MaskBEV: Joint Object Detection and Footprint Completion for Bird's-eye View 3D Point Clouds

William Guimont-Martin[1], Jean-Michel Fortin[1], François Pomerleau[1], Philippe Giguère[1]

***Abstract*— Recent works in object detection in LiDAR point clouds mostly focus on predicting bounding boxes around objects. This prediction is commonly achieved using anchor-based or anchor-free detectors that predict bounding boxes, requiring significant explicit prior knowledge about the objects to work properly. To remedy these limitations, we propose MaskBEV, a bird's-eye view (BEV) mask-based object detector neural architecture. MaskBEV predicts a set of BEV instance masks that represent the footprints of detected objects. Moreover, our approach allows object detection and footprint completion in a single pass. MaskBEV also reformulates the detection problem purely in terms of classification, doing away with regression usually done to predict bounding boxes. We evaluate the performance of MaskBEV on both SemanticKITTI and KITTI datasets while analyzing the architecture advantages and limitations.**

I. Introduction

Object detection in 3D point clouds is crucial to many applications of robotics and autonomous vehicles. Point clouds, captured by sensors such as LiDARs, provide accurate 3D information about the system's surroundings. However, it is more difficult to process them with deep neural networks in order to extract actionable semantic information. Indeed, point clouds, unlike images that are a dense regular grid of pixels, are irregular, unstructured and unordered [1]. Moreover, LiDAR point clouds suffer from multiple types of occlusion and signal miss [2]. One type of occlusion is *external-occlusion*, which is caused by obstacles blocking the laser from reaching the objects. *Self-occlusion* happens when an object's near side hides its far side. It is inevitable and will affect every object in a LiDAR scan. *Signal miss* can be caused by reflective materials reflecting the laser beam away from the sensor or by low reflectance. This often leads to objects appearing incomplete in LiDAR scans.

As such, 3D object detection models need to take into account these particularities of LiDAR point clouds to achieve good accuracy. This is especially true for object detection in an autonomous vehicle context, where only the part of the object facing the LiDAR will be visible. This makes objects incomplete and harder to detect [2].

Most recent 3D object detection architectures are based on predicting a set of 3D bounding boxes around objects. These methods can be separated into two main categories: *anchor-based* and *anchor-free* detectors. Anchor-based approaches use a predefined set of box proposals for potential object localization, along with their dimensions [2]–[8]. A notable drawback of this type of approach is the need to employ a large number of anchors, which are regulated by numerous hyperparameters related to their locations and sizes [9]. Anchor-free detectors instead directly predict the object's location and shape using keypoints from feature maps [9]–[19], such as the corners of the bounding box, or the object's center and size. Anchor-free detectors often heavily depend on post-processing steps such as non-maximal suppression (NMS), thresholding, and max pooling. These hand-crafted components (e.g., anchors in an anchor-based detector and NMS) explicitly encode prior knowledge about the objects [20] and thus require careful tuning.

To address these limitations and to further adapt object detection architectures to the particularities of point clouds, we propose MaskBEV. It is a bird's-eye view (BEV) mask-based 3D object detection neural architecture. Instead of

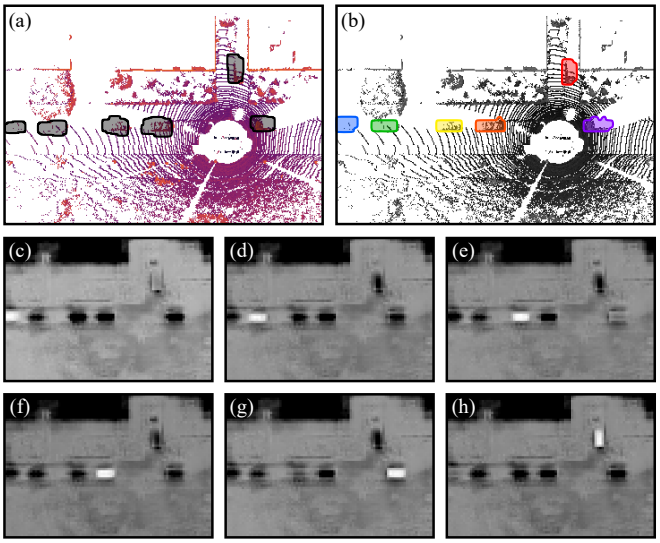


Fig. 1: **Mask prediction from MaskBEV.** (a) BEV of a point cloud from SemanticKITTI's validation set. We overlay the ground truth masks capturing each object's footprint on the point cloud. (b) Mask predictions from MaskBEV overlayed on the input point cloud, each instance is predicted on a different mask, here shown in different colors. (c-h) Mask predictions before applying the sigmoid and thresholding operation. We clipped and normalized the raw predictions to make them easier to visualize. We can notice the black outlines of cars not predicted by a particular mask. This means that each query token specializes in detecting one instance while suppressing the others.

[1] The authors are with Northern Robotics Laboratory, Université Laval, Québec City, Canada,
william.guimont-martin.1@ulaval.ca,
philippe.giguere@ift.ulaval.ca

∗ The source code for this paper is available here: https://github.com/norlab-ulaval/mask_bev

predicting bounding boxes, it predicts a BEV instance mask capturing the footprint of each object as shown in Figure 1, along with their class. Our approach allows us to *simultaneously detect and perform object completion*, making our detection framework robust to occlusions and signal misses. Importantly, our goal is to introduce a new 3D object detection paradigm based on masks instead of bounding boxes, as opposed to simply demonstrating superior accuracy over state-of-the-art models. In short, our contributions are:

- a novel architecture, MaskBEV, demonstrating the advantages of 3D object detection based on masks;
- an algorithm to generate mask labels from both bounding boxes and semantic labels; and
- an evaluation of our approach for vehicle detection on KITTI [21] and SemanticKITTI [22] datasets.

II. Related Work

The prevailing approaches for detecting 3D objects involve predicting a set of bounding boxes. These methods fall into two primary types: anchor-based and anchor-free detectors. After highlighting the challenges of both these methods, we review mask-based detection.

A. Anchor-based Detection

Anchor-based detection methods rely on a predefined set of box proposals – anchors – to predict the location and dimensions of objects. The use of anchors comes in two flavors: *single-stage* and *two-stage* methods.

Single-stage methods regress 3D bounding boxes in a single pass using anchors. These anchors serve as reference points, as the final bounding box geometry will be regressed as offsets from these [23]. In the context of 3D object detection, the anchors usually encode prior knowledge about the possible object locations per class, their dimensions, their offset with respect to the ground, and their viewing angle [7]. Multiple solutions have been proposed following this approach, namely VoTr-SSD [6], PointPillars [7], 3DSSD [17], SE-SSD [8], and GLENet [24].

Two-stage methods are largely based on Faster R-CNN [25] which starts with a first stage where regions of interest (RoIs) are extracted from the input scene. This is accomplished by a Region Proposal Network (RPN), which uses anchors to define putative regions where an object instance might be present. Based on this principle, methods such as Fast Point R-CNN [3], PV-RCNN++ [4], Voxel R-CNN [5], and BtcDet [2], work by using sub-networks to propose RoIs directly in point clouds. This allows the extraction of region-wise information that is then classified and used to regress bounding boxes. This usage of both classification and regression is typical for bounding box predicting methods.

While achieving competitive performance in 3D object detection, anchor-based methods inherently suffer from several drawbacks. First, these methods require a large number of anchor boxes to cover possible object locations and sizes. As such, it introduces numerous hyperparameters to be tuned, namely the number of boxes, their sizes, and aspect ratios [9]. Second, the detections are predicted via regression with respect to these anchors. Therefore, the quality of anchors – how well they are aligned with objects – has a significant impact on model performances [20]. Third, it complexifies the detection pipeline by adding a proxy task, the bounding box detection, compared to a single, unified model that directly predicts the presence or absence of an object in a particular location. Fourth, bounding boxes might also contain parts of other objects. This is particularly problematic if the anchors are axis-aligned, as it will have difficulties with diagonal elongated objects [26], such as large vehicles.

Our MaskBEV approach, in contrast, directly predicts masks instead of bounding boxes, circumventing the need for anchors. Consequently, it does not require any prior knowledge about the location and size of objects, while reducing the number of hyperparameters to tune. Also, we reformulate detection purely in terms of classification without resorting to regression, which is often considered problematic [27].

B. Anchor-free Detection

Anchor-free detection predicts an object's location and size using keypoints locations around objects. CenterNet [10], which detects objects in images by their center, inspired many subsequent 3D object detection architectures such as CenterNet3D [11], AFDetV2 [12] HotSpotNet [13], CenterPoint [14], SWFormer [16] and MGAF-3DSSD [15]. Other methods instead predict the corners of 3D bounding boxes [18] or points defining the characteristic L-shape of vehicles viewed from the side in a bird's-eye view point cloud [19].

Anchor-free detection avoids the limitation of anchor-based detection. Instead, they often depend on complex post-processing steps to extract peaks in feature heatmaps (e.g., NMS, thresholding and max pooling) and rules to assign ground truth to predictions to ensure the quality of the predictions [20]. These processes improve the models' prediction and include additional design decisions that need to be tuned. MaskBEV can side-step the need for such post-processing and complex ground truth assignation rules with the help of mask-based detection.

C. Mask-based Semantic and Instance Segmentation

MaskFormer [28] and Mask2Former [29] are two recent semantic and panoptic segmentation models, designed for computer vision. MaskFormer's insight, from which Mask2Former builds onto, is to consider this segmentation task as a mask classification instead of a per-pixel classification. Pixel classification seeks to predict the class of each pixel of an image. In contrast, mask classification predicts a set of binary masks, i.e., bundles of pixels or points, and a class is predicted for each mask. Ground truths are uniquely assigned to predictions using bipartite matching in a permutation-invariant manner. This differs from methods often used in anchor-free methods that match predictions to the closest ground truth or to predictions with an intersection over union (IoU) above a user-defined threshold. Masks with no matches are assigned a special `no_object` class label. Bipartite matching encourages predictions to be unique, thus eliminating the need for post-processing such as NMS [20].

MaskRange [30] reuses the same meta-architecture as Mask2Former but applies it to range images for semantic segmentation. Mask3D [31], SPFormer [32], and MaskPLS [33] all adapt the MaskFormer meta-architecture to produce point masks, i.e., a point-wise binary mask for each instance. Other approaches, not based on Mask2Former, also leverage mask prediction to do instance segmentation; 3D-BEVIS [34], 3D-BoNet [35] and DyCo3D [36] show the potential of mask-based detection for instance segmentation in LiDAR data. However, they developed their approach for reconstructed indoor point clouds, and thus require point clouds with less occlusion than single-pose LiDAR scans.

Our MaskBEV leverages mask predictions to detect vehicles in LiDAR point clouds. Consequently, instead of predicting a set of bounding boxes around each vehicle, MaskBEV predicts a bird's-eye view mask for each detected instance. This allows MaskBEV to do away with the post-processing steps required by anchor-based methods, and to be more robust to occlusion inherent to single-pose LiDAR scans. Moreover, the use of BEV instance masks allows us to do object completion simultaneously with object detection, thus finding the complete footprint of a vehicle. Masks also show themselves to be more flexible than bounding boxes: they allow capturing objects of any geometry, not just rectangular objects. MaskBEV also eliminates the need for prior knowledge about objects' location and dimensions to be baked into the network (e.g., anchors, post-processing steps); everything is learned directly from data. Mask-based object detection also removes the need for regression and instead reformulates the instance detection task purely in terms of classification.

III. MaskBEV

MaskBEV is a novel object detection and footprint completion architecture for LiDAR point clouds using BEV masks. It is based on transformer networks in order to process global information by cross- and self-attention over the entire BEV. At the moment, we strictly target vehicles, as transformer-based networks primarily help with the detection of large objects [20]. From a point cloud, we predict a binary BEV mask and a class for each detected instance. As shown on Figure 1, each mask represents the top-down view of an object's footprint. These masks are complete, meaning that they capture the full footprint of the instances, as shown in Figure 2d. This effectively reformulates 3D object detection as a classification-only task, as masks are not regressed. Moreover, it allows for object completion of arbitrary shapes. This is due to the transformer's philosophy of reducing the number of inductive biases baked into the network [37]. Consequently, the completed shape prior is provided by the ground truth annotation in the form of masks.

MaskBEV, as depicted in Figure 3, is split into two main components: an encoder and a mask prediction module. The encoder transforms 3D point clouds into BEV images, allowing us to reformulate the detection task to a computer vision one. The mask-prediction module first extracts multi-scale features and then predicts a set of up to M masks, along with their class (i.e., a binary classification of whether there is a car or not in the corresponding mask). These two components are described in more detail below.

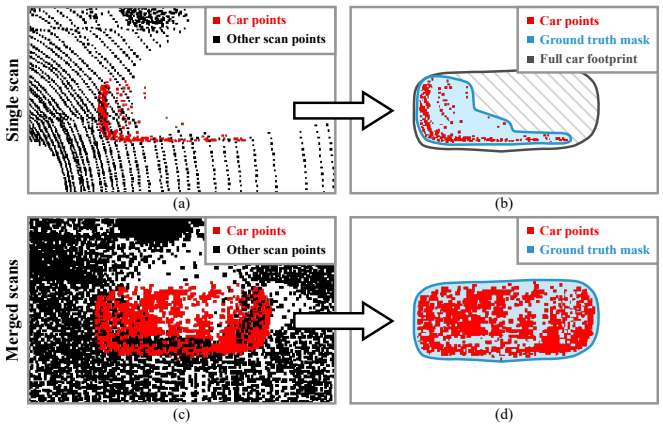


Fig. 2: **Mask generation from instance label.** (a) Single scan point clouds only show the surface of the object that directly faces the LiDAR. (b) The mask generated from a single scan is partial and does not represent the complete footprint of the vehicle. (c) Using merged sequential LiDAR scans, it is possible to gather points from all around a vehicle. (d) Masks produced from the constructed map are complete, i.e., represent the entire footprint of the vehicle.

A. Encoder

The encoder of MaskBEV transforms point clouds into BEV images in a similar fashion to PointPillars [7]. First, we sparsely voxelize the input 3D point cloud into a 2D grid along the ground plane in a region around the LiDAR, sampling up to 32 points per voxel. Then, we augment each point with voxel-relative information, resulting in a ten-dimensional vector: three for the (x, y, z) position, one for the distance of the voxel from the origin, three for the offset of each point from its voxel's center, and three for the offset from the arithmetic mean of all the points in the same voxel. We also concatenate the return strength of the laser – a value between 0 and 1 – if available. Finally, we generate the BEV image by projecting all voxels using a multilayer PointNet [38] followed by layer normalization. The resulting image is of size $(F \times H \times W)$, where F is the embedding dimension of voxels, H and W are the number of voxels along the y-axis and x-axis, respectively.

B. Mask2Former

Our mask-predicting module is based on Mask2Former [29]. Similarly, we extract multi-scale features from the point cloud BEV using a Swin-T backbone [39]. We use learnable absolute positional encoding to inform the network about the voxels' position relative to the LiDAR. The backbone produces feature maps of sizes $(S, H/2, W/2)$, $(2S, H/4, W/4)$, $(4S, H/8, W/8)$, and $(8S, H/16, W/16)$, where S is the dimension of the multi-scale feature embedding. Typically, S is larger than F (i.e., the number of channels in the BEV image) since it needs to capture the relationship between voxels.

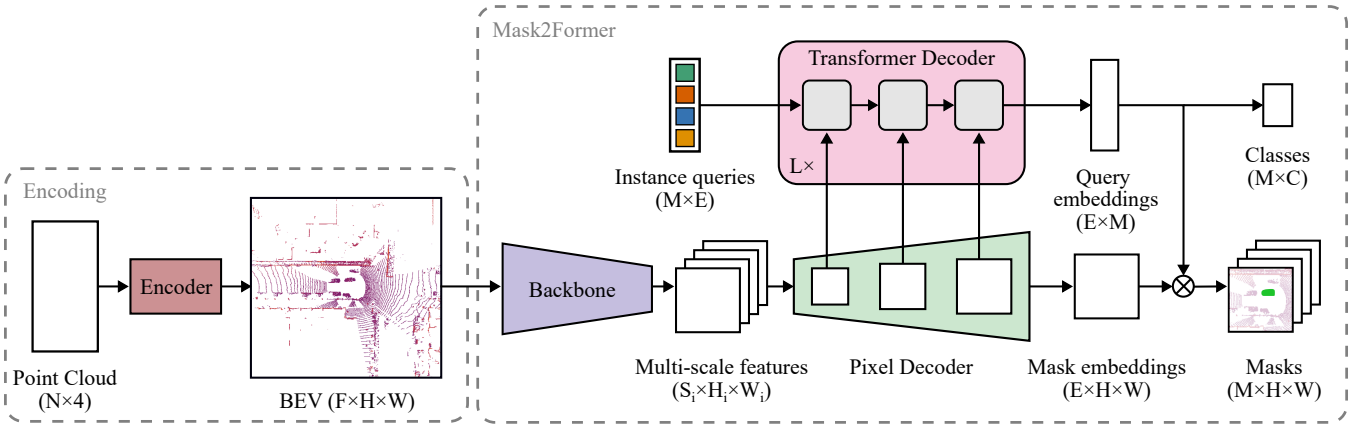


Fig. 3: **MaskBEV complete architecture.** It has two main parts: an encoder and a mask prediction network. The encoder is responsible for converting a 3D point cloud into a BEV feature map. Then, this feature map is fed into a Mask2Former [29] network that outputs a set of classes prediction and binary BEV masks. Each class and mask pair represents a detection made by the network. These masks predict the footprint of each detected instance.

The multi-scale feature maps produced by the backbone are upsampled using a multi-scale deformable attention transformer from Zhu *et al.* [40], ultimately producing a mask embedding of dimension E. The intermediary upsampled feature maps are fed into a L-layers transformer decoder and queried from a set of M queries tokens of dimension E, which in turn produce an embedding for each possible mask detection. The query embeddings are then projected and correlated with the mask embeddings to get, respectively, a set of M classes and the instance masks. We then apply a sigmoid activation on the mask embeddings followed by a threshold of 0.5 to get the binary masks representing the footprint of detected objects.

C. Mask Ground Truth Generation

Since MaskBEV requires mask supervision to both learn detection and footprint completion, we need to transform commonly available labels into a set of binary masks. We focus on bounding box and per-point instance labels. For bounding boxes from KITTI, we generate BEV mask images and draw the orthogonal projection of the bounding boxes onto the ground plane. For semantic and instance annotations, we generate a set of instance masks from labels in SemanticKITTI by voxelizing the point cloud in a BEV and using the voxels containing instance points to deduce the footprint of objects. However, since only the LiDAR-facing side of objects are visible, as shown in Figure 2a, this process would give incomplete BEV masks, as depicted in Figure 2b. To remedy this problem, we build a map from sequential scans, gathering points from multiple views around static objects, as shown in Figure 2c. We then produce a BEV mask from this map at the position of a LiDAR scan, which gives us a complete mask of objects, as illustrated in Figure 2d. We apply closing and opening morphological operations to clean the instance masks. These operations are followed by the filtering of masks whose area is smaller than a certain threshold. This removes instances that are never fully seen in the aggregated point cloud (i.e., vehicles that are too far away from any single scan). Due to this process, we limit ourselves to static vehicles in our training datasets (i.e., parked vehicles, for masks generated from semantic labels). This training method does not prevent MaskBEV from detecting mobile vehicles in single-scan point clouds from the test split. When we generate a mask label for a specific LiDAR scan, we only keep masks for instances that have at least one point in the point cloud. While the BEV instance mask labels are generated from multiple LiDAR scans, the network only receives in input a single scan.

This generation of complete mask labels enables MaskBEV to learn how to perform *object completion* in a data-driven manner. To do so, we train MaskBEV to predict the complete footprint of vehicles, even when only parts of the objects are visible.

IV. Experiments and Results

A. Datasets

We evaluate MaskBEV on two widely used datasets in the 3D object detection literature. We focus our experimentation on vehicle detection because they are larger objects that could benefit from our transformer-based architecture. We also experimentally validate the quality of the footprint completion, a key aspect of our proposed approach.

The KITTI Vision Benchmark Suite [21] is composed of a large quantity of data coming from diverse sensors installed on a standard station wagon. The dataset contains 16 142 vehicles in the training set, and 16 608 in the validation set, from all of which we generate BEV instance masks. For our research, we only use the data from the LiDAR (Velodyne HDL-64E) for car detection and the Inertial Navigation System (OXTS RT 3003) for localization. KITTI contains bounding box annotations that are limited to what is visible from the image plane, meaning that vehicles behind the LiDAR are not labeled. For this reason, the point cloud is cropped to keep the points between [0, 80] m in the x-axis and [-40, 40] m in the y-axis, from the sensor's reference

frame. We use the bounding box labels to generate our training BEV instance masks.

The SemanticKITTI dataset [22] provides dense point-wise annotations for the whole field-of-view of the LiDAR, for all sequences of the KITTI Vision Odometry Benchmark. It consists of 19 130 scans from ten sequences in the training set, totaling 136 374 static vehicle instances. The validation set contains 4071 scans all taken from a single sequence, in which there are 37 280 instances of static vehicles. Since we have 360° labels this time, the point cloud is cropped to keep the points in a 40 m range around the LiDAR sensor.

B. Implementation Details

Our implementation of MaskBEV, represented in Figure 3, uses two PyTorch [41] libraries, `mmdetection`[1] and `mmdetection3d`,[2] respectively for their implementation of Mask2Former and PointPillars. The points are split using voxels of size 0.16 m, which results in feature maps of size $H = W = 500$, for both SemanticKITTI and KITTI. The encoder is a 3-layer PointNet which outputs a BEV representation with $F = 128$ channels. From this, a Swin-T backbone generates multi-scale features with $S = 192$ channels. The pixel decoder's output has $E = 256$ channels and the transformer decoder uses $M = 45$ queries. The remaining parameters are taken from Mask2Former's default parameters [29]. It should be noted that we did not perform an extensive hyperparameters search.

C. Training Details

For training, we used the AdamW optimizer [42] with a learning rate of 1×10^{-4} and weight decay of 1×10^{-5}. We did not use a learning rate multiplier for the backbone, since we do not use pre-trained networks. Following MaskFormer, we multiply the loss of predictions assigned to `"no_object"` by 0.1. The models were trained on four Nvidia RTX A6000 GPUs, an AMD Ryzen Threadripper 3970X 32-core CPU, and 128 GB of RAM. We train on SemanticKITTI for 21 epochs for 60 hours, and 50 epochs on KITTI in 22 hours.

For the SemanticKITTI dataset, we applied the following data augmentation: we randomly drop 5 % of points, apply random flipping along the y axis (i.e., a left-right swap from the vehicle's perspective), and randomly translate individual points with noise sampled from $\mathcal{N}(0, 0.2)$. For KITTI, we needed stronger data augmentation to counter effect the lower number of vehicles per scan. In addition to the augmentation used for SemanticKITTI, we also used the following data augmentation inspired from Lang *et al.* [7]: adding up to ten instances from other scans, making sure not to collide with existing instances; randomly translating and rotating vehicle instances, limiting the displacement to 0.25 m and the rotation to 9 degrees; randomly translating the full point cloud by a maximum of 0.2 m; and randomly rotating the full point cloud by a maximum of 2.5 degrees.

[1] https://github.com/open-mmlab/mmdetection
[2] https://github.com/open-mmlab/mmdetection3d

D. Results on SemanticKITTI

Since most semantic and instance architectures evaluate their approach using a per-point mean intersection over union (mIoU) for SemanticKITTI, there is no direct comparison for MaskBEV with other competing approaches. This limitation is due to our solution being the first one predicting complete BEV masks and not a per-point prediction. Consequently, we report usual detection metrics based on binary mask IoU, thereby establishing a novel baseline for mask-based detection on 3D point clouds. We evaluated MaskBEV on all vehicle instances that are visible in the point cloud no matter the amount of occlusion (i.e., at least one point of the vehicle is present in the LiDAR scan). In Table I, we present AP50, AP70, mAP and mIoU for mask predictions on SemanticKITTI's validation split. It should be noted that the BEV prediction task is more difficult than bounding box prediction, due to the more flexible nature of masks.

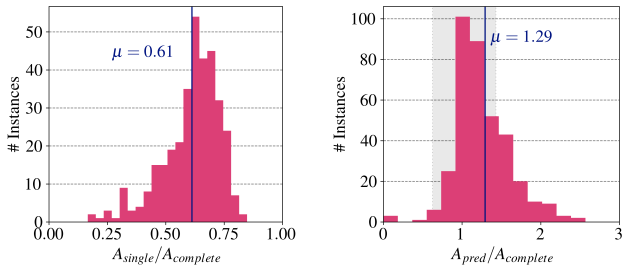
TABLE I: Results of mask detection metrics on the SemanticKITTI's validation split.

Model	AP50	AP70	mAP	mIoU
MaskBEV (ours)	77.68	63.11	48.53	69.93

1) Qualitative Results: We show in Figure 1 examples of mask predictions on a sample of point clouds from SemanticKITTI's validation split. Figure 1a presents the input point cloud with the instance mask labels overlayed. Figure 1b shows the network's predictions. Figure 1c-h show the normalized masks before the sigmoid and the thresholding operation. Interestingly, we can observe black outlines of cars in these raw mask predictions. They correspond to other vehicles that the network suppressed in order to predict a single instance. This suppression seems to indicate that MaskBEV is able to leverage the *global structure* of the scene, as well as other detections, to improve performance.

We show in Figure 5 (a-c) and (e-g) examples of predictions made by MaskBEV on SemanticKITTI. We can see that MaskBEV's predictions are more accurate on simpler point clouds, such as single roads, but struggles on more complex scene structures such as intersections.

2) Mask Completion: We provide here an analysis of the mask completion capabilities of MaskBEV. In Figure 4a, we compare ground truth masks generated from a single scan and merged scans in SemanticKITTI. To achieve this, we compute the ratio between the area of the instance masks generated from a single scan for one instance, shown in blue in Figure 2b, to the area of the complete mask, outlined in gray in the same figure. We compute this metric for all instances over all scans, and only keep the largest value per instance. In other words, we only consider the best case, i.e., the scan in which the vehicle is the most visible for each instance. This ratio demonstrates how well a single scan can capture the actual footprint of objects in the ideal case. We ignore here instances only containing a few points per scan, and thus having a low single-scan area. The histogram



(a) (b)

Fig. 4: (a) Histogram of the ratio between the area of the largest mask generated from a single scan, A_{single}, to the area of the complete mask of the same instance, $A_{complete}$, for SemanticKITTI's validation split. Lower values of ratios indicate instances that are not well captured from any single scan. (b) Histogram of the ratio between the area of our prediction, A_{pred}, to the area of the complete mask $A_{complete}$ for the same instance, for SemanticKITTI's validation split. Most predictions are larger than their corresponding ground truth (i.e., ratios bigger than one), meaning that MaskBEV tends to overestimate the footprint of instances.

in Figure 4a shows that most single-scan instance masks computed from the ground truth data are incomplete; they have completion ratios between 0.2 and 0.8. Overall, the average ratio over all instances is 0.61, meaning that most single-scan masks only cover about $61\,\%$ of the complete footprint of the vehicles in the ideal case. If we include the instances with a low single-scan area (i.e., instances suffering from severe occlusion in every scan), the average ratio slightly decreases to 0.55. Nevertheless, our mask-generation algorithm still allows using these instances for training, since we gather points over a whole sequence and generate a complete mask. Training on such instances could help with detecting vehicles that suffer from severe occlusion or that are far away. If we instead consider the average ratio of all instances over all scans, and not only the ideal case, the average drops to 0.41 with a standard deviation of 0.15. This ratio shows that in the average case, points on an instance only capture about $40\,\%$ of the complete footprint, still proving that complete masks offer a clear advantage over single-scan masks.

In Figure 4b, we compare MaskBEV's predictions to the complete BEV instance masks. The histogram shows the ratio between the area of predictions to the area of the ground truth for each instance in SemanticKITTI's validation set. This metric gives insight about the relative size of our prediction to the actual size of the object. The average ratio is 1.29, meaning that MaskBEV's predictions are on average $29\,\%$ larger than ground truth labels. If we compute the IoU using only the mask areas (i.e., assuming otherwise perfect detection) and a detection threshold of $70\,\%$ for the IoU, ratios between $\frac{70}{100} = 0.7$ and $\frac{100}{70} \approx 1.43$, shown as a gray zone in Figure 4b, are considered as correct detections. Thus, we can deduce that larger predictions are an important source of error for MaskBEV as the histogram is skewed towards larger values. Larger detections are not inherently detrimental in an autonomous vehicle context, where a detected object's enlarged footprint would result in a more conservative, safer path planning.

E. Results on KITTI

In Table II, we evaluate MaskBEV using AP70 on the different difficulty levels defined by KITTI and mIoU over mask predictions. As in the case of SemanticKITTI, our evaluation metrics are computed directly on the predicted masks, making a straightforward comparison with other methods difficult. Nevertheless, we show the performance on bounding box predictions for the state-of-the-art SE-SSD [8] approach. These results indicate that our approach is promising, but has significant room for improvement. Importantly, MaskBEV contains fewer inductive biases, either in its transformer backbone or the lack of rectangular anchors in the output. Consequently, we conjecture that its performance is greatly affected by the small size of the KITTI dataset. This phenomenon has already been seen in transformer-based neural networks in computer vision [43], where it is not uncommon to pre-train these networks on billions of images. We show qualitative examples of predictions in Figure 5 (d) and (h).

TABLE II: Mask detection metrics on KITTI. Results for MaskBEV are computed on mask predictions. Other results are from KITTI's BEV leaderboard for bounding boxes.

Model	**AP70 on cars**			**mIoU**
	Easy	**Moderate**	**Hard**	
MaskBEV (masks, ours)	72.22	71.02	54.39	64.79
SE-SSD [8]	95.68	91.84	86.72	-
PV-RCNN [44]	94.98	90.65	86.14	-
BtcDet [2]	92.81	89.34	84.55	-
PointPillars [7]	90.07	86.56	82.81	-

V. Conclusion

In this paper, we introduced a new 3D object detection paradigm for point clouds, based on BEV masks instead of bounding boxes. To do so, we developed the novel MaskBEV architecture, by combining a PointPillars encoder with Mask2Former. MaskBEV avoids common drawbacks of anchor-based and anchor-free detectors while remaining conceptually simple. Our method, by predicting complete BEV masks, allows object completion while simultaneously detecting them. This makes our approach particularly apt to deal with LiDAR scans fraught with occlusions and signal misses. We evaluated MaskBEV on SemanticKITTI and KITTI, with competitive results. Qualitative results indicate that MaskBEV is also able to incorporate global cues as well as the shape prior provided by the training data. All in all, these suggest that we can be optimistic about further developments of this paradigm of mask-based object detection in point clouds.

In terms of future work, we will extend MaskBEV to other types of objects such as pedestrians, cyclists and moving vehicles. We additionally plan on studying MaskBEV's

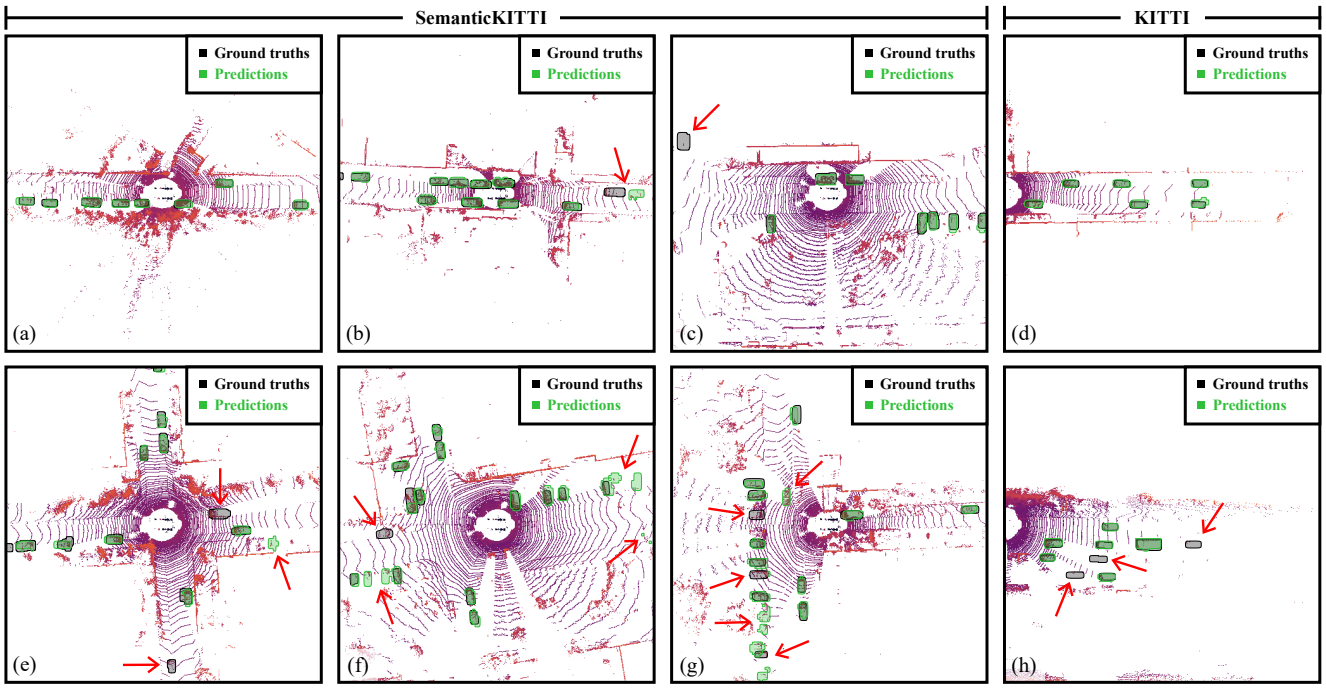


Fig. 5: **MaskBEV predictions on SemanticKITTI and KITTI datasets.** The first three columns are samples from SemanticKITTI, and the rightmost one is from KITTI. The top row shows a sample of good predictions from the network. The bottom row displays failure cases to analyze the limitations of MaskBEV. We see that the network struggles with more complex scenes such as (e), (f) and (g). Missed detections are emphasized by red arrows. Smaller ground truths (i.e., rectangles that are too small to be vehicles), are filtered out by the process described in Section III-C, and thus are not used for evaluation.

object completion capabilities, potentially developing data augmentation techniques that take advantage of this. We also need to further analyze how our transformer-based network can leverage both local and global scene structures. Since our approach has less inductive biases, we can expect it to greatly benefit from larger training datasets, e.g., KITTI-360 [45], Waymo Open Dataset [46] and nuScenes [47]. Finally, transformer-based approaches have been shown to be compatible with self-supervised learning. As such, we plan on testing Voxel-MAE [48] with our method, in order to reduce the need for labeled data.

References

[1] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, "Deep Learning on 3D Point Clouds," *Remote Sensing*, vol. 12, no. 11, p. 1729, 2020.

[2] Q. Xu, Y. Zhong, and U. Neumann, "Behind the Curtain: Learning Occluded Shapes for 3D Object Detection," in *AAAI Conference on Artificial Intelligence*, 2022.

[3] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast Point R-CNN," in *CVPR*, 2019.

[4] S. Shi, L. Jiang, J. Deng, *et al.*, "PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection," *International Journal of Computer Vision*, pp. 1–21, 2022.

[5] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection," in *AAAI Conference on Artificial Intelligence*, 2021.

[6] J. Mao, Y. Xue, M. Niu, *et al.*, "Voxel Transformer for 3D Object Detection," in *CVPR*, 2021.

[7] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in *CVPR*, 2019.

[8] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu, "SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud," in *CVPR*, 2021.

[9] H. Law and J. Deng, "CornerNet: Detecting Objects as Paired Keypoints," in *ECCV*, 2018.

[10] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," *arXiv preprint arXiv:1904.07850*, 2019.

[11] G. Wang, J. Wu, B. Tian, S. Teng, L. Chen, and D. Cao, "CenterNet3D: An Anchor Free Object Detector for Point Cloud," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 953–12 965, 2021.

[12] Y. Hu, Z. Ding, R. Ge, *et al.*, "AFDetV2: Rethinking the Necessity of the Second Stage for Object Detection from Point Clouds," in *AAAI Conference on Artificial Intelligence*, 2022.

[13] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, "Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots," in *ECCV*, 2020.

[14] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-Based 3D Object Detection and Tracking," in *CVPR*, 2021.

[15] J. Li, H. Dai, L. Shao, and Y. Ding, "Anchor-free 3D Single Stage Detector with Mask-Guided Attention for Point Cloud," in *ACM-MM*, 2021.

[16] P. Sun, M. Tan, W. Wang, *et al.*, "SWFormer: Sparse Window Transformer for 3D Object Detection in Point Clouds," in *ECCV*, 2022.

[17] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-Based 3D Single Stage Object Detector," in *CVPR*, 2020.

[18] R. Ma, C. Chen, B. Yang, *et al.*, "CG-SSD: Corner Guided Single Stage 3D Object Detection from LiDAR Point Cloud," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 191, pp. 33–48, 2022.

[19] T. Zou, G. Chen, Z. Li, *et al.*, "KAM-Net: Keypoint-Aware and Keypoint-Matching Network for Vehicle Detection From 2-D Point Cloud," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 207–217, 2021.

[20] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in *ECCV*, 2020.

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[22] J. Behley, M. Garbade, A. Milioto, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *ICCV*, 2019.

[23] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single Shot MultiBox Detector," in *ECCV*, 2016.

[24] Y. Zhang, Q. Zhang, Z. Zhu, J. Hou, and Y. Yuan, "Glenet: Boosting 3d object detectors with generative label uncertainty estimation," *arXiv preprint arXiv:2207.02466*, 2022.

[25] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *NIPS*, 2015.

[26] J.-M. Fortin, O. Gamache, V. Grondin, F. Pomerleau, and P. Giguère, "Instance Segmentation for Autonomous Log Grasping in Forestry Operations," in *IROS*, 2022.

[27] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," in *ECCV*, 2016.

[28] B. Cheng, A. Schwing, and A. Kirillov, "Per-Pixel Classification is Not All You Need for Semantic Segmentation," *NeurIPS*, 2021.

[29] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-Attention Mask Transformer for Universal Image Segmentation," in *CVPR*, 2022.

[30] Y. Gu, Y. Huang, C. Xu, and H. Kong, "MaskRange: A Mask-classification Model for Range-view based LiDAR Segmentation," *arXiv preprint arXiv:2206.12073*, 2022.

[31] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D for 3D Semantic Instance Segmentation," *arXiv preprint arXiv:2210.03105*, 2022.

[32] J. Sun, C. Qing, J. Tan, and X. Xu, "Superpoint Transformer for 3D Scene Instance Segmentation," *arXiv preprint arXiv:2211.15766*, 2022.

[33] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss, "Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving," *IEEE Robotics and Automation Letters*, 2023.

[34] C. Elich, F. Engelmann, T. Kontogianni, and B. Leibe, "3D Bird's-Eye-View Instance Segmentation," in *GCPR*, 2019.

[35] B. Yang, J. Wang, R. Clark, *et al.*, "Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds," *NeurIPS*, 2019.

[36] T. He, C. Shen, and A. Van Den Hengel, "DyCo3D: Robust Instance Segmentation of 3D Point Clouds Through Dynamic Convolution," in *CVPR*, 2021.

[37] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in Vision: A survey," *ACM Computing Surveys*, 2022.

[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *CVPR*, 2017.

[39] Z. Liu, Y. Lin, Y. Cao, *et al.*, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *ICCV*, 2021.

[40] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable DETR: Deformable Transformers for End-to-End Object Detection," *arXiv preprint arXiv:2010.04159*, 2020.

[41] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," 2019.

[42] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[43] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, "Scaling vision transformers," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2022.

[44] S. Shi, C. Guo, L. Jiang, *et al.*, "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," in *CVPR*, 2020.

[45] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[46] P. Sun, H. Kretzschmar, X. Dotiwalla, *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *CVPR*, 2020.

[47] H. Caesar, V. Bankiti, A. H. Lang, *et al.*, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *CVPR*, 2020.

[48] C. Min, D. Zhao, L. Xiao, Y. Nie, and B. Dai, "VoxelMAE: Masked Autoencoders for Self-supervised Pre-training Large-scale Point Clouds," *arXiv preprint arXiv:2206.09900*, 2022.