A. Mamba

A.1. Background

The state space (SSM) model [16] continuous system maps a 1D input $x(t) \in \mathbb{R}^L$ to an output signal $y(t) \in \mathbb{R}^L$ via a hidden state $h(t) \in \mathbb{R}^N$. This can be represented as the following set of linear differential equations:

$$\begin{cases} h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \\ y(t) = \mathbf{C}h(t) + \mathbf{D}x(t), \end{cases} \tag{4}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$, $\mathbf{B} \in \mathbb{R}^{N\times N}$, $\mathbf{C} \in \mathbb{R}^{1\times N}$ are the learnable parameters and $\mathbf{D} \in \mathbb{R}^{\mathbf{1}}$ is a residual connection. Mamba [15] discretizes the SSM model parameters $\mathbf{A}$ and $\mathbf{B}$ using the zero-order hold (ZOH) transformation and a timescale parameter Δ, where $\overline{\mathbf{A}} = \exp(\mathbf{\Delta A})$, $\overline{\mathbf{B}} = (\mathbf{\Delta A})^{-1}(\exp(\mathbf{\Delta A}) - \mathbf{I}) \cdot \mathbf{\Delta A}$. The resulting discretized equations are as follows:

$$\begin{cases} h_t = \mathbf{A}h_{t-1} + \mathbf{B}x_t, \\ y_t = \mathbf{C}h_t, \end{cases} \tag{5}$$

where $\mathbf{D}$ is omitted from the equations and incorporated as a simple residual connection in the architecture. Finally, Mamba computes its output through an efficient reformulation as a global convolution, enabling efficient training:

$$\begin{cases} \overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}}, \mathbf{C}\overline{\mathbf{A}}\overline{\mathbf{B}}, ..., \mathbf{C}\overline{\mathbf{A}}^{\mathbf{k}}\overline{\mathbf{B}}), \\ \mathbf{y} = \mathbf{x} * \overline{\mathbf{K}} \end{cases} \tag{6}$$

Mamba combines the time-varying strength of self-attention, near-linear scaling of recurrent neural networks and fast training of convolutions for efficient modeling of sequences. This enables potential benefits in long-horizon (ie. hundreds of partial sectors across multiple full rotations of the LiDAR sensor) driving scenarios, although we defer this exploration to future work.

A.2. Comparison of PFCF against existing Mamba works.

Mamba methods for LiDAR object detection include Voxel Mamba [47] and UniMamba [21]. These methods use costly geometric heuristics such as the Hilbert curve or Z-curve to impose an arbitrary order by which the voxel grid is serialized. These heuristics require recomputation for every resolution size as the curve patterns are only square-shaped. This leads to significant memory usage during inference. In comparison, our Polar Hierarchical Mamba backbone offers a simple serialization method by leveraging the natural temporal ordering of the polar space. We simply serialize according to the azimuth angles, using a bidirectional local SSM to fully capture the intra-sector relationships before employing a forward global SSM to capture the inter-sector dependencies. A visualization of the different serialization approaches is shown in Figure 6.

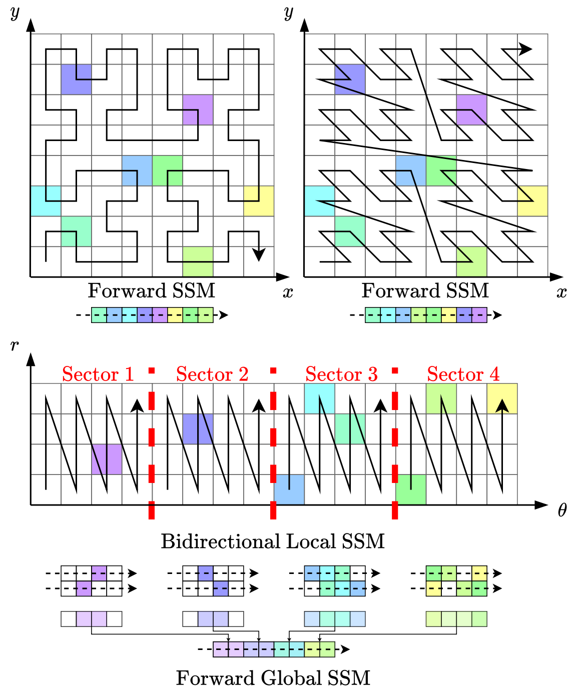

Figure 6. **Comparison with existing LiDAR Mamba works. (Top)** Previous Mamba-based LiDAR models operate on full point cloud scans and use costly geometric heuristics such as Hilbert (left) or Z curves (right) to maintain spatial locality. **(Bottom)** Our method serializes sectors simply by when they arrive, spreading memory usage and computations over the course of a sensor rotation rather than altogether in one peak.

B. Implementation and training details

Waymo Open dataset. The Waymo Open dataset [39] has a detection range of 75 meters and adopts mean average precision (mAP) and mean average precision weighted by heading accuracy (mAPH) as the main evaluation metrics. Waymo differentiates the metrics into L1 and L2 difficulty levels - L1 for objects with more than five LiDAR points and L2 for objects with at least one LiDAR point. We use a voxel size of $(0.32m, 0.32m, 0.1875m)$ and detection range of $[-74.88m, 74.88m]$ along the X and Y axes. We set the detection range to $[-2m, 4m]$ in the Z axis. We use the Adam [22] optimizer with a one-cycle learning rate policy, a weight decay of 0.05, and max learning rate of $2.5e-3$. We use a class-specific non-max-suppression during inference with IoU thresholds of $0.75, 0.6, 0.55$, corresponding to vehicle, pedestrian, and cyclist classes respectively. A faded training strategy is used for the last epoch.

nuScenes dataset. The nuScenes dataset [5] features a detection range of 54 meters along the X and Y axes, with a vertical range of $[-5m, 3m]$, and uses mean average precision and nuScenes detection score (NDS) averaged across all classes as the main metrics. mAP is averaged across the distance thresholds of $0.5m, 1.0m, 2.0m$, and $4.0m$. NDS is an average of mAP along with true positive evaluations of translation, velocity, orientation, scaling and attribute errors. We follow the official 10-class setting, including vehicles, barriers, and traffic participants. Our voxelization uses a uniform voxel size of $(0.25m, 0.25m, 0.25m)$ and detection range of $[-54m, 54m]$ along the X and Y axes. We use the Adam optimizer with a one-cycle learning rate schedule, setting a maximum learning rate of $1.5e-3$, a weight decay of 0.01, and gradient clipping of 10.

Model implementation. We use the open-source OpenPCDet [41] codebase to build our model. For our feature encoder we tweak the dynamic [49] VFE to support cylindrical coordinates, calculating cylindrical voxel sizes to output a voxel grid of shape $(468, 468, 32)$ along the (r, θ, z) axes respectively. We build our own 3D backbone using the Mamba code provided by [15, 47], and use the 2D backbone from [23] and the detection head from [44]. To maintain comparability with previous methods we use the same training schedule of 24 epochs for Waymo Open [39] and 30 epochs for nuScenes [5]. Our training on the full Waymo and nuScenes datasets are conducted on 8 H200 GPUs with total batch size of 24 and 16 respectively, and our ablation studies are conducted on 1 H200 GPU with batch size 3 and 2 respectively. Additionally, pseudocode of the forward pass of the PHiM block is provided in Alg. 1, and actual code implementations for PHiM, the sector feature buffer and our streaming training data simulation has been attached in the code supplement.

Hyperparameters. For our 3D backbone, we use an input model dimension of 128 and tensor shape of $(468, 468)$. We use 6 PHiM blocks with strides $[1, 1, 1, 2, 1, 4]$ and kernel size $[3, 3, 3, 3, 3, 5]$ for the decomposed downsampling layers. We keep hyperparameters for the 2D backbone and detection head the same as [47].

C. Metrics evaluation details

Comparison with previous methods. We use reported accuracy results from original papers in our table and scatterplot. For our latency evaluations, we use specified config files from the OpenPCDet [41] codebase for the Cartesian based methods and from each repo for the polar-based methods. Due to needing data transformations to simulate a streaming setup, all mAP results are on the validation sets. For comparisons with Cartesian based methods, we evaluate our model with length 1 sequence of $\frac{1}{1}$ LiDAR sector - equivalent to one full point cloud.

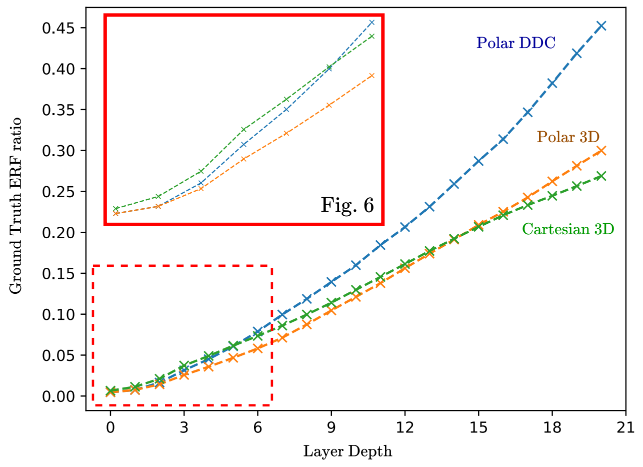

Figure 7. **Quantitative Distortion over 20 layers.** The physical volume within the effective receptive field (ERF) of ground truth objects compared to background. Over very deep upsampling/downsampling pipelines - for example stacked PHiM blocks - polar decomposed convolutions scale poorly in tracking the foreground ERF ratio of Cartesian 3D convolutions. However, with fewer stacked blocks, polar decomposed convolutions track the Cartesian 3D convolutions ERF better.

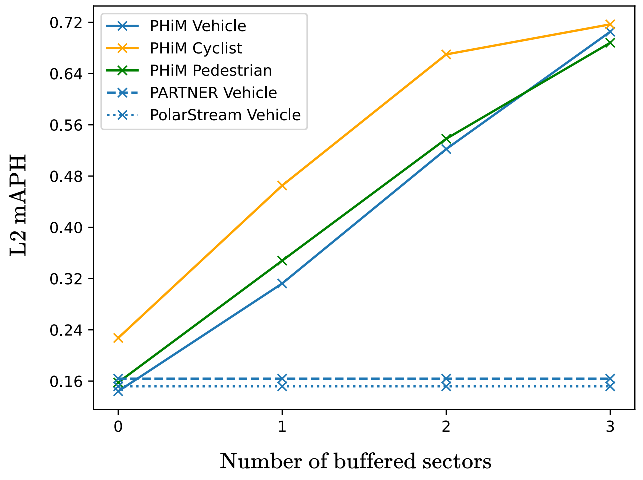

Figure 8. **Global prediction accuracy over number of buffered sectors.** Global scene prediction performance in relation to number of buffered sector features, evaluated with PHiM $\frac{1}{4}$. A buffer size of three therefore corresponds to having full-scan information available (3 buffered sectors and the new input sector). PHiM reformulates streaming prediction to generate and refine global outputs as more sectors are buffered. Unlike other streaming methods that predict only on the current sector without updating past outputs, PHiM progressively improves global accuracy and reaches full-scan performance while retaining streaming efficiency. We report the L2 mAPH for each class individually.

Ablations. For our ablations in Tab. 3 in the main paper, Fig. 8 and Tab. 5 in the Appendix, we use sector size of

Method	Num. Params	Params Mem.	FLOPs	Peak Mem. Usage
Voxel Mamba [47]	20.1 M	76.6 MB	939.0 G	3.0 GB
PHiM (ours)	11.8 M	45.0 MB	926.1 G	1.7 GB

Table 6. **Efficiency comparison with baseline Mamba.** Metrics: Number of parameters in millions (↓), parameter memory usage in MB (↓), floating point operations in GFLOPs (↓), peak memory usage in GB (↓). Another benefit of decomposing distortion heavy 3D convolutions into separate 2D convolutions is a reduction in parameters. Besides, PHiM also removes position encodings and serialization curves stored in memory, resulting in significant parameters and memory savings.

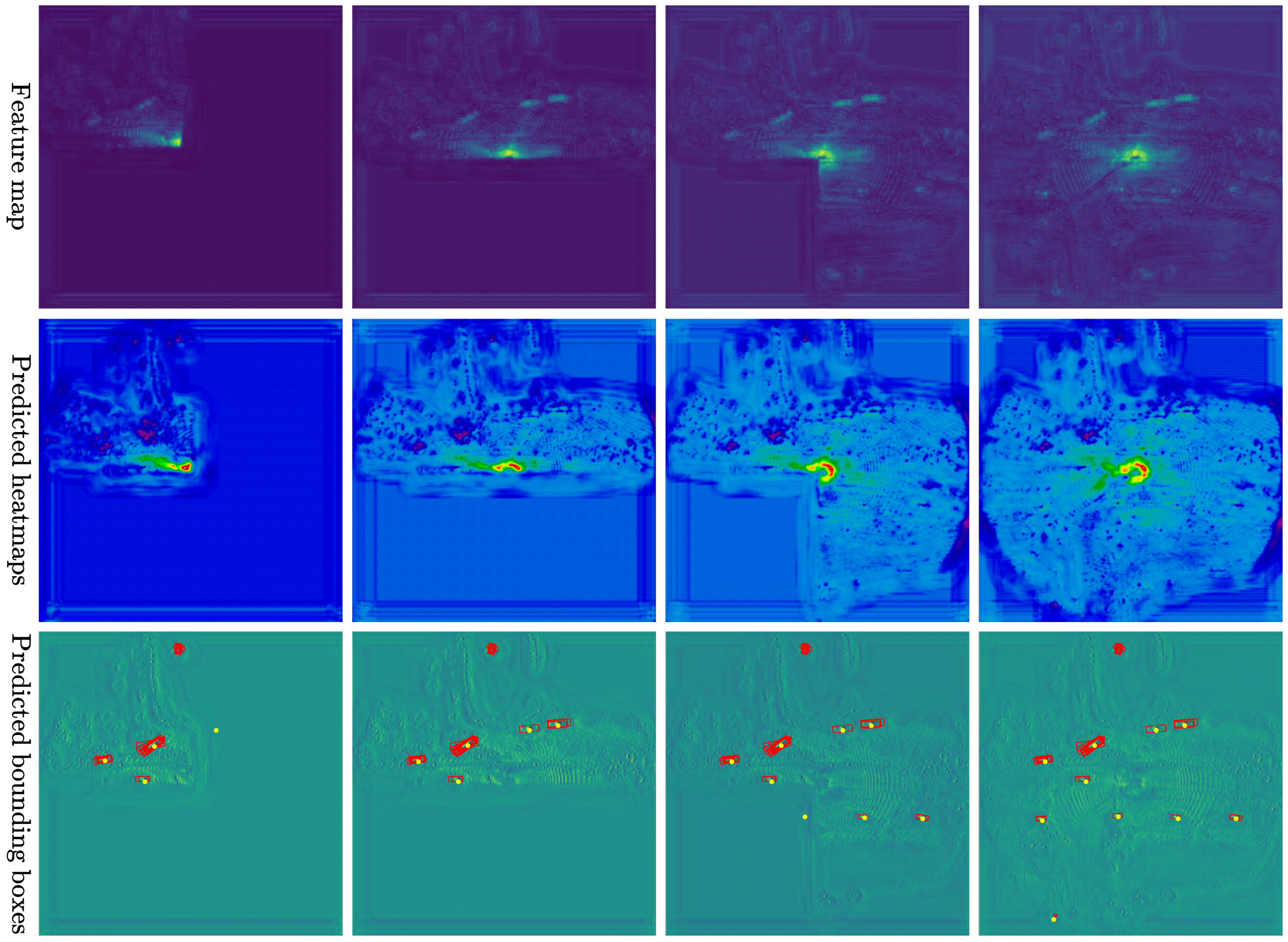

Figure 9. **Prediction refinement qualitative visualization. (Top)** PHiM feature map before detection head. From left to right, the frames correspond to the number of sectors in the sector buffer with PHiM operating on $\frac{1}{4}$ sector size. **(Middle)** Predicted Centerpoint heatmap. **(Bottom)** Predicted bounding boxes are shown in red and the yellow points indicate ground truth object centers. Ground truths on the sector borders are missed initially (first and third frames) but are able to be detected with information from the following sector (second and fourth frames).

$\frac{1}{4}$ for simplicity although any sector size can be chosen for the experiments.

Per-plane distortion. To quantify local geometric distortion between Cartesian and cylindrical coordinate representations of a 3D point cloud (visualized in Fig. 10), we compute a *local area distortion* score based on pairwise distances in 2D projections. Given two sets of corresponding points $\mathbf{P}_{cart}, \mathbf{P}_{cyl} \in \mathbb{R}^{M \times 2}$ (sampled from the full set if necessary), we define distortion as the standard deviation of the ratio of pairwise distances:

Algorithm 1: PHiM Block forward pass.

```
Input: Voxel features $V$, coordinates $C$
    batch size $B$, spatial shape $S$
Initialize local feature tensor $V_{\text{local}}$
foreach frustum $f$ do
    extract $V_f$ from $V$
    $F_{\text{fw}} \leftarrow \text{fsm}(V_f)$
    $F_{\text{bw}} \leftarrow \text{bsm}(V_f^{\text{rev}})$
    $V_{\text{local}}[f] \leftarrow F_{\text{fw}} + F_{\text{bw}}^{\text{rev}}$
end
build SparseConvTensor $X \leftarrow (V_{\text{local}}, C)$
foreach encoder $E$ do
    $X \leftarrow E(X)$
end
initialize global feature tensor $V_{\text{global}}$
merge partial sectors into same batch
foreach batch $b$ do
    extract $F_b$ from $X$
    $V_{\text{global}}[b] \leftarrow \text{fgm}(F_b)$
end
$X.\text{features} \leftarrow \text{normalize}(V_{\text{global}})$
foreach decoder $D$ do
    $X \leftarrow D(X)$
end
$X.\text{features} \leftarrow X.\text{features} + V_{\text{local}}$
return $X.\text{features}$, $X.\text{indices}$
```

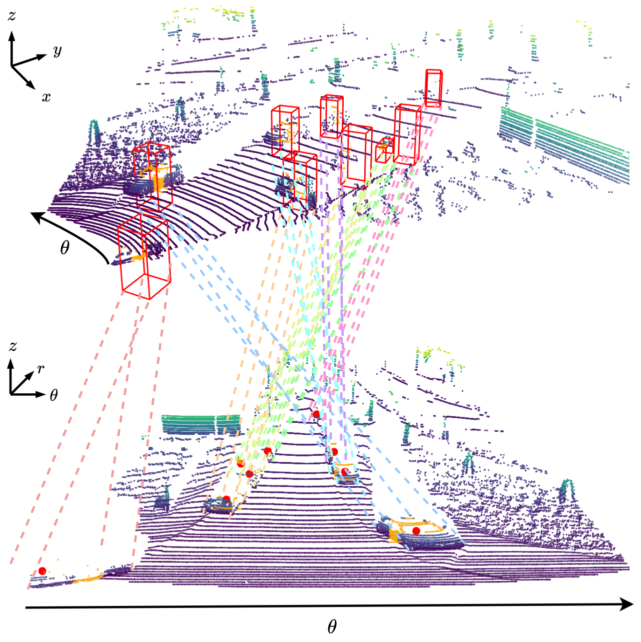

Figure 10. **Egocentric distortion in polar coordinates.** Illustration of object deformation. Objects of similar size are distorted differently relative to the ego vehicle.

$$\text{distortion} = \text{std}_{m,n}\left(\frac{\|\mathbf{p}_m^{(2)} - \mathbf{p}_n^{(2)}\|_2}{\|\mathbf{p}_m^{(1)} - \mathbf{p}_n^{(1)}\|_2}\right), \quad m, n \in \{1, \dots, M\}$$

where $\mathbf{p}_m^{(1)} \in \mathbf{P}_{cart}$, $\mathbf{p}_m^{(2)} \in \mathbf{P}_{cyl}$.

We evaluate this distortion over three 2D subspaces derived from the Cartesian and cylindrical representations of the point cloud:

- In the (r, θ) plane:

$$\mathbf{p}^{(1)} = (x, y), \quad \mathbf{p}^{(2)} = (r, \theta)$$

$$r = \sqrt{x^2 + y^2}, \quad \theta = \arctan 2(y, x)$$

- In the (r, z) plane:

$$\mathbf{p}^{(1)} = (x, z), \quad \mathbf{p}^{(2)} = (r, z)$$

- In the (θ, z) plane:

$$\mathbf{p}^{(1)} = (y, z), \quad \mathbf{p}^{(2)} = (\theta, z)$$

Each distortion score is computed using 1000 randomly sampled points to reduce computational cost. Finally, we normalize the distortions into percentages to understand the relative contribution of each subspace to the overall distortion:

$$\text{percent}_s = \frac{\text{distortion}_s}{\sum_t \text{distortion}_t} \times 100\%, \quad s \in \{r\text{-}\theta, r\text{-}z, \theta\text{-}z\}$$

Physical volume ERF. One heuristic for measuring the compounding distortion with respect to network depth is how much physical volume represents ground truth foreground objects compared to background throughout the pipeline. As shown in Fig. 10 in the supplement, ground truth objects can be either stretched or compressed when in the egocentric polar view. As such, the proportion of foreground to background is also impacted by the polar coordinate system. Moreover, sparse convolutions exacerbate this distortion as stretched foreground regions—objects near the ego vehicle—reside in denser parts of the feature map and thus receive disproportionately high attention during feature aggregation. Conversely, compressed foreground regions—objects farther from the ego—occupy fewer voxels and risk being underrepresented. This imbalance in physical representation leads to a skewed effective receptive field (ERF), where certain spatial regions dominate the learned features, introducing compounding bias as the network deepens.

This spatial imbalance motivates the need for a more uniform treatment of physical space across depths. In contrast to polar representations, Cartesian coordinate systems preserve physical distances uniformly and exhibit a near-linear scaling between foreground ERF and total ERF as the network grows deeper. To mitigate the compounding distortion introduced by non-uniform spatial resolution in polar grids and sparse convolutions, we aim to design a method that more faithfully tracks the Cartesian ERF ratio—ensuring that foreground regions maintain consistent influence in the feature hierarchy while preserving the low-latency benefits of the polar view.

We measure the foreground effective receptive field ratio as the ratio of physical volume overlapping ground truth objects compared to overall physical volume of the sparse feature map. Specifically, let the voxel grid be defined in cylindrical coordinates with shape (N_θ, N_r, N_z), and spatial bounds $[r_{\min}, r_{\max}]$, $[z_{\min}, z_{\max}]$, and $\theta \in [-\pi, \pi]$.

Let the voxel resolutions be defined as:

$$\Delta r = \frac{r_{\max} - r_{\min}}{N_r}, \quad \Delta z = \frac{z_{\max} - z_{\min}}{N_z}, \quad \Delta\theta = \frac{2\pi}{N_\theta}.$$

For a voxel at index $(i, j, k) \in \mathbb{Z}^3$, where $r_j = r_{\min} + j\Delta r$, its physical volume is:

$$V_{ijk} = \tfrac{1}{2}\left[(r_j + \Delta r)^2 - r_j^2\right]\Delta\theta\Delta z.$$

Given a set of activated voxels $\mathcal{A} \subseteq \mathbb{Z}^3$, define:

$$V_{\text{total}}^{(L)} = \sum_{(i,j,k)\in\mathcal{A}} V_{ijk}, \quad V_{\text{fg}}^{(L)} = \sum_{(i,j,k)\in\mathcal{A}\cap\mathcal{G}} V_{ijk},$$

where $\mathcal{G}$ denotes ground-truth foreground voxel indices.

Then the foreground ERF ratio is:

$$R^{(L)} = \frac{V_{\text{fg}}^{(L)}}{V_{\text{total}}^{(L)}}.$$

We evaluate the growth of the foreground ERF with respect to layer depth, where each layer is either a 3D downsampling convolution followed by a 3D upsampling convolution or two 2D decomposed convolutions followed by their corresponding upsampling convolutions. Between each convolution layer, we add batch normalization and ReLU activation, fully consistent with the pipeline within our PHiM block 3.2. Crucially, we find that despite poor scaling with deeper downsampling pipelines, decomposed convolutions closely track the Cartesian foreground ERF ratio within the range of 0 to 6 PHiM blocks. This motivates our choice for 6 stacked PHiM blocks in our final architecture.

D. Additional results

Efficiency metrics comparison with baseline Mamba. To strengthen our efficiency claims from the main paper, we include a comparison of number of floating point operations per forward pass, number of parameters, and memory usage against the representative Voxel Mamba [47] in Tab. 6. By decomposing 3D convolutions into separate 2D convolutions and removing positional encodings, PHiM saves almost 2x number of parameters. In conjunction, removing serialization curves in favor of streaming-native azimuthal ordering results in almost 2x reduction in peak memory usage. We report full-scan results on the Waymo Open dataset.

Prediction refinement ablation. We evaluate PFCF's performance on global scene predictions with varying number of buffered sectors in Fig. 8. Because PFCF makes global predictions for each new input sector, it is able to gradually refine its predictions – even on past spatiotemporal regions – and approach full-scan performance while maintaining the streaming benefit. After the initial stage of ”loading” the sector buffer, PFCF will continue performing at full-scan level. This is in contrast with previous streaming methods [6, 12, 17, 31] which only makes predictions on regions corresponding to the given input. As such, previous streaming methods always demonstrate constant global performance.

Prediction refinement qualitative visualization. Fig. 9 shows a qualitative visualization of the feature maps and output predictions for PFCF. Notably, while some ground truth objects are not detected initially (bottom row in the first and third frames), our sector buffer concatenates features together and captures the missing objects with the subsequent input sector (second and fourth frames).

Measuring efficiency. Our efficiency comparison in Tab. 6 includes measurements on total number of floating point operations per forward pass, number of total parameters, parameter memory and peak memory usage. We use the ptflops [38] tool to measure floating point operations and built-in CUDA memory tracking in Pytorch to measure peak CUDA memory usage. We report averages over 1000 samples.

E. Broader impacts discussion

This work presents Polar-Fast-Cartesian-Full, a memory-efficient and low-latency architecture for LiDAR-based perception in autonomous systems. By reducing geometric distortion in egocentric coordinates and enabling streaming inference, our method may contribute to safer, more responsive autonomous driving technologies. The reduced computational footprint of PFCF also opens the door for deployment on lower-power edge devices, potentially democratizing access to advanced perception systems.

That said, improvements in 3D perception can also amplify capabilities in domains with dual-use concerns, such as surveillance or military applications. Moreover, safety and performance gains in perception alone do not guarantee equitable outcomes; system-level considerations – such as dataset bias and deployment context – remain critical. We encourage developers to evaluate downstream uses of our method with attention to fairness, transparency, and societal impact.