Title

A LiDAR Point Cloud Generator

Authors

Yue, Xiangyu
Wu, Bichen
Seshia, Sanjit A
et al.

Publication Date

2018-06-05

DOI

10.1145/3206025.3206080

Peer reviewed

A LiDAR Point Cloud Generator: from a Virtual World to Autonomous Driving

Xiangyu Yue, Bichen Wu, Sanjit A. Seshia, Kurt Keutzer and Alberto L. Sangiovanni-Vincentelli
University of California, Berkeley
{xyyue, bichen, sseshia, keutzer, alberto}@eecs.berkeley.edu

***Abstract*— 3D LiDAR scanners are playing an increasingly important role in autonomous driving as they can generate depth information of the environment. However, creating large 3D LiDAR point cloud datasets with point-level labels requires a significant amount of manual annotation. This jeopardizes the efficient development of supervised deep learning algorithms which are often data-hungry. We present a framework to rapidly create point clouds with accurate point-level labels from a computer game. The framework supports data collection from both auto-driving scenes and user-configured scenes. Point clouds from auto-driving scenes can be used as training data for deep learning algorithms, while point clouds from user-configured scenes can be used to systematically test the vulnerability of a neural network, and use the falsifying examples to make the neural network more robust through retraining. In addition, the scene images can be captured simultaneously in order for sensor fusion tasks, with a method proposed to do automatic calibration between the point clouds and captured scene images. We show a significant improvement in accuracy (+9%) in point cloud segmentation by augmenting the training dataset with the generated synthesized data. Our experiments also show by testing and retraining the network using point clouds from user-configured scenes, the weakness/blind spots of the neural network can be fixed.**

I. Introduction

Autonomous driving requires accurate and reliable perception of the environment. Of all the environment sensors, 3D LiDARs (Light Detection And Ranging) play an increasingly important role, since their resolution and field of view exceed radar and ultrasonic sensors and they can provide direct distance measurements that allow detection of all kinds of obstacles [1]. Moreover, LiDAR scanners are robust under a variety of conditions: day or night, with or without glare and shadows [2]. While LiDAR point clouds contain accurate depth measurement of the environment, navigation of autonomous vehicles also relies on correct understanding of the semantics of the environment. Most of the LiDAR-based perception tasks, such as semantic segmentation[3], [4], [2] and drivable area detection[5], [6], require significant amount of point-level labels for training and/or validation. Such annotation, however, is usually very expensive.

To facilitate the manual annotation process, much work has been done on interactive annotation. Annotation methods have been proposed for labeling 3D point clouds of both indoor scenes [7] and outdoor driving scenes [8]. These methods utilize little computer assistance during the annotation process and thus need a significant amount of human effort. In [9], [10], approaches have been proposed to enhance the man-machine interaction to improve annotation efficiency. In [11], [12], annotation suggestions for indoor RGBD scenes are proposed by the system that are interactively corrected or refined by the user. In order to provide faster interactive labeling rates, [13] proposes a group annotation approach for labeling objects in 3D LiDAR scans. Active learning has also been introduced in the annotation process to train a classifier with fewer interactions [14], [15], yet it requires users to interact with examples one-by-one. Other frameworks further take into account the risk of mislabeling and cost of annotation. [16] proposes a model of the labeling process and dynamically chooses which images will be labeled next in order to achieve a desired level of confidence.

Recently, video games have been used for creating large-scale ground truth data for training purposes. In [17], a video game is used to generate ground truth semantic segmentation for the synthesized in-game images. However, human effort is still required in the annotation process. In [18], the same game engine is used to generate ground truth 2D bounding boxes of objects in the images. [19] further extends the work of [17] so that various ground truth information(e.g. semantic segmentation, semantic instance segmentation, and optical flow) can be extracted from the game engine. In addition, many driving simulation environments[20], [21], [22] have been built in order to obtain various kinds of labeled data for autonomous driving purposes. Many of these work[18], [17], [19], [22] show the effectiveness of synthetic data in image-based learning tasks by showing improved performance after training with additional synthetic data. However, little work has been done on extracting annotated 3D LiDAR point clouds from simulators, not to mention showing the efficacy of the synthetic point clouds during the training process of neural networks.

Note that even if we could provide large amounts of training data, it is still almost impossible for any algorithms to achieve 100% accuracy. For Cyber-Physical Systems used for safety critical purposes, such as autonomous driving, verifying neural networks is of extreme importance [23]. In [24], a framework is proposed to systematically analyze Convolutional Neural Networks (CNNs) used in objection detection in autonomous driving systems. However, the framework only takes into account cars from direct front/back view and thus has a very limited modification space. In addition, each background image needs to be manually annotated, making it expensive to generate a dataset with large diversity. To the

(a) (b) (c)

Fig. 1: Sample data extracted from an in-game scene. (a): Image of the scene; (b): Point cloud of car (Blue dots) mapped to image after calibration matches car in image; (c): Extracted point cloud from the same scene.

best of our knowledge, no similar work has been done on LiDAR point clouds. In this paper, we propose an extraction-annotation-CNN testing framework based on a popular video game. The main contributions of this framework are as follows:

- The framework can automatically extract point-cloud data with ground truth labels together with the corresponding image frame of the in-game scene, as shown in Fig. 1.
- The framework can do automatic calibration between collected point clouds and images which can then be used together for sensor fusion tasks.
- Users can construct specified scenarios in the framework interactively and the collected data(point clouds and images) can then be used to systematically test, analyze and improve LiDAR-based and/or image-based learning algorithms for autonomous driving.

We conducted experiments on a Convolutional Neural Network (CNN)-based model for 3D LiDAR point cloud segmentation using the data collected from the proposed framework. The experiments show 1) significantly improved performance on KITTI dataset[8] after retraining with additional synthetic LiDAR point clouds, and 2) efficacy of using the data collected from user-configured scenes in the framework to test, analyze and improve the performance of the neural network. The performance improvements come from the fact that the data collected in the rich virtual world contains a lot of information that the neural network failed to learn from the limited amount of original training samples.

II. Technical Approach

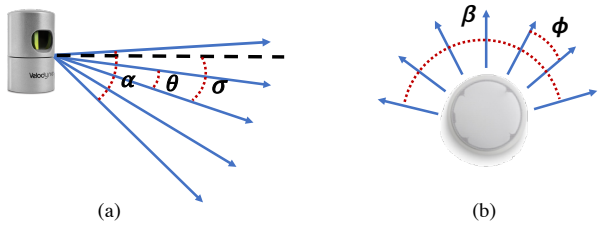
A. In-Game Simulation Setup and Method for Data Collection

We choose to utilize the rich virtual world in Grand Theft Auto V (GTA-V), a popular video game, to obtain simulated point clouds as well as captured in-game images with high fidelity[1]. Our framework is based on DeepGTAV[2], which uses Script Hook V[3] as a plugin.

[1]The publisher of GTA-V allows non-commercial use of footage of gameplay [17].

[2]https://github.com/aitorzip/DeepGTAV

[3]http://www.dev-c.com/gtav/scripthookv/



(a) (b)

Fig. 2: Sample configurable parameters of the virtual LiDAR. (a) shows front view of the virtual LiDAR: black dotted line is the horizontal line, α is the vertical field of view (FOV), θ is the vertical resolution, σ is the pitch angle; (b) shows top view of the virtual LiDAR, β is the horizontal FOV, and ϕ is the horizontal resolution.

In order to simulate realistic driving scenes, an ego car is used in the game with a virtual LiDAR scanner mounted atop, and it is set to drive autonomously in the virtual world with the AI interface provided in Script Hook V. While the car drives on a street, the system collects LiDAR point clouds and captures the game screen, simultaneously. We place the virtual LiDAR scanner and the game camera at the same position in the virtual 3D space. This set-up offers two advantages: 1) a sanity check can be easily done on the collected data, since point clouds and corresponding images must be consistent; 2) calibration between the game camera and the virtual LiDAR scanner can be done automatically, and then collected point clouds and scene images can be combined together as training dataset for neural networks for sensor fusion tasks. Details of the proposed calibration method will be described in Section II-B.

Ray casting is used to simulate each laser ray emitted by the virtual LiDAR scanner. The ray casting API takes as input the 3D coordinates of the starting and ending point of the ray, and returns the 3D coordinates of the first point the ray hits. This point is used, with another series of API function calls, to calculate, among other data, the distance of the point, the category and instance ID of the object hit by the ray, thus allowing automatic annotation on the collected data.

In our framework, users can provide configurations of the LiDAR scanner including vertical field of view (FOV), vertical resolution, horizontal FOV, horizontal resolution,

pitch angle, maximum range of laser rays, and scanning frequency. Some of the configurable parameters are shown in Fig. 2.

B. Automatic Calibration Method

The goal of the calibration process is to find the corresponding pixel in the image for each LiDAR point. In our framework, the calibration process can be done automatically by the system based on the parameters of the camera and LiDAR scanner. In addition, the centers of the camera and LiDAR scanner are set to the same position in the virtual world, making the calibration projection similar to the camera perspective projection model, as shown in Fig. 3.

The problem is formulated as follows: for a certain laser ray with $azimuth$ angle ϕ and $zenith$ angle θ, calculate the index (i,j) of the corresponding pixel on image. $\mathcal{F}_c$, $\mathcal{F}_o$, P, P' and P_{far} are 3D coordinates of a) center of camera/LiDAR scanner, b) center of camera near clipping plane. c) point first hit by the virtual laser ray (in red), d) pixel on image corresponding to P, and e) a point far away in the laser direction, respectively. m and n are the width and height of the near clipping plane. γ is $1/2$ vertical FOV of camera while ψ is $1/2$ vertical FOV of the LiDAR scanner. Note that LiDAR scanner FOV is usually smaller than camera FOV, since there is usually no object in the top part of the image, and the emitting laser to open space is not necessary. After a series of 3D geometry calculation, we can get:

$$\begin{aligned} i &= \frac{R_m}{m} \cdot (f \cdot \tan\gamma \cdot \frac{m}{n} - \frac{f}{\cos\theta} \cdot \tan\phi), \\ j &= \frac{R_n}{n} \cdot (f \cdot \tan\gamma + f \cdot \tan\theta), \end{aligned} \tag{1}$$

where $f = \left\| \overrightarrow{\mathcal{F}_c \mathcal{F}_o} \right\|$, and (R_m, R_n) is the pixel resolution of the image/near clipping plane.

Further, in order for the ray casting API to work properly, the 3D coordinates of P_{far} are also required. Using similar 3D geometry calculations, we obtain:

$$\begin{aligned} P' &= \mathcal{F}_c + f \cdot \overrightarrow{x_c} - \frac{f}{\cos\theta} \cdot \tan\phi \cdot \overrightarrow{y_c} - f \cdot \tan\theta \cdot \overrightarrow{z_c}, \\ P_{far} &= \mathcal{F}_c + k \cdot (P' - \mathcal{F}_c), \end{aligned} \tag{2}$$

where k is a large coefficient, and $\overrightarrow{x_c}$, $\overrightarrow{y_c}$, $\overrightarrow{z_c}$ are unit vectors of the camera axis in the world coordinate system.

An example of the calibration result is shown in Fig. 1. After simulation, both image and point cloud of the specified in-game scene are collected by the framework (Fig. 1 (a, c)). Then with the proposed calibration method, we map all the points with category "Car" to the corresponding image. As shown in Fig. 1 (b), the mapped car point cloud (blue dots) matches the car in the image fairly accurately.

C. Configurable In-game Scene

Besides the auto-driving mode for large-scale data collection, our framework offers a configurable mode, where the user can configure desired in-game scenes and collect data from them. One advantage of configurable scenes is generating training data of driving scenes that are dangerous or rare in real world. Another advantage is that we can systematically sample the modification space(e.g. number of cars, position and orientation of a car) of an in-game scene. The data can then be used to test neural network, expose its vulnerabilities and improve its performance throuth re-training. Our framework offers a large modification space of the in-game scene. As shown in Fig. 4, the user can specify and change 8 dimensions of in-game scene: car model, car location, car orientation, number of cars, scene background, color of car, weather, and time of day. The first 5 dimensions affect both LiDAR point cloud and scene image, while the last three dimensions affect only the scene image. An example of sampling is shown in Fig. 5, where the scenes are only sampled from the spatial dimensions (X, Y) with only one car in each scene. X and Y are the location offset of the car relative to the camera/LiDAR location in the left-right and forward-backward directions. Fig.5 (b) shows collected point cloud of the samples shown in Fig.5 (a). The red points represent car points while the blue points represent the scene background. The collected point clouds match the scenes well thus allowing the use of the data to test neural nets systematically.

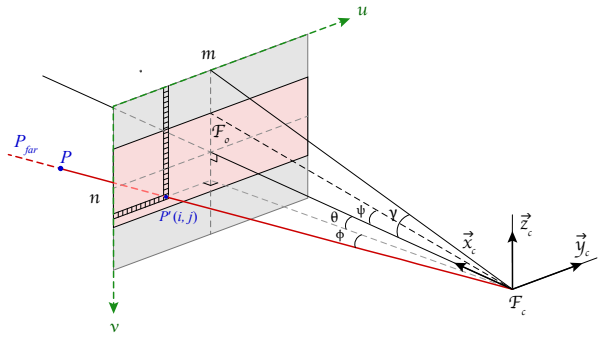


Fig. 3: Projection for Calibration. $\mathcal{F}_o$ is the center of the near clipping plane of the camera; $\mathcal{F}_c$ is the center of the camera and of the LiDAR scanner; the red line is the laser ray and P is the point hit by the ray; the calibrated on-image point has pixel index (i,j) and 3D coordinates P'; γ is the $1/2$ camera vertical FOV and ψ is the $1/2$ LiDAR vertical FOV; ϕ and θ are the $azimath$ and $zenith$ angles of the laser ray.

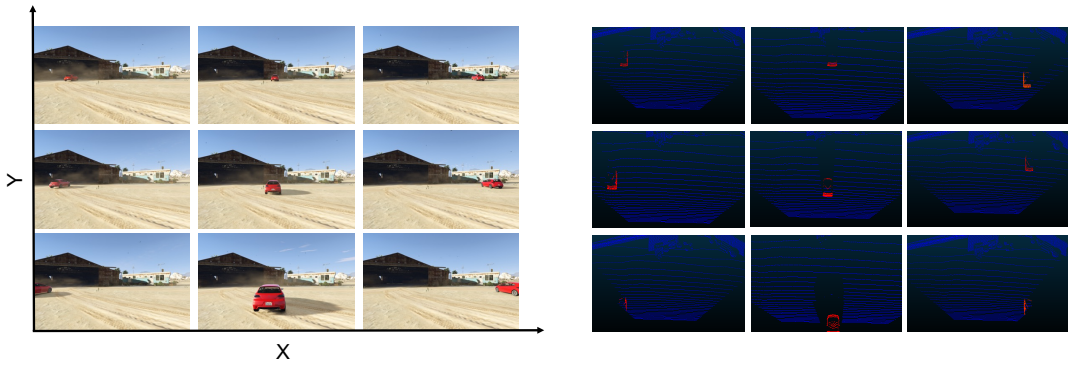
III. Experiments and Results

We performed experiments to show the efficacy of our data synthesis framework: 1) Data collected by the framework can be used in the training phase and help improve the validation accuracy; 2) Collected data can be used to systematically test a neural network and improve its performance via retraining.

A. Evaluation Metrics

Our experiments are performed on the task of LiDAR point cloud segmentation; specifically, given a point cloud detected by a LiDAR sensor, we wish to perform point-wise classification, as shown in Fig. 6. This task is an essential step for autonomous vehicles to perceive and understand the environment, and navigate accordingly.

Fig. 4: Modification dimensions of the framework with image in center showing the reference scene.



(a) (b)

Fig. 5: Scenes with one car sampled from spatial dimensions and corresponding point cloud. (a) shows the scene image while changing the location of the car on X(left-right) and Y(forward-backward) directions; (b) shows point clouds (red for car and blue for background) of scenes in (a).

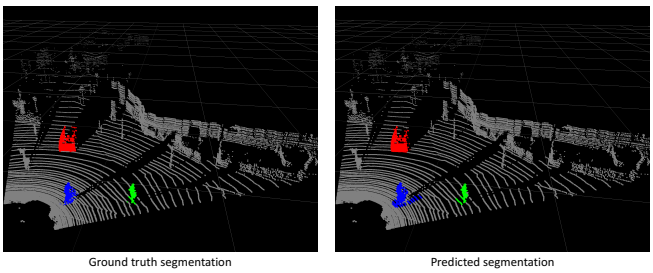


Fig. 6: LiDAR point cloud segmentation

To evaluate the accuracy of the point cloud segmentation algorithm, we compute *Intersection-over-Union* (IoU), *Precision* and *Recall* as:

$$IoU_c = \frac{|\mathcal{P}_c \cap \mathcal{G}_c|}{|\mathcal{P}_c \cup \mathcal{G}_c|}, Pr_c = \frac{|\mathcal{P}_c \cap \mathcal{G}_c|}{|\mathcal{P}_c|}, Recall_c = \frac{|\mathcal{P}_c \cap \mathcal{G}_c|}{|\mathcal{G}_c|}.$$

Here, $\mathcal{P}_c$ denotes the set of points that our model predicted to be of class-c, $\mathcal{G}_c$ denotes the ground-truth set of points belonging to class-c, and $|\cdot|$ denotes the cardinality of a set. *Precision* and *Recall* measures accuracy with regard to false positives and false negatives, respectively; while *IoU* takes both into account. For this, ***IoU*** is used as the primary accuracy metric in our experiments.

B. Experimental Setup

Our analysis is based on SqueezeSeg [2], a convolutional neural network based model for point cloud segmentation.

To collect the real-word dataset, we used LiDAR point cloud data from the KITTI dataset and converted its 3D bounding box labels to point-wise labels. Since KITTI dataset only provides reliable 3D bounding boxes for front-view LiDAR point clouds, we limit the horizontal field of view(FOV) to the forward-facing $90°$. This way, we obtained 10,848 LiDAR scans with manual labels. We used 8,057 scans for training and 2,791 scans for validation. Each point in a KITTI LiDAR scan has 3 cartesian coordinates (x, y, z) and an intensity value, which measures the amplitude of the signal received. Although the intensity measurement as an extra input feature is beneficial to improve the segmentation accuracy, simulating the intensity measurement is very difficult and not supported in our current framework. Therefore we excluded intensity as an input feature to the neural network for GTA-V synthetic data. We use NVIDIA TITAN X GPUs for the experiments during both the training and validation phases.

C. Experimental Results

For the first set of experiments, we used our data synthesis framework to generate 8,585 LiDAR point cloud scans in autonomous-driving scenes. The generated data contain (x, y, z) measurements but do not contain intensity. The horizontal FOV of the collected point clouds are set to be $90°$ to match the setting of KITTI point clouds described in Section III-B.

To quantify the effect of training the model with synthetic data, we first trained two models on the KITTI training set with intensity included and excluded, and validated on the KITTI validation set. The performance is shown in the first 2 rows of Table I as the baseline. The model with intensity achived better result. Then we trained another model with only GTA-V synthetic data. As shown in the third row of Table I, the performance drops a lot. This is mostly because the distributions of the synthetic dataset and KITTI dataset are quite different. Therefore, through training purely on synthetic dataset, it is hard for the neural network to learn all the required details for the KITTI dataset, which might be missing or insufficient in the synthetic training dataset. Finally, we combined the KITTI data and GTA-V data together as the training set and train another model. As shown in the last row of Table I, the performance is improved significantly, almost 9% better than the accuracy achieved only using real-world data. Despite the loss of the intensity channel, the GTA+KITTI dataset gives better accuracy than if intensity is included. This demonstrates the efficacy of the synthetic data extracted in our framework.

Then we used our framework to systematically test SqueezeSeg. As an illustrative experiment, we only performed sampling in the car location X-Y dimensions as in Fig. 5, rather than the whole modification space. 555 scenes were sampled to test SqueezeSeg, with the IoU results shown in Fig. 7. The blue and green dots show the car locations resulting in low IoU. Most of the "blind spot" are locations far from the LiDAR scanner, but there are also closer locations that result in low IoU scores. Close locations with low IoUs are dangerous in autonomous driving, since they can mislead the decision-making system of the autonomous vehicles and result in immediate accident.

TABLE I: Segmentation Performance Comparison on the Car Category. Only data used in the first row has Intensity channel.

	Precision	Recall	**IoU**
KITTI w/ Intensity	66.7	95.4	64.6
KITTI w/o Intensity	58.9	95.0	57.1
GTA-V only	30.4	86.6	29.0
KITTI w/o Intensity + GTA-V	69.6	92.8	**66.0**

All numbers are in percentage.

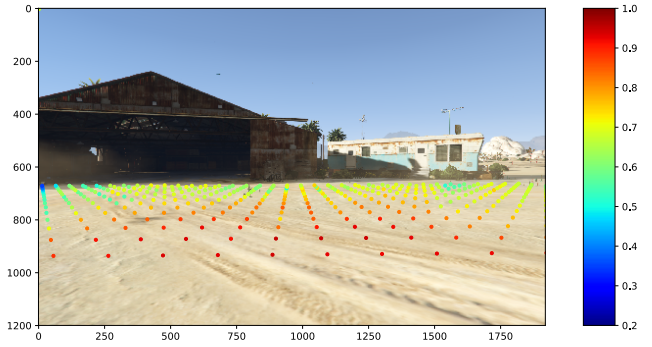


Fig. 7: IoU scatter with the change of car location

Further experiments are then done to show the efficacy of using synthetic data from the proposed framework to possibly improve performance of the network on bad sample points in the modification space. We synthesized totally 2,250 LiDAR point cloud scans in 15 different scene backgrounds. In each scene background, only one car is placed with the same orientation as the camera view. We obtained 150 point cloud scans in each scene background by changing the position of the car (X, Y) in the sampled space: $S = \{(x, y) \mid x \in \{-5, \cdots, 4\}, y \in \{5, \cdots, 19\}\}$, where X, Y are respectively the left-right and forward-backward offset relative to the position of the camera. For each scene background, the position and orientation of the camera were fixed.

We split the collected point cloud scans based on the scene background. 1200 point cloud scans in the first 7 backgrounds are used as validation set $\mathcal{V}$, and the rest 1050 scans, which we call retraining set $\mathcal{R}$, are used for retraining purpose. First, we train a neural network with purely KITTI data and do evaluation on the synthetic 1200 scans in the validation set. We define *mean IoU(mIoU)* for each point in the 15×10 X-Y modification space as averaging IoUs over all the 7 scene backgrounds:

$$mIoU(i, j) = \frac{1}{n} \sum_{k=1}^{n} IoU(i, j, k),$$

where n is the number of scene backgrounds ($n = 7$ in this experiment), (i, j) is in $\{(i, j) \mid i \in [-5, 4], j \in [5, 19], i, j \in$

$\mathbb{Z}\}$ and $IoU(i, j, k)$ refers to the IoU of the point cloud sc sampled at (i, j) in the X-Y modification with the k_{th} sce background.

The mIoU map of the validation set is computed, as show in Fig. 8. We can see that the pre-trained network perforn poorly on positions that are far away, at the boundary the FOV. But more surprisingly, we also observed that position (-3, 5), which is fairly close to the ego-vehicle, t mIoU score is also very low. Detection errors at such ne distance can be very dangerous.

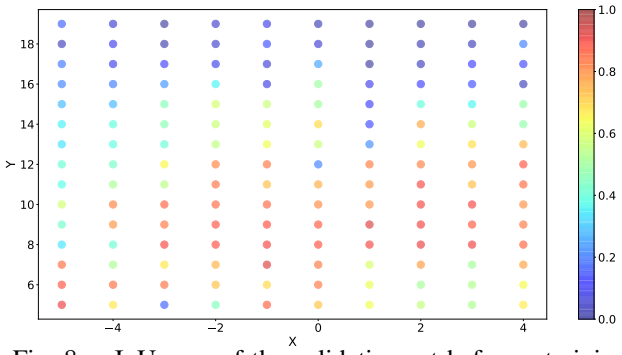


Fig. 8: mIoU map of the validation set before retraining

Based on the mIoU map, we choose positions with mIoU smaller than a threshod to form a retraining set, as shown in Fig. 9.

Then all the point clouds in the retraining set $\mathcal{R}$ with a selected position are added to the original training set. After the retraining process, we re-evaluate the validation set, with the new mIoU map shown in Fig. 10. As the figure shows, at almost all the close-to-center positions originally with low mIoU, the neural network performs much better than before the retraining. In order to visualize the performance improvements better, we plot the *mIoU* improvement after the retraining process for each position. The mIoU improvements are sorted and plotted in Fig. 11. We see that after retraining, performance on point clouds at most of the positions gets much better, with slightly

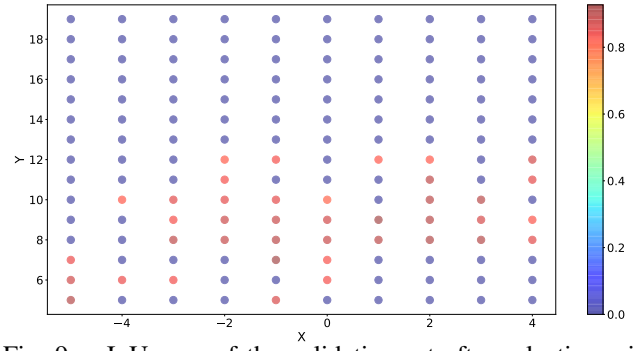


Fig. 9: mIoU map of the validation set after selection with mIoU less than 0.65 set to 0. All the point clouds in the retraining set $\mathcal{R}$ corresponding to the blue positions in the new mIoU map will be added to the original training set.

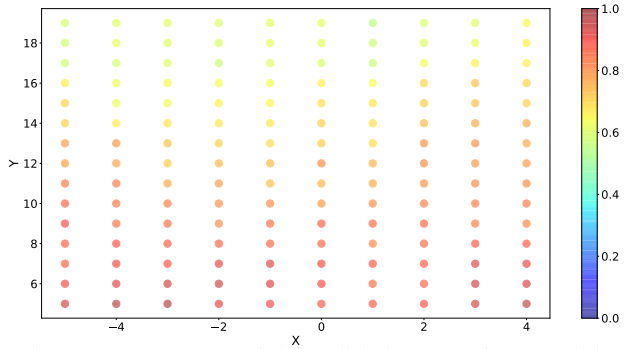


Fig. 10: mIoU map of the validation set after retraining

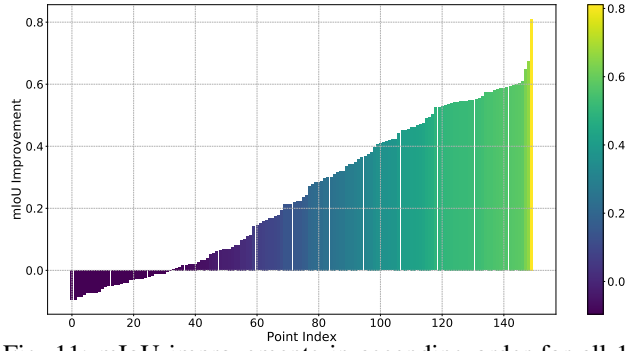


Fig. 11: mIoU improvements in ascending order for all 150 positions

degraded performance at only a small fraction of positions. Meanwhile, the performance on KITTI dataset remained almost the same with *IoU* changing from 60.8% to 60.6%. These experiments show the efficacy of using synthetic data from user-configured scenes of the proposed framework to test, analyze and improve the performance of neural networks through retraining.

IV. Conclusions and Future Work

In this paper, we proposed a framework that synthesizes annotated LiDAR point clouds from a virtual world in a game, with a method to automatically calibrate the point cloud and scene image. Our framework can be used to: 1) obtain a large amount of annotated point cloud data, which can then be used to help neural network training; 2) systematically test, analyze and improve performance of neural networks for tasks such as point cloud segmentation. Experiments show that for a point cloud segmentation task, synthesized data help improve the validation accuracy (IoU) by 9%. Furthermore, the systematical sampling and testing framework can help us to identify potential weakness/blind spots of our neural network model and fix them. The first set of experiments also show the effectiveness of the intensity channel in LiDAR point clouds. In the future, we will work on simulating the intensity information, which we believe will definitely help the research in this field.

REFERENCES

[1] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *IEEE Intelligent Vehicles Symposium*, 2009, pp. 215–220.

[2] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *IEEE International Conference on Robotics and Automation*, 2018.

[3] A. Dewan, G. L. Oliveira, and W. Burgard, "Deep semantic classification for 3d lidar data," *Computing Research Repository*, vol. abs/1706.08355, 2017.

[4] D. Dohan, B. Matejek, and T. Funkhouser, "Learning hierarchical semantic segmentations of LIDAR data," in *International Conference on 3D Vision (3DV)*, Oct. 2015.

[5] Z. Liu, S. Yu, X. Wang, and N. Zheng, "Detecting drivable area for self-driving cars: An unsupervised approach," *arXiv preprint arXiv:1705.00451*, 2017.

[6] R. Fernandes, C. Premebida, P. Peixoto, D. Wolf, and U. Nunes, "Road detection using high resolution lidar," in *IEEE Vehicle Power and Propulsion Conference (VPPC)*, Oct 2014, pp. 1–6.

[7] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 746–760.

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[9] R. Kopper, F. Bacim, and D. A. Bowman, "Rapid and accurate 3d selection by progressive refinement," in *IEEE Symposium on 3D User Interfaces (3DUI)*, March 2011, pp. 67–74.

[10] M. Veit and A. Capobianco, "Go'then'tag: A 3-d point cloud annotation technique," in *IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 2014, pp. 193–194.

[11] Y.-S. Wong, H.-K. Chu, and N. J. Mitra, "Smartannotator: An interactive tool for annotating RGBD indoor images," *arXiv preprint arXiv:1403.5718*, 2014.

[12] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo, "An interactive approach to semantic modeling of indoor scenes with an rgbd camera," *ACM Trans. Graph.*, vol. 31, no. 6, pp. 136:1–136:11, 2012.

[13] A. Boyko and T. Funkhouser, "Cheaper by the dozen: Group annotation of 3d data," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '14. ACM, 2014, pp. 33–42.

[14] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell, "Active learning with gaussian processes for object categorization," in *IEEE 11th International Conference on Computer Vision (ICCV)*, 2007, pp. 1–8.

[15] A. Top, G. Hamarneh, and R. Abugharbieh, "Active learning for interactive 3d image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2011, pp. 603–610.

[16] P. Welinder and P. Perona, "Online crowdsourcing: rating annotators and obtaining cost-effective labels," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW),*. IEEE, 2010, pp. 25–32.

[17] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 102–118.

[18] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 746–753.

[19] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *International Conference on Computer Vision (ICCV)*, 2017.

[20] D. Biedermann, M. Ochs, and R. Mester, "Evaluating visual adas components on the congrats dataset," in *IEEE Intelligent Vehicles Symposium (IV)*, June 2016, pp. 986–991.

[21] V. Haltakov, C. Unger, and S. Ilic, "Framework for generation of synthetic ground truth data for driver assistance applications," in *German Conference on Pattern Recognition*. Springer, 2013, pp. 323–332.

[22] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[23] T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional falsification of cyber-physical systems with machine learning components," *Computing Research Repository*, vol. abs/1703.00978, 2017.

[24] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Systematic testing of convolutional neural networks for autonomous driving," *arXiv preprint arXiv:1708.03309*, 2017.