Multi-Modal Streaming 3D Object Detection

Mazen Abdelfattah, Kaiwen Yuan, Z. Jane Wang, Fellow, IEEE, and Rabab Ward, Life Fellow, IEEE

Abstract—Modern autonomous vehicles rely heavily on mechanical LiDARs for perception. Current perception methods generally require 360° point clouds, collected sequentially as the LiDAR scans the azimuth and acquires consecutive wedge-shaped slices. The acquisition latency of a full scan (~100 ms) may lead to outdated perception which is detrimental to safe operation. Recent streaming perception works proposed directly processing LiDAR slices and compensating for the narrow field of view (FoV) of a slice by reusing features from preceding slices. These works, however, are all based on a single modality and require past information which may be outdated. Meanwhile, images from high-frequency cameras can support streaming models as they provide a larger FoV compared to a LiDAR slice. However, this difference in FoV complicates sensor fusion. We propose an innovative camera-LiDAR streaming 3D object detection framework that uses camera images instead of past LiDAR slices to provide an up-to-date, dense, and wide context for streaming perception. The proposed method outperforms prior streaming models and powerful full-scan baselines on the challenging NuScenes benchmark in detection accuracy and end-to-end runtime. Our method is shown to be robust to missing camera images, narrow LiDAR slices, and small camera-LiDAR miscalibration.

Index Terms—RGB-D Perception, sensor fusion, streaming 3D object detection.

I. INTRODUCTION

SAFE navigation for autonomous vehicles (AVs) requires minimal latency in perception, especially in dynamic environments. Dominant LiDAR-based 3D object detectors often rely on rotational LiDARs to provide a 360° point cloud of the scene around an AV [1], [2], [3]. A full LiDAR sweep is composed of a stream of slices/packets collected sequentially as the LiDAR scans the azimuth. These detectors must wait for a full LiDAR scan (~100 ms) before inference which presents a bottleneck for perception latency. This delay can lead to outdated detections which is detrimental to safe autonomous driving [4], [5], as shown in Fig. 1.

Instead of waiting for a full scan, [4] proposed directly taking a slice (once it arrives) as input for detection models. While processing LiDAR slices reduces latency, the narrow field of view (FoV) of a slice results in less available context for the model to learn and infer from, thus resulting in lower accuracy. This is especially problematic for objects close to the ego-vehicle (the most safety-critical). A slice may only contain a fragment of an object, as shown in Fig. 1. To tackle this problem, all prior work on streaming 3D detection [4], [5], [6] have utilized memory mechanisms and features from past slices to provide a global context.

The use of past features is not ideal (especially in dynamic scenes) since it expands the current context with outdated information. A vehicle with a speed of 60 km/h moves nearly 1.7 m during a single scan by a 10 Hz LiDAR. An object of interest in previous slices may no longer be in the same position when processing the current slice due to the fast movement of objects, sudden occlusion, or a multitude of different factors inherent in a dynamic environment. These are issues that simple ego-motion compensation cannot fix, since it does not address the motion of other objects and it can be error prone. Errors resulting from using past LiDAR features will aggregate and affect downstream tasks.

This concern has not been addressed previously in streaming 3D object detection [4], [5], [6]. The use of features from past slices relies on two assumptions that may not hold in practice: 1) that slice inference time is shorter than the arrival rate of slices, and 2) that past slices truly represent the current world. If 1) fails, then a sequential bottleneck arises where arriving slices wait for prior slices to be processed. If 2) fails, then we get outdated or misinformed perception. Prior streaming detection works simulate a streaming input by slicing a relatively static LiDAR scan (collected by slow vehicles) into n slices. For example, the popular NuScenes dataset uses vehicles that...
move at an average speed of 16 km/h [1], which is far below the normal driving speed in neighborhoods or highways (up to 50 and 100 km/h respectively). As objects zoom past each other at such high speeds, previous LiDAR slices would capture an outdated state of the world. If both the ego-vehicle and another nearby car move at a speed of 100 km/h in opposite directions, they would move nearly 6 meters apart as the LiDAR scans the azimuth.

Meanwhile, cameras are widely deployed on AVs and usually acquire images at high rate. Images also provide a wider FoV than a LiDAR slice, which makes them ideal in providing a global context for streaming detection. Common fusion methods [3], [7], [8], [9], [10] either rely on point-wise operations or project image features to Birds Eye View (BEV) directly. The former method requires a matching FoV between LiDAR and cameras. In streaming perception, the FoV overlap between a camera and a LiDAR slice is small. The latter project image features to BEV directly without really learning from these features in 3D. Thus, existing fusion methods are not trivially applicable to our problem.

Here we design a streaming 3D object detection framework that aims to 1) utilize camera images to provide a current global context to a LiDAR slice, 2) maximize streaming detection accuracy, 3) minimize end-to-end latency, and 4) demonstrate robustness to calibration issues and sensor drop. Through extensive experiments we show how our method achieves these goals. The features of each modality are extracted independently and in parallel, thus avoiding the matching FoV requirement. Image features are projected to 3D voxels and so they populate the space around a LiDAR slice. After learning separately from image and point cloud features in 3D, we perform pooling to get a unified spatial BEV representation. This representation is then encoded and sent to a detection head for 3D bounding box prediction. Our contributions are as follows:

1) We propose the first framework for multi-modal streaming detection that effectively uses camera images to complement narrow LiDAR slices.
2) The proposed method outperforms prior streaming methods in detection accuracy and end-to-end latency without relying on past LiDAR slices. It also outperforms some single and multi-modal full-scan detectors while taking only a LiDAR slice as input.
3) Our method removes multiple sequential bottlenecks, i.e., waiting on prior slices or other modalities to be encoded, which can be exploited for faster runtime.
4) Experiments show the benefits of the proposed method and its robustness to small camera-LiDAR miscalibration, missing camera images, and very narrow LiDAR slices (see Fig. 1). We also discuss the runtime and computational cost of the proposed framework.

II. RELATED WORK
a) LiDAR-Based 3D Detection: Processing a LiDAR point cloud for 3D object detection usually involves one of the following representations: 1) Voxelizing the input into 3D grids and extracting features using 3D CNNs or sparse convolutions [11], [12]. While accurate, these models are computationally expensive and depend on voxel resolution. 2) Processing the point cloud as a dense range image [13], [14]. This representation, while efficient, suffers from object overlap and depth-varying size. 3) The BEV representation is the most popular as it well-balances accuracy and efficiency [15], [16], [17]. In our work, we employ this representation as we focus on fast and accurate 3D detection.

b) LiDAR-Camera Fusion: Early fusion work relied on point-wise sampling of image features [7], [8]. Recently, simpler and more effective methods proposed decorating each 3D point with image features [3] or segmentation scores [9]. These point-wise fusion methods require matching FoV between the two modalities and thus are not suited for our streaming problem where the camera images should provide a wider FoV than the narrow LiDAR slices. Moreover, these sequential methods must wait for the encoding of image features, and require extremely accurate camera-LiDAR calibration which can degrade with operation [18]. Our method is designed specifically for streaming detection, and thus does not use point-wise fusion or require matching FoVs between modalities. Avoiding sequential fusion speeds up our method and makes it robust to small miscalibration.

Concurrently, full-scan fusion models that concatenate the BEV-encoded point clouds and projected image features started to gain attention. Most of such methods use depth prediction to ‘lift’ [10], [19] image features to 3D and then voxel pooling to get pooled BEV features. However, 1.) lifting depends on the accuracy of depth prediction; and 2.) it results in a non-uniform point cloud of image features that is difficult to process or learn from and thus most works just use pooling to directly get the BEV features. Alternatively, our method of projection by voxel sampling is lighter as it needs no depth prediction, and it results in a structured uniform space that allows for learning in 3D using convolutions, a key innovation in our work. We empirically show that sampling outperforms lifting for streaming perception. Some recent works use attention or complicated fusion modules to achieve high accuracy [20], [21], however, they often require high runtimes and are thus unsuitable for streaming perception.

c) Streaming 3D Object Detection: Streaming detection from LiDAR slices is a relatively new direction with limited prior work. Nonetheless, it was shown to be effective in replacing full 360° scans for more latency-efficient perception. Han et al. [4] used recurrent neural networks to learn from prior slices; thus expanding the available context and restoring the degraded accuracy due to streaming. Also, [5] proposed storing multi-scale feature maps from past slices and concatenating them with current slice features. Finally, [6] represented slices in polar coordinates using “polar pillars” and padded past slice features to current features along the azimuth dimension.

We argue that such past-aware methods may be ill-suited for the very dynamic environments of autonomous driving as discussed in Section I. This is a main motivation behind our work: we only take the most recently observed state of the 3D world from both camera and LiDAR slices for accurate, fast perception. In the experiments, we also highlight runtime advantages by removing the sequential dependency of waiting for the encoding of prior slices.

III. MULTI-MODAL STREAMING 3D DETECTION
As shown in Fig. 2, our streaming detector takes as input a LiDAR point cloud slice and camera images encompassing this slice and outputs 3D detections. Each modality goes through a separate feature extraction pipeline (Section III-A). We present a novel method to project 2D image features to BEV using an intermediate 3D volumetric representation. The
Fig. 2. Overview of the proposed framework. With a point cloud slice and its corresponding images as the input (see cars and pedestrian), two modalities are encoded separately and in parallel, producing two top-down BEV feature maps. These maps are fused and sent to a BEV encoder and a detection head. The example shows the importance of complimentary and spatially-related feature maps: Images provide the context needed to accurately detect the fragmented bottom vehicle with very few points, and the person is detected despite occlusion and shade with the support of the point cloud input.

resulting spatially-aligned multi-modal BEV representations are then fused, encoded, and sent through a detection head (Section III-B). Finally, we propose ways to speed up inference in Section III-C.

A. Multi-Modal Feature Extraction

Features from both input modalities are extracted independently and in parallel. The goal is to have these features in a common spatial (BEV) and feature space for fusion.

1) Point Cloud Encoding: First, the 3D point cloud slice is transformed into a BEV feature map using a PointPillars [17] encoder as it well-balances accuracy and speed. Each point is represented with a vector that includes its 3D location, reflectance, relative timestamp, and its slice’s index \( s \). The scene is divided into a top-down BEV \( V \times V \times V \) grid. Points within each pillar (grid pixel) are encoded using a PointNet [22] giving each pillar a feature vector of size \( c \). This produces a 2D BEV feature map \( P_{bev} \in \mathbb{R}^{V \times V \times c} \).

2) Camera Image Encoding: Let \( X_j \in \mathbb{R}^{H \times W \times 3} \) be the \( j \)-th input image from \( J \) images. A pre-trained ResNet-50 [23] followed by a feature pyramid network (FPN) [24] are used to generate feature maps \( F_j \in \mathbb{R}^{H \times W \times c} \). These are \( J \) semantically-rich image features surrounding a LiDAR slice.

B. Multi-Modal Fusion

1) Projection and 3D Convolutions: To widen the 3D context around a LiDAR slice, 2D image features must be projected to a common spatial and feature space with the point cloud features. Given \( J \) image feature maps \( F_j \), projection is done as follows: The scene is divided into a 3D grid \( V \in \mathbb{R}^{V_x \times V_y \times V_z \times c} \) using the same voxel resolution used to grid the point cloud input in the \( xy \) plane. Thus both modalities are spatially-aligned in BEV. Projection is done using perspective mapping \( \Pi \) from the pinhole camera and the intrinsic \( K \) and extrinsic \( R_t \) matrices of each camera. We fill 3D voxels \((x, y, z) \in V\) with image features \( F_j \) by sampling using (1). If multiple pixels project to the same 3D voxel (intersecting camera FoVs), their features are averaged. Now image features are in 3D space \( V \).

\[
[u, v]^T = \Pi K R_t [x, y, z, 1]^T
\]  

We use 3D convolutions to extract rich 3D information from the projected image features encompassing a LiDAR slice. We then use efficient pooling methods to flatten the z-axis and obtain a dense BEV representation \( I_{bev} \in \mathbb{R}^{V_x \times V_y \times c} \) as shown in Fig. 3. We use \( 1 \times 1 \) convolutions followed by non-linear activation (ReLU) to act as a projection layer from the image feature space to the point cloud feature space. Then a residual module consisting of a single 3D convolution with kernel size 3 is used along with batch norm and ReLU followed by a similar module but with stride 2 in the z-axis halving the number of z-axis voxels and giving them a height of 1 m. Afterwards, average pooling with stride 2 in the z-axis is used to obtain 2 m-high voxels. Two meters usually cover the height of common objects of interest (i.e. cars, or pedestrians). Finally, the max operator is used along the z dimension to reduce it to 1, effectively obtaining a BEV feature map \( I_{bev} \). We use very few 3D convolutions and rely on pooling to mitigate the computational cost.
2) **Fusion:** The resulting image $I_{bev}$ and point cloud $P_{bev}$ BEV representations are spatially-identical in BEV with dimensions $V_x \times V_y$. Fusion is done as a concatenation of the BEV feature maps channel-wise. This contextually-rich combined representation $S_{bev} \in \mathbb{R}^{V_x \times V_y \times c \times 2}$ is then sent through a lightweight CNN-based BEV encoder. This encoder outputs 3 multi-scale feature maps which are then aggregated with an FPN [24] and sent to the detection head.

3) **Detection Head:** We use a CenterPoint head [2], which is an anchor-free method to localize and regress 3D bounding boxes. While the original implementation uses separate heads each focused on classes of a common size, we follow [6] and use a single head for all classes. This significantly speeds up inference as discussed in Section V-C. It has also proven sufficient when coupled with per-class non-maximum suppression (NMS) for post-processing.

C. **Focusing on Populated Regions**

While only the points within a slice are encoded, PointPillars produces a BEV pseudo-image of the whole map with the ego-vehicle at the center. Similarly, when one or two cameras are used (total FoV $\sim 125^\circ$), most of the voxels $V$ are empty after projection. Doing convolution on these empty regions wastes memory and computation. To save these resources, we extract the populated regions of our $B$ BEV maps and stack them into a mini-batch. We divide the map into four quarters, only one of which is non-empty and contains our slice. Using tensor slicing we extract only this populated quarter to form a tensor with shape $(B, C, \frac{V_x}{2}, \frac{V_y}{2})$. The same procedure is used with the volumetric representation to maintain the spatial correspondence between modalities. It also speeds up projection and 3D convolutions since we skip empty regions.

Different slice feature maps that are now stacked along the batch dimension are likely to share some information. This can be exploited using the batch norm operation [25] which has been shown to greatly stabilize and speed up training convergence. Discarding empty regions also means the model needs to attend only to a LiDAR slice. This can make the detection task more tractable and speed up training. This was proven empirically since our models often converge after 1 epoch while comparable models require 12–20 epochs [2].

During inference, discarding empty regions significantly reduces end-to-end latency. Results on speed and FLOPs are reported empirically and discussed further in Section V-C. After fusion and encoding, the BEV feature maps are restored back into the original BEV map size by zero-padding.

IV. EXPERIMENTAL SETUP

A. **Streaming LiDAR Input**

Most LiDAR datasets provide full $360^\circ$ scans. Following prior work [4], [6], we simulate a streaming LiDAR input by dividing a full LiDAR sweep into $n$ non-overlapping slices (with an angle of $\frac{360^\circ}{n}$). While LiDAR slices are captured at different poses [27], public full-scan datasets usually transform all slices to the coordinate frame of the last slice and annotate data accordingly [1]. We follow prior work [4], [6] in not undoing this pose correction for two reasons: 1) Evaluation on the full-scan benchmark’s hidden test sets requires detections in the coordinate frame of the last slice. 2) Our method does not rely on prior slices (posed or not) and this “un-correction” does not greatly affect the underlying geometry of a LiDAR slice except by slight translation. To highlight our method’s robustness to small LiDAR corrections, we experiment with adding small random 3D translation to each point in a slice. As shown in Table IV(c) our model’s detection accuracy is not much affected. In contrast, past-aware streaming detection requires certain ego-transformation of the encoded features from prior frames to the current frame [6].

B. **Dataset and Implementation Details**

We evaluate on the popular large-scale NuScenes [1] dataset as it provides the $360^\circ$ LiDAR scans and camera images needed for our streaming framework. The dataset contains 700 sequences for training, 150 for validation and 150 for testing. Each training slice is annotated with ground truth bounding boxes if one or more corners are within the slice sector. This trains the model on cases where an object is heavily fragmented or even without points in a slice.

The dataset is built using a 32-beam LiDAR and 6 cameras with intersecting FoVs that cover all sides of an AV, each producing an image with a resolution of $1600 \times 900$. Each camera covers an angle of $70^\circ$ except the back camera which covers $110^\circ$. It also provides transformation matrices for LiDAR-to-Camera projection and vice versa. Most annotated pedestrians in the NuScenes dataset are moving with an average speed of $\sim 110$ km/h while 23% of cars move at $\sim 24$ km/h with very few vehicles moving at a maximum of $\sim 70$ km/h.

The ResNet backbone is obtained from a hybrid task cascade model [28] that is pre-trained on 2D detection and segmentation for autonomous driving scenes [1].

The scene’s grid range is set to $[-51.2, 51.2] m$ for the $x$ and $y$ axes, and $[-3, 5] m$ for the $z$ axis. For the point cloud pipeline, the voxel size is set to $0.2 \times 0.2 \times 8$, producing a $512 \times 512$ feature map in BEV. For the 3D image pipeline, we use voxels of size $0.2 \times 0.2 \times 0.5$, producing a $512 \times 512 \times 16$ volumetric representation of image features. After 3D convolutions, we obtain a BEV feature map of shape $512 \times 512$ matching the point cloud BEV feature map.

Of the limited prior work on streaming detection, [4] used a public LiDAR dataset which does not provide $360^\circ$ camera images, [5] used an in-house dataset, and [6] used NuScenes and re-trained prior work [4], [5] on it. Thus, we evaluate and compare with prior work on the NuScenes benchmark. For evaluation, the dataset uses the mean average precision (mAP) metric based on the accuracy of center detection within 4 thresholds: $[0.5, 1, 2, 4] m$. The overall mAP metric is the average over classes and center thresholds.

At inference time, the top 500 detections are gathered and then per-class NMS is performed within a slice. The dataset we use is not made for streaming detection or per-slice evaluation. For a fair comparison with other methods, detections are aggregated from previous slices in a scene as a post-processing step and then per-class NMS is performed to filter out duplicated detections between slices [4].

V. **RESULTS AND DISCUSSION**

The aim of our method is to outperform prior streaming perception methods in 3D detection accuracy and end-to-end latency while demonstrating robustness. In this section we highlight how we achieve these goals. We compare the proposed
model with both streaming and non-streaming LiDAR-only, camera-only, and fusion models (as far as we know, we’re the only streaming fusion work). We conduct ablation studies with different image backbones and projection methods, as well as experiments showing our model’s robustness to camera drop, camera-LiDAR misalignment, and point cloud jitter. Finally, we study the runtime and computation complexity of our method. Qualitative examples are shown in Figs. 4 and 6.

A. Detection Benchmark Results

Evaluated on the NuScenes test set, our 8-slice model achieves the state-of-the-art (SOTA) detection accuracy (in mAP) for streaming models without using any past features (see Table I).

This accuracy surpasses the 4-slice PolarStream [6] model, showing that even with more fragmentation and without access to past information, our model performs better for detection. Focusing on end-to-end runtime, we report FPS from observation to model prediction, considering LiDAR input acquisition time. Our model outperforms popular streaming and full-sweep detectors and baselines in both detection accuracy and end-to-end runtime.

The NDS score is a catch-all metric for detection and other auxiliary tasks such as attribute estimation. While our work is specifically designed for fast (FPS) and accurate (mAP) streaming detection we still report our NDS. The only other streaming work reporting NDS on this test set is a 4-slice PolarStream [6]. Our 8-slice (smaller context) model performs better in detection (mAP), but our NDS is lower due to slightly higher estimation errors in these auxiliary tasks. Our analysis focuses on the metrics relevant to our work (runtime and detection mAP). We outperform prior streaming works and many non-streaming baselines in both.

1) Comparisons With Streaming Models: Without using memory mechanism or outdated features from past slices, our mAP is superior or on par to prior past-aware streaming methods, as shown in Table II. We show that current image features are able to effectively replace past LiDAR features and provide sufficient context for accurate streaming detection. Moreover, our 8-slice (smaller context) model performs better than models that take the whole scene or wider slices (4) as input. We believe 8-slice is generally a good middle ground between too few or too many slices. It is enough to reduce latency and still contains enough contextual information.

To highlight the challenges of streaming perception, we implement a streaming variant of a comparable and powerful LiDAR-only detector (Pillar-based CenterPointP [2]) by limiting its input to one slice during training and evaluation. As shown in Table II, streaming generally causes a severe drop in mAP. Our method was able to restore and surpass all lost accuracy due to streaming. Even at detrimental heavy fragmentation (32 slices) our model increases mAP from 2.2 to 51.1. This 32-slice mAP exceeds that of the full-scan CenterPointP shown

### Table I

| Model               | Input | mAP | Car | Ped | Bicycle | mATE | mASE | NDS | FPS |
|---------------------|-------|-----|-----|-----|---------|------|------|-----|-----|
| PointPillars [17]   | L     | 30.5| 68.4| 59.7| 1.1     | 0.52 | 0.29 | 45.3| 11.8|
| PointPillars [9]    | L+C   | 46.4| 77.9| 73.3| 24.1    | 0.39 | 0.27 | 58.1| 4.2 |
| CenterPoint [2]     | L     | 52.7| 85.0| 74.2| 30.4    | 0.30 | 0.25 | 62.3| -   |
| BEVP Fusion [10]    | L+C   | 60.3| 85.2| 84.6| 30.7    | 0.26 | 0.24 | 67.3| 7.65|
| Ours (8)            | L     | 70.2| 88.6| 89.2| 51.0    | 0.261| 0.239| 72.9| 5.9 |

### Table II

| Model               | Modality | Past Features | 1  | 4  | 8  | 16 | 32 |
|---------------------|-----------|---------------|----|----|----|----|----|
| Han et al. [4]      | L         | ✓             | 52.9| 53.8| 52.7| 50.6|
| Strobe [5]          | L         | ✓             | 46.9| 49.4| 47.9| 45.4| 42.0|
| PolarStream [6]     | L         | ✓             | 53.2| 52.7| 53.9| 52.4|
| CenterPointP [2]    | L         | ✓             | 49.1| 30.4| 27.1| 22.6| 2.2 |
| Ours (8)            | L+C       | ✓             | 57.6| 55.3| 54.7| 53.8| 51.1|

This accuracy surpasses the 4-slice PolarStream [6] model, showing that even with more fragmentation and without access to past information, our model performs better for detection. Focusing on end-to-end runtime, we report FPS from observation to model prediction, considering LiDAR input acquisition time. Our model outperforms popular streaming and full-sweep detectors and baselines in both detection accuracy and end-to-end runtime.

The NDS score is a catch-all metric for detection and other auxiliary tasks such as attribute estimation. While our work is specifically designed for fast (FPS) and accurate (mAP) streaming detection we still report our NDS. The only other streaming work reporting NDS on this test set is a 4-slice PolarStream [6]. Our 8-slice (smaller context) model performs better in detection (mAP), but our NDS is lower due to slightly higher estimation errors in these auxiliary tasks. Our analysis focuses on the metrics relevant to our work (runtime and detection mAP). We outperform prior streaming works and many non-streaming baselines in both.
Fig. 6. Qualitative example of our streaming model’s performance on the NuScenes validation set. Detections from 8 slices are aggregated (each slice is colored) and NMS is used to filter out duplicates. Though the cars in the ‘back’ and ‘back right’ cameras (as well as other objects) are fragmented between slices, they are still accurately detected.

### TABLE III

**Comparison With Non-Streaming Models on NuScenes Val Set**

| Model          | Modality | Stream | mAP | Car | Ped | Bicycle |
|----------------|----------|--------|-----|-----|-----|---------|
| PointPillars [17] | L        | ✗      | 29.5| 70.5| 59.9| 1.60    |
| ImVoIceNet [29] | C        | ✗      | 51.8|      |      |         |
| 3D CVF [28]     | L+C      | ✗      | 42.2| 79.7| 71.3|         |
| MoCa [30]       | L+C      | ✗      | 47.9| 82.4| 79.1| 27.2    |
| CenterPointp [2] | L        | ✗      | 49.1| 83.8| 77.4| 12.7    |
| Ours (all)      | L+C      | ✔️     | 57.6| 83.8| 80.9| 51.6    |
| CenterPointp (8s) | L        | ✔️     | 27.1| 66.1| 59.7| 4.9     |
| Ours (8s)       | L+C      | ✔️     | 54.7| 83.1| 81.8| 47.3    |
| Ours (16s)      | L+C      | ✔️     | 53.8| 82.3| 81.2| 52.1    |
| Ours (32s)      | L+C      | ✔️     | 51.1| 80.6| 79.0| 47.3    |
| Ours (8s)†      | L+C      | ✔️     | 47.8| 80.4| 76.9| 38.0    |

* †Pillar-based. ‡Without the back camera image.

in Table III. Also, our full-scan model was able to increase mAP for CenterPointp by 8.5, showing the benefit of our fusion method even for full-scan models. Finally, maintaining mAP > 50 between all slicing scenarios shows that the image features provide valuable and stable context that can withstand point cloud fragmentation (see Fig. 1).

Our model produces high quality bounding boxes for each slice once it arrives. However, until fresh sensor observations arrive, these detections may themselves go stale. Generally, post-processing depends on downstream tasks, which is outside the scope of this work (e.g. tracking or prediction). One good way to maintain detections through time until fresh observations arrive is to use the ego-vehicle’s velocity as well as the predicted velocity of an object to calculate a shift in its location until the LiDAR scans that object again.

2) **Comparisons With Non-Streaming Models:** We highlighted the safety issue associated with waiting for a full-scan, as well as the large drop in mAP caused by trivially using the streaming input with non-streaming models. The only works that operate under similar constraints to ours are prior streaming works [4], [6]. Nonetheless, we compare with non-streaming models to highlight how our framework compensates for the reduced context of streaming LiDAR. Our 32-slice model consumes only 3% of the LiDAR scan and has an mAP higher than some powerful full-scan baselines such as PointPainting [9] and CenterPointp [2]. Our proposed fusion method provides enough context to replenish an impoverished streaming point cloud input. Most notable, our streaming models are able to improve the accuracy for small object detection, especially for the bicycle class. This is likely because: a) bicycles have a small point cloud footprint with inconsistent shapes compared to other objects like cars; and b) bicycle is a tail class on NuScenes [31]. For both reasons, it benefits greatly from rich image features.

### B. Ablation Studies

1) **Robustness to Missing images/few Points:** We experiment with training without the image from the back camera, which has a wider FoV than other cameras (see Table IV(d)). Our mAP degrades only slightly for the critical car and pedestrian classes as shown in Table III. This shows that our model is somewhat robust to missing camera images. This is also shown qualitatively by visualizing accurate detections behind the ego-vehicle in Fig. 4. Moreover, at heavy fragmentation and with a limited number of points, our 32-slice model maintains high...
accuracy. We also demonstrate robustness to random point jitter in Table IV(c). This shows the benefits of separate encoding for robust streaming perception.

2) Robustness to Calibration Degradation: Point-wise fusion models depend on accurate camera-LiDAR calibration which can be difficult to maintain during operation due to mechanical vibrations or thermal strains [18]. In contrast, our model’s decoupling of point and image encoding and its sparse volumetric representation for image features lead to better robustness to small calibration errors. To study the effect of mis-calibration, we survey SOTA work on online self-calibration (traditional [32] and learnable [18]) and find that they can bring the mean Euler angle error down to around ±1° and mean translation error to ±10 cm from the ground truth calibration depending on the level of noise.

To simulate the error, we follow prior work in self-calibration by uniformly sampling (within a ± range) a translation vector and Euler angles, producing a noisy rotation matrix. These noisy transformations are applied to the ground truth extrinsic matrix $R_t$. We test on 4 thresholds within the expected error and on 2 worst case scenario errors at (3°, 30 cm) and (5°, 50 cm). As shown in Fig. 5, even the (1°, 10 cm) calibration error can greatly change the pixel-point correspondence where small and critical objects like pedestrians are almost completely covered by road features. The work in [3] has shown that even with accurate calibration, false negatives in segmentation masks can be detrimental for models that use point-wise decoration with segmentation scores [9]. Point-wise features projected with (1°, 10 cm) noise would be deeply problematic. In contrast, our model shows robustness by maintaining strong performance at that noise level with only a 2.7 degradation in mAP. However, at very high noises, accuracy degrades indicating that image features do provide valuable context for streaming multi-modal detection. At all noise levels, our mAP remains higher than the full-scan baseline [17].

3) Using Different Projection Methods: Our method for bringing image features to 3D involves sampling using projected voxel coordinates. An alternative proposed in [19] is per-pixel lifting, where a small network predicts the depth for each pixel along specified bins and project accordingly. This is often followed by a voxel pooling operation to get the BEV feature map. We compare these two projection methods in Table IV. We chose sampling over lifting not just because it is more accurate, but also it is parameter-free and avoids the learnable depth prediction task which can be challenging and a bottleneck for accuracy. Moreover, sampling produces the uniform 3D space required for 3D convolutions, which is a core innovation in our work. These convolutions allow our models to learn from image features in 3D and compensate effectively for the narrow point cloud slices.

4) Using Different Image Backbones: Using ResNet101 (2x the size of ResNet50) as an image encoder resulted in modest increases in mAP but greater increases in NDS which are attributed to better velocity and orientation estimation. However, it is much slower than the ResNet50 backbone.

C. On Sequential Dependencies

Our work facilitates faster runtimes by removing multiple sequential dependencies. First, in the proposed framework, point encoding does not depend on image features. Therefore encoding of both modalities can run in parallel. Second, our framework does not require past slice features which allows for processing slices in parallel as they arrive. While other concurrent full-scan fusion methods may enjoy the former benefit, the latter benefit is unique to our proposed streaming framework. Moreover, our method has much faster runtime and learns from image features in 3D compared to concurrent BEV-lifting methods. The closest work to ours is BEVFusion which has a runtime of 8.4 FPS (thus not real-time) compared to our 45 FPS (>5x speedup).

Considering the parallelism inherent in the proposed method and the fast frame rate of cameras, our streaming framework can be efficient in terms of end-to-end runtime. For example, an AV can take advantage of this parallelism by maintaining our proposed 3D volumetric representation of the world and simply updating it as images arrive from cameras. In that case, our framework only has to fetch the most up-to-date features from the image’s BEV representation. Thus the end-to-end runtime (starting from a point cloud slice in the input, and ending with detections in the output) of our 8-slice streaming model is only 15.8 ms with SOTA accuracy. Also, since our framework is fully convolutional, it benefits from advances in neural network quantization and pruning. Finally, this parallel streaming approach allows the computational burden of perception to be distributed along the scan instead of processing the whole scene at once, or in a sequence.

We study runtime and FLOPs of various components of our (8-slice) model on a Tesla V100 GPU and report the results in Table V. A slice (45°) can be covered by a single image based on the FoV of Nuscenes cameras (70°) and contains about 30000 points. These values are used during runtime testing. Moreover, runtime is reported at full and half precision (fp16) since many latency-critical applications use fp16 to speed up inference and reduce storage requirements. Table V shows the significant impact of our cropping operation on 3D convolutions (~75% reduction in runtime and FLOPs) and on our LiDAR-only pipeline.

Our 15.8 ms (8-slice) streaming framework is faster than all prior 8-slice streaming models. After adding 6.25 ms for a 20 Hz LiDAR to collect a 1/8 slice, prior work mostly runs at 37 frames per second (FPS) [6]. On the other hand, our framework runs at 45 Hz. If a slice arrives every 6.25 ms, and processing by prior work takes about 20 ms [6], then the current slice has to wait until the previous one is processed. This is the pitfall of depending on past slices. In our case, multiple slices can be processed simultaneously, which can be exploited by accelerators like FPGAs or GPUs. Our mAP gains are mainly from processing images. Image feature extraction is a process that is done anyway on most AV perception stacks. Image information is often indispensable, and we use a standard ResNet-50 backbone which is pre-trained on typical perception tasks. We study the effect of smaller image resolutions on the overall runtime. While images

### Table V

| Module | GFLOPs | Runtime (ms) | Runtime/fp16 (ms) | mAP |
|--------|--------|--------------|------------------|-----|
| Pts. + BEV Enc. + Multi-Head | 180.2 | 44.5 | 33.8 | 54.7 |
| Pts. + BEV Enc. + Single-Head | 161.6 | 27.4 | 18.2 | 54.7 |
| Above w/ cropping | 40.43 | 15.8 | 14.0 | 54.7 |
| 3D Convolutions | 715.7 | 157.6 | 61.6 | - |
| Above w/ cropping | 178.9 | 38.5 | 15.9 | - |

**Img Encoding (1600x900)**
- Above w/ (1200x678) | 126.5 | 28.9 | 14.8 | 54.7 |
- Above w/ (800x450) | 71.6 | 17.2 | 11.7 | 53.9 |

**Using Different Projection Methods**

- LiDAR-Only Benchmarked on 45 FPS (45°)

**Using ResNet101 (2x)**
- LiDAR+Camera (8-slice) model on a Tesla V100 GPU and report the results in Table V. A slice (45°) can be covered by a single image based on the FoV of Nuscenes cameras (70°) and contains about 30000 points. These values are used during runtime testing. Moreover, runtime is reported at full and half precision (fp16) since many latency-critical applications use fp16 to speed up inference and reduce storage requirements. Table V shows the significant impact of our cropping operation on 3D convolutions (~75% reduction in runtime and FLOPs) and on our LiDAR-only pipeline.
with smaller resolution lose some details, they still maintain high mAP at a much faster runtime, as shown in Table V. Fp16 seems saturated at a certain low resolution. This implies that the overhead of operating at fp16 precision overcomes its benefits at lower resolutions.

Future works can extend on our letter by exploiting the streaming nature of LiDAR for other tasks, e.g. tracking or prediction. Other works can offer further analysis on streaming perception at high speeds. Moreover, there is still space to build novel representations for streaming perception beyond BEV which is more tailored towards full-scan methods.

VI. CONCLUSION

This study proposes a multi-modal framework for accurate and fast 3D detection from streaming LiDAR point cloud data. To reduce latency in detection, our approach processes LiDAR slices as they come and expands their context using camera images. Parallel encoding and independence from past slices enable further reduction in latency, especially when coupled with our proposed cropping operation. The robustness of the proposed framework to small calibration errors, missing camera images, and heavily fragmented point clouds was studied and demonstrated. Our streaming framework is shown to outperform streaming as well as representative full-scan baselines in terms of detection accuracy and runtime on a challenging benchmark.

REFERENCES

[1] H. Caesar et al., “nuScenes: A multimodal dataset for autonomous driving,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11621–11631.
[2] T. Yin, X. Zhou, and P. Krahenbuhl, “Center-based 3D object detection and tracking,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 11784–11793.
[3] C. Wang, C. Ma, M. Zhu, and X. Yang, “PointAugmenting: Cross-modal augmentation for 3D object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 11794–11803.
[4] W. Han et al., “Streaming object detection for 3-D point clouds,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 423–441.
[5] D. Frossard et al., “StrObE: Streaming object detection from LiDAR packets,” in Proc. Conf. Robot Learning., 2021, pp. 1174–1183.
[6] Q. Chen, S. Vora, and O. Beijbom, “PolarStream: Streaming object detection and segmentation with polar pillars,” in Proc. Adv. Neural Inf. Process. Syst., 2021, vol. 34, pp. 26871–26883.
[7] M. Liang et al., “Deep continuous fusion for multi-sensor 3D object detection,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 641–656.
[8] T. Huang et al., “EPNet: Enhancing point features with image semantics for 3D object detection,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 35–52.
[9] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, “PointPainting: Sequential fusion for 3D object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 4604–4612.
[10] Z. Liu et al., “BEVFusion: Multi-task multi-sensor fusion with unified bird’s-eye view representation,” in Proc. IEEE Int. Conf. Robot. Autom., 2023, pp. 2774–2781.
[11] Y. Zhou and O. Tuzel, “VoxelNet: End-to-end learning for point cloud based 3D object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 4490–4499.
[12] Y. Yan, Y. Mao, and B. Li, “SECOND: Sparceley embedded convolutional detection,” Sensors, vol. 18, no. 10, 2018, Art. no. 3337.
[13] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, “LaserNet: An efficient probabilistic 3D object detector for autonomous driving,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 12677–12686.
[14] L. Fan, X. Xiong, F. Wang, N. Wang, and Z. Zhang, “RangeDet: In defense of range view for LiDAR-based 3D object detection,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 2918–2927.
[15] A. Barrera, C. Guindel, J. Beltrán, and F. García, “BirdNet+: End-to-end 3D object detection in LiDAR bird’s eye view,” in Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst., 2020, pp. 1–6.
[16] S. Mohapatra, S. Yomamanhi, H. Gotzigg, S. Milz, and P. Mader, “BEVDetNet: Bird’s eye view LiDAR point cloud based real-time 3D object detection for autonomous driving,” in Proc. IEEE Int. Conf. Intell. Transp. Syst., 2021, pp. 2809–2815.
[17] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “PointPillars: Fast encoders for object detection from point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 12697–12705.
[18] K. Yuan, Z. Guo, and Z. J. Wang, “RGNNet: Tolerance aware LiDAR-camera online calibration with geometric deep learning and generative model,” IEEE Robot. Automat. Lett., vol. 5, no. 4, pp. 6956–6963, Oct. 2020.
[19] J. Philion and S. Fidler, “Lift, splat, shoot: Encoding images from arbitrary camera rig by implicitly unprojecting to 3D,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 194–210.
[20] Z. Chen et al., “AutoAlignV2: Deformable feature aggregation for dynamic multi-modal 3D object detection,” in Proc. Eur. Conf. Comput. Vis., 2022, pp. 628–644.
[21] Y. Li et al., “Voxel field fusion for 3D object detection,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2022, pp. 1120–1129.
[22] C. R. Qi, H. Su, M. Kaichun, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 652–660.
[23] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[24] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2117–2125.
[25] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating deep network training by reducing internal covariate shift,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 448–456.
[26] J. H. Yoo et al., “3D-CVF: Generating joint camera and LiDAR features using cross-view spatial feature fusion for 3D object detection,” in Proc. Eur. Conf. Comput. Vis., Springer, 2020, pp. 720–736.
[27] Y. Li, C. Wen, F. Juefei-Xu, and C. Feng, “Fooling LiDAR perception via adversarial trajectory perturbation,” in Proc. IEEE/CVF Int. Conf. Comput. Vis., 2021, pp. 7898–7907.
[28] K. Chen et al., “Hybrid task cascade for instance segmentation,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2019, pp. 4974–4983.
[29] D. Rukhlovich et al., “ImVoxelNet: Image to voxels projection for monocular and multi-view general-purpose 3D object detection,” in Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis., 2022, pp. 2397–2406.
[30] W. Zhang, Z. Wang, and C. C. Loy, “Improving data augmentation for multi-modality 3D object detection,” in Proc. Int. Conf. Learn. Representations 2023 Workshop Scene Representations Auton. Driving, 2023, [Online]. Available: https://openreview.net/forum?id=8dYU49xcK9
[31] B. Zhu et al., “Class-balanced grouping and sampling for point cloud 3D object detection,” 2019, arXiv:1908.09492.
[32] Z. Taylor and J. Nieto, “Motion-based calibration of multimodal sensor extrinsics and timing offset estimation,” IEEE Trans. Robot., vol. 32, no. 5, pp. 1215–1229, Oct. 2016.