Direct LiDAR-based object detector training from automated 2D detections

Robert McCraith
Visual Geometry Group
University of Oxford
robert@robots.ox.ac.uk

Eldar Insafutdinov
Visual Geometry Group
University of Oxford
eldar@robots.ox.ac.uk

Lukas Neumann
Visual Recognition Group
Czech Technical University in Prague
neumalu1@fel.cvut.cz

Andrea Vedaldi
Visual Geometry Group
University of Oxford
vedaldi@robots.ox.ac.uk

Abstract

3D Object detection (3DOD) is an important component of many applications, however existing methods rely heavily on datasets of depth and image data which require expensive annotation in 3D thus limiting the ability of a diverse dataset being collected which truly represents the long tail of potential scenes in the wild.

In this work we propose to utilise a readily available robust 2D Object Detector and to transfer information about objects from 2D to 3D, allowing us to train a 3D Object Detector without the need for any human annotation in 3D. We demonstrate that our method significantly outperforms previous 3DOD methods supervised by only 2D annotations, and that our method narrows the accuracy gap between methods that use 3D supervision and those that do not.

1 Introduction

3D Object Detection in LiDAR data (3DOD) is an important component of many applications, ranging from autonomous cars to robotics. However compared to the more traditional 2D Object Detection, which is an extensively studied field of Computer Vision, its 3D counterpart on the other hand has seen slower development, mainly due to challenging data acquisition and even more challenging human annotation requirements, as labeling objects in 3D is substantially more expensive and more time consuming while also being error prone. As a result in the context of autonomous driving, there is only a handful of annotated datasets Geiger et al. [2012], Caesar et al. [2019], Sun et al. [2020] for 3D Object Detection, but still they only contain data for one or two cities, that reduces their ability to generalise to other locations or to capture rare events (aka the *long tail problem*), which is especially important in such a safety-critical application.

In order to address the aforementioned 3D annotation requirements, we instead propose to exploit a readily available 2D Object Detector on a *image sequence* as a form of weak supervision to create the training signal for the 3D Object detector in *LiDAR data*, thus completely removing the need for 3D human annotations. The cross-modal training approach has several advantages: Firstly, image object detection is a well-studied problem with robust methods on very diverse data with minimal bias towards specific models of car which may be more prominent in some locations. Secondly, by only requiring a 2D detection at training time, we reduce drastically the amount of annotation required to label new data. And last but not least, transferring detections between domains means that our network must learn to reason about the 3D shape of objects, rather than presuming that an object is present, which can cause problems for 3DOD methods which rely on image-based detectors at test time as well Zakharov et al. [2020a], McCraith et al. [2022].

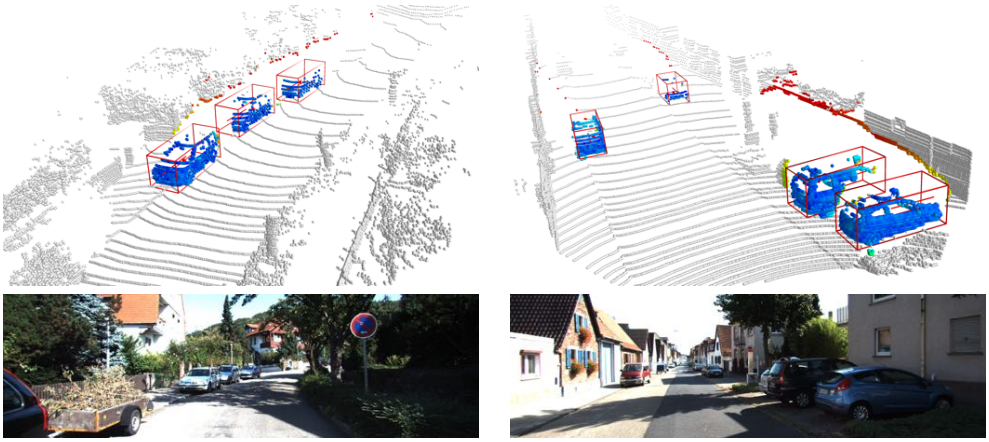

Figure 1: Qualitative examples of our LiDAR 3D Object Detection method. LiDAR point cloud and detected cars (top row) and scene image (for visualization purposes only - bottom row)

In this paper, we introduce three key contributions to weakly-supervised 3DOD training: i) Unlike previous methods, the 2D Object Detector is only required at training time, which not only simplifies the pipeline and removes need for additional sensors in the car or a robot, but it also eliminates sensitivity to missed detections by the image object detector, which is especially prevalent for small far-away objects. ii) We introduce a new *Soft Inlier Count* (SIC) loss function which allows us to train the whole system end-to-end using standard back-propagation. iii) A new multi-cell voting scheme is introduced to relax the original hard-choice training loss of the fully-supervised method, allowing the model to gradually improve its estimates as the training progresses.

2 Related

2.1 Supervised 3D Object Detection

Image object detection methods are quite mature, which resulted in many early 3D object detection methods making detections in RGB and using these to select LiDAR data on which to reason about 3D location. A notable example of this is Frustum PointNets Qi et al. [2017]. In this method an image based detector produces a set of bounding boxes who's frustums are used to select a subset of the LiDAR point cloud. These points are then segmented to further reduce down to only points inside the 3D bounding box and a final stage produces location and rotation estimates.

Another popular technique is to perform detection and localisation on LiDAR data alone. A method popularised in Zhou and Tuzel [2018] is to gather LiDAR points which fall into a voxel and learn features voxel-by-voxel, then take these features and organize them to construct a BEV image of features upon which further convolutional layers derive location. This has been expanded on in Lang et al. [2019], Yin et al. [2021] where voxels are expanded in the vertical axis to contain all points in a square patch (in BEV) and the resulting prediction are decoded by anchors or heatmaps.

2.2 Weakly Supervised 3D Object Detection

Recent works have attempted to resolve annotation difficulties by utilising a coarse annotation strategy for a large set with fine annotations for a smaller set thus making annotation less time consuming/costly. In Meng et al. [2020] they use a small number of weak annotations of bounding box centres in BEV and a small amount of exact 3D box annotations. A detector is trained in a two-stage manner by first predicting object proposals in BEV and then regressing accurate 3D localisation. Such a coarse labeling strategy also can result in some of the problems mentioned in Feng et al. [2020] such as inability to accurately localise car center under heavy occlusion, partial scans of objects based on their perceived orientation and a poor ability to determine object shape from some viewpoints. Qi et al. [2021] run a pre-trained detector (on a small subset of labeled data) on an entire LiDAR sequence to produce a set of 3D boxes which are then passed into a multi-target tracker. Utilising complimentary information from multiple frames allows them to improve detection accuracy and apply their auto-labeler in a semi-supervised scenario.

Cross-modal supervision. Other formulations of this problem utilise additional datasets to supervise components of 3D detection pipelines. Qin et al. [2020] trains a 3D object proposal network based on distance-normalised point cloud density. In the second stage, an image-based network pretrained on a different dataset to classify proposals and predict viewpoint image bounding box is used as a teacher for a LiDAR-only model. An alternative approach is used in Zakharov et al. [2020b] where a network is pretrained on Parallel Domain McNamara [2020] – a synthetic dataset to provide a good initial estimate of translation and rotation. This network then takes Mask R-CNN He et al. [2017] detections that have a high overlap with a ground truth bounding box to refine the parameters on read data. Finally, McCraith et al. [2022] also takes Mask R-CNN detections and uses a Frustum PointNet style architecture and attempts to learn which points are inliers to remove noise, as with the other methods however this method depends on the availability of camera and LiDAR at test time (and their coordinate transformation remains consistent). Compared to these approaches we rely on the same 2D detection but only a training time. This has an advantage that our trained detector operates on LiDAR input and can recover objects that an RGB-based detector may fail to predict, for example when the car is reflected from a glass wall. Training a detector end-to-end allows to exploit regularities in the input LiDAR space and effectively utilise available 2D supervision.

3 Method

Our goal is to determine the 3D bounding boxes without the need for expensive, time consuming human annotation which severely limits the construction of large scale datasets for 3DOD. To achieve this we leverage mature 2D object detection methods for which the annotation is much simpler and more large scale datasets with a diverse set of data exist from around the globe rather than a small area as is typically seen in 3D Object Detection datasets.

Cross modal supervision such as this however has several challenges. 2D instance segmentation in pixel space is a crude approximation of segmentation in 3D point clouds, with many pixels at the boundary of objects being mis-classified (or coming from the region in the objects frustum from one sensor but not the other), the transparency of windows results in LiDAR points correctly attributed to a vehicle in image space to actually be located on surfaces behind the vehicle, and in many cases LiDAR scans may only have very small parts of a vehicle visible leading to ambiguity or orientation and translation.

3D Object Detection is typically reduced to regressing values for (X, Y, Z) center, (l, w, h) size and θ yaw, rather than the other option of predicting the 3D location of the 8 corners of the 3D bounding box. In many cases object size is a hard to determine quantity, as noted in Feng et al. [2020] in many cases only one side of a car is visible making annotations for size unreliable, with this in mind we use a generic car shape with a fixed size to train out model to predict location and orientation which we feel are much more important quantities for downstream tasks.

3.1 Exploiting 2D Labels in a 3D Point Cloud

In our method, the training signal is generated by an off-the-shelf 2D object detection and segmentation system, such as Mask R-CNN He et al. [2017]. Given an RGB image of the scene $I \in \mathbb{R}^{3\times H\times W}$, the 2D detector creates a set of detections D in the form of a segmentation mask m for each detected car

$$D = \left\{ m \in \{0,1\}^{H\times W} \right\} \tag{1}$$

Taking a corresponding LiDAR point cloud of the same scene $L = \left\{(x, y, z, r) \in \mathbb{R}^4\right\}$, we define the point cloud subset L_m for each detection m as

$$L_m = \left\{ p \subset L : \text{proj}(p) \in m \right\} \tag{2}$$

$$\text{proj}(u) = K T_{\text{cam}} u \quad u \in \mathbb{R}^3 \tag{3}$$

where $\text{proj}(u)$ is the projection of the world point u onto the 2D image given by known camera intrinsics K and a transformation from LiDAR to camera co-ordinate system $T_{\text{cam}} : \mathbb{R}^3 \to \mathbb{R}^3$.

While in some cases it may be possible to directly reconstruct the 3D shape of an object given enough LiDAR points exist in the L_m subset, in our context the object is only typically observed from one side, resulting in partial scans at best. This is especially a problem for cars whose apparent angle (how it appears in the image rather than its global orientation) suggests that the object it moving

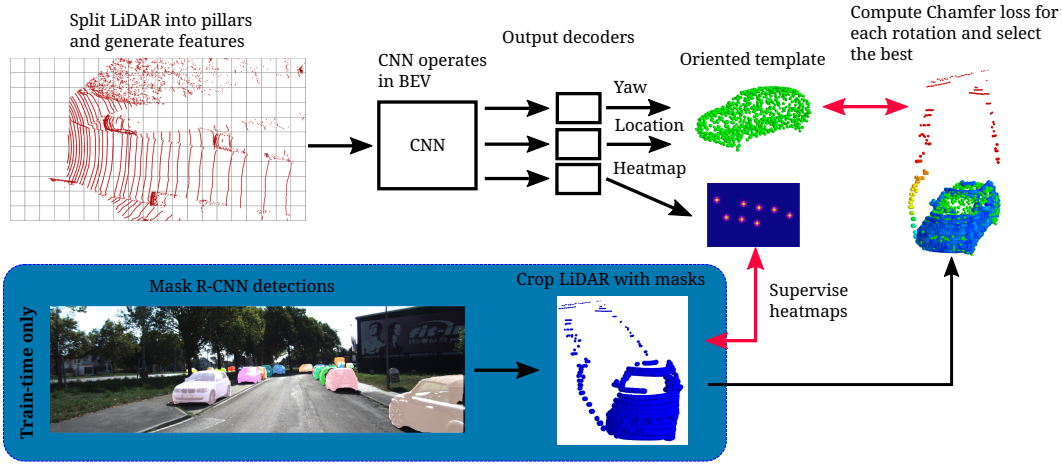

Figure 2: Our method only uses an off-the-shelf 2D object detector (bottom) to train a LiDAR 3D object detector (top), yet still achieving comparable accuracy to full supervision by laborious and costly 3D annotations.

away from the ego vehicle. As a result of this ambiguity we choose to utilise a 3D rigid model M of a car with a typical car size and shape, which we fit to observed LiDAR points L_m. The model $M = \{p \in \mathbb{R}^3\}$ is translated to a world location and rotated using the translation T and orientation θ, assuming that the translation and the orientation is assumed to be the car position and rotation (yaw) respectively.

We then define the distance measure between the translated model $\mathcal{T}_{T,\theta}(M)$ and the observed object point cloud L_m as

$$d\big(L_m, M \mid T, \theta\big) = \frac{1}{|L_m|} \sum_{p \in L_m} \min_{p' \in \mathcal{T}_{T,\theta}(M)} \|p - p'\|^2 \tag{4}$$

This distance measurement is similar to Chamfer distance which measures the distance between point clouds A and B by taking the closest point in set B for each point in A and vice versa, however in our case we only measure distance for observed every point in L_m as measuring distance also for every model point M would not work when only part of the object is captured as is often the case.

Note that the distance in Eq. 4 is fully differentiable with respect to the translation/rotation parameters T, and therefore it might be natural to ask if we could just iteratively search locations and compute this metric then picking the location which minimizes the distance. This however results in poor performance as outlier LiDAR points shift our prediction away from the expected location and when the available points represent a small section of the the target vehicle this distance can be unstable.

We therefore propose sharing information on object locations across frames by using a deep neural network Ψ which takes a LiDAR point cloud of the whole scene and generates a set of object detections $\Psi : L \rightarrow \{(T_i, \theta_i)\}$, where each detected object i is encoded by a 3D position of its center T_i and an orientation θ_i. We then define a loss $\mathcal{L}$ for one scene as

$$\mathcal{L}(\Psi \mid L, D, M) = \frac{1}{|D|} \sum_{m \in D} d(L_m, M \mid T_m, \theta_m) \quad (T_m, \theta_m) \in \Psi(L) \tag{5}$$

and we train the network Ψ by optimizing the loss over the whole dataset.

3.2 Soft Inlier Count metric

While sharing knowledge of locations across frames allows for a more robust estimation with partial point clouds, another issue still exists: often additional LiDAR points not belonging to the car are mistakenly included in the L_m subset, which skews the distance measurement and the resulting

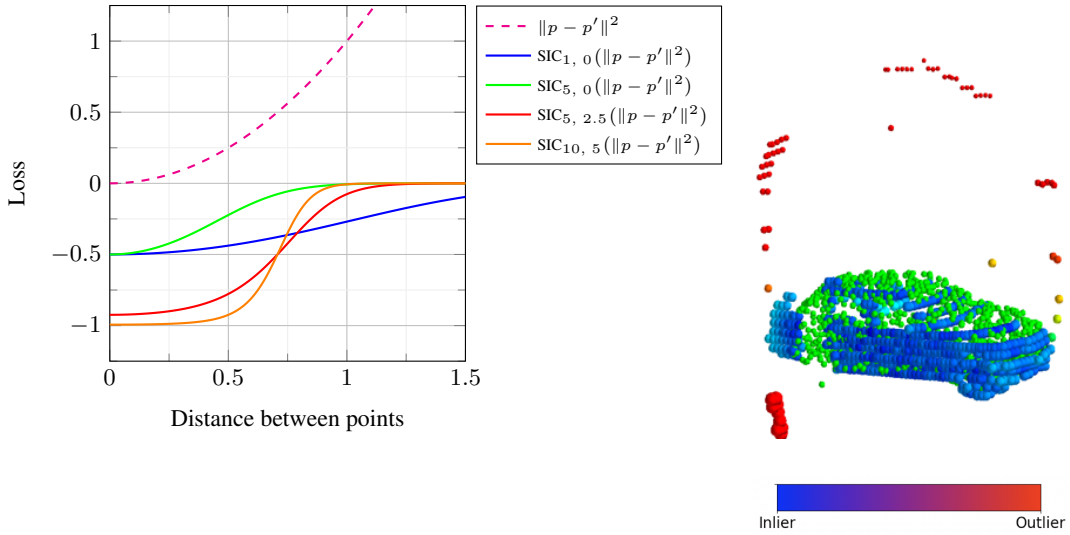

Figure 3: *Soft Inlier Count (SIC) loss*. Using L2 norm directly results in outlier points having a disproportionate influence on the loss minimising location. Different parametrisations of the SIC loss (left). An example matching of the 3D rigid model (green) to the detected object point cloud, with SIC loss value illustrated by the colour of each LiDAR point (right).

location/rotation estimate. Even the best 2D Object Detectors will make some erroneous predictions at a pixel level, caused by occluders of similar appearance, or LiDAR not reflecting properly. These points have a large value in our loss function for a correctly positioned model M which results in the outliers pulling the prediction away from the correct location even if they are small in number (see Fig. 3 - dashed line).

To address the outlier issue, we propose a new *Soft Inlier Count* (SIC) loss to soften the L2 metric of Eq. 4 with a sigmoid function as

$$\bar{d}\big(L_m, M \mid T_m, \theta_m\big) = \frac{1}{|L_m|} \sum_{p \in L_m} \sum_{p' \in \mathcal{T}_{T_m,\theta_m}(M)} \mathrm{SIC}_{\alpha,\beta}(p,p') \tag{6}$$

$$\mathrm{SIC}_{\alpha,\beta}(p,p') = -\frac{1}{1+\exp\big(-\alpha\|p-p'\|^2+\beta\big)} \tag{7}$$

where α and β are method parameters whose value is determined empirically (see Sec. 4.2) and again $\mathcal{T}_{T,\theta}(M)$ denotes the rigid model M translated by T and rotated by θ.

As a result, instead of using the raw L2 distance between the LiDAR and template rigid model we maximize the number of points sufficiently close to the template at a given location, with the assumption that most of the LiDAR points in the mask fall on the object we are interested in. We then sum across the entire set of LiDAR points L_m selected by the 2D Object Detector, which gives a soft count of how many points LiDAR points are close to the model M as well as giving an idea of quality, while still being fully differentiable.

3.3 Model Architecture

Our model is based on the CenterPoint architecture Yin et al. [2021], where LiDAR points are first separated into voxels of infinite height, then are fed through a network to generate features for this spatial volume. These feature maps are then scattered into a sparse Birds Eye View map of features upon which a CNN is run. We make two notable improvements to the CenterPoint architecture:

Multi-cell Voting Scheme. The original CenterPoint architecture uses the 3D ground truth center location to supervise the center heatmap head, where it selects a single cell of the heatmap as the positive target while the rest of the heatmap is taken as negative. This hard choice makes sense when 3D ground truth is available, however in our case as the 3D information is unavailable and therefore we need to take into account that the initial estimate of the object center might be well off, especially when the object point cloud is incomplete as this will skew the center estimate towards the visible

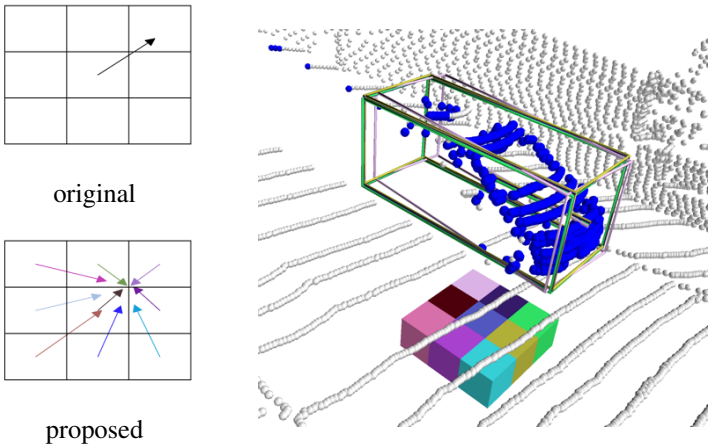

Figure 4: *Multi-cell Voting Scheme.* Rather than making a single hard prediction (top-left), our method produces multiple hypotheses by also making predictions in neighbouring cells of the feature map (bottom-left). A sample scene point cloud with the neighbouring cells and their corresponding predicted 3D bounding-boxes (right); colour of each bounding box encodes the source cell which produced it.

part. To mitigate this issue instead of making a hard choice of center we instead initially pick the median of the object point cloud and use this to select a neighbourhood of cells (see Fig. 4). During a forward pass the prediction made by each of these cells is evaluated and the center of the best prediction (= the prediction with the lowest loss value) is used as an updated center.

Yaw Estimation. Estimating rotation angle of the object (= yaw) is a surprisingly complex task with a selection of methods to predict such quantities available. Regression is problematic as a prediction near $-\pi$ for an object with orientation near π causes a loss value much higher than the actual error. CenterPoint Yin et al. [2021] instead predicts a vector $\big(\sin\theta, \cos\theta\big) \in [-1, 1]^2$ to address this boundary issue. Whilst such a formulation works well if the target (true) orientation is known at training time, when the ground truth is not available inherent ambiguities create deep local minima of the loss function which make the training unstable. The most common ambiguity is caused by the fact that for partial object point cloud all four $90°$-step rotations typically have a very similar loss.

In order to help the model not to get stuck in these local optima, at training time in every iteration we instead forcibly try all possible orientations and pick the orientation with lowest loss value as the target. To make this computationally feasible, we quantise the space of all orientations $[-\pi, \pi]$ into 64 distinct bins and transform the yaw prediction into a 64-way classification. More formally, the final loss function $\bar{\mathcal{L}}(\Psi \mid L, D, M)$ is

$$\theta^*_m = \underset{\theta' \in \mathcal{R}}{\operatorname{argmin}}\ \bar{d}\big(L_m, M \mid T_m, \theta'\big) \tag{8}$$

$$\bar{\mathcal{L}}(\Psi \mid L, D, M) = \frac{1}{|D|} \sum_{m \in D} \Big(d(L_m, M \mid T_m, \theta^*_m) + \mathrm{CE}(\theta_m \mid \theta^*_m) \Big) \quad (T_m, \theta_m) \in \Psi(L) \tag{9}$$

where $\mathcal{R}$ represents the set of possible rotations and CE denotes the standard cross-entropy loss.

To summarize, our model takes a set of point cloud detections $\big\{(x, y, z, r) \in \mathbb{R}^4\big\}$ as the input (where r denotes the reflectance value) and it outputs a dense feature map $\mathcal{M} \in \mathbb{R}^{H \times W \times F}$, consisting of a center heatmap head $[H \times W \times 1] \in (0, 1)$, a regression head $[H \times W \times 3] \in \mathbb{R}^3$ encoding the object center 3D location offset from the corner of the cell, and the orientation head $[H \times W \times 64] \in \mathbb{R}^{64}$. The size of the feature map is $H = 200$ and $W = 176$, therefore one cell corresponds to the size of 40cm in world coordinates when 10cm voxel edge length is used (as the convolutional network downsamples 4 times).

4 Experiments

4.1 Training and Inference

The KITTI Object Detection dataset Geiger et al. [2012] has 7481 publicly available 3D annotated frames. This is usually split into roughly equal-sized training and validation sets with no sequence of frames present in both originally created in Chen et al. [2017]. This allows us to compare to both fully-supervised existing methods and also the weakly-supervised works Zakharov et al. [2020b], Qin et al. [2020], McCraith et al. [2022] who only present results on this dataset. The standard evaluation in Birds Eye View (BEV) IoU with a strict threshold of 70% is used.

To prepare our dataset Mask R-CNN He et al. [2017], Wu et al. [2019] is used to retrieve objects in image space and generate the masks of point clouds which each detection will be compared to. The entire collection of points is augmented with rotation, translation, left-right flipping and scaling. The points are then sorted into the relative voxel bins which are fed into the network. Each LiDAR mask is compared to the regression and yaw classifier outputs of the corresponding output cells. During inference only the LiDAR data is utilised and detection is performed by the predicted heatmap with Non-maxima Supression (NMS) to prevent each nearby cell predicting the same car.

To evaluate our method we compare to all relevant weakly-supervised 3DOD methods that provide results on the KITTI dataset (see Tab. 1). Compared to McCraith et al. [2022], our method performs significantly better and it only requires the availability of LiDAR at test time, the benefit of this being that our detector is forced to reason about the shape of cars in the input LiDAR, rather than simply attempting to fit the part of the frustum with the greatest density. Zakharov et al. [2020b] pre-train their network on synthetic data and have a slow optimisation step for each instance detected by Mask R-CNN. Because of this they use their method to generate autolabels and feed these into PointPillars Lang et al. [2019], a very similar architecture to CenterPoint Yin et al. [2021], however they ignore more difficult image detections and still struggle with high IoU thresholds. For reference and to evaluate the gap between full and weak supervision, we also include results using the same underlying backbone Yin et al. [2021] as our method. We observe that compared to previous weakly-supervised methods, the gap is significantly smaller. We also observe that when removing the prediction of car size our method is remarkably close to a fully supervised network in the easy case, with a drop off in the moderate and hard cases, likely owing to the often extremely low number of LiDAR Points available for such car instances making it difficult for our inlier counting method to determine a meaningful fit.

4.2 Ablations

Soft Inlier Count Loss. To evaluate how well our mesh describes the observed point cloud we use must ensure the locations predicted result in an object at locations where LiDAR has collected data and are determined to belong to a car using the image mask. Chamfer distance or its one-sided variant are commonly used to compare point clouds, but using the standard L2 loss is problematic as presence of outlier points will disproportionately change the location of the associated vehicle, resulting in suboptimal performance (see Tab. 2 - first row). By using the newly introduced Soft Inlier Count (SIC), we observe this quantitatively in Tab. 2 that the accuracy is better, and we found that although

Method	Supervision	AP_{BEV} (IoU = 0.7)		
		Easy	Moderate	Hard
McCraith et al. [2022]†	2D detections	66.7	64.7	57.9
Zakharov et al. [2020a]	2D detections + synth 3D	81.00	59.80	-
ours†	2D detections	**86.39**	**74.79**	**67.31**
CenterPoint† Yin et al. [2021]	3D ground truth	88.08	83.41	80.72
CenterPoint Yin et al. [2021]	3D ground truth	90.84	84.45	82.30

Table 1: Weakly-supervised 3D Object Detection accuracy on KITTI validation set. Fully-supervised CenterPoint Yin et al. [2021] included for reference, methods using average 3D bounding-box instead of predicting actual size denoted with †. 2D detection networks are trained on Cityscapes Cordts et al. [2016] (McCraith et al. [2022]) or COCO Lin et al. [2014] (Zakharov et al. [2020b])

the SIC loss is not extremly sensitive to specific parameter settings, the best parameter values were determined to be $\alpha = 5$ and $\beta = 0$. We also observe that for high α values the loss becomes too strict for our use case, as it becomes impossible to precisely match our rigid one-size-fits-all car model to actual LiDAR detections.

Loss Function	AP_{BEV} (IoU = 0.7)		
	Easy	Medium	Hard
$\|p - p'\|^2$	69.14	61.64	52.11
$SIC_{1,\,0}(\|p - p'\|^2)$	79.23	67.26	59.69
$SIC_{5,\,0}(\|p - p'\|^2)$	80.17	69.58	63.73
$SIC_{5,\,2.5}(\|p - p'\|^2)$	79.91	69.88	63.19
$SIC_{10,\,5}(\|p - p'\|^2)$	72.94	64.08	56.26

Table 2: Ablation of the *Soft Inlier Count* (SIC) loss with different parameter values and its comparison to the standard L2 loss on the KITTI validation set.

Window Size	AP_{BEV} (IoU = 0.7)		
	Easy	Medium	Hard
0	80.17	69.58	63.73
1	84.88	74.53	67.16
2	86.39	74.79	67.31
3	82.05	72.26	66.58

Table 3: Ablation of the multi-cell voting scheme. Only a single heatmap cell (window size $= 0$) as in CenterPoint Yin et al. [2021] performs worst and too big window sizes create ambiguities, eg. when cars are parked closely.

Multi-cell Voting. Normally in anchor/location based predictions such as PointPillars Lang et al. [2019] and CenterPoint Yin et al. [2021] the object center is known and can be used to assign a heatmap location and choose which the corresponding cell in the predictions to penalise in order to calculate the loss. In our case however the center is unknown, which means some approximation must be used. Regardless of how this center is chosen as it can only be based on the partial captures of the LiDAR is will inherently be biased in some way, depending on how much of the car is visible/detected by the mask. Quantitatively we observe in Tab. 3 that using only a single cell indeed has sub-optimal accuracy, whilst on the other end using too large window of cells becomes too large and causes ambiguities when objects are placed close together, as now multiple cars can potentially be predicted by the same cell. To summarize, in our method we opt to use window size of 2 (ie. 5×5] cells) as it provides decent accuracy boost while being significantly faster to train than larger window sizes.

5 Conclusion

In this work, we utilised a readily available robust 2D Object Detector to transfer information about objects from 2D to 3D, allowing us to train a 3D Object Detector without the need for any human annotation in 3D. We demonstrated that our method significantly outperforms previous 3DOD methods supervised by only 2D annotations, and that our method narrows the accuracy gap between methods that use laborious and costly 3D supervision and those that do not.

Acknowledgement

We are very grateful to Continental Corporation for sponsoring this research. Lukas was supported by OP VVV funded project CZ.02.1.01/0.0/0.0/16019/0000765 “Research Center for Informatics”. Robert gratefully acknowledges support from the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines & Systems.

References

Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019.

Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. CVPR*, 2016.

Di Feng, Lars Rosenbaum, Fabian Timm, and Klaus Dietmayer. Labels Are Not Perfect: Improving Probabilistic Object Detection via Label Uncertainty. *arXiv e-prints*, art. arXiv:2008.04168, August 2020.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *Proc. ICCV*, 2017.

Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Robert McCraith, Eldar Insafutdinov, Lukas Neumann, and Andrea Vedaldi. Lifting 2d object locations to 3d by discounting lidar outliers across objects and views. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2411–2418. IEEE, 2022.

Kevin McNamara. Parallel domain: Data generation for autonomy. `https://paralleldomain.com/`, 2020.

Qinghao Meng, Wenguan Wang, Tianfei Zhou, Jianbing Shen, Luc Van Gool, and Dengxin Dai. Weakly supervised 3D object detection from LiDAR point cloud. In *ECCV*, 2020.

Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *arXiv preprint arXiv:1711.08488*, 2017.

Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6134–6144, 2021.

Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3D object detection from point clouds. In *Proc. ACMM*, 2020.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

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

Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.

Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2020a.

Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *Proc. CVPR*, 2020b.

Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.