Better Monocular 3D Detectors with LiDAR from the Past

Yurong You*†, Cheng Perng Phoo*†, Carlos Andres Diaz-Ruiz‡, Katie Z Luo†,
Wei-Lun Chao§, Mark Campbell‡, Bharath Hariharan†, Kilian Q Weinberger†

***Abstract*— Accurate 3D object detection is crucial to autonomous driving. Though LiDAR-based detectors have achieved impressive performance, the high cost of LiDAR sensors precludes their widespread adoption in affordable vehicles. Camera-based detectors are cheaper alternatives but often suffer inferior performance compared to their LiDAR-based counterparts due to inherent depth ambiguities in images. In this work, we seek to improve monocular 3D detectors by leveraging unlabeled historical LiDAR data. Specifically, at inference time, we assume that the camera-based detectors have access to multiple unlabeled LiDAR scans from past traversals at locations of interest (potentially from other high-end vehicles equipped with LiDAR sensors). Under this setup, we proposed a novel, simple, and end-to-end trainable framework, termed AsyncDepth, to effectively extract relevant features from asynchronous LiDAR traversals of the same location for monocular 3D detectors. We show consistent and significant performance gain (up to 9 AP) across multiple state-of-the-art models and datasets with a negligible additional latency of 9.66 ms and a small storage cost. Our code can be found at https://github.com/YurongYou/AsyncDepth.**

I. Introduction

To drive safely, autonomous vehicles and driver assist systems must detect traffic participants and obstacles accurately. Current state-of-the-art prototypes rely on LiDAR sensors that provide accurate 3D information[1]. However, LiDAR sensors are expensive and their high cost precludes their mass adoption in consumer cars. Most commercially available driver assist systems instead rely on cheaper sensors — (360°-view) monocular cameras. Although more affordable, image-based 3D object detectors substantially underperform their LiDAR-based counterparts due to the inherent difficulty of inferring depth from images [2].

While it may be impractical and cost-prohibitive for *every* vehicle to be equipped with LiDAR sensors, *some* (e.g. high-end luxury, police, *etc.*) vehicles within a community may be outfitted with such sensors. In this setting, a few LiDAR-equipped vehicles collect data and share them (anonymously) with a large fleet of camera-only cars. If a camera-only car traverses a route for which past LiDAR data is available, it can fuse this data with its own sensor readings. A natural question to ask is: *can we improve camera-based 3D object detectors using LiDAR data from the same location, but collected in the past?*

∗ Equal contributions
† Computer Science Department, Cornell University {yy785, cp598, kzl6, bh497, kqw4}@cornell.edu
‡ Mechanical and Aerospace Engineering Department, Cornell University {cad297, mc288}@cornell.edu
§ Department of Computer Science and Engineering, Ohio State University chao.209@osu.edu

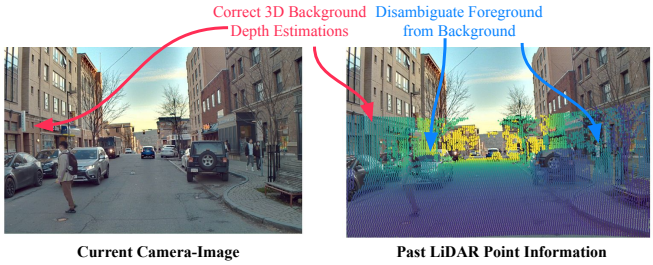

Fig. 1: **Can past LiDAR traversals help monocular 3D object detection?** Here we show a current image (left) and an asynchronous depth map rendered from a past LiDAR traversal (right). The asynchronous depth map provides accurate depth for background regions (red arrows) and helps the monocular model disambiguate foreground objects in current scene (blue arrows).

Prior work has shown that combining 3D LiDAR point clouds with 2D images can improve 3D object detection [3]–[5], but these models crucially relies on a *synchronized* LiDAR and camera sensors. In our case, however, the 3D data we have comes from a *different car* passing through the scene presumably at a *different time*. As such, vehicles and pedestrians will obviously have moved in the interim. Since these are the objects we want to detect in the current scene, these asynchronous offline LiDAR scans will not capture the shape and location of these objects.

However, even though the objects of interest may not be present in these past LiDAR scans, we argue that these scans still contain vital information for accurate 3D object detection. By aggregating data across multiple traversals, we can identify and remove transient objects [6] and thus obtain accurate 3D data about the static background. We posit that this 3D information about the background, collected from past traversals, can then be used to both *detect* foreground objects in the current image as well as *localize* them. First, because foreground objects move and are therefore *transient*, they will correspond to regions where the current image data is inconsistent with the previously collected depth (blue arrows in Figure 1). This can help the model detect ambiguous or partially occluded objects. Second, in the areas where the previously collected depth is consistent with the current image, (*i.e.*, the background), we get accurate depth for free (red arrows). This accurate depth can be used by the image detector to localize foreground objects in 3D, *e.g.*, by reasoning about where the pedestrian's feet meet the road.

Based on this insight, we propose a simple and effective approach for combining these asynchronous 3D data from past traversals with image-based detectors. We project each

of the point clouds aggregated at each location into a depth map for each camera [7], [8]. From these depth maps, features are extracted, pooled across all past traversals, and combined with the image representation as the (intermediate) input to the monocular detector. During training, the depth-map based feature extractor is trained jointly with the object detector. During inference, the camera-only model can use the features extracted from past LiDAR scans to better detect and localize the objects.

We validate our approach, termed AsyncDepth, across two real-world self driving datasets, Lyft L5 Perception [9] and Ithaca365 [10], with two representative camera-based 3D object detection models [1], [11]–[14]. Using our method, we observe a consistent improvement across both datasets, and up to 9.5 mAP over the baselines on far away ranges. Our contributions are as follows:

- We study a novel yet highly practical scenario where *asynchronous* historical LiDAR point cloud data is available to *camera-only* perception systems.
- We show the practicality by proposing a simple and general approach to integrate asynchronous point cloud data into 3D monocular object detectors.
- We empirically demonstrate that our method yields consistent performance gains with low additional latency (9.66 ms) and a tiny storage cost across different datasets, detection ranges, object types, and detectors.

II. Related Work

Perception for Autonomous Vehicles. Sensing the environment around a vehicle can be done via different input modalities, such as LiDAR or camera. LiDAR sensors are more expensive than cameras but are capable of capturing 3D geometry of the traffic scenes at high fidelity. Current state-of-the-art 3D object detectors therefore mostly rely on LiDAR sensors [15]–[19]. Camera-based 3D object detectors are a cheaper but also less accurate alternative due to depth ambiguities induced by perspective projections. The use of stereo-cameras [2], [5], [20]–[22] can close the gap, however the most common sensors in end user cars are monocular cameras [11], [13], [14], [20], [23]–[33], as they can be easily integrated within the car to capture a full 360 view around the vehicle. These monocular-based models can be roughly divided into two categories based on whether the detection is performed in 2D perspective view [13], [23], [24], [26], [27] or in 3D [11], [14], [20], [25], [28]–[33]. In this work we show that both types of these models can be vastly improved through the use of offline LiDAR scans from past traversals.

Sensor Fusion in 3D Object Detection. LiDAR sensors yield accurate 3D geometry but suffer from sparse resolution; cameras, on other other hand, provide high resolution input but are inept in capturing 3D information. Given their complementary characteristics, multiple research efforts have explored fusing LiDAR and images for better 3D object detection [3], [4], [34], [35]. In contrast to our work, these approaches typically assume *synchronous* sensors and still require expensive LiDARs during inference. You et al. [5] address the cost issue by proposing to correct camera based perception through sparser (4-beams) and therefore cheaper LiDAR sensors. We explore an alternative setup, where the current scene is only captured by cheap cameras but *asynchronous* LiDAR scans from the past are available. In principle, this is a much harder setup but it is of great practical value as it allows all cars to benefit from the LiDAR scans of a few (expensive) vehicles.

LiDAR Scans from Past Traversals. With accurate GPS/INS, LiDAR data from past traversals can be geo-located and aligned for easy retrieval. Recent work has started to explore the use of such data to aid perception and visual odometry [6]. MODEST [36] leverages past traversals to discover dynamic objects without any annotations. Rote-DA [37] utilizes previous traversals to adapt pre-trained detectors to new target domains. You et al. [38] propose the use of past LiDAR point clouds to create feature descriptors for improving LiDAR-based 3D object detectors. Nonetheless, these prior works demonstrate that past traversals are useful for various tasks in autonomous driving. We leverage this observation but in contrast, we are the first to show how to utilize past LiDAR to improve camera-only 3D object detection — a common setting in practice.

III. AsyncDepth

Setup. We follow a typical test-time *camera-only* sensor setup: the autonomous vehicle is equipped with synchronized sensors, including C calibrated cameras and localization sensors (*e.g.*, GPS/INS) but no LiDAR sensors. The C cameras are calibrated and have corresponding intrinsic and extrinsic matrices $\{(P^i, T^i)\}$. These cameras capture C images of the surrounding environment at a certain frequency. When the vehicle drives through a location p, we denote the instantaneous images captured by the cameras as $\{\boldsymbol{I}_p^i \in \mathbb{R}^{H\times W\times 3}, i = 1, \ldots, C\}$. We also record the global 6-DoF localization as a rigid transformation G_p that maps from the local to the global coordinate frame. We do not assume that the fields of view of the C cameras overlap, so work aims to develop a monocular 3D object detector that can identify objects of interest (dynamic traffic participants like cars, pedestrians, *etc.*) and infer their 3D positions, orientations, and sizes in the scene from these images.

Detector abstraction. Current state-of-the-art monocular 3D object detectors [11], [13], [23]–[25] mostly follow a "featurize-then-detect" pipeline. Given images $\{\boldsymbol{I}_p^i\}$ and the corresponding camera parameters $\{(P^i, T^i)\}$, the detector first extracts image features $\{f(\boldsymbol{I}_p^i)\}$ from each of the images via a (usually pre-trained) featurizer f; a detector head h then lifts these 2D feature maps to 3D object bounding boxes $\mathcal{B}_p = h(\{f(\boldsymbol{I}_p^i), P^i, T^i\})$ (here f and h contain learnable parameters). Such lifting from 2D to 3D is a notoriously *ill-posed* problem for monocular detectors, since accurate *depth* cannot be measured geometrically from the given 2D images [39]. Current detectors get around this issue by learning a prior over depth [11], [13], [14], [25], [32], but without any geometric or multi-view information, accurate depth estimation can be challenging.

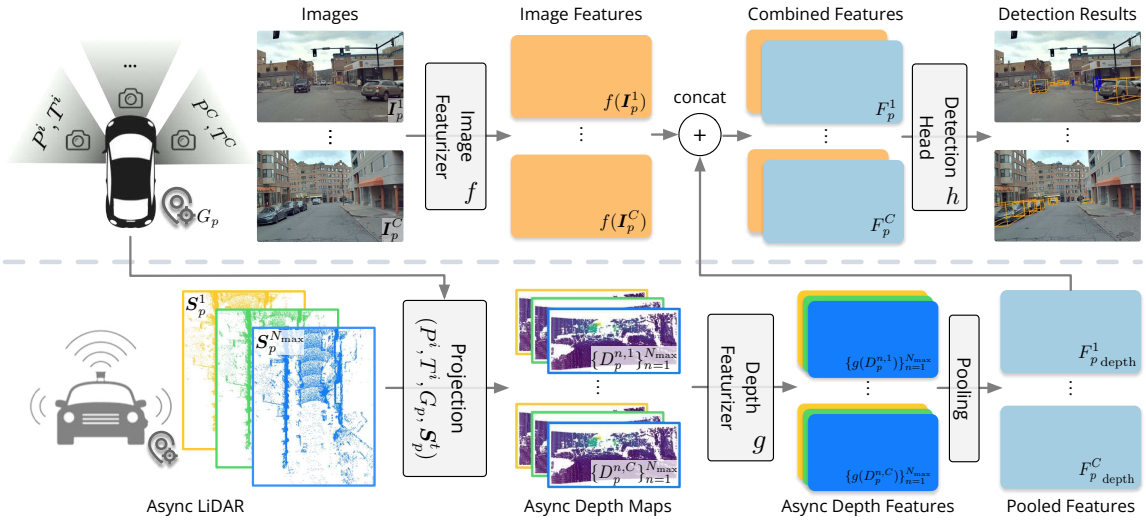

Fig. 2: **Overview of AsyncDepth.** It consists of three parts: (top left) general "featurize-then-detect" pipeline for monocular 3D detection; (bottom) extracting asynchronous depth features from past LiDAR traversals of the same location; (top right) fusing the image features with AsyncDepth features. Please refer to subsection III-C symbol definitions.

Overview of our approach. We propose a novel feature learning approach, termed AsyncDepth, to extract additional geometric information from past LiDAR scans that complements the image features $\{f(\boldsymbol{I}_p^i)\}$ for 3D monocular object detection (Figure 2). We start by constructing asynchronous depth maps from the historical LiDAR point clouds using the localization and camera parameters. We then featurize these depth maps using a featurizer g and aggregate them across traversals. The aggregated features are appended as additional channels to the image features for the detector head h. The whole pipeline is fully differentiable and can be learned end-to-end alongside almost all state-of-the-art monocular 3D object detectors.

A. Past LiDAR Traversals

Past LiDAR traversals from other vehicles. We assume offline data-sharing among vehicles equipped with *different sensor modalities*. These vehicles drive about the same areas, collect *unlabeled* sensor data, and share it with other vehicles when they are not operating. Specifically in this work, we focus on one particular setting: vehicles with camera-only setups have access to past traversal data from vehicles with LiDAR sensors (this can be realized via community sharing or as a service provided by a vehicle manufacturer). These past LiDAR point clouds, though not capturing information about the instantaneous *dynamic* objects in the current drive, can provide abundant 3D information about the *static* environment. It has been shown by [38] that LiDAR-based detection models can be enhanced by these past traversals.

Different from [38], we assume a camera-only setup for the operating vehicle. We hypothesize that the environment information within these past LiDAR traversals can greatly help monocular models detect objects. We validate this hypothesis with a generally applicable and simple framework for monocular 3D detectors.

LiDAR Densification. We follow [38] and maintain a maximum of $N_{\max} \geq 1$ LiDAR traversals for the driving locations (bottom half of Figure 2). Each traversal n is a sequence of point clouds $\{\boldsymbol{Q}_r^n \in \mathbb{R}^{k\times 3}\}$, where r index the frame and k is the number of points, obtained from the past traversals of other vehicles. We transform each point cloud into the same global coordinate system via the associated 6-DoF localization. We combine point clouds along the road to obtain a densified point cloud $\boldsymbol{S}_p^n = \bigcup_{r\in R} \boldsymbol{Q}_r^n$, where R is a subset of frames sampled every s meters along the road near location p. Of course, these point clouds contain both dynamic objects in the past and static background. But as pointed out by [6], [36], with multiple traversals of the same scene, simple statistics (*i.e.* ephemerality/persistency point score) can already help to disambiguate dynamic/static components. Rather than constructing hand-crafted statistics from these point clouds, we propose to learn a feature extractor that can extract relevant information from them.

B. Asynchronous Depth Feature from Past LiDAR

As discussed previously, inferring depth information from images is an ill-posed problem. However, the densified point clouds $\{\boldsymbol{S}_p^n\}_{n=1}^{N_{\max}}$ at location p can provide strong cues for estimating the object position. We project this point cloud into each camera's image to yield a corresponding depth map. Concretely, we use the current 6-DoF localization G_p and the i-th camera's parameters (P^i, T^i) to perform this projection. For every 3D point in the densified point cloud $\boldsymbol{S}_p^n$ (represented in homogeneous coordinates as $\boldsymbol{q}_j \equiv [x_j, y_j, z_j, 1]^\top$), we project it to the local camera coordinate,

$$\hat{\boldsymbol{q}}_j = [\hat{x}_j, \hat{y}_j, \hat{z}_j, 1]^\top \equiv T^i G_p^{-1} \boldsymbol{q}_j.$$

We then project each of these points $\hat{\boldsymbol{q}}_j$ onto the image plane of the i-th camera by perspective projection:

$$[\hat{u}_j^i \hat{z}_j, \hat{v}_j^i \hat{z}_j, \hat{z}_j]^\top \equiv P^i \hat{\boldsymbol{q}}_j$$
$$u_j^i, v_j^i = \lfloor \hat{u}_j^i \rfloor, \lfloor \hat{v}_j^i \rfloor$$

where (u_j^i, v_j^i) are the corresponding pixel indices on the i-th image. Projecting all points in $\boldsymbol{S}_p^n$ into the image plane of

camera i and filling the corresponding depth value renders a depth map $D_p^{n,i} \in \mathbb{R}^{H \times W}$ (see the right image of Figure 1)

$$D_p^{n,i}[u,v] = \max\left\{\max_{(u_j^i, v_j^i)=(u,v)} \hat{z}_j, -1\right\},$$

where we take the maximum depth when multiple points are projected into a same pixel and fill the empty pixel with -1. This implicitly favors background depth since foreground depth is usually closer. As shown in Figure 1, such a depth map provides very rich depth prior for the image. Intuitively, once the model detects an object in 2D, figuring out its depth is a much easier task with the surrounding *background* depth.

C. Feature Learning and Detection

The previous depth maps, $\{D_p^{n,i}\}$, can be noisy since they were captured under different conditions and contain different sets of prior foreground objects. To extract relevant information, we feed them through a 2D backbone g, which is designed to yield depth feature maps $g(D_p^{n,i}) \in \mathbb{R}^{H' \times W' \times d_{\text{depth}}}$ with the same size as that of the image features $f(\boldsymbol{I}_p^i)$. To aggregate the depth features from different traversals, we apply an order invariant pooling function to pool the feature maps along the traversal dimension:

$$F^i_{p\,\text{depth}}[u,v] = \text{pool}(g(D_p^{n,i})[u,v], n = 1, \ldots, N_{\max}).$$

We use mean pooling in our implementation by default. The pooled feature maps from the past LiDAR traversals are concatenated with the corresponding image features along the feature dimension as the new 2D features $F_p^i = \text{concat}(F^i_{p\,\text{depth}}, f(\boldsymbol{I}_p^i))$. We then apply the same detector head h (with slight change to input feature size) to obtain the bounding box predictions $\mathcal{B}_p = h(\{F_p^i, P^i, T^i\})$. As a result, the information from past LiDAR traversals can be incorporated into the existing camera-only object detection models with minimal changes in model architecture.

Training and Inference. We train the whole model, including (i) the depth backbone g that takes LiDAR scans of *past* traversals as input, (ii) the image featurizer f that takes images at the *current* time as input, and (iii) the detector head h, end-to-end with loss signals from annotated 3D labels of the *current* scene. We keep the loss designs of baseline camera-based detection models intact.

During inference, we assume that the model has access to the past LiDARs traversals and generates the depth map online. The depth map backbone can run in parallel with the image featurizer to reduce latency.

IV. Experiments

Datasets. We validate our approach on two large-scale datasets: Lyft L5 Perception Dataset [9] and Ithaca365 [10] Dataset. To the best of our knowledge, these are the only two publicly available autonomous driving datasets that have both bounding box annotations and multiple traversals with accurate localization (Note that nuScenes [40] contains some scenes with multiple traversals but the localization in z-axis is not accurate [41]). The Lyft dataset is collected in Palo Alto (California) and the Ithaca365 dataset is collected in Ithaca (New York). Both datasets provide camera images (6 ring-camera images in Lyft and 2 frontal-view images in Ithaca365) and 3D LiDAR scans (40-beam in Lyft and 128-beam in Ithaca365). We thus perform 360-degree detection on Lyft and frontal-view only detection on Ithaca365. The detection range is set to maximum 50m to the ego vehicle, following the setup of most camera-only detection models developed on nuScenes dataset [40]. For the Lyft dataset, to ensure fair assessment of generalizability, we re-split the dataset so that the training set and test set are geographically disjoint; we also discard locations with less than 2 traversals in the training set. This results in a train/test split of 10,499/3,412 examples. For the Ithaca365 dataset, we follow the default split of the dataset, which results in 4,445/1,644 train/test examples covering the same route but with different collection times. Adhering to our setup, synchronized LiDAR point clouds are **not used** during testing.

Localization. With current localization technology, we can achieve high localization accuracy (*e.g.*, 1-2 cm level accuracy with RTK). We assume good localization in asynchronous LiDAR traversals and the camera-only systems and study the effect of the localization error in the supplementary.

Evaluation metric. We adopt similar metrics from the nuScenes dataset [40] to evaluate the detection performance. We evaluate detection performance within 50m of the ego vehicle. The mean average precision (mAP) is the mean of average precisions (AP) of different classes under $\{0.5, 1, 2, 4\}$m thresholds that determine the match between detection and ground truth. Because Lyft and Ithaca365 datasets do not provide the objects' velocities and attributes ground-truths, we only compute 3 types of true positive metrics (TP metrics), including ATE, ASE and AOE for measuring translation, scale and orientation errors. These TP metrics are computed under a match distance threshold of 2m. Additionally, we also compute a distance based breakdown (0-30m, 30-50m) for these metrics. We evaluate 5 foreground objects (car, truck, bus, pedestrian and bicycle) on Lyft and 2 objects (car, pedestrian) on Ithaca365. Similar to NDS (nuSccenes detection score), we calculate the overall detection score (DS) for these two datasets as $\text{DS} = \frac{1}{6}[3 \cdot \text{mAP} + \sum_{\text{mTP} \in \mathbb{TP}}(1 - \min(1, \text{mTP}))]$. To showcase the most significant improvements from AsyncDepth, we mainly present mAP evaluation results on the main paper and include the rest in the supplementary due to space limitations.

Detection models. We experiment with two representative, high-performing monocular 3D object detection models: FCOS3D [13] and Lift-Splat [1], [11], [12], [14]. FCOS3D extends 2D object detection [42] to 3D by detecting objects in perspective views and regressing additional 3D targets for each of the detected objects. The Lift-Splat style model first constructs a Bird's Eye View (BEV) representation of the scene and then applies a detection head. This BEV representation is constructed by predicting the depth distribution for each pixel on the image feature map and "splatting" the corresponding weighted 2D features into BEV space via camera parameters. Thanks to their strong performance and clean

Method	mAP	Car			Truck			Bus			Bicycle			Pedestrian		
		0-30	30-50	0-50	0-30	30-50	0-50	0-30	30-50	0-50	0-30	30-50	0-50	0-30	30-50	0-50
FCOS3D [13]	14.6	47.4	23.6	37.9	4.5	3.1	4.2	5.1	3.9	5.1	20.3	1.3	10.1	25.7	3.6	15.7
+ AsyncDepth	16.0	48.3	24.5	38.8	7.2	4.1	6.0	9.2	7.7	8.9	24.4	1.5	10.8	26.3	1.7	15.8
Δ AP	+1.4	+0.9	+0.9	+0.9	+2.7	+1.0	+1.8	+4.1	+3.8	+3.8	+4.1	+0.2	+0.7	+0.6	-1.9	+0.1
Lift-Splat [11]	23.3	65.2	25.6	50.7	11.9	5.6	9.9	18.7	13.3	15.7	31.3	0.3	13.7	35.7	4.2	21.3
+ AsyncDepth	25.4	66.7	27.0	52.2	14.5	6.9	11.0	22.4	22.8	24.0	34.3	0.4	15.9	35.9	8.4	23.8
Δ AP	+2.1	+1.5	+1.4	+1.5	+2.6	+1.4	+1.1	+3.7	+9.5	+8.3	+3.1	+0.1	+2.3	+0.3	+4.2	+2.5

TABLE I: **Mean Average Precision (mAP) of two types of detectors across different ranges and object class types on the Lyft dataset.** We evaluate two types of monocular 3D object detection models (FCOS3D [13] and Lift-Splat [1], [11] We show the mAP metric and its breakdown across different ranges (in meters) and class objects. "Δ AP" indicates the gain. We observe AsyncDepth improves the reference detectors in all but one case. Other corresponding metrics (ATE, ASE, AOE and DS) are included in the supplementary material where we observe a similar trend.

Method	mAP	Car			Pedestrian		
		0-30	30-50	0-50	0-30	30-50	0-50
FCOS3D [13]	25.0	46.3	23.8	36.2	18.3	7.8	13.8
+ AsyncDepth	29.2	51.7	29.6	42.2	20.0	10.3	16.2
Δ AP	+4.2	+5.4	+5.8	+6.0	+1.7	+2.5	+2.4
Lift-Splat [11]	39.4	66.6	30.2	52.8	37.1	13.7	26.0
+ AsyncDepth	42.9	70.2	38.2	58.3	37.6	16.6	27.5
Δ AP	+3.5	+3.6	+8.0	+5.5	+0.5	+2.9	+1.5

TABLE II: **Mean Average Precision (mAP) of two types of detectors across different ranges and object types on the Ithaca365 dataset.** Please refer to Table I for naming. AsyncDepth improves reference models in all cases.

design, these two types of monocular 3D object detection models have been well-received by the community [26]–[31].

Implementation details. We strive for a clean and simple implementation to show the generalizability of the proposed approach: we adopt an official implementation [43] of these two models with minimal changes only for supporting the Lyft and Ithaca365 datasets, and use the *exact same* hyper-parameters on both datasets without bells and whistles. For the Lift-Splat / BEVDet model, we adopt the efficient camera-to-BEV transformation implementation in [1] and a detection head similar to [44]. We use a Swin-T [45] pre-trained on nuImages [40] as the image backbone, and supervise depth prediction by the smooth-L1 loss against ground-truth depth during training. For the FCOS3D model, we follow the original paper and use the official training schedule. We use a pretrained ResNet101 [46], [47] with deformable convolutions [48] for image feature extraction. We use ResNet18 [47] to extract features from the depth maps for the different 3D detectors. We deploy a feature pyramid network [49] to extract multi-scale features if the 3D detector of interest is also using multi-scale features. For both datasets, we use a maximum 5 other traversals at the reference location to obtain the depth maps. For the Lyft dataset, for each past traversal we use point clouds closest to $\{0, -20, 20\}$m to the ego vehicle along the road since we are performing 360-degree detection; for the Ithaca365 dataset, we use $\{0, 10, 20\}$m for frontal-view detection.

A. Monocular 3D Detections with AsyncDepth

We show the performance of various detectors with and without AsyncDepth on Lyft and Ithaca365 in Table I and Table II respectively. Overall, we observe that using LiDAR scans from past traversals can significantly improve the performance of monocular 3D object detectors. On Lyft, we observe an improvement of 1.8 mAP averaged across different detectors and different classes at various evaluation thresholds. On Ithaca365, we observe an even more pronounced improvement over the baselines (the AsyncDepth variants outperform the baseline by an average of 3.9 mAP).

To understand where the performance gains are from, we look at the performance of the detectors on various classes. On Lyft, we observe that the biggest improvements come from the detection of bus and bicycles (with an improvement of 6 points and 1.9 respectively); on Ithaca365, the performance gain of AsyncDepth largely reflects on car detection, with a remarkable 5.8 improvement in performance.

In addition, we also look at the performance of the detectors at various ranges. The performance improvement from using AsyncDepth is most pronounced in the challenging far-range object detection (30-50m). On Lyft, we even observe a stark improvement of 9.5 mAP in far-range bus detection over the Lift-Splat baseline; on Ithaca365, we observe an average improvement 6.9 over the baselines across the two detectors on car detection. This suggests that all the detectors benefit from the depth information encoded in the features, particularly in the far ranges where it is especially difficult to infer depth from images.

B. Ablation study

All ablations are on the Lift-Splat detector on the Ithaca365.

Effect of using synchronous depth maps. Our approach involves learning complementary features from the asynchronous depth maps constructed from historical traversals. Discerning readers might question how our method would perform if we instead used synchronous depth maps for learning the feature extractors. In Table III, we observe that learning features from synchronous depth map (+

Lift-Splat Variants	mAP	Car			Pedestrian		
		0-30	30-50	0-50	0-30	30-50	0-50
baseline	39.4	66.6	30.2	52.8	37.1	13.7	26.0
+ AsyncDepth	42.9	70.2	38.2	58.3	37.6	16.6	27.5
+ SyncDepth	51.1	72.3	44.9	62.3	48.0	29.9	39.9

TABLE III: **Mean average precision for Lift-Splat model with asynchronous/synchronous depth map on Ithaca365 dataset.** "+ AsyncDepth" stands for the proposed method using depth maps from asynchronous LiDAR. "+ SyncDepth" stands for an *oracle* scenario where we replace asynchronous depth maps with synchronized depth.

Depth Featurizer	mAP	Car			Pedestrian		
		0-30	30-50	0-50	0-30	30-50	0-50
N/A	39.4	66.6	30.2	52.8	37.1	13.7	26.0
Down. + Avg.	39.8	66.7	31.5	53.3	**37.7**	14.5	26.5
Random Init.	41.6	69.1	37.0	57.0	35.1	16.0	26.2
ImageNet Init.	**42.9**	**70.2**	**38.2**	**58.3**	37.6	**16.6**	**27.5**

TABLE IV: **Mean average precision of our methods with different depth featurizers.** "Down. + Avg." stands for directly using the downsampled projected asynchronous depth maps and averaging them across traversals; "Random Init." and "ImageNet Init." initialize the same featurizer randomly and from ImageNet pre-trained weights, respectively.

SyncDepth) indeed outperforms the baseline by a significant margin, thus validating the claim that accurate depth is crucial to 3D object detection. Though leveraging offline depth maps (+AsyncDepth) is worse than using synchronous depth maps, synchronous depth maps requires real-time LiDAR sensing which is expensive to obtain. Offline depth maps are a cheaper alternative that can significantly boost the performance of detectors, greatly improving the sensing ability of camera-only autonomous vehicles.

Effects of feature extractors. One key aspect of our approach is to deploy an image featurizer to featurize the asynchronous depth-maps before aggregating various information from different traversals. To validate such a design choice, we consider a non-learning baseline (Down. + Avg.) in which we first downsample each offline depth map to appropriate sizes using bilinear interpolation and average them to form a single channel feature that can be appended to the extracted image feature maps. We present the result in Table IV. Naively averaging the asynchronous depth-maps does bring forth some improvements over the baseline but it is far from using a learnable feature extractor. In addition, we also investigated the difference between using a pre-trained ImageNet ResNet18 and a randomly initialized ResNet18. Although the backbone has been pretrained on ImageNet, consisting of natural images, we observed improvements brought by this initialization, especially on car detections. This validates previous results in the literature [50]–[54].

Different number of historical traversals. Throughout the text, we assume the max number of past traversals for each scene $N_{\max} \leq 5$. However, due to privacy concerns or hardware failures, we might have access to less than 5 during inference. To investigate the robustness of AsyncDepth, we conduct inference with various number of upper bound for $N_{\max}$ in Table V. With just 2 traversals for each scene, AsyncDepth can outperform the baseline ($N_{\max} = 0$) by a large margin (3.3 AP for car and 1.4 AP for pedestrian).

# Traversals	mAP	Car			Pedestrian		
		0-30	30-50	0-50	0-30	30-50	0-50
$N_{\max} = 0$	39.4	66.6	30.2	52.8	37.1	13.7	26.0
$N_{\max} = 1$	40.2	66.4	34.8	54.2	35.5	16.2	26.3
$N_{\max} \leq 2$	41.8	68.2	36.0	56.1	36.6	17.0	27.4
$N_{\max} \leq 5$	42.9	70.2	38.2	58.3	37.6	16.6	27.5

TABLE V: **Mean Average Precision of using AsyncDepth with various number of past traversals during inference.** $N_{\max} = 0$ corresponds to vanilla Lift-Splat baseline model without using past LiDAR traversals. $N_{\max} \leq m$ stands for only using $\leq m$ past traversals during testing.

Data storage and latency. We provide an analysis of the additional computational and storage overhead introduced by our method. On average, the AsyncDepth part yields an extremely low *9.66 ms* latency (whole model: 70.78 ms, image featurizer: 23.94 ms). This is due to the relatively small network (ResNet-18) in AsyncDepth. The latency can be further decreased since the depth featurizer can run in parallel with the image featurizer. For data storage and transmissions, the LiDAR points for 5 past traversals of a single scene take about 17.16 MB. For context, the average American commute about 15 miles to work on average [55]. For typical usage, our method needs 13.49 GB to store 5 past traversals. The cost to store this amount of data is low — with current technology, it costs about $0.01/GB for hard drives — and it can be further reduced with compression.

Supplementary. Please refer to the supplementary for more ablation studies and qualitative visualization.

V. Conclusion

We explore using asynchronous LiDAR scans from past traversals to improve monocular 3D detectors for autonomous vehicles. Though not containing information about the location/shape of the target objects in the current scene, we show that these LiDAR scans still contain vital information that can aid 3D object detection. Specifically, we extract offline depth maps from the past traversals and use these depth maps to learn features that aid monocular 3D object detectors. Our approach is simple, lightweight, and compatible with practically all state-of-the-art monocular detectors. We show consistent enhancement of multiple detectors on multiple datasets, opening up new possibilities in improving monocular 3D detection using past traversals.

Acknowledgment

This research is supported by grants from the US NSF (IIS-1724282, TRIPODS-1740822, IIS-2107077, OAC-2118240, OAC-2112606 and IIS-2107161), the ONR DOD (N00014-17-1-2175) and the Cornell Center for Materials Research with funding from the NSF MRSEC program (DMR-1719875). Katie Luo was supported in part by an NVIDIA Graduate Fellowship.

REFERENCES

[1] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *ICRA*, 2023. 1, 2, 4, 5

[2] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "End-to-end pseudo-lidar for image-based 3d object detection," in *CVPR*, June 2020. 1, 2

[3] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2022. [Online]. Available: http://dx.doi.org/10.1109/CVPR52688.2022.00116 1, 2

[4] C. Sautier, G. Puy, S. Gidaris, A. Boulch, A. Bursuc, and R. Marlet, "Image-to-lidar self-supervised distillation for autonomous driving data," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2022. [Online]. Available: http://dx.doi.org/10.1109/CVPR52688.2022.00966 1, 2

[5] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," in *ICLR*, Apr. 2020. 1, 2

[6] D. Barnes, W. Maddern, G. Pascoe, and I. Posner, "Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments," in *ICRA*. IEEE, 2018, pp. 1894–1900. 1, 2, 3

[7] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, and H. Li, "Depth completion from sparse lidar data with depth-normal constraints," in *ICCV*, October 2019. 2

[8] Y. Zhang and T. Funkhouser, "Deep depth completion of a single rgb-d image," in *CVPR*, June 2018. 2

[9] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet, "Level 5 perception dataset 2020," https://level-5.global/level5/data/, 2019. 2, 4

[10] C. A. Diaz-Ruiz, Y. Xia, Y. You, J. Nino, J. Chen, J. Monica, X. Chen, K. Luo, Y. Wang, M. Emond, W.-L. Chao, B. Hariharan, K. Q. Weinberger, and M. Campbell, "Ithaca365: Dataset and driving perception under repeated and challenging weather conditions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 21 383–21 392. 2, 4

[11] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," *Lecture Notes in Computer Science*, p. 194–210, 2020. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-58568-6_12 2, 4, 5

[12] J. Huang, G. Huang, Z. Zhu, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," *arXiv preprint arXiv:2112.11790*, 2021. 2, 4

[13] T. Wang, X. Zhu, J. Pang, and D. Lin, "Fcos3d: Fully convolutional one-stage monocular 3d object detection," *ICCV Workshop*, Oct 2021. [Online]. Available: http://dx.doi.org/10.1109/ICCVW54120.2021.00107 2, 4, 5

[14] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander, "Categorical depth distribution network for monocular 3d object detection," in *CVPR*, 2021, pp. 8555–8564. 2, 4

[15] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *NeurIPS*, 2017. 2

[16] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *CVPR*, 2018. 2

[17] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *CVPR*, 2019. 2

[18] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *ICCV*, 2019, pp. 12 697–12 705. 2

[19] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *CVPR*, 2020, pp. 11 040–11 048. 2

[20] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00864 2

[21] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *CVPR*, June 2019. 2

[22] Y. Wang, B. Yang, R. Hu, M. Liang, and R. Urtasun, "Plumenet: Efficient 3d object detection from stereo images," *IROS*, Sep 2021. [Online]. Available: http://dx.doi.org/10.1109/IROS51168.2021.9635875 2

[23] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *CVPR*, 2016, pp. 2147–2156. 2

[24] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," *ICCV*, Oct 2019. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2019.00938 2

[25] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *ECCV*, 2022. 2

[26] T. Wang, Z. Xinge, J. Pang, and D. Lin, "Probabilistic and geometric depth: Detecting objects in perspective," in *CoRL*. PMLR, 2022, pp. 1475–1485. 2, 5

[27] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon, "Is pseudo-lidar needed for monocular 3d object detection?" in *ICCV*, October 2021, pp. 3142–3152. 2, 5

[28] Y. Zhang, Z. Zhu, W. Zheng, J. Huang, G. Huang, J. Zhou, and J. Lu, "Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving," *arXiv preprint arXiv:2205.09743*, 2022. 2, 5

[29] E. Xie, Z. Yu, D. Zhou, J. Philion, A. Anandkumar, S. Fidler, P. Luo, and J. M. Alvarez, "Mˆ 2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation," *arXiv preprint arXiv:2204.05088*, 2022. 2, 5

[30] Y. Liu, T. Wang, X. Zhang, and J. Sun, "Petr: Position embedding transformation for multi-view 3d object detection," in *ECCV*, 2022. 2, 5

[31] Y. Liu, J. Yan, F. Jia, S. Li, Q. Gao, T. Wang, X. Zhang, and J. Sun, "Petrv2: A unified framework for 3d perception from multi-camera images," in *ICCV*, 2023. 2, 5

[32] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," in *CoRL*. PMLR, 2022, pp. 180–191. 2

[33] Z. Chen, Z. Li, S. Zhang, L. Fang, Q. Jiang, and F. Zhao, "Graph-detr3d: Rethinking overlapping regions for multi-view 3d object detection," in *ACM MM*, 2022. 2

[34] X. Han, H. Wang, J. Lu, and C. Zhao, "Road detection based on the fusion of lidar and image data," *International Journal of Advanced Robotic Systems*, vol. 14, no. 6, p. 1729881417738102, 2017. 2

[35] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4603–4611, 2020. 2

[36] Y. You, K. Z. Luo, C. P. Phoo, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Learning to detect mobile objects from lidar scans without labels," in *CVPR*, June 2022. 2, 3

[37] Y. You, C. P. Phoo, K. Z. Luo, T. Zhang, W.-L. Chao, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Unsupervised adaptation from repeated traversals for autonomous driving," in *NeurIPS*, Dec. 2022. 2

[38] Y. You, K. Z. Luo, X. Chen, J. Chen, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Hindsight is 20/20: Leveraging past traversals to aid 3d perception," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=qsZoGvFiJn1 2, 3

[39] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[40] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020, pp. 11 621–11 631. 4, 5

[41] https://www.nuscenes.org/nuscenes#data-format. 4

[42] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *ICCV*, 2019, pp. 9627–9636. 4

[43] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," https://github.com/open-mmlab/mmdetection3d, 2020. 5

[44] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *CVPR*, 2021, pp. 11 784–11 793. 5
[45] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021, pp. 10 012–10 022. 5
[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255. 5
[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778. 5
[48] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *ICCV*, 2017, pp. 764–773. 5
[49] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017, pp. 2117–2125. 5
[50] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby, "Big transfer (bit): General visual representation learning," in *ECCV*. Springer, 2020, pp. 491–507. 6
[51] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, *et al.*, "A large-scale study of representation learning with the visual task adaptation benchmark," *arXiv preprint arXiv:1910.04867*, 2019. 6
[52] Y. Guo, N. C. Codella, L. Karlinsky, J. V. Codella, J. R. Smith, K. Saenko, T. Rosing, and R. Feris, "A broader study of cross-domain few-shot learning," in *ECCV*. Springer, 2020, pp. 124–141. 6
[53] C. P. Phoo and B. Hariharan, "Self-training for few-shot transfer across extreme task differences," in *ICLR*, 2021. 6
[54] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *ICLR*, 2021. 6
[55] https://www.nrc.gov/docs/ML1006/ML100621425.pdf. 6