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

Xiangyu Yue
EECS, UC Berkeley
xyyue@berkeley.edu

Bichen Wu
EECS, UC Berkeley
bichen@berkeley.edu

Kurt Keutzer
EECS, UC Berkeley
keutzer@berkeley.edu

Alberto Sangiovanni-Vincentelli
EECS, UC Berkeley
alberto@berkeley.edu

Sanjit A. Seshia
EECS, UC Berkeley
sseshia@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. 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 autonomous-driving scenes and user-configured scenes. Point clouds from auto-driving scenes can be used as training data for deep learning algorithms. We show a significant improvement in accuracy (+9%) in point cloud segmentation by augmenting the training dataset with the generated synthesized data. Point clouds from user-configured scenes can then be used to systematically test and analyze neural networks by sampling scenes in the scene modification space. We also propose a method to do automatic calibration between the point cloud and captured scene image.

1 Introduction

Autonomous driving requires accurate and reliable perception of the environment. Of all the sensors, 3D LiDARs (Light Detection And Ranging) plays an increasingly important role, as 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 [8]. Moreover, LiDAR scanners are robust under a variety of conditions: day or night, with or without glare and shadows [16]. 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 and drivable area detection, require significant amount of point-level labels for training and/or validation. Such annotation, however, is usually very expensive.

To facilitate 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 [11] and outdoor driving scenes [4]. These methods utilize little computer assistance during the annotation process and thus need a significant amount of human effort. Other methods use more computer assistance. In [7, 13], approaches have been proposed to enhance the interaction between a user and the 3D environment to improve annotation efficiency. In [15, 10], annotation suggestions for indoor RGBD scenes are proposed by the system, and then a user can interactively fix or refine the annotation. In order to provide faster interactive labeling rates, [1] proposes a group annotation

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

(a) (b) (c)

Figure 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.

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 [6, 12], 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. [14] 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 [9], 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 [5], the same game engine is used to generate ground truth 2D bounding boxes of objects in the images. However, more work is needed for the creation of 3D LiDAR point cloud datasets.

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 [2]. In [3], a framework is proposed to systematically analyze convolutional neural networks (CNNs) used in classification of cars in autonomous driving systems. However, the framework only takes into account cars from direct back view and thus has a very limited modification space. To the best of our knowledge, no similar work has been done for LiDAR point clouds.

In this paper, we propose a framework based on a popular video game that can address both issues of label generation and systematic testing. First, our framework can automatically extract point cloud data with ground truth labels together with the corresponding image frame of the in-game auto-driving scene, as shown in Figure 1. The collected point cloud dataset itself can then be used as training data, and the corresponding images can be further used for sensor fusion algorithms. Second, in our framework, the user can configure desired scenes interactively. The collected data can then be used for neural network testing and analysis. Preliminary experimental results show significant advantages over state-of-the-art methods.

2 Technical Approach

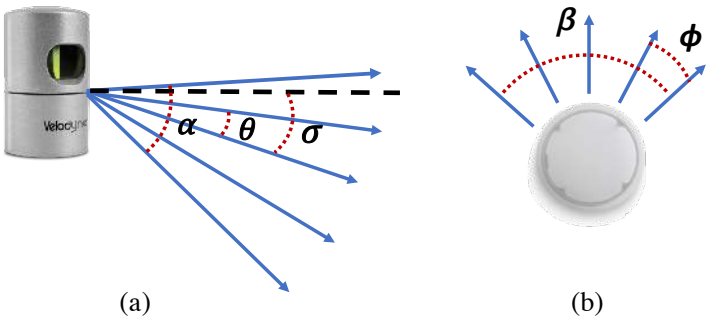
2.1 In-Game Simulation Setup and Method for Data Collection

We choose to utilize the rich virtual world in Grand Theft Auto V (GTA-V), a popular video game, to obtain simulated point cloud as well as images with high fidelity. The publisher of GTA-V allows non-commercial use of footage of gameplay [9]. Our framework is based on DeepGTAV[1], which uses Script Hook V[2] as a plugin.

In order to simulate realistic driving scenes, a test car is used to drive autonomously in a game with a virtual LiDAR scanner mounted atop. While the car drives on street, the system collects LiDAR point clouds and captures the game screen. The virtual LiDAR scanner and the game camera are placed at the same position in the virtual 3D space, offering two advantages: 1) a sanity check can be easily done on the collected data, since point cloud and images should be consistent; 2) calibration between the game camera and the virtual LiDAR scanner can be done automatically, and then point clouds

[1] `https://github.com/ai-tor/DeepGTAV`

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



(a) (b)

Figure 2: Sample configurable parameters. α is vertical field of view (FOV), θ is vertical resolution, σ is pitch angle, β is horizontal FOV, and ϕ is horizontal resolution.

and game scene images can be combined together as training dataset for neural networks for sensor fusion tasks. Details of the calibration method will be described in Section 2.2.

Ray casting is used to simulate each laser ray which the virtual LiDAR scanner emits. The ray casting API takes as input the coordinates of the starting point and ending point of the ray, and returns the coordinates of the first point the ray hits. This point is used to calculate the distance of the point, the category and instance ID of the object hit by the ray, thus allowing automatic annotation.

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, an scanning frequency. Some of the configurable parameters are shown in Figure 2.

2.2 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 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 Figure 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-clip 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-clip 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 image, and emitting laser to open space isn't quite necessary. After a series of 3D geometry calculation, we have:

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

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

Further, in order for the ray casting API to work properly, the 3D coordinates of P_{far} are also required. Then:

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

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

An example of the calibration result is shown in Figure 1. After simulation, both image and point cloud of the specified in-game scene are collected by the framework (Figure 1 (a, c)). Then with the

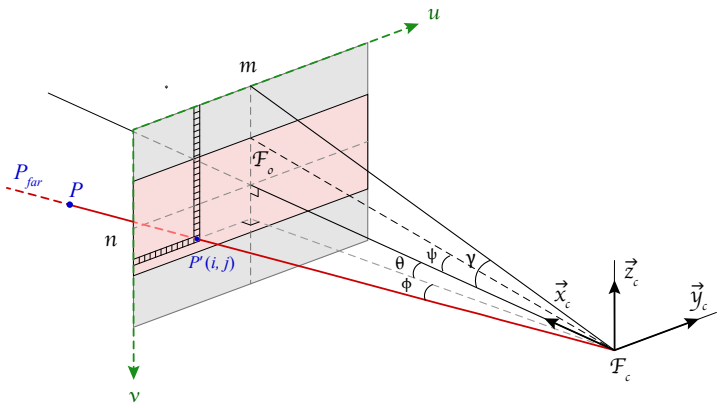


Figure 3: Projection for Calibration. $\mathcal{F}_o$ is center of near-clip plane of the camera; $\mathcal{F}_c$ is the center of camera and LiDAR scanner; red line is the laser ray and P is the point hit by the ray; the calibrated on-image point has pixel index (i, j) and 3D coordinates P'; γ is $1/2$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

proposed calibration method, we map all the points with category "Car" to the corresponding image. As shown in Figure 1 (b) The mapped car point cloud (blue dots) matches the car in the image fairly accurately.

2.3 Configurable In-game Scene

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 of an in-game scene. The data can then be used to test neural nets and expose its vulnerabilities. Our framework offers a large modification space of the in-game scene. As shown in Figure4, the user can specify and change 8 dimensions of in-game scene: car model, car location, car orientation, number of cars, scene location, 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 scene image. An example of sampling is shown in Figure 5, in which 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 left-right and forward-backward directions. Figure5 (b) shows collected point cloud of the samples shown in Figure5 (a). Red points represent car points while blue points represent the scene background. The collected point clouds match the scene well thus allowing the use of the data to test neural nets systematically.

3 Experiments and Results

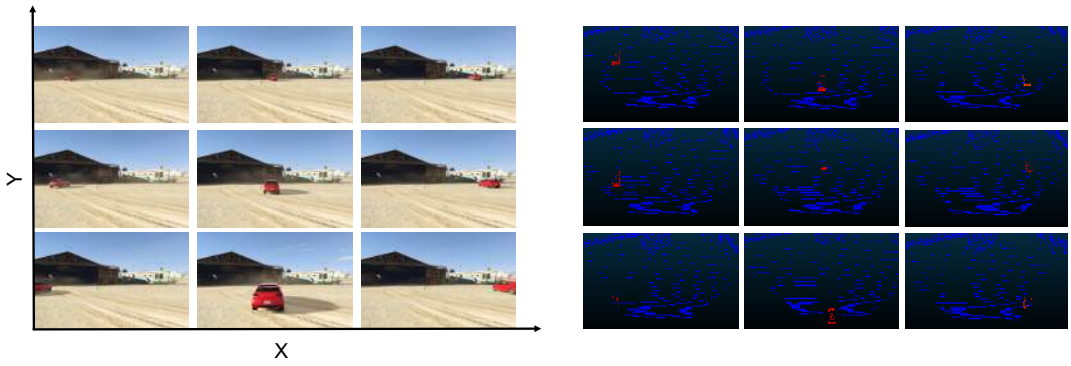
We performed experiments to show the efficacy of our data synthesis framework: 1) Data collected by the framework can be used in the training phase and help boost the validation accuracy; 2) Collected data can be used to systematically test a neural network. Our experiments are performed on the task 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.

As in [16], to evaluate the accuracy of 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.

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



(a) (b)

Figure 5: Scenes with one car sampled from spatial dimensions and corresponding point cloud. (a) shows the scene image while changing 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).

Our analysis is based on SqueezeSeg [16], 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. This way, we obtained 10,848 LiDAR scans with manual labels. We used 8,057 frames for training and 2,791 frames for validation and we used our data synthesis framework to generate 8,585 frames as extra training data.

As described in [16], we trained SqueezeSeg a) with only KITTI data, b) with only GTA synthetic data and c) with both dataset, and evaluated its accuracy on the KITTI validation set. Accuracy for these 3 cases are described in Table. 1. From the table, when augmented with synthetic data, the model achieved the best accuracy, better than if we only used real-world data.

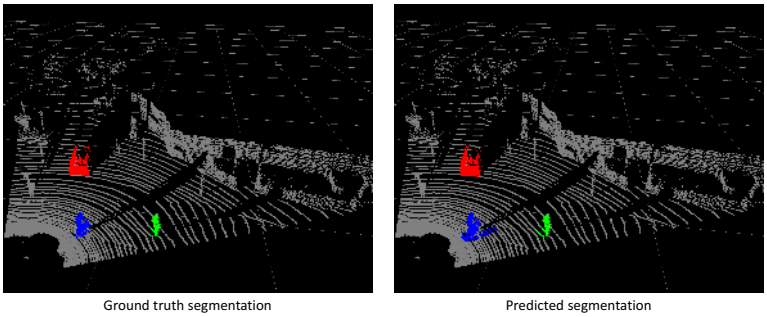


Figure 6: LiDAR point cloud segmentation

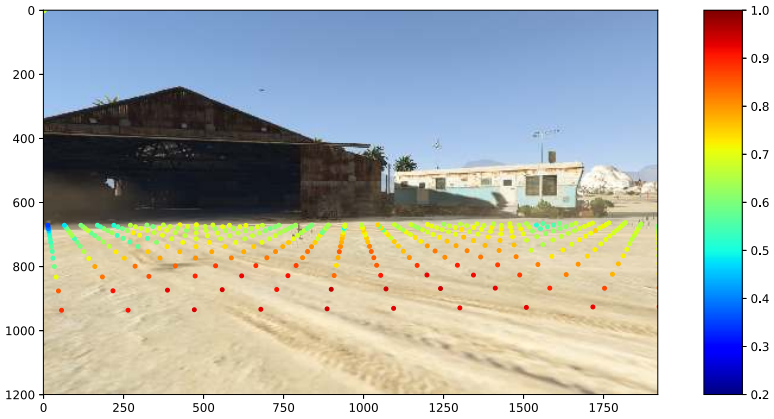


Figure 7: IoU scatter with the change of car location

Then we used our framework to systematically test SqueezeSeg. Since the total modification space of the framework is too large, we only did sampling in the car location X-Y dimensions as in Figure 5. 555 scenes are sampled to test SqueezeSeg, with the IoU results shown in Figure 7. 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 which results in low IoU scores. The segmentation accuracy can be potentially boosted by adding more samples near the "blind spot" locations detected by the testing process.

Table 1: Segmentation Performance on the Car Category with Synthesized Data

	Precision	Recall	**IoU**
KITTI	58.9	95.0	57.1
GTA	30.4	86.6	29.0
KITTI + GTA	69.6	92.8	66.0

4 Conclusions

In this paper, we propose a framework that synthesizes an annotated LiDAR point cloud 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) get a large amount of annotated point cloud data, which can then be used to help neural network training; 2) systematically test and analyze neural networks for tasks such as point cloud segmentation. Experiments show that for a point cloud segmentation task, synthesized data help boost the validation accuracy (IoU) by 9%. Further more, our systematical sampling and testing framework helps us to identify potential blind spots of our neural network model.

References

[1] Aleksey Boyko and Thomas Funkhouser. Cheaper by the dozen: Group annotation of 3d data. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 33–42, New York, NY, USA, 2014. ACM.

[2] Tommaso Dreossi, Alexandre Donzé, and Sanjit A. Seshia. Compositional falsification of cyber-physical systems with machine learning components. *CoRR*, abs/1703.00978, 2017.

[3] Tommaso Dreossi, Shromona Ghosh, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Systematic testing of convolutional neural networks for autonomous driving. *CoRR*, abs/1708.03309, 2017.

[4] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[5] Matthew Johnson-Roberson, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *CoRR*, abs/1610.01983, 2016.

[6] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, Oct 2007.

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

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

[9] Stephan R. Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *European Conference on Computer Vision (ECCV)*, volume 9906 of *LNCS*, pages 102–118. Springer International Publishing, 2016.

[10] Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 31(6):136:1–136:11, November 2012.

[11] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part V*, ECCV'12, pages 746–760, Berlin, Heidelberg, 2012. Springer-Verlag.

[12] Andrew Top, Ghassan Hamarneh, and Rafeef Abugharbieh. *Active Learning for Interactive 3D Image Segmentation*, pages 603–610. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

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

[14] P. Welinder and P. Perona. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 25–32, June 2010.

[15] Yu-Shiang Wong, Hung-Kuo Chu, and Niloy J. Mitra. Smartannotator: An interactive tool for annotating RGBD indoor images. *CoRR*, abs/1403.5718, 2014.

[16] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. *Under review for ICRA, 2018*.