary objects from the background, then fine-tune a localization network using the pseudo label. Although these methods have achieved superior results to previous methods [33, 2, 42], the performance still lags a huge gap to supervised models. Furthermore, such methods guided by contrastive learning have the problem of distinguishing different instances within the same category. Alternatively, the 3D point cloud can be decomposed into different class-agnostic instances based on proximity cues [4, 3], but it is hard to identify the foreground. These problems can be mitigated by the complementary characteristics of 2D RGB images and 3D point clouds. The point cloud data provides accurate location information, while the RGB data contains rich texture and color information. Therefore, [31] proposed to aid unsupervised object detection with LiDAR clues, but it also depends on the performance of the self-supervised model [13] to identify foreground objects. In summary, *all previous methods rely heavily on the self-supervised learning models and overlook the important information from time dimension*.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

To explore this field further, we propose a new task named *4D unsupervised object discovery*, *discovering objects utilizing 4D data——3D point clouds and 2D RGB images with temporal information* [21]. The task needs to joint discover objects on RGB images as *2D Object Detection* and 3D point clouds as *3D Instance Segmentation*. Thanks to the popularization of LiDAR sensors in autonomous driving and consumer electronics (e.g., iPad Pro), such 4D data has become much more readily available, indicating the great potential of this task for general application.

In this paper, we present the first practical solution for this task. We proposed a *joint iterative optimization* for *ClusterNet* on 3D point cloud and localization network in RGB images, utilizing the spatio-temporal consistency from multi-modality. Specifically, the ClusterNet was trained by motion cues initially, which can obtain from the consecutive point clouds using temporal information. The 3D instance segmentation output by ClusterNet can be further projected to the 2D image as supervision to train the localization network. Conversely, 2D detection can also help to refine the 3D instance segmentation. In this way, the 2D localization network and 3D ClusterNet can benefit from each other. Temporal information could serve as a constraint in the optimization.

Our main contributions are as follows: (1) we proposed a new task termed *4D Unsupervised Object Discovery*, aiming at jointly discovering the objects in the 2D image and 3D Point Cloud without manual annotations; (2) we proposed a *ClusterNet* on 3D point clouds, which is jointly iterative optimized with a 2D localization network. (3) Experiments on the Waymo Open Dataset [30] suggest the feasibility of the task and the effectiveness of our approach. We outperform the state-of-the-art unsupervised object discovery by a significant margin, superior to supervised methods with limited annotations, and even comparable to supervised methods with full annotations.

2 Related work

2.1 Supervised object detection

Object detection from Image. 2D detection has made great progress in recent years. Two-stage methods represented by the RCNN family [12, 25, 14] extract region proposals first and refine them with deep neural networks. One-stage methods like YOLO [24], SSD [20] and RetinaNet [19] predict the class-wise bounding box in one-shot based on the anchors. FCOS [32] and CenterNet [40] further detect objects without predefined anchors.

Object detection from Point Cloud. LiDAR-based 3D object detection develops rapidly along with autonomous driving. Point-based methods [23, 22, 38] directly estimated 3D bounding boxes from point clouds. The computing efficiency is affected by the number of points, so these methods are usually suitable for indoor scenes. Voxel-based methods [41, 37, 27] operate on the 3D voxelized point cloud are capable for large outdoor scenes. However, voxel resolution can greatly affect performance but is limited by computational constraints. Second [37] and PVRCNN [27] further apply sparse 3D convolutions to reduce compute. CenterPoint [39] extends the idea of anchor-free from 2D detection and proposes a center-based representation of bird-eye view (BEV).

2.2 Unsupervised object discovery

Bottom-up clustering. Clustering methods combine similar elements based on proximity cues, applicable to point clouds and image data. Selective Search [33], MCG [2] and Edge Box [42] can propose a large number of candidate objects with the help of appearance cues, but it is difficult to identify objects from the background. Similarly, point cloud data can decompose into distinct segments according to density-based methods [9, 4, 26] but is unable to determine which is foreground.

Top-down learning. Recently, self-supervised learning [7, 13, 6, 35] are capable to learn discriminate features without labels. Therefore, many methods attempt to introduce such properties to discriminate foreground objects without manual annotations. LOST [29] utilized a pre-trained DINO [6] to extract primary object attention from the background as the pseudo label and then finetuned an object detector. FreeSOLO [34] further proposed a unified framework for generating pseudo masks and iterative training. However, the performance relies heavily on the pre-trained self-supervised model, which determines the upper limits of such methods. Furthermore, such attention-based methods learned by contrastive learning also have the problem of distinguishing different instances within the same category. Our approach adopts top-down learning as well. Instead of aiding by an external self-supervised model, we look for motion cues to discover potential objects naturally.

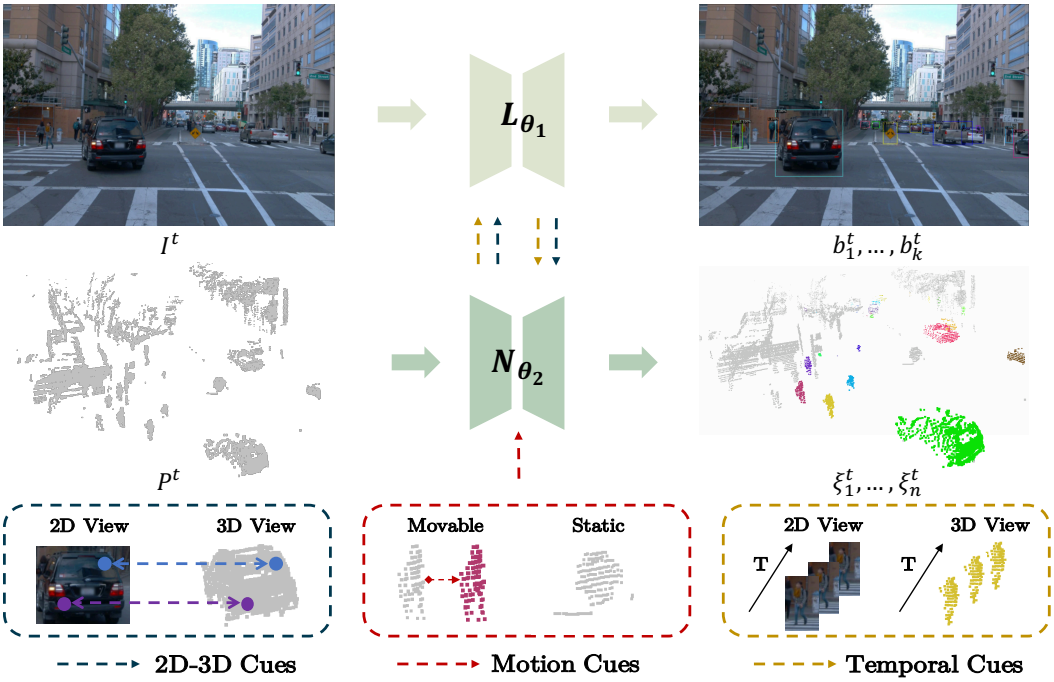

Figure 1: The pipeline of 4D unsupervised object discovery. The input is the corresponding 2D frames and 3D point clouds. The task needs to discover objects on both images and point clouds without manual annotations. Motion cues serve as the initial cues for training the ClusterNet. Then the localization network and ClusterNet are optimized jointly by 2D-3D cues and temporal cues.

3 Algorithm

3.1 Task definition and algorithm overview

The task of *4D Unsupervised Object Discovery* is defined as follows. As shown in Figure 1, the input is a set of video clips recorded in both 2D video frames I^t and 3D point clouds P^t at frame t during training. Since the point cloud and image data provide complementary information about location and appearance, they can serve as the natural cues guiding the training process mutually. During inference, the trained localization network L_{θ_1} is applied to still-image for 2D object detection, and the trained ClusterNet N_{θ_2} is applied to point cloud for 3D instance segmentation.

$$\{b_1^t, ..., b_k^t\} = L_{\theta_1}(I^t), \qquad \{\xi_1^t, ..., \xi_n^t\} = N_{\theta_2}(P^t) \tag{1}$$

$\{b_1^t, ..., b_k^t\}$ are the 2D bounding box predictions by localization network L_{θ_1} at frame t. $\{\xi_1^t, ..., \xi_n^t\}$ are the 3D instance segments output by ClusterNet N_{θ_2}. k and n denotes the instance index.

$$\theta_1^*, \theta_2^* = \arg\min_{\theta_1, \theta_2} f(L_{\theta_1}(I^t), N_{\theta_2}(P^t), t) \tag{2}$$

Our solution exploits the spatio-temporal consistency on 2D video frames and 3D point clouds. The algorithm can be formulated into a joint optimization function f in Eq.2. θ_1 and θ_2 are the parameters of the network need to optimize. Temporal information t serve as the natural constraint in function f. The localization network L_{θ_1} utilized Faster R-CNN as default. We propose a ClusterNet N_{θ_2} for 3D instance segmentation. Detail implementation will discuss in section 3.2. The major challenge is the optimization for function f without annotations. To overcome the challenge, we seek for *motion cues*, *2D-3D cues* and *temporal cues* to serve as the supervision. All these cues are extracted naturally in the informative 4D data. (1) *motion cues*, represented as 3D scene flow, can distinguish movable segments from the background. It uses to train the ClusterNet N_{θ_2} initially. (2) *2D-3D cues*, reflecting the mapping between LiDAR points and RGB pixels, can be used as a bridge to optimize the L_{θ_1} and N_{θ_2} iteratively. It indicates the output of either network can be further used to optimize another network. (3) *temporal cues*, encouraging the temporal-consistent discovery in 2D and 3D view, can serve as the constraint to optimize the function together. More details will introduce in section 3.3.

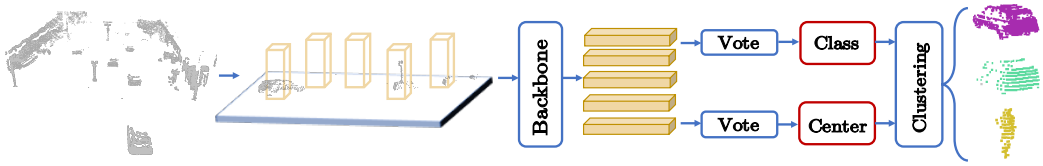

Figure 2: Overview of the ClusterNet architecture. A backbone extracts voxelized features for point clouds, given an input point cloud of N points with XYZ coordinates. Each point predicts a class type and center through a voting module. Then the points are clustered into instance segmentation.

3.2 ClusterNet

ClusterNet generates 3D instance segmentation from raw point clouds. As shown in Figure 2, given a point cloud $P \in \{(x, y, z)_i, i = 1, ..., N\}$, the network is able to give each point a class type $y_i \in \{1, 0\}$ (foreground or background) and instance ID $d_i \in \{1, ..., n\}$. Thus we can obtain n candidate segments $\xi_i = \{(x, y, z)_j | y_j = 1, d_j = i\}$ on the point cloud.

Network design. The model first voxelized 3D points $(x, y, z)_i$ and extract voxelized features by a transformer-based feature extractor [10]. We further project these voxelized features back to each point. The feature dimensions of points become $3 + C$ (3 means XYZ and C denotes the embedding dim). Inspired by the VoteNet [22], we leverage a voting module to predict the class type and center offset for each point. Specifically, the voting module is realized with a multi-layer perception (MLP) network. The voting module takes point feature $\boldsymbol{f_i} \in \mathcal{R}^{3+C}$ and outputs the Euclidean space offset $\Delta \boldsymbol{x_i} \in \mathcal{R}^3$ and class type prediction y_i. The final loss is the weighted sum of the class prediction and center regression:

$$\mathcal{L} = \mathcal{L}_{center} + \lambda \mathcal{L}_{cls} \tag{3}$$

The class prediction loss $\mathcal{L}_{cls}$ choose the focal loss [19] to balance the points of foreground and background. The predicted 3D offset $\Delta \boldsymbol{x_i}$ is supervised by a regression loss:

$$\mathcal{L}_{center} = \frac{1}{M} \Sigma_i \| \Delta \boldsymbol{x_i} - \Delta \boldsymbol{x_i^*} \|_1 \mathbb{1}[y_i^* = 1] \tag{4}$$

where $\mathbb{1}[y_i^* = 1]$ indicates whether a point belongs to the foreground according to the ground truth y_i^*. M is the total number of foreground points. $\Delta \boldsymbol{x_i^*}$ is the ground truth offset from the point position $\boldsymbol{x_i}$ to the instance center it belongs to. According to spatial proximity, we could further group the points into candidate instance segments with the predicted class type and center offset.

Training by motion cues. It is more challenging to obtain the supervision signal without manual annotation than the network design. Motion provides strong cues for identifying foreground points and grouping parts into objects since moving points generally belong to objects and have the same motion pattern if they belong to the same instance. We could estimate the 3D scene flow S^t from the sequence of point clouds P^t using the unsupervised method [17] at frame t. 3D scene flow describes the motion of all the 3D points in the scene, represented as $S^t = \{(v_x, v_y, v_z)_i^t, i = 1, ..., N\}$.

Combining the scene flow $(v_x, v_y, v_z)_i$ and point location in 2D $(u, v)_i$ and 3D $(x, y, z)_i$, we can obtain $(u, v, x, y, z, v_x, v_y, v_z)_i$ for each point p_i in P^t. Then, we cluster the points with HDBSCAN [4] to divide the scan into m segments which will be the instance candidates $\xi_1, ..., \xi_m$. However, these segments contain both foreground and background. We further assign each point p_i of segment ξ_j a binary label y_i^* to distinguish foreground points with the motion cues (scene flow), as shown in Eq. 5.

$$y_i^* = \mathbb{1}[\{\frac{1}{|\xi_j|} \sum_{p_i \in \xi_j} \mathbb{1}[\|S^t(p_i)\|_2 > \sigma]\} > \eta] \tag{5}$$

in which $\mathbb{1}[]$ is the indicator function. $|\xi_j|$ represents the total number of points belong to segment ξ_j. $p_i \in \xi_j$ is a point in segment ξ_j. $\|S^t(p_i)\|_2$ represents the velocity of the point p_i, and σ denotes the threshold for velocity. η determine the ratio of being an foreground object. $\sigma = 0.05$ and $\eta = 0.8$ by default. $y_i^* = 1$ means the point belongs to foreground segments. When the proportion of moving points in the segment is greater than the threshold η, we regard it as a foreground object, and all the points it contains are labeled as foreground. These foreground segments serve as the pseudo ground truth to train the ClusterNet initially.

3.3 Joint iterative optimization

ClusterNet trained with the *motion cues* serves as the initial weights for θ_2. Although movable objects can separate from the background, there are many static objects (e.g., parked cars or pedestrians waiting for traffic lights) in the scenes. Discovering both movable and static objects relies on further joint optimization by *2D-3D cues* and *temporal cues*. Specifically, The 3D segments output by N_{θ_2} can project to the 2D image to train the L_{θ_1}, and the 2D proposals output by L_{θ_1} can transform back to 3D view to train the N_{θ_2}. Temporal consistency ensures the objects appear continuously in 2D and 3D views, which is a critical constraint in the optimization. The joint optimization can be iterated many times since the 2D localization network and 3D ClusterNet can benefit from each other.

$$
\begin{aligned}
& \boldsymbol{\theta_1^*}, \boldsymbol{\theta_2^*} = \arg\min_{\theta_1, \theta_2} f(\boldsymbol{L_{\theta_1}}(I^t), \boldsymbol{N_{\theta_2}}(P^t), t) \\
\text{2D Step:} \quad & \boldsymbol{\theta_1^*} = \arg\min_{\theta_1} f(\boldsymbol{L_{\theta_1}}(I^t), N_{\theta_2}(P^t), t) \\
\text{3D Step:} \quad & \boldsymbol{\theta_2^*} = \arg\min_{\theta_2} f(L_{\theta_1}(I^t), \boldsymbol{N_{\theta_2}}(P^t), t)
\end{aligned}
\tag{6}
$$

In Eq. 6, our goal is to optimize the θ_1 and θ_2 without annotations. I^t and P^t denote the RGB image and point cloud at frame t. It is challenging to optimize both parameters simultaneously, so we divide the optimization process into two iterative steps: 2D step and 3D step.

2D step. In this step, the θ_2 is fixed and optimized θ_1. Since the ClusterNet N_{θ_2} are able to generate 3D instance segments $\xi_1, ..., \xi_n$ in 3D space, we can further project the 3D instance segments to 2D image plane by the transformation T_{cl} (from the LiDAR sensor to the camera) and projection matrix P_{pc} (from camera to pixels) defined by the camera intrinsic.

$$
\begin{pmatrix} \boldsymbol{u} \\ 1 \end{pmatrix} = P_{pc} T_{cl} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} \tag{7}
$$

in which $\boldsymbol{u}$ denotes the pixel location in the 2D image plane, and $\boldsymbol{x}$ represents the 3D position of LiDAR points. Hence we can obtain the object point sets $\omega_1, .., \omega_n$ in the 2D image plane by projecting the 3D instance segments $\xi_1, ..., \xi_n$. The 2D bounding boexes $b_1^*, ..., b_n^*$ derived from projected object point sets $\omega_1, .., \omega_n$ can use to optimize the weights of localization network L_{θ_1}.

3D step. In this step, the θ_1 is fixed and optimized θ_2. The localization network L_{θ_1} can generate candidate 2D proposals $b_1^a, ..., b_k^a$ based on the image appearance information. It enables us to discover more objects in the scene (e.g., parked cars regarded as background by motion cues). We can get the new 2D object set b^* by Eq. 8 (box IoU set to 0.3 in Non-Maximum Suppression).

$$
b^* = NMS(\{b_1^*, ..., b_n^*\} \cup \{b_1^a, ..., b_k^a\}) \tag{8}
$$

Since many 3D instance segments may have been labeled as background before, we later refined the label with the help of the 2D object set. Although the projection from LiDAR to the image is non-invertible without dense depth maps, we could utilize the mapping between LiDAR point and image pixels. It suggests using the LiDAR points within the 2D bounding box to relabel the 3D instance segments. However, the bounding box contains many LiDAR points corresponding to different 3D instance segments. Practically, we only consider the primary segment ξ_j (with most points) inside the bounding box and relabel the primary segment as the foreground object. With the refined label, we obtain the new set ξ^* of 3D instance segments $\xi_1^*, ..., \xi_n^*$, which utilize to optimize the weights for ClusterNet N_{θ_2}.

$$
f(L_{\theta_1}(I^t), N_{\theta_2}(P^t), t) = \underbrace{\mathcal{L}_{2D}(b^t, b_*^t) + \mathcal{L}_{smooth}(b_*^t)}_{\text{2D step}} + \underbrace{\mathcal{L}_{3D}(\xi^t, \xi_*^t) + \mathcal{L}_{smooth}(\xi_*^t)}_{\text{3D step}} \tag{9}
$$

Temporal cues. Temporal information can be integrated into the 2D step and 3D step as constraints. As shown in Eq. 9, b^t, ξ^t represent the predicted 2D bounding box set and predicted 3D segments set for frame t by L_{θ_1} and N_{θ_2}. b_*^t, ξ_*^t denote the pseudo annotation for 2D bounding boxes $(b_1^*, ..., b_n^*)$ and 3D segments $(\xi_1^*, ..., \xi_n^*)$ from previous 2D step and 3D step. $\mathcal{L}_{2D}$ is the loss for localization network, and $\mathcal{L}_{3D}$ is the loss for ClusterNet, which is the same as Eq. 3. $\mathcal{L}_{smooth}$ encourages that the same object have consistent object labels across frames. Therefore, it can help find new objects or filter out wrong annotations across time. More details are illustrated in Appendix.

4 Experiments

4.1 Dataset and implementation details

We evaluate our approach on the challenging Waymo Open Dataset (WOD) [30], which provides 3D point clouds and 2D RGB image data that is suitable for our task setting.

Dataset. Waymo [30] is a recently released large-scale dataset for autonomous driving. We utilize point clouds from the 'top' LiDAR (64 channels, a maximum distance of 75 meters), and video frames (at a resolution of 1280×1920 pixels) from the 'front' camera. The training and validation sets contain around 158k and 40k frames, respectively. All training frames and validation frames are manually annotated with 2D bounding boxes and 3D bounding boxes, which are capable of evaluating the performance of 2D object detection and 3D instance segmentation. Furthermore, Waymo also provides the scene flow annotation in the latest version [16], which can illustrate the upper potential of our method.

Evaluation protocol. Evaluation is conducted on the annotated validation set of WOD. We evaluate the performance of 2D object detection and 3D instance segmentation. The dataset contains four annotated object categories ('vehicles', 'pedestrians', 'cyclists', and 'sign'). We test the class-agnostic average precision (AP) score for vehicles, pedestrians, and cyclists. For 2D object detection, the AP score is reported at the box intersection-over-union (IoU) threshold of 0.5. For better analysis, results are also evaluated on small (area $< 32^2$ pixels), medium (32^2 pixels $<$ area $< 96^2$ pixels) and large objects (area $> 96^2$ pixels). We also calculated the average recall (AR) to measure the ability of object discovery. For 3D instance segmentation, no previous metrics have been proposed on Waymo. Referring to the 2D AP metrics, we propose to compute 3D AP score based on the IoU between predicted instance point sets and the ground truth. The ground truth for the instance segmentation can be obtained by labeling the point within 3D bounding boxes. The 3D AP score is reported at the point sets IoU threshold of 0.7 and 0.9, denoted as AP^{70} and AP^{90}, respectively. We also calculated the recall and foreground IoU for better analysis, which can measure the ability of object discovery from more perspectives. Note here the AP^{2D} denotes 2D object detection AP^{50} score. AP^{3D} denotes the 3D instance segmentation AP^{70} score.

Implementation details. Our implementation is based on the open-sourced code of mmdetection3d [8] for 3D detection and detectron2 [36] for 2D detection. For 2D localization network, we utilize Faster R-CNN [25] with FPN [18] by default, where ResNet-50 [15] is used as the backbone. The network is trained on 8 GPUs (A100) with 2 images per GPU for 12k iterations. The learning rate is initialized to 0.02 and is divided by 10 at the 6k and the 9k iterations. The weight decay and the momentum parameters are set as 10^{-4} and 0.9, respectively. For 3D ClusterNet, the input raw point clouds removed ground points first by [3] and remained the points that can only be seen on the front camera. The cluster range is $[0m, 74.88m]$ for the X-axis, $[-37.44m, 37.44m]$ for the Y-axis and $[-2m, 4m]$ for the Z-axis. The voxel size is $(0.32m, 0.32m, 6m)$. The feature extractor for voxelized points is [10], and the embedding dim for C is 128. In the focal loss for class prediction, we set $\gamma = 2.0, \alpha = 0.8$. The balance weight λ for Eq. 3 is set to 5. During inference, we set the minimum number of points to 5 for clustering. The ClusterNet is trained on 8 GPUs (A100) with 2 point clouds per GPU for 12 epochs. The learning rate is initialized to 10^{-5} and adopts the cyclic cosine strategy (target learning rate is 10^{-3}). For hyper-parameters in HDBSCAN [4], we set the min cluster size to 15, and the others follow the default. For more implementation details, please refer to Appendix.

4.2 Main results

2D object detection. Table 1 compares the results of manual annotation and ClusterNet for class-agnostic 2D object detection. For the 2D bounding box annotations, the distant boxes are LiDAR invisible. The manual-annotated supervised baseline is trained with LiDAR visible 2D box annotations for a fair comparison. Even though our method still has the gap **43.2 vs. 54.4** to supervised baseline using fully manual annotation (1137k bounding boxes), we can outperform the case when the annotation is limited. Limited annotation is frequent in real-world applications. Compared with 10% manual annotation (127k bounding boxes), our method could achieve 43.2 AP without any manual annotations and beat the 33.8 AP by a large margin. Since our method relies on the motion cues estimated unsupervised, we proved that the performance could increase to an incredible 51.8 AP with ground truth scene flow, which is very close to the performance of fully manual annotation but **without bounding box annotation**.

Table 1: Class-agnostic 2D object detection

annotation setting	#images	#bboxes	AP^{50}	AP^{50}_S	AP^{50}_M	AP^{50}_L	AR^{50}	AR^{50}_S	AR^{50}_M	AR^{50}_L
supervised										
fully manual annotation	158k	1137k	54.4	20.5	72.4	90.9	62.8	35.5	80.8	94.0
10% manual annotation	15k	127k	33.8	5.5	45.3	74.9	36.1	9.7	48.6	76.7
unsupervised										
Felzenszwalb Segmentation [11]	158k	0	0.4	0.0	0.5	1.1	11.1	0.6	14.5	30.7
LOST [29]	158k	0	1.9	0.0	1.0	7.6	5.0	0.0	0.4	27.9
ClusterNet (w/ gt sceneflow)	158k	0	51.8	21.3	70.2	89.5	60.8	30.2	81.2	94.8
ClusterNet	158k	0	43.2	18.4	56.5	81.8	55.4	26.7	71.9	93.1

Since the previous unsupervised methods only focused on still 2D images and could not extract the objects from the background accurately, it is no surprise they could only achieve poor results in such challenging scenes. LOST [29] can only extract one primary object from the background, which does not apply to our driving scenes. Felzenszwalb Segmentation [11] generate potential proposals by graph-based segmentation, but lacks the ability to identify foreground. Our method has great performance advantages over these still-image methods.

3D instance segmentation. Table 2 illustrates the effectiveness of our ClusterNet on 3D instance segmentation. Our ClusterNet achieved 26.2 AP^{70} and 19.2 AP^{90} without any annotation, superior to 10% supervised baseline 23.6 AP^{70} and 15.5 AP^{90} with 397k 3D bounding boxes annotation. We proved that our method with accurate motion cues (ground truth scene flow) could achieve 42.0 AP^{70} and 33.2 AP^{90}, even comparable to that supervised baseline with fully manual annotation (4268k 3D bounding boxes). No previous method can achieve such high performance under an unsupervised setting. Figure 3 illustrates the object prediction of our approach on the Waymo validation set.

Table 2: Class-agnostic 3D instance segmentation

annotation setting	#point clouds	#3d bboxes	AP^{70}	AP^{90}	$Recall^{70}$	$Recall^{90}$	IoU
supervised							
fully manual annotation	158k	4268k	45.7	37.3	75.1	65.1	92.2
10% manual annotation	15k	397k	23.6	15.5	61.8	48.7	81.6
unsupervised							
ClusterNet (w/ gt sceneflow)	158k	0	42.0	33.2	61.7	52.3	88.1
ClusterNet	158k	0	26.2	19.2	40.0	32.8	64.9

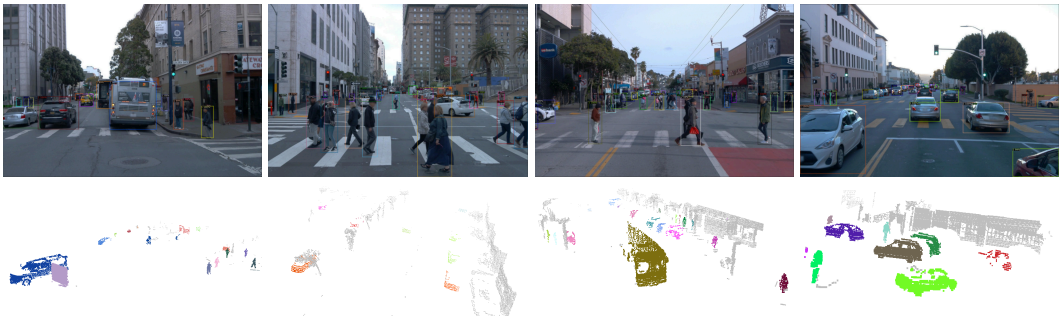

Figure 3: Visualization for 2D object detection and 3D instance segmentation on the Waymo validation set. Our approach could achieve such incredible results without any annotations.

2D instance segmentation. We can also conduct instance segmentation by projecting the LiDAR points of 3D instance segments to the 2D image plane. The key difference is that instance segmentation masks other than object bounding boxes deriving from 3D instance segments as pseudo annotations. We utilized alpha shape [1] to generate the mask of object points (projected to the 2D image). The localization network can change conveniently to Mask R-CNN [14] for the instance segmentation

without manual annotations. Some predictions on validation set are illustrated in Figure 4 and Appendix. Because the Waymo Open Dataset did not provide the annotation for instance mask, the performance of instance segmentation cannot be evaluated quantitatively.

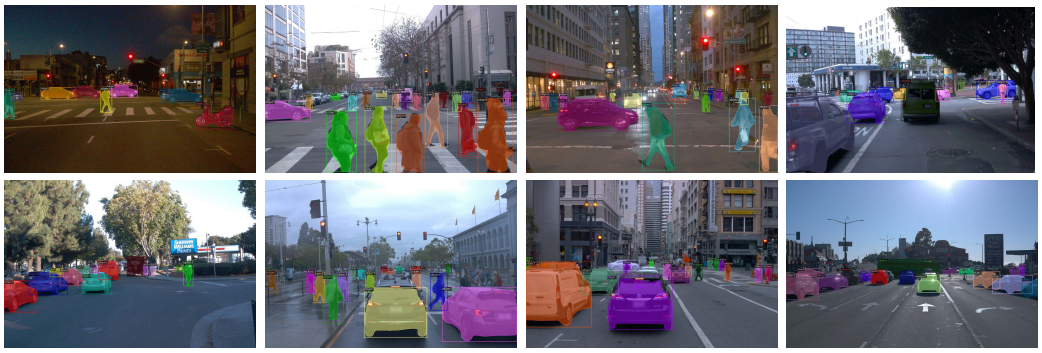

Figure 4: Instance segmentation by our approach when using Mask R-CNN [14] as our localization network, without manual annotations. Our method can generate high-quality instance masks.

4.3 Ablation studies

Analysis of multi cues for training. Table 3 analyze the contributions of multi cues in our approach to the Waymo validation set. Three iterations of joint optimizing train the final results. AP^{2D} denotes the AP^{50} score for 2D object detection, and AP^{3D} denotes the AP^{70} score for 3D instance segmentation. The ClusterNet was trained with the pseudo-annotations obtained by HDBSCAN Clustering [4] for the first time. So a simple baseline is directly using the HDBSCAN for 3D instance segmentation and project on 2D for localization network training. In comparison, we demonstrate the effectiveness of using ClusterNet; the performance increased by 4.9 AP^{2D} (from 25.1 to 30.0) and 2.2 AP^{3D} (from 4.6 to 6.8). Furthermore, the performance improved significantly by joint optimizing for the ClusterNet and localization network, with 2D-3D cues and temporal cues.

Table 3: Analysis of multi cues for training.

Method	point cloud	motion cues	2D-3D cues	temporal cues	AP^{2D} ↑	AP^{3D} ↑
HDBSCAN [4]	✓				14.9	2.1
	✓	✓			25.1	4.6
ClusterNet	✓	✓			30.0	6.8
	✓	✓	✓		40.4	25.7
	✓	✓	✓	✓	**43.2**	**26.2**

Training strategy for localization network. Training the localization network with the pseudo annotations generated by ClusterNet is quite different from training with manual annotations. Specifically, the pseudo annotations will be noisy and incomplete before joint iterative training. Therefore, two points we found crucial for the initial training: (1) sampling strategy in RPN (Region Proposal Network) and (2) early stopping for training. The following ablation experiments are conducted for the first-time training the localization network. Table 4 compares three different strategies: (a) sample anchors with box IoU > 0.7 as positive example and box IoU < 0.3 as negative example (denoted as $IoU_+ > 0.7, IoU_- < 0.3$), as in the standard Faster R-CNN; (b) sample anchors with box IoU > 0.6 as positive example and box IoU < 0.4 as negative example (denoted as $IoU_+ > 0.6, IoU_- < 0.4$); (c) sample anchors with box IoU > 0.6 as positive example, and box $0.1 <$ IoU < 0.4 as negative example (denoted as $IoU_+ > 0.6, 0.1 < IoU_- < 0.4$). In this way, the strategy considerably reduces the chance of sampling missed objects as negative examples. Table 5 compares different training iterations and shows that early stopping improves the generalization performance. Since the pseudo annotations have noise, training for a long time may overfit the noise in the training set, leading to the degradation of generalization performance.

Table 4: Ablation on sampling strategy.

sampling strategy	AP^{50}	AP^{50}_S	AP^{50}_M	AP^{50}_L
(a) IoU_+>0.7, IoU_-<0.3	27.8	3.2	37.2	70.0
(b) IoU_+>0.6, IoU_-<0.4	28.2	3.3	36.9	71.7
(c) **IoU_+>0.6, 0.1<IoU_-<0.4**	**30.0**	**4.3**	**39.4**	**73.2**

Table 5: Ablation on early stopping.

iterations	AP^{50}	AR^{50}
3000	**30.0**	**43.3**
6000	27.9	40.4
12000	25.3	35.0

Joint iterative optimization. Table 6 presents the effectiveness of our joint iterative optimization for the 2D localization network and 3D ClusterNet. Iteration 0 represents the initial performance of ClusterNet trained by motion cues (estimated scene flow). Next, each iteration means a 2D step and a 3D step. Even though the model did not perform well at the beginning, with joint iterative optimization, both AP^{2D} and AP^{3D} improved rapidly. Applying more than one iteration improves the results, indicating that the 2D localization network and 3D ClusterNet can benefit from each other. We set the iteration number as 3 by default.

Minimum points for ClusterNet. Minimum points determine the minimum number of LiDAR points for a 3D instance. Table 7 analyze the model performance under different parameters during the inference. We set the min points to 5 by default.

Table 6: Joint iterative optimization.

iterations	AP^{2D}	AP^{3D}
0	/	6.8
1	30.0	20.2
2	37.4	25.4
3	**43.2**	**26.2**
4	42.8	25.9

Table 7: Minimum points for ClusterNet.

min points	AP^{3D}
2	25.3
5	**26.2**
10	26.0
20	25.5

5 Discussion and conclusions

Discussion. Unsupervised object discovery used to believe infeasible due to the ambiguity of objects and complexity of scenarios. However, 4D data with the sequence of image frames and point clouds provide enough cues to discover the movable objects, even without manual annotation. The complementary information behind the 3D LiDAR points and 2D image and constraints from temporal are the critical factors for the success of unsupervised object discovery. With 4D sensor data readily available onboard, our approach shows extraordinary potential for scenarios with limited or no annotation. The only limitation is that our method is suitable for movable objects; static things (never move) like beds or chairs can not be discovered. After all, the manual annotation is limited, and the costs are expensive. We seek the real intelligence for the machine to naturally understand the scene by mutual supervision of different modalities rather than human annotations.

Conclusions. In this work, we propose a new task named *4D Unsupervised Object Discovery*. The task needs to discover objects both on the image and point clouds without manual annotations. We present the first practical approach for this task by proposing a *ClusterNet* for 3D instance segmentation and joint iterative optimization. Extensive experiments on the large-scale Waymo Open Dataset demonstrate the effectiveness of our approach. So far as we know, we are the first work to achieve such high performance for unsupervised 2D object detection and 3D instance segmentation, bridging the gap between unsupervised methods and supervised methods. Our work sheds light on a new perspective on the future study of unsupervised object discovery.

Societal Impacts. The development of unsupervised object discovery requires large datasets, introducing privacy issues. The technology of unsupervised detection dramatically reduces the labeling cost; it may affect the people engaged in the labeling industry in the future. The elimination of human intervention may also cause some data annotators to lose their current jobs. Our approach only tests in driving scenes for effectiveness, which may lead to some wrong detection in other scenes.

References

[1] Nataraj Akkiraju, Herbert Edelsbrunner, Michael Facello, Ping Fu, EP Mucke, and Carlos Varela. Alpha shapes: definition and software. In *Proceedings of International Computational Geometry Software Workshop*, 1995. 7

[2] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *CVPR*, 2014. 1, 2

[3] Igor Bogoslavskyi and Cyrill Stachniss. Efficient online segmentation for sparse 3d laser scans. *PFG–Journal of Photogrammetry Remote Sensing and Geoinformation Science*, 2017. 1, 6

[4] 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*, 2013. 1, 2, 4, 6, 8

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1, 2

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2

[8] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 6

[9] Bertrand Douillard, James Underwood, Noah Kuntz, Vsevolod Vlaskine, Alastair Quadros, Peter Morton, and Alon Frenkel. On the segmentation of 3d lidar point clouds. In *ICRA*, 2011. 2

[10] 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. *arXiv preprint arXiv:2112.06375*, 2021. 4, 6

[11] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 7

[12] Ross Girshick. Fast r-cnn. In *ICCV*, 2015. 2

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1, 2, 7, 8

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[16] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *RAL*, 2021. 6

[17] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *NeurIPS*, 2021. 4

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 6

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2, 4

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[21] AJ Piergiovanni, Vincent Casser, Michael S Ryoo, and Anelia Angelova. 4d-net for learned multi-modal alignment. In *ICCV*, 2021. 2

[22] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, 2019. 1, 2, 4

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[24] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *CVPR*, 2017. 1, 2

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 2015. 1, 2, 6

[26] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *ICRA*, 2011. 2

[27] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020. 1, 2

[28] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1

[29] 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, 2, 7

[30] 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 *CVPR*, 2020. 2, 6

[31] Hao Tian, Yuntao Chen, Jifeng Dai, Zhaoxiang Zhang, and Xizhou Zhu. Unsupervised object detection with lidar clues. In *CVPR*, 2021. 1

[32] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019. 1, 2

[33] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013. 1, 2

[34] Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. Freesolo: Learning to segment objects without annotations. *arXiv preprint arXiv:2202.12181*, 2022. 1, 2

[35] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In *CVPR*, 2021. 1, 2

[36] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[37] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2

[38] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020. 2

[39] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021. 1, 2

[40] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2

[41] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018. 1, 2

[42] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 1, 2

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] Code will be released upon acceptance
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1 and Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]