Weakly Supervised 3D Object Detection from Lidar Point Cloud

Qinghao Meng[1], ✉Wenguan Wang[2], Tianfei Zhou[3],
Jianbing Shen[3,1], Luc Van Gool[2], and Dengxin Dai[2]

[1]School of Computer Science, Beijing Institute of Technology
[2]ETH Zurich   [3]Inception Institute of Artificial Intelligence
https://github.com/hlesmqh/WS3D

Abstract. It is laborious to manually label point cloud data for training high-quality 3D object detectors. This work proposes a *weakly supervised* approach for 3D object detection, only requiring a small set of weakly annotated scenes, associated with a few precisely labeled object instances. This is achieved by a two-stage architecture design. Stage-1 learns to generate cylindrical object proposals under weak supervision, *i.e.*, only the horizontal centers of objects are click-annotated in bird's view scenes. Stage-2 learns to refine the cylindrical proposals to get cuboids and confidence scores, using a few well-labeled instances. Using only 500 weakly annotated scenes and 534 precisely labeled vehicle instances, our method achieves $85-95\%$ the performance of current top-leading, fully supervised detectors (requiring $3,712$ exhaustively and precisely annotated scenes with $15,654$ instances). Moreover, with our elaborately designed network architecture, our trained model can be applied as a 3D object annotator, supporting both automatic and active (human-in-the-loop) working modes. The annotations generated by our model can be used to train 3D object detectors, achieving over 94% of their original performance (with manually labeled training data). Our experiments also show our model's potential in boosting performance when given more training data. Above designs make our approach highly practical and introduce new opportunities for learning 3D object detection at reduced annotation cost.

Keywords: 3D Object Detection · Weakly Supervised Learning

1 Introduction

Over the past several years, extensive industry and research efforts have been dedicated to autonomous driving. Significant progress has been made in key technologies for innovative autonomous driving functions, with 3D object detection being one representative example. Almost all recent successful 3D object detectors are built upon *fully supervised* frameworks. They provided various solutions to problems arising from monocular images [1, 2], stereo images [3] or point clouds [4–7]; gave insight into point cloud representation, introducing

---

✉ Corresponding author: *Wenguan Wang* (wenguanwang.ai@gmail.com).
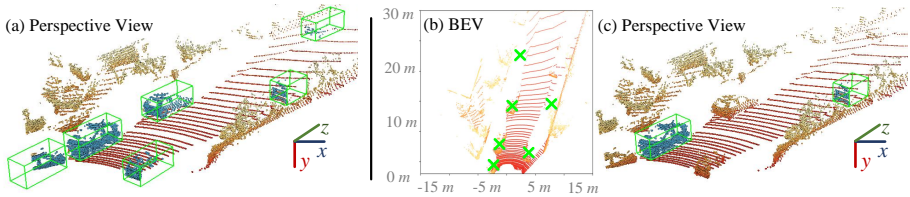
Fig. 1. Comparison between the *full* supervision used in prior arts **(a)** and our *inaccurate, inexact* **(b)** and *incomplete* **(c)** supervision. Previous fully supervised methods are trained from massive, exhaustively-labeled scenes ($3,712$ precisely annotated scenes with $15,654$ vehicle instances), while our model uses only 500 weakly annotated scenes with center-annotated BEV maps as well as 534 precisely labeled vehicle instances.

techniques such as voxelization [8, 9] and point-wise operation [10]; and greatly advanced the state-of-the-arts. However, these methods necessitate *large-scale, precisely-annotated* 3D data to reach performance saturation and avoid overfitting. Unfortunately, such data requirement involves an astonishing amount of manual work, as it takes hundreds of hours to annotate just one hour of driving data. The end result is that a corpus of 3D training data is not only costly to create, but also limited in size and variety. In short, the demand for massive, high-quality yet expensive labeled data has become one of the biggest challenges faced by 3D object detection system developers.
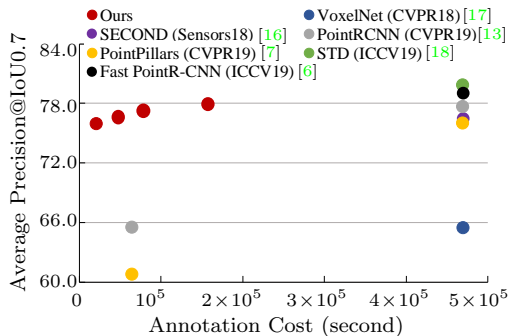
In order to promote the deployment of 3D object detection systems, it is necessary to decrease the heavy annotation burden. However, this essential issue has not received due attention so far. To this end, we propose a weakly supervised method that learns 3D object detection from less training data, with more easily-acquired and cheaper annotations. Specifically, our model has two main stages. Stage-1 learns to predict the object centers on the (x, z)-plane and identity foreground points. The training in this stage only requires a small set of weakly annotated bird's eye view (BEV) maps, where the horizontal object centers are labeled (Fig. 1(b)). Such *inexact* and *inaccurate* supervision greatly saves annotation efforts. Since the height information is missing in BEV maps, we generate a set of cylindrical proposals whose extent along the y-axis is unlimited. Then, Stage-2 learns to estimate 3D parameters from these proposals and predict corresponding confidence scores. The learning paradigm in this stage is achieved by a few, precisely-annotated object instances as *incomplete* supervision (Fig.1(c)), in contrast to prior arts [5, 11], which consume massive, exhaustively-labeled scenes (*full* ground-truth labels, Fig. 1(a)).

Our weakly supervised framework provides two appealing characteristics. First, it learns 3D object detection by making use of a small amount of weakly-labeled BEV data and precisely-annotated object instances. The weak supervision from BEV maps is in the form of click annotations of the horizontal object centers. This enables much faster data labeling compared to strong supervision requiring cuboids to be elaborately annotated on point clouds (about 40∼50× faster; see §3). For the small set of well-annotated object instances, we only label

25% of objects in the weakly-labeled scenes, which is about 3% of the supervision used in current leading models. Such a weakly supervised 3D object detection paradigm not only provides the opportunity to reduce the strong supervision requirement in this field, but also introduces immediate commercial benefits.

Second, once trained, our detector can be applied as an annotation tool to assist the laborious labeling process. Current popular, supervised solutions eagerly consume all the training data to improve the performance, while pay little attention to how to facilitate the training data annotation. Our model design allows both automatic and active working modes. In the automatic mode (no annotator in the loop), after directly applying our model to automatically re-annotate KITTI dataset [12], re-trained PointPillars [7] and PointRCNN [13] can maintain more than 94% of their original performance. In the active setting, human annotators first provide center-click supervision on the BEV maps, which is used as privileged information to guide our Stage-2 for final cuboid prediction. Under such a setting, re-trained PointPillars and PointRCNN reach above 96% of their original performance. More essentially, compared with current strongly supervised annotation tools [14,15], our model is able to provide more accurate annotations with much less and weaker supervision, at higher speed (§5.4).

For KITTI [12], the experiments on **Car** class show that, using only 500 weakly annotated scenes and 534 precisely labeled vehicle instances, we achieve 85−95% of the performance of fully supervised state-of-the-arts (which require 3,712 precisely annotated scenes with 15,654 vehicle instances). When using more training data, our performance is further boosted (Fig. 2). For **Pedestrian** class with fewer annotations, our method even outperforms most existing methods, clearly demonstrat-

Fig. 2. Performance *vs.* annotation efforts, tested on KITTI [12] `val` set (**Car**), under the *moderate* regime. Our model yields promising results with far less annotation demand (§5.2).

ing the effectiveness of our proposed weakly supervised learning paradigm.

2   Related Work

Learning Point Cloud Representations: Processing sparse, unordered point cloud data from LiDAR sensors is a fundamental problem in many 3D related areas. There are two main paradigms for this: voxelization or point based methods. The first type of methods [8,9,19,20] voxelize point clouds into volumetric grids and apply 2D/3D CNNs for prediction. Some of them [4,11,21,22] further improve volumetric features with multi-view representations of point clouds. Voxel based methods are computationally efficient but suffer from information loss (due to quantitization of point clouds with coarse voxel resolution). The second-

type, point based methods [10, 13, 17, 23–25], which directly operate on raw point clouds and preserve the original information, recently became popular.

3D Object Detection: A flurry of techniques have been explored for 3D object detection in driving scenarios, which can be broadly categorized into three classes. *(1) 2D image based methods* focus on camera based solutions with monocular or stereo images, by exploring geometries between 3D and 2D bounding boxes [1, 3, 26, 27], or similarities between 3D objects and CAD models [2, 28]. Though efficient, they struggle against the inherent difficulty of directly estimating depth information from images. *(2) 3D point cloud based methods* rely on depth sensors such as LiDAR. Some representative ones project point clouds to bird's view and use 2D CNNs [4, 5] to learn the point cloud features. Some others [17, 21] apply 3D CNNs over point cloud voxels to generate cuboids. They tend to capture local information, due to the limited receptive fields of CNN kernels. Thus sparse convolutions [16] with enlarged receptive fields are adopted later. To avoid losing information during voxelization, some efforts learn point-wise features directly from raw point clouds, using PointNet [10]-like structures [6, 7]. *(3) Fusion-based methods* [10, 13, 24, 25, 29, 30] attempt to fuse information from different sensors, such as cameras and LiDAR. The basic idea is to leverage the complementary information of camera images and point clouds, *i.e.*, rich visual information of images and precise depth details of point clouds, to improve the detection accuracy. However, fusion-based methods typically run slowly due to the need of processing multi-modal inputs [16].

Click Supervision: Click annotation schemes were used to reduce the burden of collecting segmentation/bounding box annotations at a large scale. Current efforts typically leverage center-click [31, 32], extreme-point [33, 34], or corrective-click [31] supervision for semantic segmentation [31, 35] or object detection [32–34] in 2D visual scenarios. However, in this work, we explore center clicks, located on BEV maps, as weak supervision signals for 3D object detection.

3D Object Annotation: Very few attempts were made to scale up 3D object annotation pipelines [14, 15]. [15] lets an annotator place 2D seeds from which to infer 3D segments and centroid parameters, using fully supervised learning. [14] suggests a differentiable template matching model with curriculum learning. In addition to different annotation paradigms, model designs and level of human interventions, our model is also unique in its weakly supervised learning strategy and dual-work mode, and achieves stronger performance.

3   Data Annotation Strategy for Our Weak Supervision

Before detailing our model, we first discuss how to get our weakly supervised data.
Traditional Precise But Laborious Labeling Strategy: Current popular 3D object detectors are fully supervised deep learning models, requiring precisely annotated data. However, creating a high-quality 3D object detection dataset is more complex than creating, for example, a 2D object detection dataset. For precise labeling [36, 37], annotators first navigate the 3D scene to find an object with the help of visual content from the camera image (Fig. 3(a)). Later, an initial
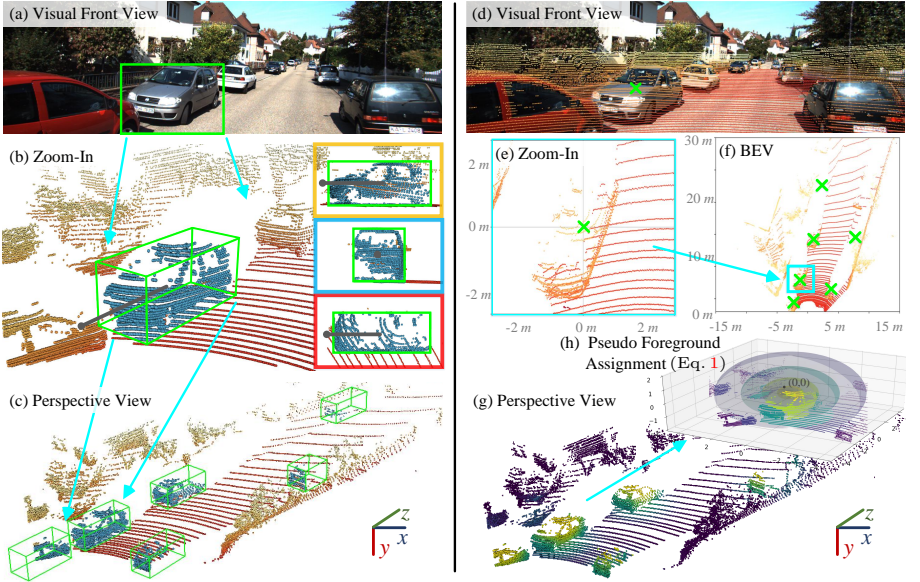
Fig. 3. **(a-c)**: Precise annotations require extensive labeling efforts (see §3). **(d-f)**: Our weak supervision is simply obtained by clicking object centers (denoted by ✖) on BEV maps (see §3). **(g-h)**: Our pseudo groundtruths for fore-/background segmentation (yellower indicates higher foreground score; see §4.1).

rough cuboid and orientation arrow (Fig. 3 (b)) are placed. Finally, the optimal annotation (Fig. 3 (c)) is obtained by gradually adjusting the 2D boxes projected in orthographic views. As can be seen, although this labeling procedure generates high-quality annotations, it contains several subtasks with gradual corrections and 2D-3D view switches. It is thus quite laborious and expensive.

Our Weak But Fast Annotation Scheme: Our model is learned from a small set of weakly annotated BEV maps, combined with a few precisely labeled 3D object instances. The weakly annotated data only contains object center-annotated BEV maps, which can be easily obtained. Specifically, human annotators first roughly click a target on the camera front-view map (Fig. 3 (d)). Then the BEV map is zoomed in and the region around the initial click is presented for a more accurate center-click (Fig. 3 (e)). Since our annotation procedure does not refer to any 3D view, it is very easy and fast; most annotations can be finished by only two clicks. However, the collected supervision is weak, as only the object centers over (x, z)-plane are labeled, without height information in y-axis and size of cuboids.

Annotation Speed: We re-labeled KITTI `train` set [12], which has 3,712 driving scenes with more than 15K vehicle instances. This took about 11 hours, *i.e.*, 2.5 s per instance. As KITTI does not report the annotation time, we refer to other published statistics [37, 38], which suggest around 114 s per instance in a fully manual manner [38] or 30s with extra assistance of a 3D object detector [37].

Thus our click supervision provides a 15∼45× reduction in the time required for traditional precise annotations.

Annotation Quality: To assess our annotation quality, Fig. 4 depicts the average distance between our annotated centers and the KITTI groundtruths on BEV maps. The average errors on x- and z-axes are about 0.25 m and 0.75 m, respectively, bringing out the limitation of LiDAR sensors in capturing the object better to its side than the back.
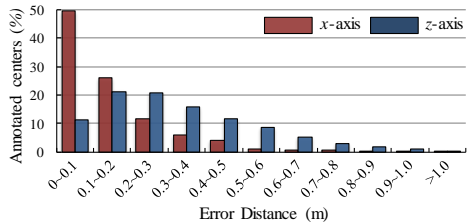
Fig. 4. Distance distributions (**Car**) on x- and z-axes of our weak BEV annotations.

4   Proposed Algorithm

Our object detector takes raw point clouds as input and outputs oriented 3D boxes. It has a cylindrical 3D proposal generation stage (§4.1, Fig. 5(a-b)), learning from click supervision, and a subsequent, proposal-based 3D object localization stage, learning from a few, well-annotated object instances (§4.2, Fig. 5(c-d)). Below we will focus on **Car** class. However, as evidenced in our experiments (§5.2), our model can also easily be applied to other classes, such as **Pedestrian**.

### 4.1   Learn to Generate Cylindrical Proposals from Click Annotations

There are two goals in our first stage: 1) to generate foreground point segmentation; and 2) to produce a set of cylinder-shaped 3D object proposals. The fore-/background separation is helpful for the proposal generation and provides useful information for the second stage. Because only the horizontal centers of objects are labeled on the BEV maps, our proposals are cylinder-shaped.

Pseudo Groundtruth Generation. Since the annotations in the BEV maps are weak, proper modifications should be made to produce pseudo, yet stronger supervision signals. Specifically, for a labeled vehicle center point $o \in \mathcal{O}$, its horizontal location (x_o, z_o) in the LiDAR coordinate system can be inferred according to the projection from BEV to point cloud. We set its height y_o (over y-axis) to the LiDAR sensor's height (the height of the ego-vehicle), *i.e.*, $y_o=0$. The rationale behind such a setting will be detailed later. Then, for each unlabeled point p, its pseudo foreground value $f^p \in [0, 1]$ is defined as:

$$f^p = \max_{o \in \mathcal{O}}(\iota(p, o)), \quad \text{where } \iota(p, o) = \begin{cases} 1 & \text{if } d(p, o) \leq 0.7, \\ \frac{1}{\kappa}\mathcal{N}(d(p, o)) & \text{if } d(p, o) > 0.7. \end{cases} \quad (1)$$

Here, $\mathcal{N}$ is a 1D Gaussian distribution with mean 0.7 and variance 1.5, and $\kappa=\mathcal{N}(0.7)$ is a normalization factor. And $d(p, o)$ is a distance function: $d(p, o) = [(x_p-x_o)^2 + \frac{1}{2}(y_p - y_o)^2 + (z_p-z_o)^2]^{\frac{1}{2}}$, where (x_p, y_p, z_p) is the 3D coordinate of
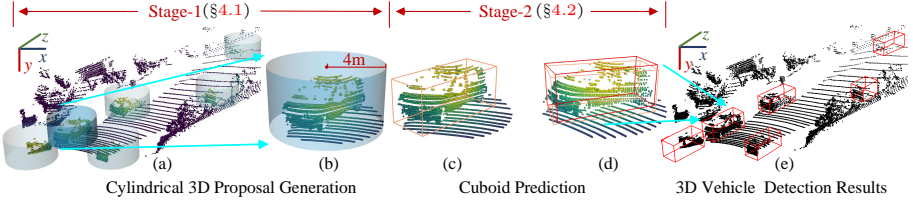
Fig. 5. Our 3D object detection pipeline (§4). (a-b) Cylindrical 3D proposal generation results from Stage-1 (§4.1). Yellower colors correspond to higher foreground probabilities. (c-d) Cuboid prediction in Stage-2 (§4.2). (e) Our final results.

p. The coefficient $(=\frac{1}{2})$ is used to due to the large uncertainty over y-axis. The foreground probability assignment function $\iota(p, o)$ gives high confidence $(=1)$ for those points close to o (*i.e.*, $d(p, o) \leq 0.7$), and attenuates the confidence for distant ones (*i.e.*, $d(p, o) > 0.7$) by following the Gaussian distribution $\mathcal{N}$. The reason why we set the heights of the labeled center points $\mathcal{O}$ as 0 is because most object points are at lower altitudes than the LiDAR sensor (at the top of the ego-vehicle) and they will gain high foreground values in this way. For those background points even with similar altitudes to the LiDAR sensor, they are very sparse and typically far away from the vehicle centers in (x, z)-plane (see Fig. 3(h)), and can thus be ignored. Plane detection [3] can be used for more accurate height estimation, but in practice we find our strategy is good enough.
Point Cloud Representation. Several set-abstraction layers with multi-scale grouping are applied to directly learn discriminative point-wise features from raw point cloud input [24]. Then, two branches are placed over the backbone for foreground point segmentation and vehicle (x, z)-center prediction, respectively.
Foreground Point Segmentation. With the point-wise features extracted from the backbone network and pseudo groundtruth f^p generated in Eq. 1, the foreground segmentation branch learns to estimate the foreground probability $\tilde{f}^p$ of each point p. The learning is achieved by minimizing the following loss:

$$\mathcal{L}_{\text{seg}} = \alpha(1 - \hat{f}^p)^\gamma \log(\hat{f}^p), \quad \text{where} \quad \hat{f}^p = \tilde{f}^p \cdot f^p + (1 - \tilde{f}^p) \cdot (1 - f^p). \quad (2)$$

This is a soft version of the focal loss [39], where the binary fore-/background label is given in a probability formation. As in [39], we set $\alpha = 0.25$ and $\gamma = 2$.
Object (x, z)-Center Prediction. The other branch is for object (x, z)-center regression, as the weakly annotated BEV maps only contain horizontal information. As in [13], a bin-based classification strategy is adopted. For each labeled object center $o \in \mathcal{O}$, we set the points within 4 m distance as *support points* (whose pseudo foreground probabilities ≥ 0.1). These support points are used to estimate the horizontal coordinates of o. For each support point p, its surrounding area ($L \times L$ m^2) along x- and z- axes is the searching space for o, which is split into a series of discrete bins. Concretely, for x- and z- axis, the search range $L (=8\,\text{m})$ is divided into 10 bins of uniform length $\delta (=0.8\,\text{m})$. Therefore, for a support point p and the corresponding center o, the target bin assignments

(b_x, b_z) along x- and z-axis can be formulated as:

$$b_x = \lfloor \frac{x_p - x_o + L}{\delta} \rfloor, \qquad b_z = \lfloor \frac{z_p - z_o + L}{\delta} \rfloor. \tag{3}$$

Residual (r_x, r_z) is computed for further location refinement within each assigned bin:

$$r_{u \in \{x,z\}} = \frac{u}{\varepsilon}(u_p - u_o + L - (b_u \cdot \delta + \frac{\delta}{2})), \qquad \text{where} \quad \varepsilon = \frac{\delta}{2}. \tag{4}$$

For a support point p, the center localization loss $\mathcal{L}_{\text{bin}}$ is designed as:

$$\mathcal{L}_{\text{bin}} = \sum\nolimits_{u \in \{x,z\}} \mathcal{L}_{\text{cls}}(\tilde{b}_u, b_u) + \mathcal{L}_{\text{reg}}(\tilde{r}_u, r_u), \tag{5}$$

where $\tilde{b}$ and $\tilde{r}$ are predicted bin assignments and residuals, and b and r are the targets. $\mathcal{L}_{\text{cls}}$ is a cross-entropy loss for bin classification along the (x, z)-plane, and $\mathcal{L}_{\text{reg}}$ refers to the $\ell 1$ loss for residual regression w.r.t the target bins.

Cylindrical 3D Proposal Generation. During inference, the segmentation branch estimates the foreground probability of each point. Then, we only preserve the points whose foreground scores are larger than 0.1. As we only have horizontal coordinates of the centers, we cannot directly generate 3D bounding box proposals. Instead, for each center, we generate a cylindrical proposal with a 4 m radius over (x, z)-plane and unlimited extent along y-axis (Fig. 5 (a, b)).

Center-Aware Non-Maximum Suppression. To eliminate redundant proposals, we propose a center-aware non-maximum suppression (CA-NMS) strategy. The main idea is that, it is easier to predict centers from center-close points than far ones, and center-close points gain high foreground scores under our pseudo groundtruth generation strategy. Thus, for a predicted center, we use the foreground probability of its sourced (support) point, as its confidence score. That means we assume that a point with a higher foreground score is more center-close and tends to make a more confident center prediction. Then we rank all the predicted centers according to their confidence, from large to small. For each center, if its distance to any other pre-selected centers is larger than 4 m on the (x, z)-plane, its proposal will be preserved; otherwise it is removed.

4.2   Learn to Refine Proposals from A Few Well-Labeled Instances

Stage-2 is to estimate cuboids from proposals and recognize false estimates. We achieve this by learning a proposal refinement model from a few well-annotated instances, motivated by two considerations. **(i)** The proposal refinement is performed instance-wise, driving us to consume instance-wise annotations. **(ii)** Our initial cylindrical proposals, though rough, contain rich useful information, which facilitates cuboid prediction especially when training data is limited.

Overall Pipeline. Our method carries out refinement of cuboid predictions over two steps. First, an initial cuboid generation network takes cylindrical proposals as inputs, and outputs initial cuboid estimations (Fig. 5 (b-c)). Then, a final cuboid refinement network takes the initial cuboid estimations as inputs, and outputs final cuboid predictions (Fig. 5 (c-d)) as well as confidence.

Initial Cuboid Generation. The initial cuboid generation network stacks several set abstraction layers, intermediated with a single-scale grouping operation, to collect contextual and pooled point features as the cylindrical proposal representations [24]. Then, a multilayer perceptron based branch is appended for initial cuboid estimation. Let us denote the groundtruth of an input cuboid as $(x, y, z, h, w, l, \theta)$, where (x, y, z) are the object-center coordinates, (h, w, l) object size, and θ orientation from BEV. A bin-based regression loss $\mathcal{L}_{\text{bin}}$ is applied for estimating θ, and a smooth $\ell 1$ loss $\mathcal{L}_{\text{reg}}$ is used for other parameters:

$$\mathcal{L}_{\text{ref}} = \mathcal{L}_{\text{bin}}(\tilde{\theta}, \theta) + \sum_{u \in \{x,y,z,h,w,l\}} \mathcal{L}_{\text{reg}}(\tilde{u}, u), \tag{6}$$

where $(\tilde{x}, \tilde{y}, \tilde{z}, \tilde{h}, \tilde{w}, \tilde{l}, \tilde{\theta})$ are the estimated cuboid parameters.

Final Cuboid Refinement. The final cuboid refinement network has the similar network architecture of the initial cuboid generation network. It learns to refine initial cuboid estimations with the same loss design in Eq. 6. In addition, to predict cuboid's confidence, an extra confidence estimation head is added, which is supervised by an IoU-based regression loss [27,40]:

$$\mathcal{L}_{\text{con}} = \mathcal{L}_{\text{reg}}(\tilde{C}_{\text{IoU}}, C_{\text{IoU}}), \tag{7}$$

where the targeted confidence score C_{IoU} is computed as the largest IoU score between the output cuboid and groundtruths.

In the first cuboid generation step, for each groundtruth 3D bounding box, cylindrical proposals whose center-distances (on (x, z)-plane) are less than 1.4 m away are selected as the training samples. Then, the output cuboids from those cylindrical proposals are further used as the training samples for the groundtruth in the final refinement step.

4.3   Implementation Detail

Detailed Network Architecture. In Stage-1 (§4.1), to align the network input, 16K points are sampled from each point-cloud scene. Four set-abstraction layers with multi-scales are stacked to sample the points into groups with sizes (4096, 1024, 256, 64). Four feature propagation layers are then used to obtain point-wise features, as the input for the segmentation and center prediction branches. The segmentation branch contains two FC layers with 128 and 1 neuron(s), respectively. The (x, z)-center prediction branch has two FC layers with 128 and 40 neurons, respectively. In Stage-2 (§4.2), 512 points are sampled from each cylindrical proposal/cuboid, and each point is associated with a 5D feature vector, *i.e.*, a concatenation of 3D point coordinates, 1D laser reflection intensity, and foreground score. Before feeding each proposal/cuboid into the generation/refinement network, the coordinates of points are canonized to guarantee their translation and rotation invariance [6]. The corresponding groundtruth is modified accordingly. In addition, for cylindrical proposals, only a translation transformation is performed over (x, z)-plane, *i.e.*, the horizontal coordinates of the proposal center are set as $(0, 0)$. For each cuboid, the coordinates of points

within a 0.3 m radius are cropped to include more context. For the cuboid generation/refinement network, four set-abstraction layers with single-scale grouping are used to sample the input points into groups with sizes (256, 128, 32, 1). Finally, a 512D feature is extracted for cuboid and confidence estimation.

Data Preparation. KITTI `train` set has 3,712 precisely annotated training scenes with 15,654 vehicle instances. Unless otherwise noted, we use the following training data setting. The first 500 scenes with our weakly annotated BEV maps are used for training our Stage-1 model, and 25% of the vehicle instances (=534) in the 500 scenes are associated with precise 3D annotations and used for training our Stage-2 model[1]. We make use of this weak and limited training data to better illustrate the advantage of our model. This also allows us to investigate the performance when using our model as an annotation tool (see §5.4).

Data Augmentation. During training, we adopt several data augmentation techniques to avoid overfitting and improve the generalization ability. In Stage-1, left-right flipping, scaling from [0.95, 1.05], and rotation from [-10°, 10°] are randomly applied for each scene. In addition, to diversify training scenarios, we randomly sample a few annotated vehicle centers with surrounding points within a cylinder with a 4 m radius, and insert them into the current sample. Furthermore, to increase the robustness to distant vehicles, which typically contain very few points, we randomly drop the points within the cylindrical space (with a 4 m radius) of labeled centers. In Stage-2, for each proposal, we randomly conduct left-right flipping, scaling from [0.8, 1.2], and rotation from [-90°, 90°]. We shift each proposal by small translations, following a Gaussian distribution with mean 0 and variance 0.1, for x-, y-, and z-axis each individually. We randomly change the foreground label of the points. To address large occlusions, we randomly omit part of a proposal (1/4−3/4 of the area in BEV). Finally, for each proposal, we randomly remove the inside points (at least 32 points remain).

Inference. After applying CA-NMS for the cylindrical proposals generated in Stage-1 (§4.1), we feed the remaining ones to Stage-2 (§4.2) and get final 3D predictions. We then use an oriented NMS with a BEV IoU threshold of 0.3 to reduce redundancy. Our model runs at about 0.2 s per scene, which is on par with MV3D [4] (0.36 s), VoxelNet [17] (0.23 s) and F-PointNet [25] (0.17 s).

5   Experiment

5.1   Experimental Setup

Dataset. Experiments are conducted on KITTI [12], which contains 7,481 images for train/val and 7,518 images for testing. The `train/val` set has 3D bounding box groundtruths and is split into two sub-sets [4,17]: `train` (3,712 images) and `val` (3,769 images). We train our detector only on a weakly labeled subset of `train` set, while the `val` set is used for evaluation only. Detection outcomes are evaluated in the three standard regimes: *easy, moderate, hard*.

Evaluation Metric. Following [4], average precisions for BEV and 3D boxes are reported. Unless specified, the performance is evaluated with a 0.7 IoU threshold.

---

[1] The instances are randomly selected and the list will be released.

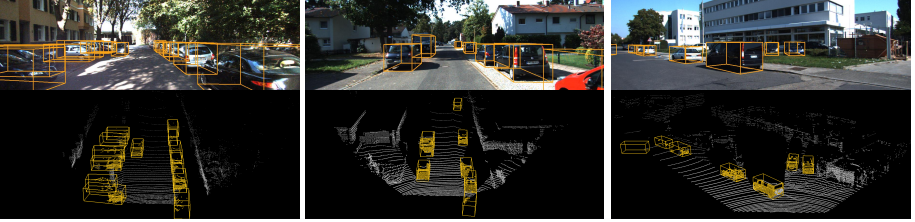**Table 1. Evaluation results on KITTI `val` set (Car).** See §5.2 for details.

Learning Paradigm	Detector	Modality	BEV@0.7			3D Box@0.7		
			Easy	Moderate	Hard	Easy	Moderate	Hard
Trained with the whole KITTI `train` set: 3,712 precisely labeled scenes with 15,654 vehicle instances								
Fully supervised	VeloFCN [11]	LiDAR	40.14	32.08	30.47	15.20	13.66	15.98
	PIXOR [5]	LiDAR	86.79	80.75	76.60	-	-	-
	VoxelNet [17]	LiDAR	89.60	84.81	78.57	81.97	65.46	62.85
	SECOND [16]	LiDAR	89.96	87.07	79.66	87.43	76.48	69.10
	PointRCNN [13]	LiDAR	-	-	-	88.45	77.67	76.30
	PointPillars [7]	LiDAR	89.64	86.46	84.22	85.31	76.07	69.76
	Fast PointR-CNN [6]	LiDAR	90.12	88.10	86.24	89.12	79.00	77.48
	STD [18]	LiDAR	90.50	88.50	88.10	89.70	79.80	79.30
Trained with a part of KITTI `train` set: 500 precisely labeled scenes with 2,176 vehicle instances								
Fully supervised	PointRCNN [13]	LiDAR	87.21	77.10	76.63	79.88	65.50	64.93
	PointPillars [7]	LiDAR	86.27	77.13	75.91	72.36	60.75	55.88
Trained with a part of KITTI `train` set: 125 precisely labeled scenes with 550 vehicle instances								
Fully supervised	PointRCNN [13]	LiDAR	85.09	74.35	67.68	67.54	54.91	51.96
	PointPillars [7]	LiDAR	85.76	75.30	73.29	65.51	51.45	45.53
Trained with a part of KITTI `train` set: 500 weakly labeled scenes with 534 precisely annotated instances								
Weakly supervised	**Ours**	LiDAR	88.56	84.99	84.74	84.04	75.10	73.29

Table 2. Evaluation results on KITTI `test` set (Car). See §5.2 for details.

Learning Paradigm	Detector	Modality	BEV@0.7			3D Box@0.7		
			Easy	Moderate	Hard	Easy	Moderate	Hard
Trained with the whole KITTI `train` set: 3,712 precisely annotated scenes with 15,654 vehicle instances								
Fully supervised	PIXOR [5]	LiDAR	87.25	81.92	76.01	-	-	-
	VoxelNet [17]	LiDAR	89.35	79.26	77.39	77.47	65.11	57.73
	SECOND [16]	LiDAR	88.07	79.37	77.95	83.13	73.66	66.20
	PointRCNN [13]	LiDAR	89.47	85.68	79.10	85.94	75.76	68.32
	PointPillars [7]	LiDAR	88.35	86.10	79.83	79.05	74.99	68.30
	Fast PointR-CNN [6]	LiDAR	88.03	86.10	78.17	84.28	75.73	67.39
	STD [18]	LiDAR	94.74	89.19	86.42	87.95	79.71	75.09
Trained with a part of KITTI `train` set: 500 weakly labeled scenes + 534 precisely annotated instances								
Weakly supervised	**Ours**	LiDAR	90.11	84.02	76.97	80.15	69.64	63.71

5.2 Quantitative and Qualitative Performance

Quantitative Results on KITTI `val` Set (Car). In Table 1, we compare our method with several leading methods, which all use fully-labeled training data (*i.e.*, 3,712 precisely-labeled scenes with 15,654 vehicle instances). However, despite using far less, weakly labeled data, our method yields comparable performance. In addition, as there is no other weakly supervised baseline, we re-train two outstanding detectors, PointRCNN [13] and PointPillars [7], under two relatively comparable settings, *i.e.*, using **(i)** 500 precisely labeled scenes (containing 2,176 well-annotated vehicle instances); and **(ii)** 125 precisely labeled scenes (containing 550 well-annotated instances). This helps further assess the efficacy of our method. Note that, for training our method, we use 500 scenes with center-click labels and 534 precisely-annotated instances. We find that our method significantly outperforms re-trained PointRCNN and PointPillars, using whether the same amount of well-annotated scenes (500; setting **(i)**) or the similar number of well-annotated instances (534; setting **(ii)**).

Fig. 6. Qualitative results of 3D object detection (Car) on KITTI `val` set (§5.2). Detected 3D bounding boxes are shown in yellow; images are used only for visualization.

Table 3. Evaluation results on KITTI `val` set (Pedestrian). See §5.2 for details.

Learning Paradigm	Detector	Modality	BEV@0.5			3D Box@0.5		
			Easy	Moderate	Hard	Easy	Moderate	Hard
Trained with: 951 precisely labeled scenes with 2,257 pedestrian instances								
Fully supervised	PointPillars [7]	LiDAR	71.97	67.84	62.41	66.73	61.06	56.50
	PointRCNN [13]	LiDAR	68.89	63.54	57.63	63.70	69.43	58.13
	Part-A^2 [40]	LiDAR	-	-	-	70.73	64.13	57.45
	VoxelNet [17]	LiDAR	70.76	62.73	55.05	-	-	-
	STD [18]	LiDAR	75.90	69.90	66.00	73.90	66.60	62.90
Trained with: 951 weakly labeled scenes with 515 pedestrian instances								
Weakly supervised	**Ours**	LiDAR	74.79	70.17	66.75	74.65	69.96	66.49

Quantitative Results on KITTI `test` Set (Car). We also evaluate our algorithm on KITTI `test` set, by submitting our results to the official evaluation server. As shown in Table 2, though current top-performing methods use much stronger supervision, our method still gets competitive performance against some of them, such as PIXOR [5] and VoxelNet [17]. We can also observe that there is still room for improvement between weakly- and strong-supervised methods.

Qualitative Results. Fig. 6 depicts visual results of a few representative scenes from KITTI `val` set, showing that our model is able to produce high-quality 3D detections of vehicles that are highly occluded or far away from the ego-vehicle.

Quantitative Results on KITTI `val` Set (Pedestrian). We also report our performance on **Pedestrian** class (with 0.5 IoU threshold). In this case, our model is trained with 951 click-labeled scenes and 25% (515) of precisely annotated pedestrian instances. More training details can be found in the supplementary material. As shown in Fig. 3, our method shows very promising results, demonstrating its good generalizability and advantages when using less supervision.

5.3   Diagnostic Experiment

As the ground-truth for KITTI `test` set is not available and the access to the test server is limited, ablation studies are performed over the `val` set (see Table 4).

Robustness to Inaccurate BEV Annotations. As discussed in §3, the annotations over the BEV maps are weak and inaccurate. To examine our robustness

Table 4. Ablation study on KITTI `val` set (Car). See §5.3 for details.

Aspects	Training Setting	BEV@0.7			3D Box@0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Full model	500 weakly labeled scenes +25% (534) precisely annotated instances	88.56	84.99	84.74	84.04	75.10	73.29
More precisely annotated BEV maps	500 weakly labeled scenes +25% (534) more precisely annotated instances	88.52	84.57	85.02	85.67	75.13	73.92
More training data	3,712 weakly labeled scenes +534 precisely annotated instances	88.81	86.98	85.76	86.08	76.04	74.97
	1,852 weakly labeled scenes +25% precisely annotated instances	89.11	85.95	85.52	87.14	76.78	76.56
	3,712 weakly labeled scenes +25% precisely annotated instances	89.32	86.17	86.31	87.57	77.62	76.94

to inaccurate BEV annotations, we retrain our model with precise BEV annotations inferred from groundtruth 3D annotations. From Table 4, only marginal improvements are observed, verifying our robustness to noisy BEV annotations.
More Training Data. To demonstrate the potential of our weakly supervised 3D object detection scheme, we probe the upper bound by training on additional data. As evidenced by the results in the last three rows in Table 4, with the use of more training data, gradual performance boosts can indeed be achieved.

5.4 Performance as An Annotation Tool

Our model, once trained, can be used as an 3D object annotator. It only consumes part of KITTI `train` set, allowing us to explore its potential for assisting annotation. Due to its specific network architecture and click-annotation guided learning paradigm, it supports both automatic and active annotation modes.
Automatic Annotation Mode. For a given scene, it is straightforward to use our predictions as pseudo annotations, resulting in an automatic working mode. In such a setting, our method takes around 0.1 s for per car instance annotation. Previous 3D detection methods can also work as automatic annotators in this way. However, as they are typically trained with the whole KITTI `train` set, it is hard to examine their annotation quality.
Active Annotation Mode. In the active mode, human annotators first click on object centers in BEV maps, following the labeling strategy detailed in §3. Then, for each annotated center, 25 points are uniformly sampled from the surrounding 0.4 m×0.4 m region (0.1 m interval). These points are used as the centers of cylindrical proposals and the foreground masks around them are generated according to Eq. 1. Then, we use our Stage-2 model to predict the cuboids, from which the one with largest confidence score is selected as the final annotation. About 2.6 s is needed for annotating each car instance in our active annotation mode, whereas humans take 2.5 s for center-click annotation on average.
Annotation Quality. Table 5 reports the evaluation results for our annotation quality on KITTI `val` set. Two previous annotation methods[14,15] are included. [15] is a fully supervised deep learning annotator, trained with the whole KITTI `train` set. It only works as an active model, where humans are required to

Table 5. Comparison of annotation quality on KITTI `val` set (see §5.4).

Learning Paradigm	Method	Mode	Speed (sec./inst.)	BEV@0.5			3D Box@0.5		
				Easy	Moderate	Hard	Easy	Moderate	Hard
Trained with the whole KITTI `train` set: 3,712 well-labeled scenes with 15,654 vehicle instances									
Fully Supervised	[15]	Active	3.8	-	-	-	-	-	88.33
Trained with KITTI `train`+`val`: 7,481 scenes (implicitly using 2D instance segmentation annotations)									
Fully-Supervised	[14]	Auto	8.0	80.70	63.36	52.47	63.39	44.79	37.47
Trained with a part of KITTI `train` set: 500 weakly labeled scenes + 534 precisely annotated instances									
Weakly Supervised	**Ours**	Auto	0.1	96.33	89.01	88.52	95.85	89.14	88.32
		Active	2.6	99.99	99.92	99.90	99.87	90.78	90.14

Table 6. Performance of PointRCNN [13] and PointPillars [7] when trained using different annotations sources. Results are reported on KITTI `val` set (§5.4).

Detector	Annotation Source	BEV@0.7			3D Box@0.7		
		Easy	Moderate	Hard	Easy	Moderate	Hard
PointRCNN [13]	Manual	90.21	87.89	85.51	88.45	77.67	76.30
	Automatic (ours)	88.02	85.75	84.27	83.22	74.54	73.29
	Active (ours)	88.64	85.41	84.94	84.21	76.08	74.91
PointPillars [7]	Manual	89.64	86.46	84.22	85.31	76.07	69.76
	Automatic (ours)	88.55	85.62	83.84	84.79	74.18	68.52
	Active (ours)	88.94	85.88	83.86	84.53	75.03	68.63

provide object anchor clicks. [14] requires synthetic data for training and relies on MASK-RCNN[41], so it implicitly uses 2D instance segmentation annotations. It works in an automatic mode. The scores for these models are borrowed from the literature, as their implementations are not released. Following their settings[14, 15], scores with 0.5 3D IoU criterion are reported. As seen, our model produces high-quality annotations, especially in the active mode. Our annotations are more accurate than[14,15], with much less and weak supervision. Considering our fast annotation speed (0.1−2.6 s per instance), the results are very significant.
Suitability for 3D Object Detection. To investigate the suitability of our labels for 3D object detection, we use our re-labeled KITTI `train` set (**Car**) to re-train PointPillars [7] and PointRCNN [13], which show leading performance with released implementations. During training, we use their original settings. From Table 6, we can observe that the two methods only suffer from small performance drops when using our labels.

6   Conclusion and Discussion

This work has made an early attempt to train a 3D object detector using limited and weak supervision. In addition, our detector can be extended as an annotation tool, whose performance was fully examined in both automatic and active modes. Extensive experiments on KITTI dataset demonstrate our impressive results, but also illustrate that there is still room for improvement. Given the massive number of algorithmic breakthroughs over the past few years, we can expect a flurry of innovation towards this promising direction.

References

1. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3D object detection for autonomous driving. In: CVPR. (2016) 1, 4
2. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In: CVPR. (2017) 1, 4
3. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3D object proposals for accurate object class detection. In: NeurIPS. (2015) 1, 4, 7
4. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: CVPR. (2017) 1, 3, 4, 10
5. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3D object detection from point clouds. In: CVPR. (2018) 1, 2, 4, 11, 12
6. Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point R-CNN. In: ICCV. (2019) 1, 3, 4, 9, 11
7. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR. (2019) 1, 3, 4, 11, 12, 14
8. Maturana, D., Scherer, S.: Voxnet: A 3D convolutional neural network for real-time object recognition. In: IROS. (2015) 2, 3
9. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3D shapenets: A deep representation for volumetric shapes. In: CVPR. (2015) 2, 3
10. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In: CVPR. (2017) 2, 4
11. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3D Lidar using fully convolutional network. In: Robotics: Science and Systems. (2016) 2, 3, 11
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: CVPR. (2012) 3, 5, 10
13. Shi, S., Wang, X., Li, H.: PointRCNN: 3D object proposal generation and detection from point cloud. In: CVPR. (2019) 3, 4, 7, 11, 12, 14
14. Zakharov, S., Kehl, W., Bhargava, A., Gaidon, A.: Autolabeling 3D objects with differentiable rendering of SDF shape priors. arXiv preprint arXiv:1911.11288 (2019) 3, 4, 13, 14
15. Lee, J., Walsh, S., Harakeh, A., Waslander, S.L.: Leveraging pre-trained 3D object detection models for fast ground truth generation. In: ITSC. (2018) 3, 4, 13, 14
16. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors (2018) 3, 4, 11
17. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3D object detection. In: CVPR. (2018) 3, 4, 10, 11, 12
18. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: STD: Sparse-to-dense 3D object detector for point cloud. In: ICCV. (2019) 3, 11, 12
19. Xiang, Y., Choi, W., Lin, Y., Savarese, S.: Data-driven 3D voxel patterns for object category recognition. In: CVPR. (2015) 3
20. Xie, J., Zheng, Z., Gao, R., Wang, W., Zhu, S.C., Nian Wu, Y.: Learning descriptor networks for 3d shape synthesis and analysis. In: CVPR. (2018) 3
21. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3D shape recognition. In: ICCV. (2015) 3, 4
22. Qi, C.R., Su, H., Niessner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3D data. In: CVPR. (2016) 3

23. Rethage, D., Wald, J., Sturm, J., Navab, N., Tombari, F.: Fully-convolutional point networks for large-scale point clouds. In: ECCV. (2018) 4
24. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS. (2017) 4, 7, 9
25. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3D object detection from RGB-D data. In: CVPR. (2018) 4, 10
26. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3D bounding box estimation using deep learning and geometry. In: CVPR. (2017) 4
27. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: GS3D: An efficient 3D object detection framework for autonomous driving. In: CVPR. (2019) 4, 9
28. Mottaghi, R., Xiang, Y., Savarese, S.: A coarse-to-fine model for 3D pose estimation and sub-category recognition. In: CVPR. (2015) 4
29. Liang, M., Yang, B., Wang, S., Urtasun, R.: Deep continuous fusion for multi-sensor 3D object detection. In: ECCV. (2018) 4
30. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3D object detection. In: CVPR. (2019) 4
31. Bearman, A., Russakovsky, O., Ferrari, V., Fei-Fei, L.: What's the point: Semantic segmentation with point supervision. In: ECCV. (2016) 4
32. Papadopoulos, D.P., Uijlings, J.R., Keller, F., Ferrari, V.: Training object class detectors with click supervision. In: CVPR. (2017) 4
33. Maninis, K.K., Caelles, S., Pont-Tuset, J., Van Gool, L.: Deep extreme cut: From extreme points to object segmentation. In: CVPR. (2018) 4
34. Papadopoulos, D.P., Uijlings, J.R., Keller, F., Ferrari, V.: Extreme clicking for efficient object annotation. In: ICCV. (2017) 4
35. Benenson, R., Popov, S., Ferrari, V.: Large-scale interactive object segmentation with human annotators. In: CVPR. (2019) 4
36. Xie, J., Kiefel, M., Sun, M.T., Geiger, A.: Semantic instance annotation of street scenes by 3D to 2D label transfer. In: CVPR. (2016) 4
37. Wang, P., Huang, X., Cheng, X., Zhou, D., Geng, Q., Yang, R.: The apolloscape open dataset for autonomous driving and its application. IEEE TPAMI (2019) 4, 5
38. Song, S., Lichtenberg, S.P., Xiao, J.: SUN RGB-D: A RGB-D scene understanding benchmark suite. In: CVPR. (2015) 5
39. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollar, P.: Focal loss for dense object detection. In: ICCV. (2017) 7
40. Shi, S., Wang, Z., Shi, J., Wang, X., Li, H.: From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. IEEE TPAMI (2020) 9, 12
41. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV. (2017) 14