Single-Click 3D Object Annotation on LiDAR Point Clouds

Trung Nguyen **Binh-Son Hua** **Duc Thanh Nguyen** **Dinh Phung**
VinAI Research, Vietnam
{v.trungnd19, v.sonhb, v.thanhnd50, v.dinhpq2}@vinai.io

Abstract

We present a simple and effective tool for performing interactive 3D object annotation for 3D object detection on LiDAR point clouds. Our annotation pipeline begins with a pre-labeling stage that infers 3D bounding boxes automatically by using a pre-trained deep neural network. While this stage can largely reduce manual effort for annotators, we found that pre-labeling is often imperfect, e.g., some bounding boxes are missing or inaccurate. In this paper, we propose to enhance the annotation pipeline with an interactive operator that allows users to generate a bounding box for a 3D object missed by the pre-trained model. This user interaction acts in the bird's eye view (BEV), i.e., top-down perspective, of the point cloud, where users can inspect the existing annotation and place additional boxes accordingly. Our interactive operator requires only a single click and the inference is done by an object detector network trained on the BEV space. Experimental results show that, compared with existing annotation tools, our method can boost up the annotation efficiency by conveniently adding missing bounding boxes with more accurate dimensions using only single clicks.

1 Introduction

3D object detection is a crucial task for perception in autonomous driving. In recent years, many deep learning architectures have been proposed to solve this task and have achieved promising results [7]. However, an inherent challenge is that existing 3D object detection models are built upon supervised learning, and hence require a massive scale of annotated data for training the models. Large-scale data collection, preprocessing, and annotation are thus critical steps in the production pipeline in additional to model development and engineering. In the public domain, some open-source tools for annotating 3D objects have been developed such as 3D-BAT [16], LATTE [14], SAnE [1], and SUSTechPOINTS [8]. Those tools provide useful features to support user interaction during annotation such as one-click annotation and tracking. Having a robust annotation tool can lead to better data turnaround time, improving overall efficiency and effectiveness of the AI-based product development pipeline.

In this paper, we propose a single-click annotation feature for 3D object annotation on LiDAR data. In our data annotation pipeline, LiDAR data can be pre-annotated automatically using a pre-trained point cloud neural network, resulting 3D bounding boxes. Despite such, a common limitation is that 3D object detectors are not perfect; there remains unannotated objects that need to be inspected and validated manually. Existing single-click annotation in the current open-source tools could potentially handle remaining unannotated objects. However, this feature is often performed poorly as existing methods are mainly based on traditional unsupervised machine learning methods including clustering and shape fitting [14, 1]. These approaches are inaccurate and slow, and do not take advantage of publicly available annotated LiDAR data such as those from the public datasets like KITTI and nuScenes [5, 2].

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Table 1: Feature summary of 3D object annotation tools.

	3D-BAT [16]	LATTE [14]	SAnE [1]	SUSTechPOINTS [8]	Ours
Pre-labeling	Y	Y	Y	Y	Y
Box interaction	Manual	Cluster & fit	Denoise	Prototype boxes	BEV detector

Here we propose a different approach based on supervised learning. The basic idea is to leverage existing supervised learning 2D object detectors and the bird's eye view (BEV) for single-click annotation. Specifically, we propose to train a neural network that takes a BEV image as input and returns 2D bounding boxes that can be subsequently transformed into 3D bounding boxes for further annotation and editing. To this end, we make the following contributions: (1) An object detector in BEV for single-click annotation; (2) Evaluations and comparisons to previous interactive methods.

2 Related works

There have been some modern publicly available annotation tools for 3D point clouds in the recent years. Here we focus on methods that allow user interaction to generate 3D bounding boxes. For indoor scenes with point clouds captured by RGB-D data, Nguyen et al. [12] performed bottom-up clustering on the point cloud to construct 3D bounding boxes. For LiDAR point clouds especially for autonomous driving context, 3D-BAT [16] is a basic tool that only allows users to manually adjust the box dimensions. LATTE [14] adds more interactivity in the BEV by gathering points around the cursor position to form a cluster of points, and then fits a rectangle to that cluster to form a predicted box. SAnE [1] improves single-click annotation in LATTE by removing noisy points from the cluster before fitting the rectangle. SUSTechPOINTS [8] places a box prototype at the position users click, and then uses region growing and fitting algorithms to adjust the box.

A common limitation of these methods is that they are prone to errors when objects are partially captured. In such cases, it is still desirable to predict a bounding box that fits an object as if it were fully captured, often known as amodal detection. Clustering-based algorithms tend to wrongly predict the dimensions of bounding boxes while region growing algorithms often result in boxes that loosely fit the object of interest. In this work, we base on supervised learning and take advantage of available annotated 3D bounding boxes on LiDAR data so that amodal detection can be addressed by deep learning. Annotated 3D boxes are also available in many public datasets, e.g., [5, 2] and thus can be used for training detectors to support user interaction. A summary of existing open-source tools for 3D object annotation is shown in Table 1.

3 Method

Problem statement. The main operation of single-click annotation can be defined as follows: when a user performs a click on a target object, a 3D box is automatically generated to bound the target object. Mathematically, given a point cloud $P = \{x_i\}$ and a click position z, we fit the 3D bounding box parameterized by (c, d, θ) on the target object, where $c \in \mathbb{R}^3$ is the center of the box, $d \in \mathbb{R}^3$ represents the dimensions (i.e., width, height, length) of the box, and $\theta \in [0, 2\pi)$ is the yaw angle of the box. In the BEV of a point cloud, coordinates are in meters. In this work, we use a different convention for BEV representation by rasterizing the BEV and store it as *a BEV image*, where coordinates are measured by pixels.

3.1 BEV object detector

Our goal to train an object detector in the BEV. To do so, we propose to rasterize the point clouds in BEV into an image and then train an image-based object detector. Bounding boxes, generated by the image-based object detector, are then transformed back to the point cloud space using scaling and translation. The benefit of doing so is twofold. First, object detection from images is a well studied problem, and the performance of state-of-the-art detectors is good enough for practical use [6, 15]. Second, orthogonal projection of point clouds in BEV fits well conventional image representation, and

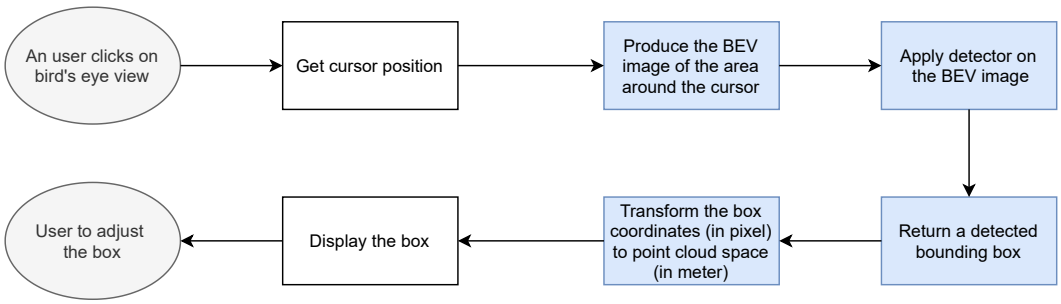


Figure 1: Our annotation pipeline. At the core of the interaction is an object detector that takes a small patch around the cursor on the BEV image of the point cloud as input, and infers the 3D bounding box at the cursor location.

we can utilize translation-invariant property of convolutions in this domain for performing single-click object detection on cropped images.

Particularly, we follow the method proposed by Chen et al. [3] for multi-view 3D object detection to produce a grayscale BEV image from an input LiDAR point cloud. First, we perform orthographic projection of an input point cloud to the BEV. Then, we discretize the projected point cloud into a 2D regular grid with resolution of 0.1 meter. After that, for each cell in the grid, we set the color value to be the intensity of the point that has the maximum height in that cell. We also transform 3D annotated bounding boxes on the original LiDAR point cloud into 2D bounding boxes on BEV. Let us denote this transformation as M. To detect objects, we train YOLOv3 object detector [11] on BEV images and 2D labels we obtained from the previous step. We selected YOLOv3 as it is lightweight and standard with implementations supported by many frameworks.

3.2 Inference

Figure 1 illustrates the use case when an user performs a click on the BEV of a raw lidar point cloud. We determine the cropped area around the position of the cursor, then rasterize the points in the cropped area and perform projection to generate the BEV image of that area. After that, we input the BEV image into the detector trained on BEV images to predict bounding boxes in BEV. To make box prediction easy to control, in most cases, we output a single bounding box per cropped area. To align bounding boxes in the original LiDAR space, we translate the boxes from the local coordinates in their cropped areas to the BEV image coordinate and invert the transformation M^{-1}. We set the height of the transformed 3D bounding boxes to predefined value. After this step, annotator can edit and refine the boxes in 3D.

To facilitate the annotation pipeline with more alternatives, we also allow users to override the predictions made by the BEV detector, and to add in missing boxes by using existing techniques such as point clustering and rectangle fitting.

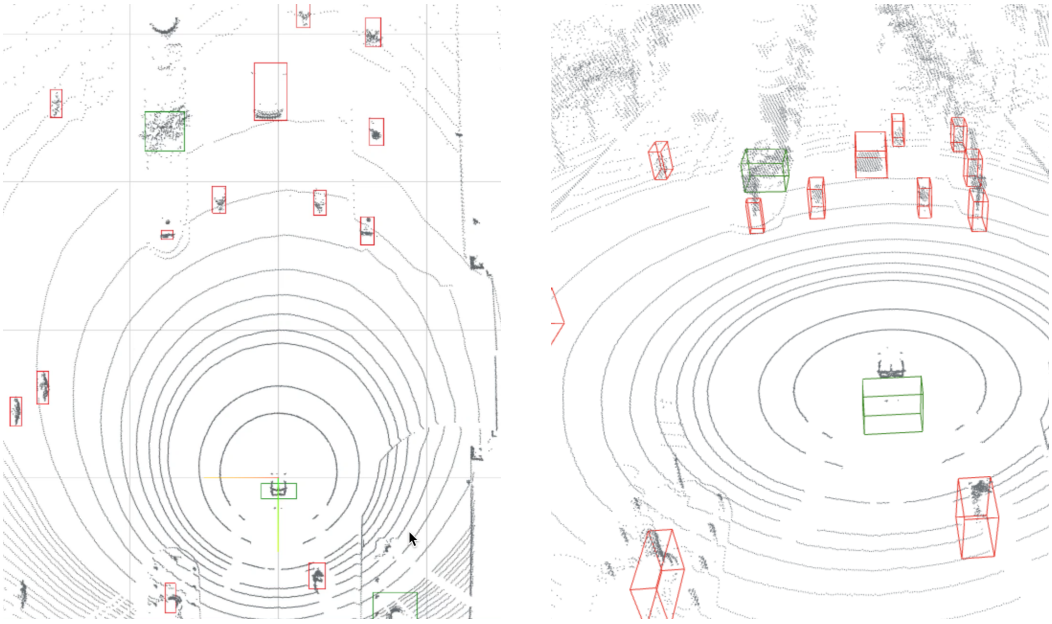
4 Experiments

Implementation. We implemented our system with two modules: a backend with OpenCV [9] and Python Flask [4] and a frontend with three.js [13] and react [10]. Our frontend is lightweight and only responsible for visualization and user interaction. All processing operations are done at the backend and results are streamlined as JSON format to the frontend.

Evaluation results. We conducted an experiment using a dataset of our inhouse data consisting of $3,600$ LiDAR point clouds. We manually annotated objects in 3D (i.e., 3D bounding boxes) in our dataset. We also generated BEV images and 2D bounding boxes. We empirically found that annotators tend to detect one object in a local area in the BEV image. Therefore, we devise a more optimized way to train the BEV detector. Specifically, for each object of interest in the BEV image, we crop a 100×100 area potentially containing the object. By this way, we obtain a dataset of about

Table 2: Quantitative results of our BEV detector and the statistics of our train and test dataset.

	Car	Truck	Trailer	Bus	Bicycle	Motorcycle	Pedestrian	Barrier	Total/Avg.
Train set	12,028	1,142	32	134	409	27,281	2,898	204	44,128
Val set	1,715	154	4	16	63	3,973	391	31	6,347
Test set	3,455	359	7	34	106	7,718	820	62	12,561
mAP@0.5	99.95	95.61	17.86	89.93	98.61	96.54	85.41	54.97	79.86



(a) BEV (b) 3D perspective view

Figure 2: Visualization of a LiDAR driving scene in BEV and in 3D, with the single-click boxes highlighted in red. The boxes by clustering and rectangle fitting is in green. As can be seen, our method has more accurate box prediction. Due to spurious points, rectangle fitting tends to result in overfit (too tight) or underfit (too loose) boxes.

$63,000$ images, and each image has only an object of interest. Our dataset was randomly split into train/validation/test sets in a 70/10/20 ratio.

We trained and tested our BEV object detector on an NVIDIA V100 GPU. Our inference ran at 81 FPS, and required 1.8 GB of GPU memory. We report the performance of our method (in mAP) on our test set in Table 2. We visualize several of our results in Figure 2.

As shown in the results, the detection of vehicles like car, truck, bicycle, and motorbike achieve high performance, e.g., more than 95% mAP. Other classes like bus, and pedestrian have lower mAP but the mAP is still in a good range. The trailer and barrier classes have lowest performance, which could potentially be due to the lack of training data. We aim to further improve our model by adding more training data and consider this for our future work.

5 Conclusion

In this paper, we propose an interactive method for annotation of 3D objects on LiDAR point clouds by performing single-click object detection on BEV. We found that the approach of using 2D object detector on BEV increases the efficiency of 3D annotation, leading to improved annotation pipeline. Benchmarking this method in a comprehensive annotation setting would be our future work.

References

[1] Hasan Asyari Arief, Mansur Arief, Guilin Zhang, Zuxin Liu, Manoj Bhat, Ulf Geir Indahl, Havard Tveite, and Ding Zhao. Sane: Smart annotation and evaluation tools for point cloud data. *IEEE Access*, 2020.

[2] 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 1903.11027*, 2019.

[3] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017.

[4] Flask: A lightweight wsgi web application framework in python. `https://flask.palletsprojects.com/en/2.0.x/`. Online.

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[6] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *arXiv 1907.09408*, 2019.

[7] 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.

[8] E Li, Shuaijun Wang, Chengyang Li, Dachuan Li, Xiangbin Wu, and Qi Hao. Sustech points: A portable 3d point cloud interactive annotation platform system. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020.

[9] Opencv: Open source computer vision library. `https://opencv.org/`. Online.

[10] React: A javascript library for building user interfaces. `https://reactjs.org/`. Online.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv 1804.02767*, 2018.

[12] Duc Thanh Nguyen, Binh-Son Hua, Lap-Fai Yu, and Sai-Kit Yeung. A robust 3d-2d interactive tool for scene segmentation and annotation. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2017.

[13] Three.js: Javascript 3d library. `https://threejs.org/`. Online.

[14] Bernie Wang, Virginia Wu, Bichen Wu, and Kurt Keutzer. Latte: Accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking. *arXiv 1904.09085*, 2019.

[15] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Naveed Asghar, and Brian A. Lee. A survey of modern deep learning based object detection models. *arXiv 2104.11892*, 2021.

[16] W. Zimmer, Akshay Rangesh, and M. Trivedi. 3d bat: A semi-automatic, web-based 3d annotation toolbox for full-surround, multi-modal data streams. *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1816–1821, 2019.