IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Efficient Multi-Camera Tokenization with Triplanes for End-to-End Driving

Boris Ivanovic[1], Cristiano Saltori[1], Yurong You[1], Yan Wang[1], Wenjie Luo[1], and Marco Pavone[1,2]

***Abstract*—Autoregressive Transformers are increasingly being deployed as end-to-end robot and autonomous vehicle (AV) policy architectures, owing to their scalability and potential to leverage internet-scale pretraining for generalization. Accordingly, tokenizing sensor data *efficiently* is paramount to ensuring the real-time feasibility of such architectures on embedded hardware. To this end, we present an efficient triplane-based multi-camera tokenization strategy that leverages recent advances in 3D neural reconstruction and rendering to produce sensor tokens that are agnostic to the number of input cameras and their resolution, while explicitly accounting for their geometry around an AV. Experiments on large-scale AV datasets and a state-of-the-art neural simulator demonstrate that our approach yields significant savings over current image patch-based tokenization strategies, producing up to 72% fewer tokens, resulting in up to 50% faster policy inference while achieving the same open-loop motion planning accuracy and improved offroad rates in closed-loop driving simulations.**

***Index Terms*—Representation Learning, Deep Learning for Visual Perception, Computer Vision for Transportation**

I. Introduction

GENERALIZING to novel environments remains a grand challenge for robots and autonomous vehicles (AVs), requiring advanced reasoning capabilities to robustly handle unseen scenarios. Accordingly, coinciding with the rapid advancement of token-based autoregressive (AR) Transformers [1] such as LLMs and VLMs, there has been a significant growth in research aiming to endow robots and AVs with general reasoning capabilities by deploying internet-pretrained foundation models on-vehicle. However, such architectures oft contain billions of parameters, complicating their real-time deployment in embedded systems [2].

Many approaches are currently being investigated to reduce model size and inference time, including model compression [3] and efficient sensor tokenization strategies. In particular, image tokenizers aim to represent images with as few latent embeddings (i.e., "tokens") as possible. Current methods largely focus on representing single images and employ autoencoding architectures [4], [5], [6] or directly encode patches of pixels [7]. VLMs primarily employ Vision Transformers (ViTs) [7] and adopt the latter, partitioning images into patches that are encoded to form a 1D token sequence. While this approach is practical, it produces token counts that scale linearly with image resolution and the number of cameras [8]. To obtain a 360-degree view of their surroundings, AVs often use 6 to 10 cameras, the patch-based tokenization of which would yield thousands of tokens per timestep, precluding real-time inference.

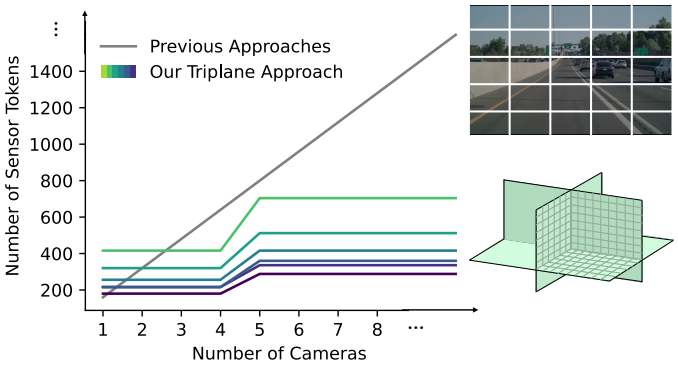


Fig. 1. Many Transformer-based AV policies employ patch-based image tokenization, whose token counts scale linearly with the number of cameras, complicating real-world AV deployment. In contrast, our proposed triplane-based approach produces a *fixed* number of tokens, irrespective of the number of cameras or their resolution, significantly reducing online inference time.

Contributions. To address this, we present a novel multi-camera image tokenization scheme (Fig. 1) based on triplanes, a recent advancement in 3D neural reconstruction and rendering, that produces tokens in a resolution-agnostic, camera-number-agnostic, and geometrically-aware manner (Section III). Experiments on a large-scale dataset consisting of 20,000 hours of driving data from 25 countries demonstrate the efficacy of our approach over patch-based tokenization, producing up to 72% fewer tokens, resulting in up to 50% faster inference while achieving the same open-loop motion planning accuracy and improved offroad rates in closed-loop driving simulations (Section IV).

II. Related Work

End-to-End Driving with Tokens. The first approaches applying token-based Transformer [1] architectures to end-to-end (E2E) driving primarily focused on connecting previously-existing modules (e.g., detection, prediction, planning) along a differentiable pathway, treating their outputs as queries, keys, and values into subsequent modules. Initial works UniAD [9] and VAD [10] encode multi-camera observations into a birds-eye view (BEV) feature grid, where each cell is treated as a token and processed through a cascade of task-specific Transformer decoders (e.g., for object tracking, map segmentation, and motion prediction). PARA-Drive [11] further tokenizes

Manuscript received: June 26, 2025; Accepted: August 30, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments.

[1]All authors are with NVIDIA, Santa Clara, CA 95051, USA bivanovic@nvidia.com

[2]Marco Pavone is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA pavone@stanford.edu

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

BEV feature grid cells into parallel multi-task decoding heads through a shared Transformer backbone. Each of these works are trained from scratch on AV-specific data and all model outputs are supervised with ground-truth (GT) labels.

Inspired by progress in language modeling, a parallel line of work aims to develop generalist embodied agents that directly use an AR Transformer [1] to process input text, actions, and sensor data into downstream task-specific tokens. RT-1 [12], Gato [13], RT-2 [14], RT-X [15], and VIMA [16] broadly introduce, extend, and deploy vision-language-action models (VLAMs) onto multiple robotic tasks and embodiments. Initial works in the AV domain focused on AR video generation [17] and predicting text explanations alongside driving actions [18], [19] from a front-facing camera, tokenizing images with the patch-based embedding scheme from ViT [7].

Multiple cameras have recently been incorporated in VLM-based E2E AV models such as DriveVLM [20], OmniDrive [21], and EMMA [22]. DriveVLM adopts the ViT-based SigLIP [23] to tokenize images, OmniDrive develops a sparse 3D query architecture to encode multiple cameras, and EMMA feeds camera images into Gemini Nano-1 [24]. However, these approaches require a separate token reduction model (e.g., LDPNetv2 [25]) to reduce inference times, auxiliary training signals that are costly to obtain, or produce a large number of tokens (500+) per image. In contrast, our approach is fully self-supervised (it can optionally use auxiliary signals if provided) and produces much fewer tokens (Section IV).

Latent Image Representations. While current approaches mostly employ ViTs [7], another common tokenization strategy is to use the latent space of a pretrained image autoencoder. VAE [4], VQ-VAE [5], and VQ-GAN [6] have demonstrated that encoder-decoder architectures can successfully compresses high-dimensional images into a low-dimensional (optionally discrete) latent space, which can be used as tokens for E2E driving models. MoST [26] adopts this approach, tokenizing each image into 256 tokens using a VQ-GAN [6] encoder (alongside other ViT-based models). TiTok [27] combines ViTs and VQ-GANs into a unified framework that further pares down the number of tokens needed to as low as 32. More broadly, while autoencoders can represent images parsimoniously, they require a fixed input resolution and focus on single images (neglecting camera geometries and scaling linearly with camera count and resolution).

More recent works explore alternative image representations (e.g., tokenizing the frequency spectrum [28]) and adaptive methods that allocate varying token amounts to images [29], [30]. Such advancements are exciting, and orthogonal to our approach since it can be paired with any backbone image encoder or online token reducer.

Volumetric Latent Representations. Accounting for multi-camera geometry is a critical aspect of the AV domain, due to both the number of cameras and their orientation (pointing outwards). To this end, there have been many volumetric latent representations proposed for the neural reconstruction and rendering of 3D environments that can also be used for tokenization.

Explicit representations like voxels and 3D Gaussian Splatting [31] can be produced in a feedforward, differentiable manner from input images, but they produce too many elements (ie., voxels and Gaussians) for direct tokenization. These can be alleviated with sparse voxel hierarchies [32] or Gaussian merging and pruning strategies, but they still result in thousands of tokens per scene. On the other hand, fully-implicit approaches such as NeRF [33] and SDF [34], [35] require little memory, but lack the ability to be produced from input images in a feedforward manner.

Hybrid explicit-implicit approaches such as K-planes [36], [37] lie in the middle, factorizing time and space into K feature grids, with $K = 3$ (triplanes) being the most common choice [38]. Their main advantage is efficiency; the bulk of modeling capacity is shifted to explicit features, allowing for a smaller feature decoder at rendering time without losing expressiveness [38]. Accordingly, triplanes have been widely used to represent faces [38], objects [39], [40], indoor environments [41], and, more recently, urban driving scenes [42], [43], [44]. While these works mainly use triplanes for neural rendering or occupancy modeling, we focus on their tokenization for use by an AR Transformer [1].

III. Multi-Camera Tokenization with Triplanes

A. Background: Triplanes

Triplanes are a volumetric latent representation comprised of three axis-aligned orthogonal feature planes $P_{xy}, P_{xz}, P_{yz} \in \mathbb{R}^{S_i \times S_j \times D_f}$ (Fig. 2), where $S_i \times S_j$ denotes each plane's spatial dimensions, with each grid cell corresponding to a $s_i \times s_j$ m^2 region of space, and D_f is the feature dimension. In this work, triplanes serve as a resolution-agnostic, camera-count-agnostic, and geometry-aware representation of multi-camera images, enabling efficient tokenization for use by AR Transformers. Our approach for doing so is illustrated in Fig. 2.

B. Encoding Multi-Camera Images into Triplanes

At time t, we assume an AV obtains an image $I_c \in \mathbb{R}^{H \times W \times D_{\text{Ch}}}$ from each of its N cameras $c \in \{C_1, \ldots, C_N\}$. To encode these N images into a triplane $\{P_{xy}, P_{xz}, P_{yz}\}$, we first featurize images with a backbone encoder network

$$f_c = \text{ImageEnc}(I_c) \;\forall\; c \in \{C_1, \ldots, C_N\}, \tag{1}$$

where $f_c \in \mathbb{R}^{H_f \times W_f \times D_f}$ and $H_f \times W_f$ denotes the features' spatial size. Then, similar to TPVFormer [42], a grid of 3D query points $q \in \mathbb{R}^{S_x \times S_y \times S_z \times D_f}$ (where S_x, S_y, S_z are the number of triplane grid cells per spatial dimension) attend to the image features through a series of per-image and cross-image deformable attention operations, leveraging each camera's intrinsic and extrinsic parameters for 3D-to-2D projections. Internally, a sinusoidal positional encoding [1] is used to represent triplane grid locations. Finally, the updated queries are averaged along each spatial dimension to produce the desired triplane $\{P_{xy}, P_{xz}, P_{yz}\}$.

Crucially, since the triplane sizes S_x, S_y, S_z are fixed, this process decouples the number of cameras N and their resolution $H \times W$ from the resulting number of tokens.

Finally, as driving scenes are unbounded, we employ a nonlinear scene parametrization as in Mip-NeRF 360 [45] to model far-away objects without increasing triplane sizes.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

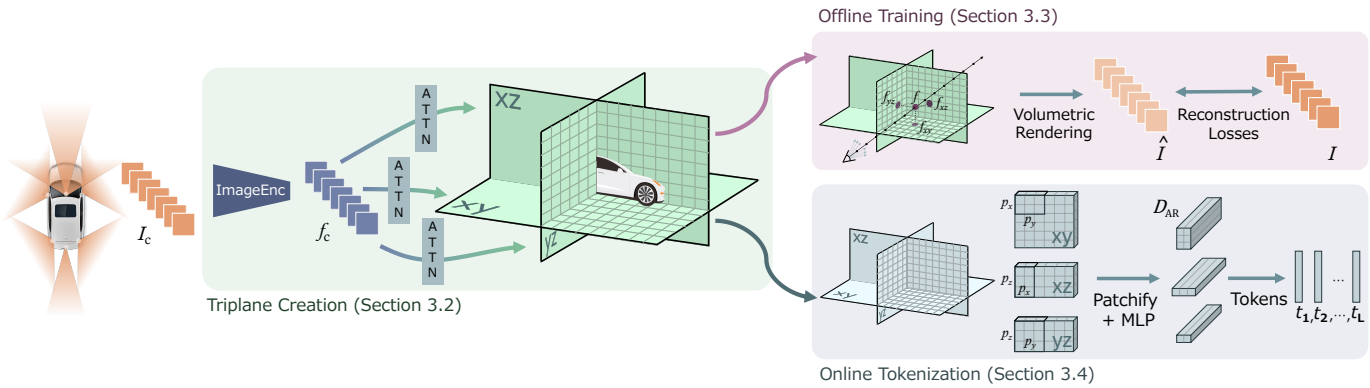


Fig. 2. Our triplane-based multi-camera tokenization strategy. Images from multiple cameras are first encoded through ImageEnc, producing features which form triplanes through a set of geometric per-image and cross-image attention operations. The model is trained in a self-supervised manner, and yields triplanes that can be effectively tokenized for downstream use by an autoregressive Transformer.

Specifically, using the x axis as an example, a grid coordinate $p_{g,x} \in \mathbb{R}$ is mapped to ego-relative coordinates $p_{\text{ego},x} \in \mathbb{R}$ with a symmetric bilinear grid resolution:

$$p_{\text{ego},x} = \begin{cases} R_{o,x}(p_{g,x} + S_{\text{in},x}) - R_{i,x}S_{\text{in},x}, & \text{if } p_{g,x} < -S_{\text{in},x} \\ R_{i,x}p_{g,x}, & \text{if } -S_{\text{in},x} \le p_{g,x} \le S_{\text{in},x} \\ R_{o,x}(p_{g,x} - S_{\text{in},x}) + R_{i,x}S_{\text{in},x}, & \text{if } p_{g,x} > S_{\text{in},x} \end{cases}$$

where $R_{i,x}, R_{o,x} \in \mathbb{R}$ denote the resolutions (m / cell) of the inner and outer cells of the triplane, respectively, and $S_{\text{in},x} \in \mathbb{N}$ is the number of inner grid cells.

C. Scalable Training via Volumetric Rendering

In contrast to prior works which leverage multiple complex (e.g., Hessian-based [43]) self-supervised losses, our work is trained to minimize only two pixel-wise reconstruction losses,

$$\mathscr{L}(I, \hat{I}) = \lambda_{\text{LPIPS}}\text{LPIPS}(I, \hat{I}) + \lambda_1 \|I - \hat{I}\|_1 \tag{2}$$

where LPIPS is the learned perceptual image patch similarity metric [46], λ are scalar weighting factors, $\|\cdot\|_1$ is the ℓ_1 norm, $\hat{I}$ is the predicted (rendered) image from the triplane, and I denotes the ground truth (GT) image from data.

Images $\hat{I}$ are rendered from triplanes via ray sampling and aggregation. Triplanes are queried for 3D positions $x \in \mathbb{R}^3$ by first projecting x onto each of the three feature planes, retrieving the corresponding feature vectors f_{xy}, f_{xz}, f_{yz} via bilinear interpolation, and aggregating the three feature vectors into f via elementwise product. A lightweight MLP then decodes the 3D features f into color and density, which are rendered into RGB images using volumetric rendering [33].

Notably, in contrast to current autoencoder-based methods [6], [47], [48], our approach does not use any GAN losses (whose training process is particularly sensitive). As we will show in Section IV, even without these losses, our triplane-based driving model matches/exceeds the driving performance of autoencoder-based approaches.

D. Tokenizing Triplanes for Downstream Use

With a trained model in hand, multi-camera input images can be converted into triplanes in a feedforward manner. Once a triplane is obtained, it can then be tokenized and ingested by a downstream AR Transformer.

Continuous Tokens. Triplanes can be converted into a 1D token sequence in multiple ways. In this work, we patchify them as in ViT [7], splitting triplanes into patches of features and encoding them into a set of feature vectors with the desired number of dimensions for downstream use.

Formally, each feature plane $P_{ij} \in \mathbb{R}^{S_i \times S_j \times D_f}$ is converted to a token sequence $\{t_\ell\}$ by first reshaping it into a set of patches $P'_{ij} \in \mathbb{R}^{S_i/p_i \times S_j/p_j \times D_f p_i p_j}$. Then, a single-layer MLP converts the resulting feature dimension $D_f p_i p_j$ into D_{AR}, producing a set of tokens $\{t_\ell\} \in \mathbb{R}^{L_{ij} \times D_{\text{AR}}}$ where $L_{ij} = \frac{S_i S_j}{p_i p_j}$ is the number of tokens needed to represent plane P_{ij} and D_{AR} is the feature dimension of the downstream AR Transformer. Performing this for each plane results in an overall token sequence $\{t_\ell\} \in \mathbb{R}^{L \times D_{\text{AR}}}$ where $L = L_{xy} + L_{xz} + L_{yz}$.

Discrete Tokens. While the above produces continuous tokens, discrete tokens can also be output. In particular, a Finite Scalar Quantization (FSQ) [49] layer can be added after triplane creation to learn a discrete codebook and produce discrete tokens. However, in initial experiments, we found that discrete tokens underperformed continuous tokens. Thus, only continuous tokens are used in this work.

IV. Experiments

Dataset. To train and evaluate driving performance, we leverage a large internal dataset consisting of 20,000 hours of driving data from multiple ego-vehicles in 1700+ cities and 25 countries. Accordingly, it contains a variety of driving scenarios including highway and urban driving, multiple weather conditions, day and night times, and varying amounts of traffic, with a geographically-separate 90% trainval and 10% test split. Each ego-vehicle has 7 cameras which we downsample to a per-camera resolution of $H \times W = 320 \times 512$ with 10 Hz observation frequency. Fig. 4 visualizes an example from the dataset.

Triplane Model. In all experiments, we set $S_x = S_y = 96$, $S_z = 48$, and $D_f = 192$. To account for driving in both high-speed freeways and slower urban streets, each triplane metrically represents 180m ahead, behind, left, and right of the ego-vehicle, as well as 45m above and 3m below the ego-vehicle, visualized in Fig. 3. As described in Section III-B, we employ a symmetric bilinear grid resolution with $S_{\text{in},x} = S_{\text{in},y} = 36$ cells and, since we cannot see under ground, the

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

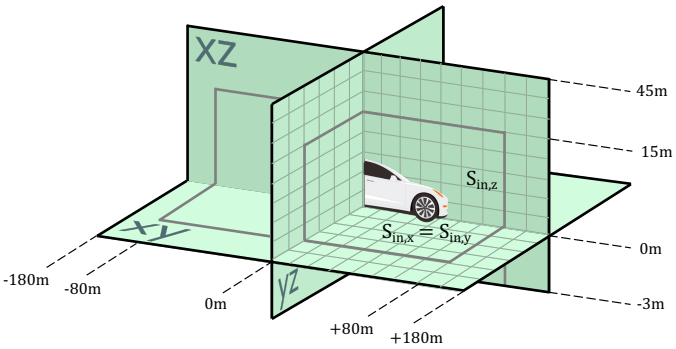


Fig. 3. We employ a bilinear triplane resolution, representing regions outside of $S_{\text{in},x}, S_{\text{in},y}, S_{\text{in},z}$ with fewer grid cells.

TABLE I. Irrespective of the choice of image encoder or number of cameras, triplanes can accurately model outdoor driving scenes, matching state-of-the-art neural reconstruction performance.

Choice of ImageEnc in Eq. (1)	PSNR ↑	SSIM ↑
DINOv2-small [50] (4 cameras)	29.15	0.85
ResNet50 [52] (4 cameras)	29.58	0.86
DINOv2-small [50] (7 cameras)	28.94	0.84
ResNet50 [52] (7 cameras)	29.34	0.85

z-axis asymmetrically represents $[-3,15]$m with 36 cells and $(15,45]$m with the remaining 12 cells.

We use the 22M-parameter DINOv2-small [50] model for ImageEnc in Eq. (1) and two sets of per-image and cross-image attention layers (4 total attention layers, totaling 6.5M parameters) to convert image features to triplanes.

AR Transformer. An LLM-like backbone serves as our representative AR Transformer model [51]. It takes sensor tokens and past ego trajectory information as input to produce future ego-vehicle positions. Future trajectories are represented as discrete tokens and decoded to positions at 10 Hz.

Baselines. Since image autoencoders and ViTs are the most commonly-used image tokenizers (Section II), we compare to VQGAN [6] and DINOv2-small [50] as representative models.

Training. We employ a multi-stage training pipeline. Tokenizers are first pre-trained to model driving data on our internal dataset (except for DINOv2 as it is already internet-pretrained), followed by training the randomly-initialized backbone on the resulting tokens. The overall tokenizer-LLM combination is trained to minimize a next-trajectory-token prediction loss. We set $\lambda_{\text{LPIPS}} = \lambda_{\text{l}} = 0.5$ in Eq. (2). Additional training details can be found in [51].

Metrics. To evaluate triplane quality, we compare the reconstructed images to GT images using peak signal-to-noise ratio (PSNR) and structural similarity index measure (SSIM). To evaluate the combined tokenizer-LLM open-loop driving performance, we leverage the commonly-used minADE_6 metric (denoting the minimum average displacement error among 6 sampled trajectories), formulated as

$$\text{minADE}_k(\hat{o}, o) = \min_{k=1,\ldots,6} \frac{1}{T} \sum_{t=1}^{T} \|\hat{o}_k^{(t)} - o^{(t)}\|_2 \qquad (3)$$

TABLE II. When paired with a 1B AR backbone, our triplane-based tokenizer achieves similar open-loop motion planning performance as baseline tokenizers, while producing significantly fewer sensor tokens. Patch sizes are denoted as $(p_x - p_y - p_z)$.

Image Tokenizer	Tokens per Image ↓	minADE_6 (m) ↓ @1s	@3s	@5s
VQGANEnc	160	**0.08**	0.33	0.74
DINOv2-small	160	**0.08**	0.32	0.69
Ours (8-8-8)	**45**	**0.08**	0.32	0.72
Ours (4-6-6)	104	**0.08**	**0.31**	**0.67**

TABLE III. Our approach is competitive with state-of-the-art approaches, even with a much smaller backbone (1B) and much fewer input tokens (35% to 72% less). Triplane patchification sizes are denoted with $(p_x - p_y - p_z)$.

nuScenes E2E Planning	Traj L2 (m) ↓ @1s	@2s	@3s	$\text{Ave}_{1,2,3s}$
UniAD [9]	0.48	0.89	1.47	0.95
VAD-Base [10]	0.41	0.86	1.46	0.91
PARA-Drive [11]	0.26	0.59	1.12	0.66
TOKEN 7B [54]	0.26	0.70	1.46	0.81
DiMA 7B (VAD-Tiny) [55]	0.20	0.53	1.10	0.61
DiMA 7B (VAD-Base) [55]	0.18	0.50	1.02	0.57
DINOv2-small + 1B	0.34	0.69	1.17	0.73
Ours (8-8-8) + 1B	0.31	0.63	1.10	0.68
Ours (4-6-6) + 1B	0.29	0.62	1.08	0.66

A. *Triplane Representation Quality*

To evaluate whether triplanes can represent driving scenarios collected from outward-facing cameras, we first evaluate their reconstruction performance after the first stage of training. Fig. 4 visualizes triplane-rendered images, GT images, and a 3-color triplane feature visualization (obtained with PCA). As can be seen in Table I, triplanes are able to represent both nearby and far-away elements of the scene, achieving 28.94 dB PSNR and 0.84 SSIM when modeling all 7 cameras, comparable to state-of-the-art neural rendering performance [53]. If modeling fewer cameras (e.g., only the front 4 cameras), reconstruction performance increases to 29.15 dB PSNR and 0.85 SSIM, as more modeling capacity can be used per-image.

Alternative Choices for ImageEnc. Our work can operate with any image backbone that produces 2D features. To demonstrate this, Table I also shows triplane reconstruction performance with a CNN-based image encoder (an ImageNet-pretrained ResNet50 [52]). As can be seen, this choice yields even better PSNR and SSIM values, although differences in model optimization result in an increase in inference runtime compared to DINOv2-small. Accordingly, we use DINOv2-small for ImageEnc in Eq. (1) in all subsequent experiments.

B. *Open-Loop Driving Performance*

To evaluate driving performance with model sizes that are suitable for deployment, we train a 1B-parameter LLM-like model ($D_{\text{AR}} = 2048$) to drive from the 4 front-facing cameras, with their past 6 frames as scene context (totaling 24 frames of input). In addition to the VQGAN and DINOv2-small tokenizer baselines, we evaluate two patch size p_x, p_y, p_z ablations for our triplane tokenizer. The first, $(p_x, p_y, p_z) = (4,6,6)$,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 4. Triplanes can accurately represent environments observed by outward-facing cameras (**top**), producing accurate reconstructions (**bottom**) and semantically-meaningful features after only being trained with self-supervised pixel reconstruction losses (**right**, a PCA visualization of triplane features).

yields 416 tokens per timestep (104 tokens per image, 35% fewer than the baselines) and matches the inference runtime of the baseline approaches. The second employs more aggressive triplane patchification, with $(p_x, p_y, p_z) = (8,8,8)$, resulting in 180 tokens per timestep (45 tokens per image, 72% fewer than the baselines) and a significantly faster runtime. As can be seen in Table II, the backbone trained with our approach performs similarly to the baselines across all timescales while requiring significantly fewer tokens per image.

In Table III, we provide a comparison of our approach and the DINOv2 baseline to a variety of other state-of-the-art (SotA) approaches for E2E planning on the nuScenes [56] validation set. We follow the standardized evaluation setup from PARA-Drive [11]: predicting 3s of future ego-motion from 2s of past image frames, training only on the nuScenes train set, and using the same displacement metric definitions. As can be seen, our work is competitive with SotA approaches. Further, our approach performs similarly to works like DiMA [55] and TOKEN [54] (which use much larger 7B LLM backbones compared to our 1B), while running much faster due to our reduced input token counts. Importantly, both the (4-6-6) and (8-8-8) versions of our approach outperform the baseline DINOv2-based tokenizer.

Halfplane Token Reduction. Note that, in this particular setup (modeling only the front-facing cameras), one can further reduce the number of tokens produced by our triplane-based approach by removing half of the xy and xz planes (they model space behind the car, and are thus unused, as visualized in Fig. 6). The results in Table II use this token-reduction technique, as do all subsequent experiments modeling only the 4 front-facing cameras. Depending on the chosen patch sizes, this can further reduce token counts by 30-40% (specifically, 37.5% for $(p_x, p_y, p_z) = (4,6,6)$ and 40.9% for $(p_x, p_y, p_z) = (8,8,8)$).

C. Inference Time Scaling

Our main hypothesis for using a triplane-based multi-camera tokenization strategy is that, while adding any layers to create an intermediate representation will increase tokenization runtime, the corresponding decrease in sensor tokens will lead to greater savings in backbone runtime, resulting in an overall inference time decrease since billion-parameter Transformers are virtually always the main inference bottleneck in such AV stacks.

To confirm this hypothesis, we profile the runtimes of combined tokenizer-backbone models with a variety of backbone model sizes, number of historical frames, and numbers of cameras. In addition to the 1B-parameter version above, we also profile the runtime of 3B-parameter and 7B-parameter backbones. Such model sizes were previously considered for AV deployment, and even successfully demonstrated on real AVs in works such as [20]. We report the average inference times of different parts of the combined model over 100 forward passes on a single A100 80GB GPU in Fig. 5. We omit error bars as there is very little variation across runs (95% confidence intervals are within 0.5ms of means).

As can be seen in Fig. 5, the majority of overall inference time is spent in the backbone (blue and green bars) and we observe the expected linear scaling of inference time as the number of cameras increases (from the top row to the bottom row) and as the number of context frames per camera increases (within each plot).

Fig. 5 confirms that our method demonstrates the tradeoff described above: an increase in tokenization time (larger purple bar) yields a much faster AR prefill stage (much smaller blue bar), with additional gains if using a more aggressive patchification scheme (increasing p_x, p_y, p_z). This results in our method achieving significant inference time reductions compared to baselines, often *halving* model runtime with larger context sizes and number of cameras.

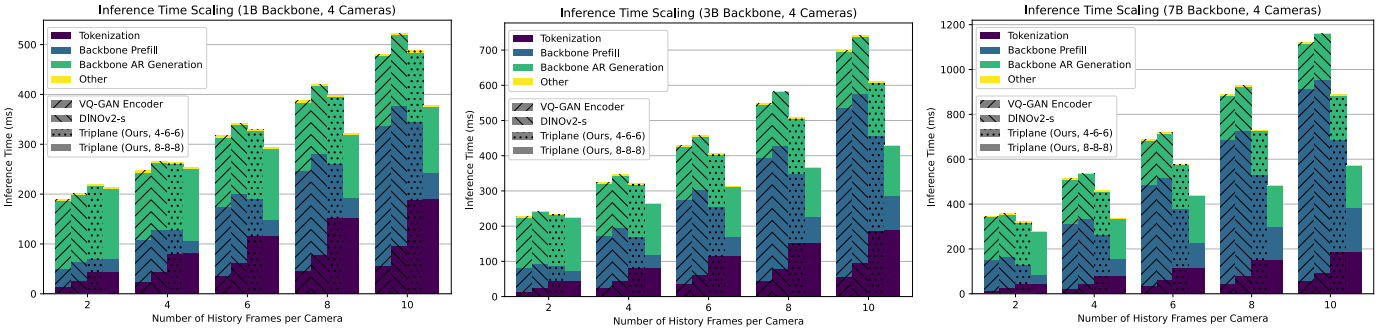
If targeting a 3 Hz inference frequency (e.g., as in [20]), it is difficult to incorporate more than 2-4 frames of context with model sizes above 1B parameters when using baseline tokenization approaches. In contrast, our triplane-based tokenizer enables running a 7B-parameter model with 7 cameras and 4 frames of context at 3 Hz!

An additional benefit of triplane-based tokenization is that patchification can be modified after training, since it is executed *after* triplane creation, enabling easier tuning of overall model performance without retraining the triplane.

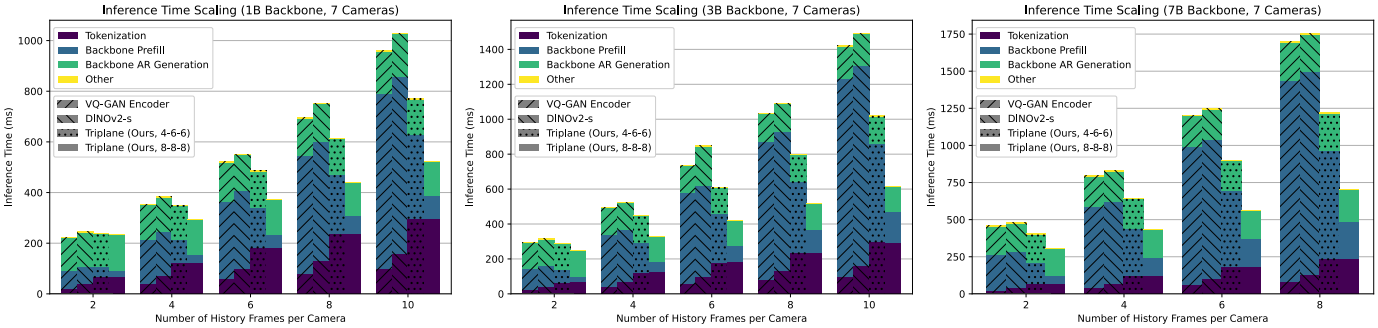
D. Resolution and Camera Count Agnosticism

Higher Resolutions. To validate that our approach can seamlessly handle higher image resolutions, we resample images from our dataset with a 3.6× higher resolution, $H' \times W' = 576 \times 1024$, and retrain our triplane-based approach with them. The resulting model achieves 28.00 dB PSNR and 0.82 SSIM, which is lower than the lower-resolution results in Table I. This is to be expected, however, as we do not alter the triplane size, meaning 3.6× more pixels are being modeled by the same number of features. Fig. 7 confirms qualitatively

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.



(a) Inference time scaling results when taking the 4 front-facing cameras as input.



(b) Inference time scaling results when taking all 7 cameras as input.

Fig. 5. Our triplane-based multi-camera tokenization approach scales much more favorably than baseline approaches as the number of cameras and number of context frames per camera increases. This trend holds across multiple autoregressive Transformer sizes.

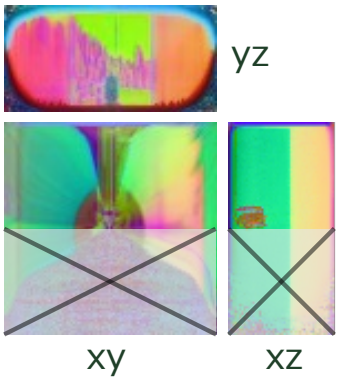


Fig. 6. If modeling cameras that face a certain direction (e.g., all front-facing cameras, as in this figure), the number of tokens can be further reduced by removing unused triplane regions, indicated with an "x". Depending on the chosen patch sizes p_x, p_y, p_z, this can further reduce token counts by up to 40%.

that triplanes can faithfully represent higher resolution data, without requiring any changes to triplane sizes.

To fully verify that the resulting tokens are still useful for driving, we retrain the 1B backbone on the resulting tokens (using $(p_x, p_y, p_z) = (4,6,6)$) and observe identical performance to our previous $(4,6,6)$ model in Table II.

Camera Count Agnosticism. To confirm our method's agnosticism to the number of cameras, as initially demonstrated in Table I, we measure the motion planning performance of the 1B backbone with 7 cameras and compare it to the 4-camera equivalent from above. Fig. 8 shows an interesting trend: When paired with our 7-camera triplane-based approach, the 1B backbone is able to achieve performance within 2.5% of its 4-camera performance. However, when paired with the DINOv2-small baseline, it yields 18% worse open-loop motion planning performance. We also observed training instabilities with the 7-camera DINOv2-small baseline, potentially due to the long sequence length brought by modeling 7 cameras each with 6 context frames (totaling 42 frames of input).

E. Performance in Closed-Loop Simulation

Finally, it is well-known that open-loop driving performance does not necessarily correlate to closed-loop driving performance [57]. Accordingly, we additionally validate the driving performance of the combined tokenizer-LLM models in a closed-loop simulation environment.

We conduct closed-loop evaluation using a state-of-the-art in-house neural reconstruction (NR)-based simulator, capable of reconstructing logged driving events and rendering novel views if the ego-vehicle departs from its originally-logged trajectory [51]. We place the combined tokenizer-LLM models within 75 challenging scenarios, mined specifically due to the amount of ego-agent and agent-agent interactions within them, each of which are 20s-long. Fig. 9 shows one such rollout with the currently-predicted trajectories superimposed over the image. Within these scenarios, we have access to high-quality GT map labels, and so we compare models by their offroad rates (in what proportion of closed-loop rollouts did the ego-vehicle leave the road). Owing to the significant speedups achieved by the $(p_x, p_y, p_z) = (8,8,8)$ version of our triplane-LLM combined model, we evaluate it against the DINOv2-LLM combined baseline. Table IV shows that our approach achieves a slightly lower offroad rate compared to the baseline.

V. Conclusion

In conclusion, we present a novel multi-camera image tokenizer based on triplanes that can produce sensor to-

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 7. Triplanes can faithfully represent higher resolution data without requiring any changes to triplane sizes. Since these cameras are all front-facing, the rear half of the *xy* and *xz* planes are unused.

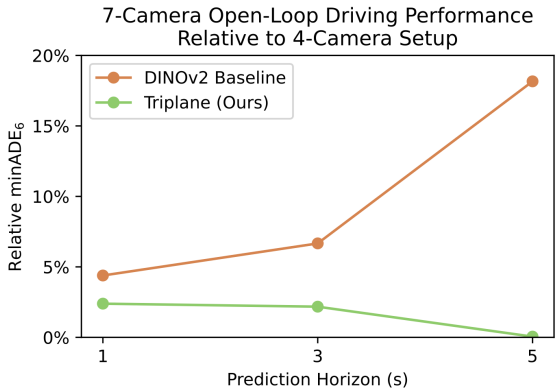


Fig. 8. Our triplane model performs similarly with 7 cameras and 4 cameras, whereas the DINOv2-small baseline struggles.

TABLE IV. When paired with a 1B AR backbone, our triplane-based tokenizer achieves slightly better closed-loop driving performance as the DINOv2-small baseline, while producing significantly fewer sensor tokens. Patch sizes are denoted $(p_x - p_y - p_z)$.

Image Tokenizer	Tokens per Image ↓	Offroad Rate ↓ @6s	@12s	@20s
DINOv2-small	160	2.7%	2.7%	4.0%
Ours (8-8-8)	45	**1.4%**	**1.4%**	**2.7%**

kens in a resolution-agnostic, camera-number-agnostic, and geometrically-aware manner. In comparison to autoencoder and ViT-based baselines, our triplane-based approach is able to represent multi-camera images with much fewer tokens, yielding significant inference time reductions while achieving comparable open-loop motion planning performance and slightly improved offroad rates in closed-loop simulations.

A core limitation of the proposed approach is that it focuses on per-timestep triplane modeling, leaving across-timestep modeling strategies to future work. Other exciting areas of future work include further triplane size reduction and feature dimension compression, and additional reductions in tokenizer latency (potentially leveraging model optimization tools such as pruning and optimized runtime environments, e.g., TensorRT).

Acknowledgments. We thank Jiawei Yang, Yue Wang, Seung Wook Kim, Sérgio Agostinho, Amrita Mazumdar, Omer Shapira, and Shalini De Mello for many helpful technical discussions and guidance, as well as Michael Watson, Maximilian Igl, Michal Tyszkiewicz, Aaron Smith, Peter Karkus, and Zan Gojcic for their support in executing closed-loop simulations.

REFERENCES

[1] A. Vaswani, *et al.*, "Attention is all you need," in *Conf. on Neural Information Processing Systems*, 2017.
[2] H. Gao, *et al.*, "A survey for foundation models in autonomous driving," *arXiv preprint arXiv:2402.01105*, 2024.
[3] X. Zhu, *et al.*, "A survey on model compression for large language models," *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1556–1577, 2024. [Online]. Available: https://aclanthology.org/2024.tacl-1.85/
[4] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Conf. on Neural Information Processing Systems*, 2015.
[5] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Conf. on Neural Information Processing Systems*, 2017.
[6] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2021.
[7] A. Dosovitskiy, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," in *Int. Conf. on Learning Representations*, 2020.
[8] F. Wang, *et al.*, "Scaling laws in patchification: An image is worth 50,176 tokens and more," *arXiv preprint arXiv:2502.03738*, 2025.
[9] Y. Hu, *et al.*, "Planning-oriented autonomous driving," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.
[10] B. Jiang, *et al.*, "VAD: Vectorized scene representation for efficient autonomous driving," in *IEEE Int. Conf. on Computer Vision*, 2023.
[11] X. Weng, *et al.*, "PARA-Drive: Parallelized architecture for real-time autonomous driving," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.
[12] A. Brohan, *et al.*, "RT-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022. [Online]. Available: https://arxiv.org/abs/2212.06817
[13] S. Reed, *et al.*, "A generalist agent," *Transactions on Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=1ikK0kHjvj
[14] A. Brohan, *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023. [Online]. Available: https://arxiv.org/abs/2307.15818
[15] E. Collaboration, "Open X-Embodiment: Robotic learning datasets and RT-X models," *arXiv preprint arXiv:2310.08864*, 2023. [Online]. Available: https://arxiv.org/abs/2310.08864
[16] Y. Jiang, *et al.*, "VIMA: General robot manipulation with multimodal prompts," in *Int. Conf. on Machine Learning*, 2023.
[17] A. Hu, *et al.*, "GAIA-1: A generative world model for autonomous driving," *arXiv preprint arXiv:2309.17080*, 2023.
[18] Wayve, "LINGO-1: Exploring natural language for autonomous driving," Wayve, 2023, Available at https://wayve.ai/thinking/lingo-natural-language-autonomous-driving/.
[19] Z. Xu, *et al.*, "DriveGPT4: Interpretable end-to-end autonomous driving via large language model," *IEEE Robotics and Automation Letters*, 2024.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Fig. 9. In a challenging closed-loop simulation of a construction scene, the baseline DINOv2-based model (**left**) stops in the middle of the road and does not start driving again, whereas our triplane-based model (**right**) has no difficulty nudging around the parked excavator and driving behind the cement mixer. On the front camera views are the predicted ego-vehicle trajectories (selected driving trajectory in orange, with the other 5 predicted trajectories in blue).

[20] X. Tian, *et al.*, "DriveVLM: The convergence of autonomous driving and large vision-language models," in *Conf. on Robot Learning*, 2024.

[21] S. Wang, *et al.*, "OmniDrive: A holistic llm-agent framework for autonomous driving with 3d perception, reasoning and planning," *arXiv preprint arXiv:2405.01533*, 2024.

[22] J.-J. Hwang, *et al.*, "EMMA: End-to-end multimodal model for autonomous driving," *arXiv preprint arXiv:2410.23262*, 2024. [Online]. Available: https://arxiv.org/abs/2410.23262

[23] X. Zhai, *et al.*, "Sigmoid loss for language image pre-training," in *IEEE Int. Conf. on Computer Vision*, 2023.

[24] Gemini Team, Google, "Gemini: A family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023. [Online]. Available: https://arxiv.org/abs/2312.11805

[25] X. Chu, *et al.*, "MobileVLM: A fast, reproducible and strong vision language assistant for mobile devices," *arXiv preprint arXiv:2312.16886*, 2023.

[26] N. Mu, *et al.*, "MoST: Multi-modality scene tokenization for motion prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.

[27] Q. Yu, *et al.*, "An image is worth 32 tokens for reconstruction and generation," in *Conf. on Neural Information Processing Systems*, 2024.

[28] C. Esteves, M. Suhail, and A. Makadia, "Spectral image tokenizer," *arXiv preprint arXiv:2412.09607*, 2024.

[29] S. Duggal, *et al.*, "How many tokens is an image worth?" in *Int. Conf. on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=mb2ryuZ3wz

[30] W. Yan, *et al.*, "ElasticTok: Adaptive tokenization for image and video," in *Int. Conf. on Learning Representations*, 2025.

[31] B. Kerbl, *et al.*, "3d gaussian splatting for real-time radiance field rendering," *Proc. of SIGGRAPH*, vol. 42, no. 4, July 2023. [Online]. Available: http://www-sop.inria.fr/reves/Basilic/2023/KKLD23

[32] L. Wang, *et al.*, "DistillNeRF: Perceiving 3d scenes from single-glance images by distilling neural fields and foundation model features," in *Conf. on Neural Information Processing Systems*, 2024.

[33] B. Mildenhall, *et al.*, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conf. on Computer Vision*, 2020.

[34] J. J. Park, *et al.*, "DeepSDF: Learning continuous signed distance functions for shape representation," in *IEEE Conf. on Computer Vision and Pattern Recognition*, June 2019.

[35] T. Takikawa, *et al.*, "Neural geometric level of detail: Real-time rendering with implicit 3D shapes," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2021.

[36] S. Fridovich-Keil, *et al.*, "K-Planes: Explicit radiance fields in space, time, and appearance," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[37] A. Cao and J. Johnson, "HexPlane: A fast representation for dynamic scenes," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[38] E. R. Chan, *et al.*, "Efficient geometry-aware 3D generative adversarial networks," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[39] J. R. Shue, *et al.*, "3D neural field generation using triplane diffusion," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[40] Y. Hong, *et al.*, "LRM: Large reconstruction model for single image to 3d," in *Int. Conf. on Learning Representations*, 2024.

[41] Y. Han, *et al.*, "Frankenstein: Generating semantic-compositional 3d scenes in one tri-plane," in *Proc. of SIGGRAPH Asia*, 2024.

[42] Y. Huang, *et al.*, "Tri-perspective view for vision-based 3d semantic occupancy prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2023.

[43] ——, "SelfOcc: Self-supervised vision-based 3d occupancy prediction," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2024.

[44] H. Xu, *et al.*, "A survey on occupancy perception for autonomous driving: The information fusion perspective," *Information Fusion*, vol. 114, p. 102671, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253524004494

[45] J. T. Barron, *et al.*, "Mip-NeRF 360: Unbounded anti-aliased neural radiance fields," *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[46] R. Zhang, *et al.*, "The unreasonable effectiveness of deep features as a perceptual metric," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[47] H. Chang, *et al.*, "MaskGIT: Masked generative image transformer," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2022.

[48] J. Yu, *et al.*, "Vector-quantized image modeling with improved VQGAN," in *Int. Conf. on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=pfNyExj7z2

[49] F. Mentzer, *et al.*, "Finite scalar quantization: VQ-VAE made simple," in *Int. Conf. on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=8ishA3LxN8

[50] M. Oquab, *et al.*, "DINOv2: Learning robust visual features without supervision," *Transactions on Machine Learning Research*, 2024, featured Certification. [Online]. Available: https://openreview.net/forum?id=a68SUt6zFt

[51] X. Wu, "Accelerate the future of AI-defined vehicles and autonomous driving," NVIDIA, 2025, Available at https://www.nvidia.com/en-us/on-demand/session/gtc25-dd40000/.

[52] K. He, *et al.*, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[53] J. Yang, *et al.*, "STORM: Spatio-temporal reconstruction model for large-scale outdoor scenes," in *Int. Conf. on Learning Representations*, 2025.

[54] R. Tian, *et al.*, "Tokenize the world into object-level knowledge to address long-tail events in autonomous driving," in *Conf. on Robot Learning*, 2024.

[55] D. Hegde, *et al.*, "Distilling multi-modal large language models for autonomous driving," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2025.

[56] H. Caesar, *et al.*, "nuScenes: A multimodal dataset for autonomous driving," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[57] D. Dauner, *et al.*, "NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking," in *Conf. on Neural Information Processing Systems*, 2024.