CS230

Point of View General Adversarial Network

Kutluhan Buyukburc
Department of Computer Science
Stanford University
kutluhan@stanford.edu

Abstract

After the deep learning revolution our perception algorithms have become data hungry and developers face difficulties meeting this demand. Synthetic data generation methods are developed to cover this demand and this work demonstrates another way of generating synthetic data from LiDAR point clouds and this method allows us to generate synthetic images of different viewpoints.

1 Introduction

Autonomous Vehicles (AV) are ubiquitous in our lives today. With robotaxis, robot vacuum cleaners and drones a part and parcel of our daily routine, this project will be beneficial to the development of AV. Robots and AVs have many sensors, but unfortunately, changing or adding new camera sensors requires recording new dataset and it consumes time and resources. We propose using Point of View General Adversarial Network (POVGAN), which can make use of the dataset that we currently have to generate new dataset without adding a sensor or changing position of sensors. To obtain the learnable features of different viewpoints, we are using LiDAR point clouds which are easily projectable to any viewpoint. With the power of the General Adversarial Network (GAN) [3], we can generate synthetic data from different viewpoints. Thus synthetic data [1; 2] can contribute to reducing the time and resources spent on conventional data capturing.

2 Related work

Generative Adversarial Networks for synthesizing AV dataset is a growing area in automotive companies and in the Artificial Intelligence community. Most works focus on label to image translation but a recent paper suggests a new method of LiDAR point cloud to image translation. We build our work on the paper "Generating Photo-realistic Images from LiDAR Point Clouds with Generative Adversarial Networks"[8] which demonstrates a way of generating camera images from LiDAR point clouds. In our paper on POVGAN, we are proposing a better way of projecting points of LiDAR to an image coordinate system which is using coordinate system transformation between a LiDAR and a camera, instead of projecting all LiDAR points to a single image. This solution leads to having a paired representation of a LiDAR and a camera. Furthermore, being able to change viewpoint by transformation gives us the freedom of generating synthetic data of different viewpoints and this is our proposed idea in synthetic image generation.

CS230: Deep Learning, Spring 2022, Stanford University, CA.

3 Dataset and Features

3.1 Dataset

Fortunately, there are many online autonomous vehicle dataset available but we have some requirements and preferences in a dataset. First of all, the dataset must include at least a camera and a LiDAR sensor. To obtain a more general result from this project, we prefer multiple cameras with different fields of view (front, side, and back). Secondly, we need to have paired input and output, and to achieve that, sensors must be calibrated and the camera coordinate system to the LiDAR coordinate system transformation or any sensor to world coordinate system at any given frame need to be known and included in the dataset. With these constraints, we found the nuScenes [12] dataset is most suitable for our project.

	Lidar	Camera	FOV Overlap	Calibration
Kitti[6]	1	4	120°	✓
Nuscenes	1	**6**	**360°**	✓
Cityscapes[11]	0	1	✗	✗
Waymo[13]	1	5	270°	✓

Table 1. Dataset comparison.

3.2 Pre-Processing

In pre-processing, in order to obtain corresponding data pairs that represent the same moment and view from a camera and a LiDAR sensor, we projected LiDAR 3D points to the image coordinate system by applying some transformation and projection steps. The transformation takes place in this order, from the LiDAR's coordinate system to the LiDAR's world coordinate system, from LiDAR's world coordinate system to the camera's world coordinate system, and finally from the camera's world coordinate system to the camera's coordinate system. Finally, to change the final form of our 3D points to pixels, we projected 3D points into the image coordinate system by multiplying the camera's intrinsic matrix and kept distance and reflection values in a 2 by the image height by the image width shaped tensor. Final tensor scaled to 512 by 512 and distance values are normalized by dividing by maximum detectable point distance of LiDAR sensor of 250 and reflection values are normalized by dividing by maximum reflection unit of 255.

Fig.1. Projected LiDAR points depth (left column), projected LiDAR points reflection (middle left column), front camera image (middle right column), LiDAR points projected on the camera image (right column).

3.3 Training Dataset

Using the nuScenes dataset we made our POVGAN training dataset which uses 6,573 LiDAR point clouds, 6,573 images from each camera which are mounted at the front, front left, front right, back, back left and back right of the vehicle. The point clouds were captured by average automotive grade LiDAR in 2022 which has 32 layers, 360° horizontal FOV, $-30°$ to $+10°$ vertical FOV and ± 2 cm accuracy within 70 meters. Images were captured by 1600 x 900 resolution cameras and cameras are used in auto-exposure mode. We applied described pre-possessing steps on LiDAR point clouds to obtain projected LiDAR points tensor so our POVGAN training dataset includes 39,438 projected LiDAR points and 39,438 corresponding images.

4 Methods

4.1 The pix2pixHD Baseline

Pix2pixHD [10] is a state-of-the-art image-to-image translation framework which solves photo-realism and resolution problems of the Pix2pix[7] model. Pix2pixHD is able to perform image-to-image transformation on High-definition images by using Coarse-to-fine Generator architecture. To forward our pre-processed projected LiDAR point to our Pix2pixHD network, we set initial layer channels of the Pix2pixHD network to 2 channels which contain the depth of point in 1st channel and reflection of the point in the 2nd channel.

Coarse-to-fine generator Instead of using one generator to produce high resolution images, this method uses multiple generators in different scales. Each generator can learn different scale features to produce photo-realistic images. The number of required generators is dependent on the output image resolution. The generators are named global generator and local generator and the global generator works on low resolution input while local generator improves on results from the global generator and outputs 2 times scaled image.

Multi-scale Discriminators High resolution images can be challenging for a single discriminator. Using a big neural network for a discriminator most likely leads to over fitting. This problem can be easily solved by using multiple discriminators with different scales. Each discriminator performs better in their scales thus we are able to differentiate real and fake images from coarse to fine scale. Furthermore this solution requires smaller footprints for training and uses less resources.

4.2 Improving Obstacle Photo-realism

Deep learning perception algorithms[1] are an important part of AV development and they solve many different detection challenges. In detail, obstacle detection is one of the crucial parts of autonomous vehicles which helps the vehicle to avoid collisions by detecting cars, pedestrians and cyclists and helps the vehicle to follow traffic rules by detecting traffic lights and signs. Knowing that, we are inclined to improve photo-realism of obstacles. We think that the use of masks for training is useful to identify obstacle pixels and background pixels so we projected 3D bounding boxes of obstacles provided by nuScenes dataset to the image coordinate system and we filled all surfaces of bounding boxes with masks.

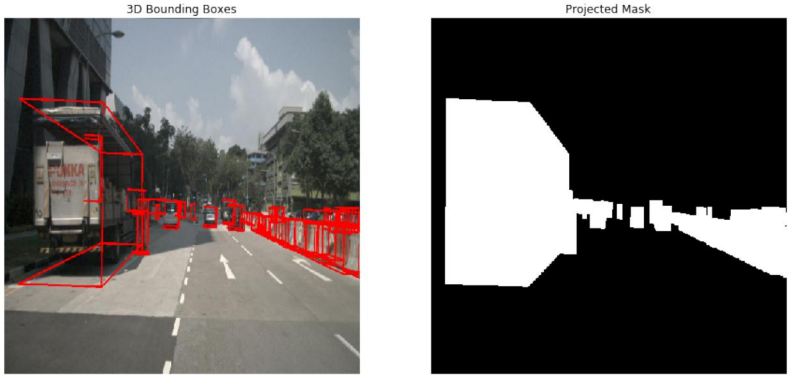

Fig.2. Projected 3D bounding boxes and filled 2D boxes of obstacles.

To enforce the network to learn more features of obstacles than background features, we penalized the generator network more on predicting masked pixels by applying weighted mean square error between generated image and real image. This loss function will be added onto the Pix2pixHD loss functions. The pre-processed LiDAR point cloud is represented by I^{ldr}, the camera image represented by I^{cam} and obstacle mask represented by I^{msk}. This obstacle loss function for image with N pixels can be expressed as:

$$L_O(G) = \mathbb{E}_(I^{ldr}, I^{cam})[\frac{1}{N}\frac{\sum_{n=1}^{N} w_n \cdot (I_n^{cam} - G(I_n^{ldr}))^2}{\sum_{n=1}^{N} w_i}]$$

where λ_{mask} is a hyper parameter and adjusts weight of masked pixels.

$$w_n = (\mathbb{1}[I_n^{mask}] + 1)\lambda_{mask}$$

In the rest of the paper we will use Mask-POVGAN for POVGAN trained with obstacle loss function to differentiate effects and results on generated images.

5 Experiments/Results/Discussion

5.1 POVGAN

In our experiment, all networks are trained from scratch by using the recommended Pix2pixHD optimizer settings which is Adam solver [5] with 0.0002 learning rate. We have trained 2 POVGAN and 1 Mask-POVGAN network to compare and evaluate our results. The first POVGAN was trained to epoch 40 with batch size of 6, the second POVGAN was trained to epoch 20 with batch size of 1 and Mask-POVGAN was trained to epoch 20 with batch size of 1, $_mask$ set to 5. Preparing our results, we pre-processed another recording of the nuScenes dataset and chose 500 random pre-processed data pairs from each of the vehicle cameras and the vehicle LiDAR. To qualify the quality of our synthetic images generated by POVGAN, we used mask-RCNN [9] to detect obstacles on the real images and the synthetic images in order to compare the number of total detected obstacles between real and synthetic images.

Fig.3. Synthetic image generated by POVGAN.

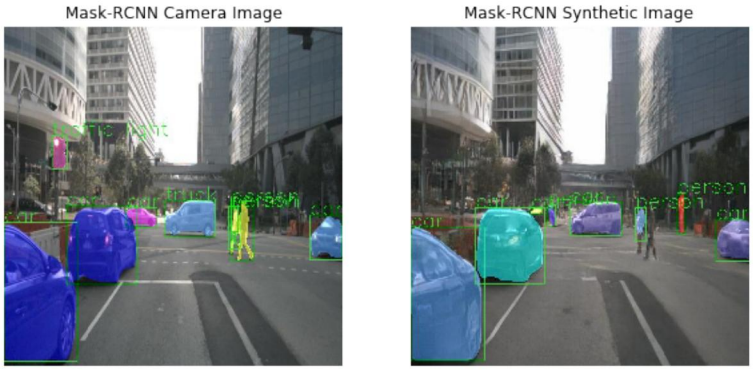

Fig.4. Obstacles are detected by Mask-RCNN on synthetic and real image.

	car	person	bus	truck
Synthetic	7271	1361	344	966
Real	9922	2940	366	1400
	77.81%	46.29%	**93.98%**	69.00%

Table 2. Number of objets detected on synthetic images generated by POVGAN(epoch 40) and number of object detected on real images captured by vehicle cameras.

From our result, we conclude that generating synthetic images of big obstacles is much easier than small obstacles. Additionally, we found that traffic lights and signs are a challenging problem for POVGAN because they are small and mounted at high places thus LiDAR is not able to obtain any point or there are not enough points to learn the features.

5.2 Camera Transformation

The most important feature of POVGAN is changing the point of view to generate synthetic images from transformed camera viewpoints. In this experiment, we applied different camera transformations on viewpoints to generate synthetic images.

Fig.5. POVGAN generated on transformation applied view points.

5.3 Mask-POVGAN

We compare POVGAN (20 epoch) and Mask-POVGAN (20 epoch) on an obstacle detection task by using Mask-RCNN to qualify the performance of the obstacle loss function.

	car	person	bus	truck
Mask-POVGAN	5439	822	255	800
POVGAN	5944	1329	293	858
Real	9922	2940	366	1400

Mask-POVGAN did not perform better than POVGAN on the obstacle detection task. There might be multiple reasons for it. Firstly, our obstacle mask not only covers car, person, bus and truck it also covers other traffic obstacles and this might prevent the network from learning features of car, person, bus and truck. Secondly, we need to fine-tune the hyper parameters of Mask-POVGAN.

6 Conclusion/Future Work

In all, we demonstrated another way of synthetic image generation from LiDAR point clouds and showed how to use sensor transformations to generate paired LiDAR and camera data to train GAN. We presented a new synthetic data generation method that is much more flexible than generating synthetic images from labels by being able to change viewpoints. Traditional labels to synthetic data generation networks suffer from identifying rotation of obstacles and only perform on front camera view, however, POVGAN is able to perform on all cameras around the vehicle and is able to learn rotation of obstacles. Additionally, we have tested the obstacle loss function which did not perform well on the recommended Pix2pixHD hyper parameters setting and we will be fine-tuning our hyper parameters and increasing the number of epochs in our future work. Finally, with our method, AV

developers might not need to record data for training their perception networks and this can speed up the development of AV.

References

[1] Deepak Talwar, Sachin Guruswamy, Naveen Ravipati, Magdalini Eirinaki Evaluating Validity of Synthetic Data in Perception Tasks for Autonomous Vehicles, 2020;

[2] Tomas Bubenicek, Supervised by: Jiri Bittner Using Game Engine to Generate Synthetic Datasets for Machine Learning, 2020;

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio. Generative Adversarial Networks, 2014; arXiv:1406.2661.

[4] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, 2017; arXiv:1711.11585.

[5] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, 2014; arXiv:1412.6980.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset.

[7] Jun-Yan Zhu, Taesung Park, Phillip Isola and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, 2017; arXiv:1703.10593.

[8] Nuriel Shalom Mor. Generating Photo-realistic Images from LiDAR Point Clouds with Generative Adversarial Networks, 2021; arXiv:2112.11245.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár and Ross Girshick. Mask R-CNN, 2017; arXiv:1703.06870.

[10] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz and Bryan Catanzaro. High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, 2017; arXiv:1711.11585.

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding, 2016; arXiv:1604.01685.

[12] 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, 2019; arXiv:1903.11027.

[13] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset, 2019; arXiv:1912.04838.