Multi-Echo LiDAR for 3D Object Detection

Yunze Man[1], Xinshuo Weng[1], Prasanna Kumar Sivakumar[2], Matthew O'Toole[1], Kris Kitani[1]
[1]Carnegie Mellon University, [2]DENSO
{yman, xinshuow, mpotoole, kkitani}@cs.cmu.edu, prasanna.kumar.sivakumar@na.denso.com

Supplementary

A. Synthetic Dataset Simulation

Though we briefly describe the synthetic multi-signal LiDAR measurement generation process and its data format in the main paper, here we provide details about the entire data simulation process. We will introduce our simulation of ambient signals (Sec A.2) as well as multi-echo point cloud and reflectance signals (Sec A.3).

Our simulation is based on a new large-scale synthetic dataset AIODrive [5] which is in submission to another conference. We cite the paper anonymously and provide it as supplementary "*AIODrive.pdf*".

A.1. Input Coordinate Transformation

As shown in Algorithm 1, we take RGB, depth and surface normal images, denotes as I_{rgb}, I_d, I_n, as the input of the simulation process, where the RGB and depth signals are directly captured by top-mounted RGBD cameras [5], and the surface normal is directly converted from depth images – The process is accurate because of the perfect synthetic depth values.

In order to mimic the spinning mechanism of LiDAR sensor, we perform a Polar-to-Image coordinate transformation on all input images. Specifically, we approximate the LiDAR sensor as a point in the 3D space and define a LiDAR sensor array $A[R_V, R_H]$ in the polar coordinate (*i.e.* elevation and azimuth). For all $i \in [1, R_V], j \in [1, R_H]$,

$$A(i,j) = (\theta_i, \gamma_j), \qquad \theta_i, \gamma_j \in [0, 2\pi) \tag{1}$$

where θ and γ are vectors representing the LiDAR sensor detector arrangement, and are predefined according to the desired vertical and horizontal FoV and resolution, and R_H denotes the number of vertically aligned LiDAR detectors on the sensor, and R_V denotes the number of horizontally aligned detection a detector performs during one single sweep. Then, we project the polar coordinate map $A[R_V, R_H]$ onto the 2D image space, resulting in non-linearly arranged positional map $A_{2D}[R_V, R_H]$ on the 2D image plane. Then, we generate new images from the input images by sampling and interpolating pixels according to the 2D positional map $A_{2D}[R_V, R_H]$. For any input image $I_{im} \in \{I_{rgb}, I_d, I_n\}, \forall i \in [1, R_V], j \in [1, R_H]$,

$$I'_{im}(i,j) = \text{interp}(I_{im}(x,y)), \;\; (x,y) = A_{2D}(i,j) \tag{2}$$

Then we get new images I'_{rgb}, I'_d, I'_n where pixels are arranged according to the arrangement of LiDAR detector orientations.

A.2. Ambient Illumination Signals

The ambient illumination signals are sunlight reflected by objects. Thus, from the imaging perspective, the information encoded in the ambient signals is very similar to that collected by RGB camera. Collected using CARLA [1] simulator, the AIODrive dataset [5] captures RGB images in the scene by deploying RGB cameras on top of the ego-vehicles. Since LiDAR sensor usually captures light with infrared wavelength, we take R-channel from the RGB images and treat it as a simulation of the ambient signal. The ambient signal is represented in the form of images I_{ambient},

$$I_{\text{ambient}} \sim I'_r$$

where I'_r denotes the R-channel of image I' after coordinate transformation.

A.3. Multi-echo Point Cloud and Reflectance Signals

Generate Raw 3D Tensor Data. Because the LiDAR sensor is essentially a time-of-flight measurement of photons, to simulate multi-signal LiDAR sensing measurements, we first simulate the photon measurements. Specifically, without considering the random false detection which happens occasionally in real LiDAR sensors, we formulate the number of photons $\mathbf{N_p}$ received by a LiDAR detector in response to an illumination period of a signal light pulse by a temporal histogram:

$$\mathbf{N_p}[n] \sim \mathcal{P}(N_{\text{signal}}[n] + N_{\text{ambient}}[n]), \tag{3}$$

where n is the n-th time interval along the temporal axis. Function $\mathcal{P}(\cdot)$ models a Poisson distribution, $N_{\text{signal}}[n]$ is the number of detected signal photons at the time interval n, and N_{ambient} models the number of ambient photons.

Algorithm 1 Multi-echo LiDAR Simulation

Inputs: $\mathbf{I_{rgb}}$: RGB image
$\mathbf{I_d}$: Depth image
$\mathbf{I_n}$: Surface normal image

Parameters: $\mathbf{N}$: Number of time bins
$\mathbf{SBR}$: Signal-Background-Ratio
$\mathbf{R_H}, \mathbf{F_H}$: Horizontal Resolution and FoV
$\mathbf{R_V}, \mathbf{F_V}$: Vertical Resolution and FoV
$\mathbf{S}$: Neighborhood aggregation kernel size
$\mathbf{K}$: Number of echo groups

Outputs: $\mathbf{I}_{\text{ambient}}[R_V, R_H]$: Ambient image
$\mathbf{I}_{\text{reflectance}}[R_V, R_H, K]$: Reflectance image
$\mathbf{P_K}$: Multi-echo Point Cloud Returns

Sample LiDAR sensor array $A[R_H, R_V]$ in the polar coordinate according to FoV F_H, F_V
Project $A[R_H, R_V]$ onto the 2D image space and get the positional map $A_{2D}[R_H, R_V]$
Transform the input images I_{im} into polar coordinate $I_{\text{im}'}$ as Eq. 2 ▷ Get ambient image $\mathbf{I}_{\text{ambient}}[R_V, R_H]$

Calculate signal photons N_{signal} for each pixel (h, w) from Eq. 4;
Calculate ambient photons N_{ambient} for each pixel (h, w) from Eq. 5;
Calculate ambient photons $N_{\text{p}}[R_V, R_H]$ for each pixel (h, w) from Eq. 3;

Simulate multi-echo mechanism and generate $\mathbf{N}^*_{\mathbf{p}}[R_H, R_V, N]$ by neighborhood aggregation as in Eq. 6
Get Top-K returns along the temporal axis with Eq. 7
Project the valid bins into 3D space and generate Top-K point cloud returns ▷ Get multi-echo point cloud $\mathbf{P_K}$
Simulate the reflectance by number of points insides its corresponding bins
Rearrange reflectance values of points into 'LiDAR Image' space ▷ Get reflectance image $\mathbf{I}_{\text{reflectance}}[R_V, R_H, K]$

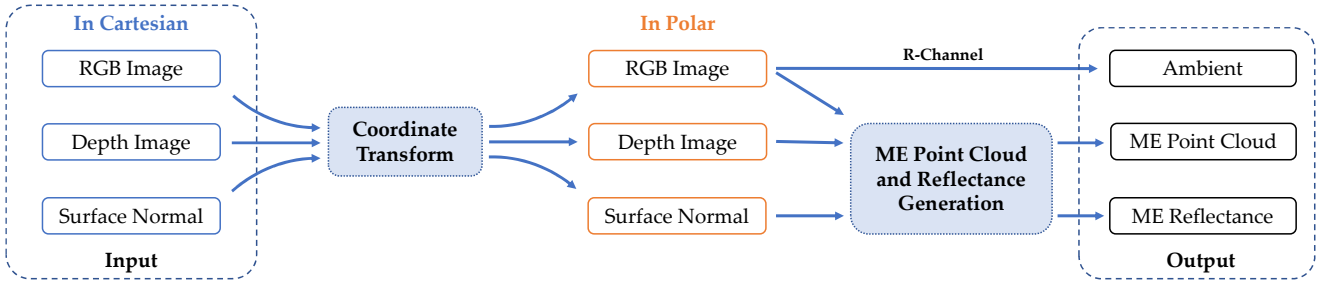

Figure 1: **Multi-signal LiDAR measurements Simulation.** Taking the inputs from AIODrive [5], we first transform the images from Cartesian space into polar space to mimic the LiDAR spinning mechanism. Then infrared ambient illumination is approximated by taking R channel of the RGB image. Multi-echo point cloud and reflectance signals are generated from trasformed RGB, Depth and surface normal images.

Based on the Eq. 3, the first step of our raw multi-echo LiDAR data generation is to generate a 3D tensor of photon counts $\mathbf{N_p}[R_V, R_H, N]$ representing the number of photons detected by the sensor. Here (R_V, R_H) is the vertical and horizontal resolution of the LiDAR (height and width of the 'LiDAR Image') and N represents the number of time intervals.

To model the number of signal photons $N_{\text{signal}}[n]$, we consider the surface reflectance, angle of incidence during reflection and radial attenuation. We model the relative photon number by assuming all LiDAR transmitters emit lasers with the same energy (same number of photons). Then, according to Lambert's cosine law, the reflected energy is proportional to $\cos(\theta)$ where θ is the incidence angle of the light with respect to the surface. This information is given by surface normal image I'_n. We use a near infrared signal light, *i.e*, the R channel of the RGB image I'_r, to approximate the reflectance of the surface. Also, the radial falloff (attenuation) of light energy is proportional to the square of travel distance. We can directly take advantage of the accu-

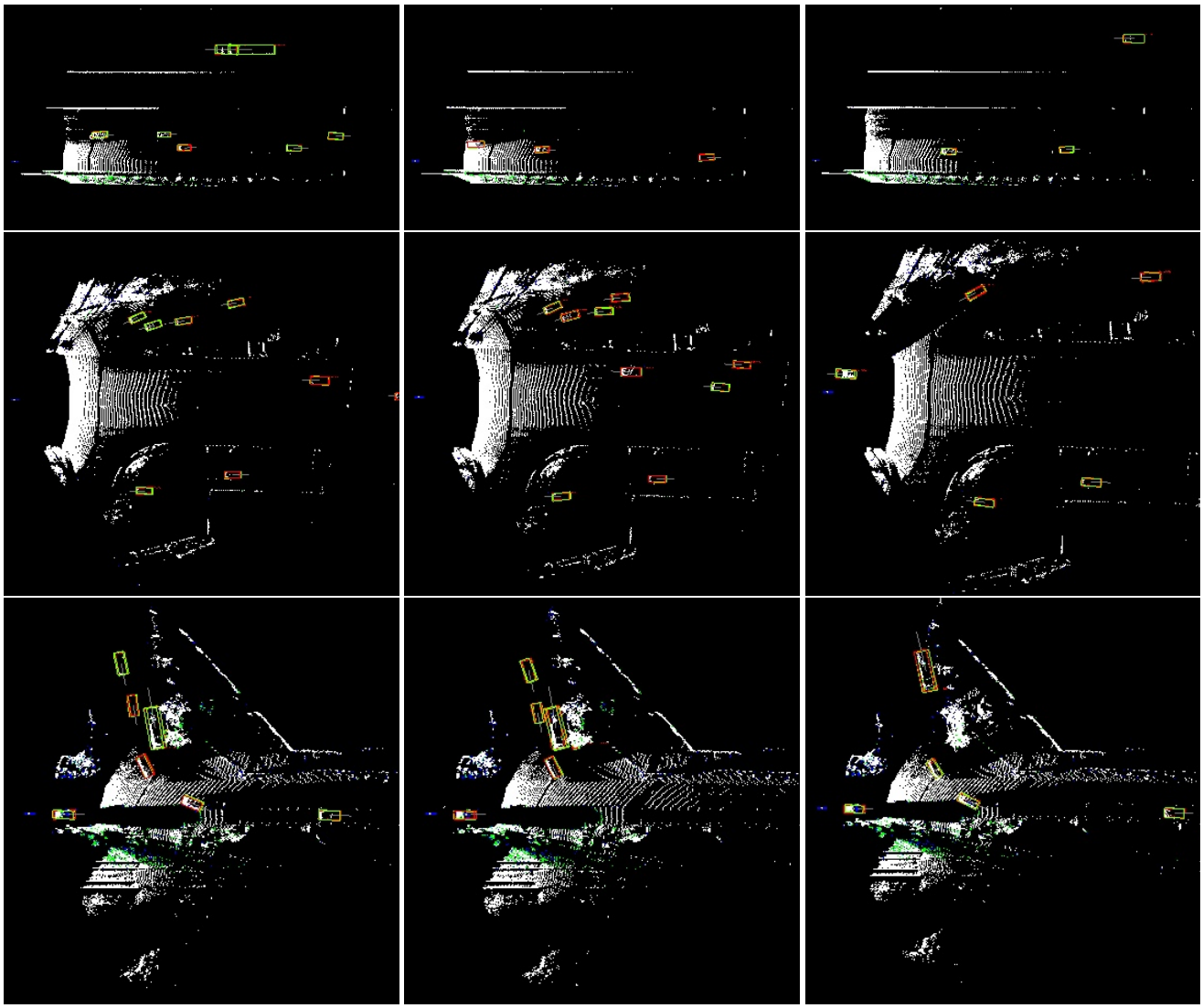

Figure 2: **Qualitative results of our MSLiD on real dataset.** The ground truth bounding boxes and predicted bounding boxes are labeled as red and green respectively. Note that 1st echo points are shown as white, 2nd echo points are shown as green, and 3rd echo points are shown as blue.

rate depth image I'_d. Then, the number of signal photons is modeled as:

$$\mathbf{N}_{\mathsf{signal}}(h,w,n) \sim \begin{cases} \mathsf{Norm}\left(\mathsf{SBR} \times \frac{I'_r(h,w)\cdot\cos\theta}{I'_d(h,w)^2}\right) & \text{If } n = n^* \\ 0 & \text{If } n \neq n^* \end{cases} \quad (4)$$

where the Norm operation means to normalize over the whole image, (*i.e.* divided by the average value in the entire image), SBR is the Signal-Background-Ratio used to control the relative strength between signal and background light, and n^* is the time bin during which the signal light is reflected by the surface.

To model the number of ambient photons $N_{\mathsf{ambient}}[n]$, we simply takes the R-channel of the RGB images and normalize over the whole image,

$$N_{\mathsf{ambient}}(h,w,n) \sim \mathsf{Norm}\left(I_r[h,w]\right), \quad \forall n \in [1,N] \quad (5)$$

Then using Eq. 3 together with Eq. 4 and Eq. 5, we can simulate the 3D tensor of photon number $\mathbf{N_p}[R_V, R_H, N]$.

Given the 3D tensor of photon numbers, the next step is to model the multi-echo mechanism. As explained in the main paper, multiple echoes happen because laser beams have a wider coverage of the 3D space instead of a perfect 2D line. In the 2D 'LiDAR Image' space, this can be explained as – Neighbor pixels overlap with each other.

We use a kernel function $G(\cdot)$ to simulate the spatial coverage, *i.e.*, the number of photons in a given time bin will be a weighted sum of its spatial neighborhood bins (not temporal ones), with nearer neighbors contributing more:

$$\mathbf{N}^*_{\mathbf{p}}[h,w,n] = \sum_{(h,w)\in\mathcal{N}(h,w)} \sum G(k_h,k_w)\cdot\mathbf{N_p}[h,w,n], \quad (6)$$

where $\mathcal{N}$ is the neighborhood of a given position on the

Table 1: **Performance comparison of 3D object detection with SOTA methods on our collected real-wold dataset**.

Method		Car - IoU = 0.7			Car - IoU = 0.5			Person - IoU = 0.5			Person - IoU = 0.25		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SECOND [6]	full echo	75.0	42.9	30.6	86.8	65.5	65.1	47.4	29.9	20.3	58.5	39.2	25.6
	+ambient	74.2	42.0	30.1	86.5	65.6	62.9	46.9	29.7	19.8	58.9	39.1	25.4
	+echo label	75.0	43.2	30.5	86.9	65.9	65.0	48.2	30.2	20.0	59.1	39.5	25.6
PV-RCNN [3]	full echo	76.9	44.1	31.2	88.1	67.2	65.3	52.7	30.9	20.8	62.0	41.2	26.2
	+ambient	76.3	43.8	**31.3**	87.9	67.1	65.0	52.6	30.3	20.4	61.7	41.3	25.8
	+echo label	77.4	44.0	30.9	88.4	66.9	64.9	53.5	31.4	20.6	62.8	41.7	26.0
MSLiD		**79.5**	**45.3**	30.7	**89.7**	**68.1**	**65.3**	**57.5**	**34.2**	**21.5**	**66.5**	**43.2**	**27.7**
Improvement		*+2.1*	*+1.2*	*-0.6*	*+1.3*	*+0.9*	*+0.0*	*+4.0*	*+2.8*	*+0.7*	*+3.7*	*+1.5*	*+1.5*

image plane, and $G(k_h, k_w)$ is the weight function over the distance between a given 2D position (r_h, r_w) and its neighbor position (k_h, k_w). Specifically, we use a Gaussian function for $G(\cdot)$. By controlling the parameters of the kernel function, we can control the spatial coverage of the lasers.

Generate Top-K Point Cloud Returns. Because standard LiDAR-based perception systems [4, 6, 2] take point cloud data as input (not raw photon count data $\mathbf{N}^*_{\mathbf{p}}[R_V, R_H, N]$), we take one step further to convert our raw multi-echo LiDAR data into point clouds so that they can be easily used in modern perception systems. Note that, with a large spatial coverage rate, each laser beam is able to cover a large 3D volume and is more likely to hit more than one target (object). This is represented as multiple strong peaks along the temporal histogram of a sensor beam. Specifically, when generating the top-K point cloud returns, we take the top-K maximum bins along the temporal axis for each sensor beam, *i.e.*, we select the bin with the top-K number of photons that exceeds a threshold and obtain $\mathbf{N}^*_{\mathbf{p}}[R_V, R_H, K]$ as follows:

$$\mathbf{N}^*_{\mathbf{p}}[R_V, R_H, K] = T\left(S\left(\mathbf{N}^*_{\mathbf{p}}[R_V, R_H, N]\right)[:, :, :\mathrm{K}]\right), \quad (7)$$

where $S(\cdot)$ is a sort function that descendingly sorts the number of photon counts along the temporal axis N. Then we only take the top-K channels in the temporal axis (*i.e.*, the top-k maximum bins). Also, $T(\cdot)$ is a threshold function that masks out bins with number of photons less than a threshold, *i.e.*, reject bins that receive noise instead of a light signal. Once we have obtained the $\mathbf{N}^*_{\mathbf{p}}[R_H, R_V, K]$, we can then transform it to K point clouds as each valid bin (non-rejected) can be back-projected to a point in 3D space. As we use a threshold to mask out invalid points, the number of valid points will be fewer when K is higher, *i.e.*, the 1st strongest point cloud has more points than the 2nd strongest point cloud and so on.

After getting the multi-echo point cloud, we can simulate the reflectance of each point by normalizing the number of points inside the bins, because we assume that each LiDAR sensor transmitter emits same number of photons. The reflectance values of the multi-echo point cloud can be rearranged into the 'LiDAR Image' space $\mathbf{I}_{\mathsf{reflectance}}[R_V, R_H, K]$.

We summarize our multi-echo LiDAR simulation process in Algorithm 1 and in Figure 1. In terms of the implementation details, we use our all five camera viewpoints and create a 360° FoV of the multi-echo LiDAR point cloud. We use 10240 numbers of time bins to voxelize 1000 meters of depth range in each view. When we sample the LiDAR sensor array in the polar coordinate, each view has a 45° vertical FoV and 140° horizontal FoV. For frontal view, we sample in $[-20^\circ, 25^\circ]$ vertical FoV with 0.2° of resolution, and $[-70^\circ, 70^\circ]$ horizontal FoV with 0.1° of resolution. To simulate a relatively large spatial coverage, we define a $[5, 5]$ patch in image coordinate centered around the projected position as its neighborhood.

B. Qualitative results of MSLiD

In the Sec. 4.3 of the main paper, we give a quantitative analysis of our method on the real-world dataset. Here, we shown some qualitative results of our MSLiD in Fig 2. Note that 1st echo points are shown as white, 2nd echo points are shown as green, and 3rd echo points are shown as blue. Our method achieves good results on the real-world dataset in different traffic and scenarios.

C. Comparisons with SOTA methods on extra input dimension

In Table 1, we show results for [6, 3] using points with extra input dimensions (ambient and echo index label), and validate that the improvement of MSLiD is from our technical contributions. Multi-echo points are regarded as independent points and grouped into one point cloud. In 'full echo', points have 3D coordinates and reflectance, with the dimension of $[x, y, z, r]$. In '+ambient', an additional ambient channel a is added to each point. In '+echo label', both the ambient value a and the echo label e (indicating the echo group of each point) are added, resulting in the point input dimension $[x, y, z, r, a, e]$.

From the results, we observed that adding ambient a

does not significantly affect performance, and adding echo label only shows incremental improvements to [6, 3] on "easy" and "moderate". This means that trivially adding extra input dimensions (ambient and echo label) cannot provide significant improvements while the technique proposed in our MSLiD model can better leverage these extra inputs.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017. 1

[2] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 4

[3] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4, 5

[4] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 4

[5] Xinshuo Weng, Yunze Man, Dazhi Cheng, Jinhyung Park, Matthew O'Toole, and Kris Kitani. All-In-One Drive: A Large-Scale Comprehensive Perception Dataset with High-Density Long-Range Point Clouds. *arXiv*, 2020. 1, 2

[6] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 4, 5