Supplementary Material for EMPERROR: A Flexible Generative Perception Error Model for Probing Self-Driving Planners

Niklas Hanselmann[1,2,3], Simon Doll[1,2], Marius Cordts[1], Hendrik P.A. Lensch[2] and Andreas Geiger[2,3]
https://lasnik.github.io/emperror/

Abstract— **In this supplementary document, we provide additional details on the architecture and implementation of EMPERROR, the baseline perception error models (PEMs) and the imitation learning (IL)-based planner in Section I, as well as additional quantitative results in Section III. We also provide additional qualitative results in the supplementary video.**

I. Implementation & Architecture Details

In this section, we provide additional technical details on the three PEMs considered in this work, as well as the IL-based planner. In the following, all multilayer perceptrons (MLPs) use the ELU activation function and layer normalization after each hidden layer, except for the MLP integrated in the transformer layers in EMPERROR and the planner, which perform normalization at the output.

A. Emperror

Input Encodings: To obtain the latent object features $\mathbf{Q}_H$ and $\mathbf{Q}_D$ used as input to the prior encoder, posterior encoder and deterministic decoder, we first project the bird's-eye view (BEV)-position (x, y) of the input $\mathbf{S}$ and $\hat{\mathbf{B}}$ to polar coordinates $(r, \cos(\psi), \sin(\psi))$. This then results in a 23-dimensional vector containing polar BEV position and height of the object center, spatial dimensions, heading angle, two-dimensional velocity vector, per-class sigmoid confidence scores and a one-hot vector indicating the input object as ground-truth $\mathbf{s}_n \in \mathbf{S}$ or detector output $\hat{\mathbf{b}}_n \in \hat{\mathbf{B}}$, respectively. For ground-truth objects, the vector additionally contains the first- and second-order angular and longitudinal dynamics. Both object descriptions are then encoded to the model's feature space via two separate two-layer MLPs with input-, hidden and output dimensions of [$23 \rightarrow 128 \rightarrow 256$] and [$27 \rightarrow 128 \rightarrow 256$] respectively. The latent map features $\mathbf{Q}_\mathcal{M}$ are obtained from a rasterized BEV map $\mathcal{M} \in \mathbb{R}^{256\times256\times5}$ providing information on 5 different aspects of the scene layout at a cell resolution of $0.5\,\text{m}$: The driveable area, car park area, road- and lane dividers, as well as the base layer provided by the nuScenes devkit [1]. The latter consists of a ground projection of LiDAR points on static infrastructure aggregated through multiple drives. The raw input map is then processed by a 5-layer convolutional neural network (CNN) similar to the configuration in [2]. We use kernel sizes of [7, 5, 5, 3, 3], with [16, 32, 64, 128, 256] filters, where the first layer operates at stride one, while the remaining layers operate at stride two. Each layer uses ReLU activations and group normalization. To retain spatial information after tokenization to the set of features $\mathbf{Q}_\mathcal{M}$, we add a two-dimensional sinusoidal positional encoding to the final feature grid.

Transformer: For the encoders and decoder operating on $\mathbf{Q}_H$, $\mathbf{Q}_D$ and $\mathbf{Q}_\mathcal{M}$, the same general transformer layer configuration consisting of multi-head cross-attention, multi-head self-attention (each with 8 heads) and a two-layer MLP with dimensions [$256 \rightarrow 512 \rightarrow 256$]. Each attention operation, as well as the MLP, is followed by layer normalization. Finally, the output of the layer is summed with its input. As described in the main paper, the privileged approximate posterior encoder uses two cross-attention operations to attend to both $\mathbf{Q}_D$ and $\mathbf{Q}_\mathcal{M}$, while the prior encoder and decoder only cross-attend to $\mathbf{Q}_\mathcal{M}$. For both encoders and the decoder, we set $L_e = L_d = 4$.

Latent & Output Projections: The probabilistic encoders use a shared two-layer MLP with dimensions [$256 \rightarrow 256 \rightarrow 2 \times 32$], layer normalization and ELU activations to decode their output to the mean and diagonal covariance of the respective Gaussian. To ensure strictly positive values for the variances, an exponential function is applied to the raw output of the model. Each sampled per object latent variable $\mathbf{z}_n \in \mathbb{R}^{32}$ is then concatenated with the corresponding $\mathbf{q}_H^n$ and processed by two-layer MLP with hidden dimensions [$256 + 32 \rightarrow 256 \rightarrow 256$] to form the input to the decoder. Finally, the output of the decoder is processed to form the final regression parameters and sigmoid class scores as in [3], [4], [5], [6], [7], [8] via two-layer MLPs with dimensions [$256 \rightarrow 128 \rightarrow 10$].

Additional Implementation Details: We train EMPERROR on 4 NVIDIA Tesla V100 GPUs using the Adam [9] optimizer with a batch size of 8, a learning rate of $1\text{e}{-4}$, weight decay of $1\text{e}{-2}$, and clip the norm of the gradients to a maximum value of 35. We pad ground truth-seeded queries $\mathbf{Q}_H^{gt}$ and detector outputs $\hat{\mathbf{B}}$ to a maximum of 300 objects per scene, and do not perform self- or cross-attention for padded positions. During evaluation, we draw random samples from the model's latent prior to fairly evaluate the learned distribution. For the greedy matching of ground truth-

[1] Mercedes-Benz AG R&D, Sindelfingen, Germany
[2] University of Tübingen, Tübingen, Germany
[3] Tübingen AI Center, Tübingen, Germany

seeded queries, a threshold of $4\,\mathrm{m}$ is used. We train for a maximum of 300 epochs and select the model checkpoint which is closest to the respective target detector in terms of mean average precision on the validation set.

B. Baseline Perception Error Models

Architecture Details: We consider two main PEM configurations as baselines. We take inspiration from the ResNet-based models in [10], [11] and extend them to our setting. As these models expect an indicator of the degree of occlusion for each object, we extend the ground-truth input representation with the 4-dimensional discrete visibility score provided by the nuScenes dataset [1]. Similar to EMPERROR, we then use a similar two-layer MLP of dimensions $[30 \rightarrow 128 \rightarrow 256]$ to project the input to the feature space of the model. We then apply a ResNet18 [12] as in [10], [11] and finally use two-layer MLPs to estimate the sigmoid classification scores as well as the parameters of either a Student-T- [11] or Gaussian [10] distribution for the regression parameters. To stabilize convergence, we ensure that the degrees of freedom for the Student-T distribution are ≥ 2 and omit the dropout layers. Note that for the Gaussian, a diagonal covariance matrix is estimated, while for the Student-T the model estimates a full scale matrix, allowing the distribution to explicitly capture correlations between regression parameters. We also design a simpler configuration, where we replace the ResNet backbone with a small MLP of dimensions $[256 \rightarrow 256 \rightarrow 256]$. For the experiments incorporating map information, we use the same convolutional map encoder architecture as EMPERROR. We then concatenate the flattened spatial feature grid to each per-agent input projection, adopting the input dimensionality of the MLP- and ResNet backbones accordingly.

Additional Implementation Details: We train each baseline PEM on 4 NVIDIA Ampere A100 GPUs using the Adam optimizer with a batch size of 144, a learning rate of $1\mathrm{e}{-3}$ and the same weight decay and gradient clipping settings used for EMPERROR. Similarly, we also sample from the models estimated distribution over the output regression parameters during evaluation. We train for a maximum of a 1000 epochs, and select the model checkpoint with the same logic as for EMPERROR after a warmup period of 400 epochs to ensure sufficient convergence.

C. Imitation Learning Planner

Architecture Details: We use similar components as in EMPERROR to build the planner, with a similar two-layer MLP of dimensions $[18 \rightarrow 128 \rightarrow 256]$ encoding input BEV detections as well as sharing the architecture for the convolutional map encoder (but omitting the basemap layer in the input). Each of the three options for the navigational command c^{nav} is modeled via a learnable embedding and summed with the planning query $\mathbf{q}^{\pi}_{\text{ego}}$ along with a projection of the ego vehicle speed ϑ^{ego} obtained via an MLP with dimensions $[1 \rightarrow 256]$. After adding this planning context, $\mathbf{q}^{\pi}_{\text{ego}}$ is processed by $L_{\pi} = 3$ transformer layers of similar

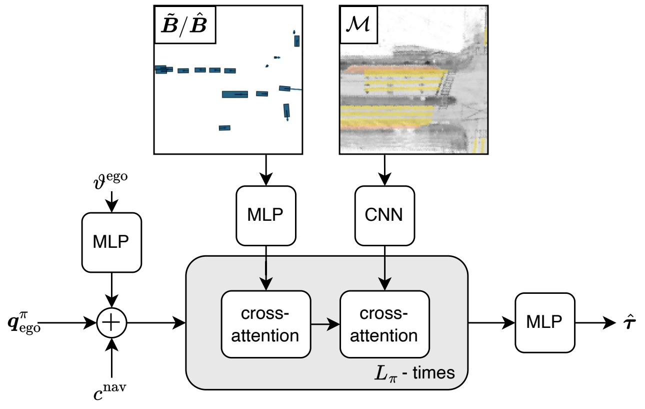


Fig. 1. **Planner Architecture.** We design a simple IL-based planner that incorporates several high-level design choices proposed in recent literature [13], [14], [15]. It refines a learned planning query $\mathbf{q}^{\pi}_{\text{ego}}$ through repeated cross-attention to latent object- and map features and decodes the result to an output trajectory $\hat{\boldsymbol{\tau}}$ using an MLP. To provide the necessary context for planning, embeddings of a discrete high-level navigational command c^{nav} and the current speed of the vehicle ϑ^{ego} are also given as input.

structure as in EMPERROR, performing cross-attention to the map and input detection results. The final trajectory is then decoded by a two-layer MLP of dimensions $[256 \rightarrow 128 \rightarrow 6 \times 3]$.

Objective: The planner parameters ω are learned via imitation learning (IL) from tuples $(\mathbf{B}, \mathcal{M}, \vartheta^{\text{ego}}, c^{\text{nav}}, \boldsymbol{\tau})$ containing expert demonstrations $\boldsymbol{\tau}$. Specifically, we minimize the absolute difference between the planned- and expert waypoints:

$$\mathcal{L}_{\text{plan}} = \frac{1}{T_{\text{plan}}} \sum_{t=0}^{T_{\text{plan}}} \| \hat{\boldsymbol{\tau}}^{t} - \boldsymbol{\tau}^{t} \|_1 \qquad (1)$$

Additional Implementation Details: We train the planner for 100 epochs on a single NVIDIA Tesla V100 GPU using the Adam Optimizer with a batch size of 16, a learning rate of $1\mathrm{e}{-4}$, weight decay of $1\mathrm{e}{-2}$ and clip the norm of the gradients to a maximum value of 1.

II. ADDITIONAL QUALITATIVE RESULTS

We provide additional qualitative results in the **supplementary video**, which allows to gauge the consistency of errors sampled over time. We compare to both the MLP + Gauss and ResNet + StudT baseline configurations. Our method is able to sample error patterns that are more consistent both across the entire scene and over multiple time steps. Besides not being able to explicitly model scene-consistent error patterns, the baseline methods also show difficulty modeling plausible errors that correctly capture all correlations on a per-object level in comparison to EMPERROR, despite their capability to implicitly (MLP + Gauss) or explicitly (ResNet + StudT) model them. This is especially evident for the velocities. We hypothesize that it is beneficial to model this consistently when sampling the error patterns in latent space, as in EMPERROR, rather than directly in output space.

III. Additional Quantitative Results

In Fig. 2, we plot the precision over different recall levels as well as the regression errors for true-positives as they relate to increasing recall, for the 'car' class with Detr3d as the target detector. This shows how the accuracy of a detection hypothesis relates to the model's confidence value assigned to it. These plots illustrate the necessity to also consider the calibration quality when evaluating PEMs, which is addressed by our Cumulative Absolute Difference Area (CD) metrics. In Figures 3 to 5, we show the aggregated metrics for all classes and target detectors in comparison to the MLP + Gauss baseline configuration.

References

[1] https://github.com/nutonomy/nuscenes-devkit, accessed: 02.03.24.

[2] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating useful accident-prone driving scenarios via a learned traffic prior," in *CVPR*, 2022.

[3] Y. Wang, V. . Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. M. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," in *CoRL*, 2021.

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

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

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

[7] S. Doll, R. Schulz, L. Schneider, V. Benzin, E. Markus, and H. P. Lensch, "Spatialdetr: Robust scalable transformer-based 3d object detection from multi-view camera images with global cross-sensor attention," in *ECCV*, 2022.

[8] S. Wang, Y. Liu, T. Wang, Y. Li, and X. Zhang, "Exploring object-centric temporal modeling for efficient multi-view 3d object detection," in *ICCV*, 2023.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[10] J. Sadeghi, B. Rogers, J. Gunn, T. Saunders, S. Samangooei, P. K. Dokania, and J. Redford, "A step towards efficient evaluation of complex perception tasks in simulation," in *NeurIPS Workshops*, 2021.

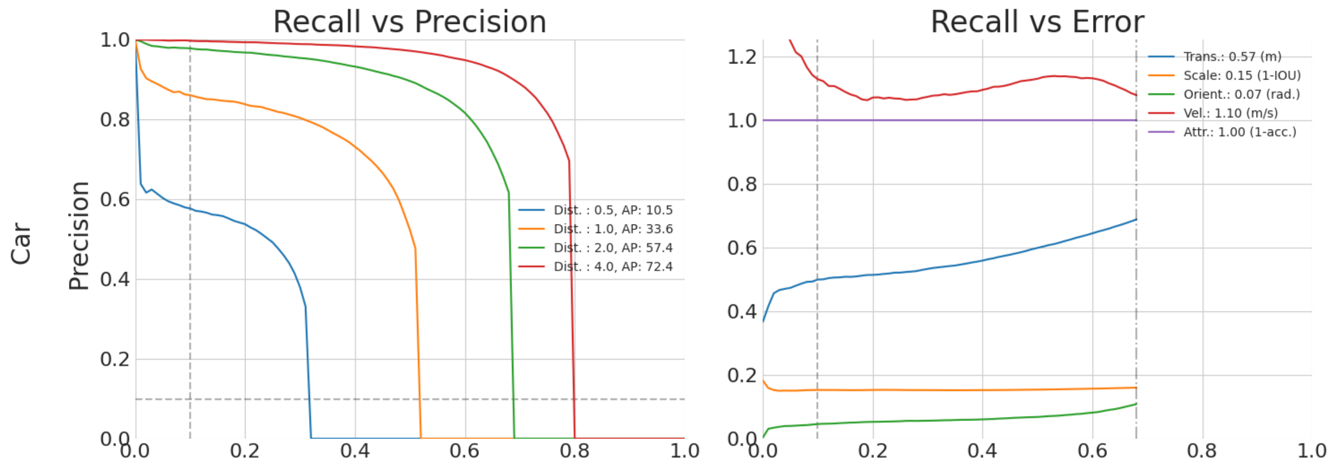
[11] J. Sadeghi, N. A. Lord, J. Redford, and R. Mueller, "Attacking motion planners using adversarial perception errors," *arXiv.org*, vol. 2311.12722, 2023.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

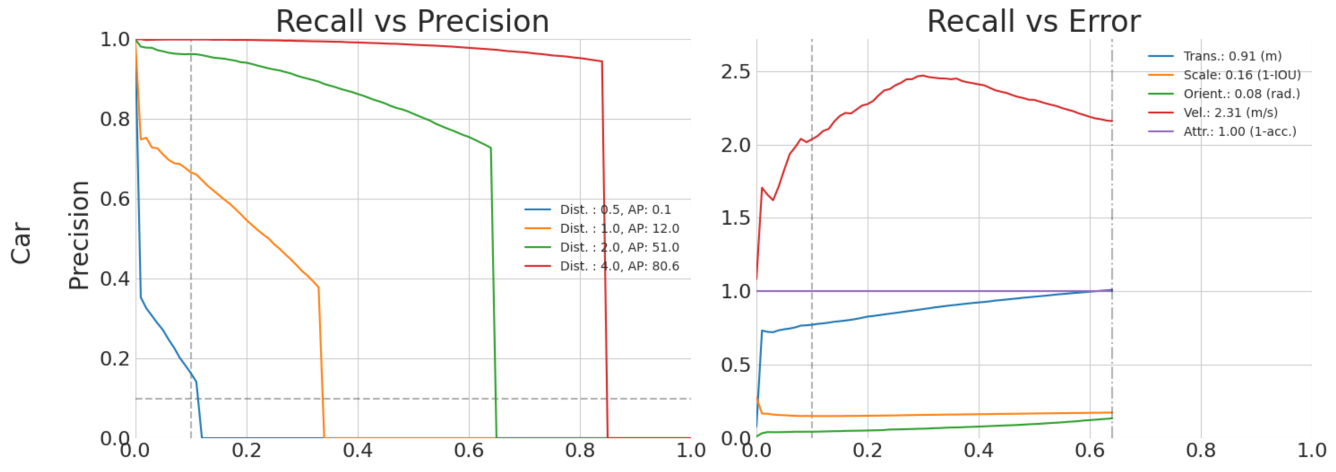
[13] K. Renz, K. Chitta, O.-B. Mercea, A. S. Koepke, Z. Akata, and A. Geiger, "Plant: Explainable planning transformers via object-level representations," in *CoRL*, 2022.

[14] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, "Planning-oriented autonomous driving," in *CVPR*, 2023.

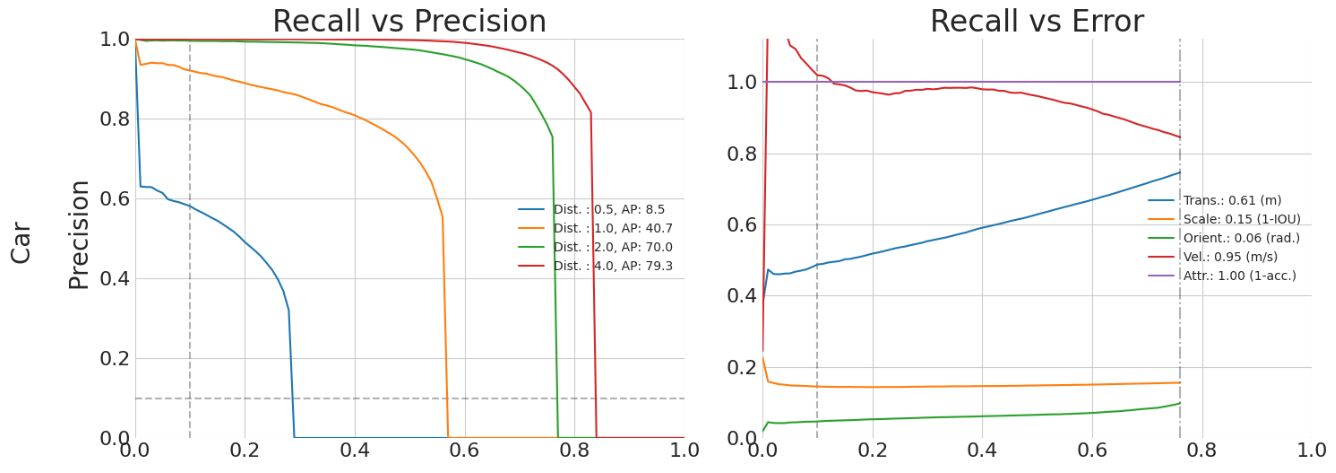
[15] B. Jiang, S. Chen, Q. Xu, B. Liao, J. Chen, H. Zhou, Q. Zhang, W. Liu, C. Huang, and X. Wang, "Vad: Vectorized scene representation for efficient autonomous driving," *ICCV*, 2023.



(a) DETR3D

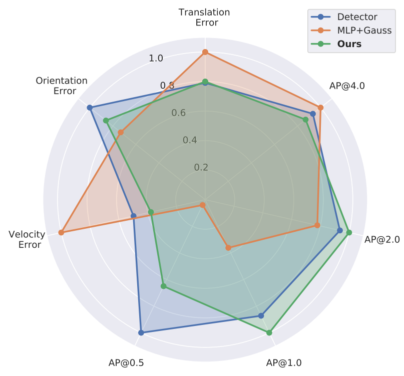


(b) MLP + Gauss

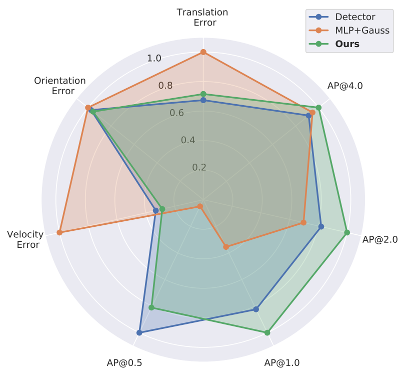


(c) Ours

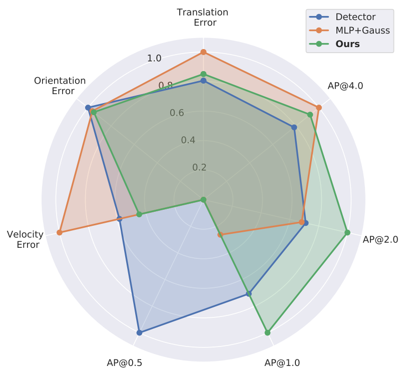
Fig. 2. **Calibration Quality.** Precision over recall and true positive errors for the 'car' class with Detr3d as the target detector.



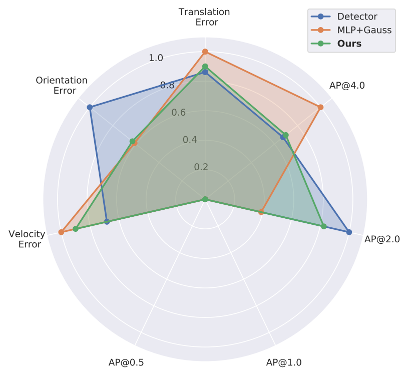
(a) All Classes



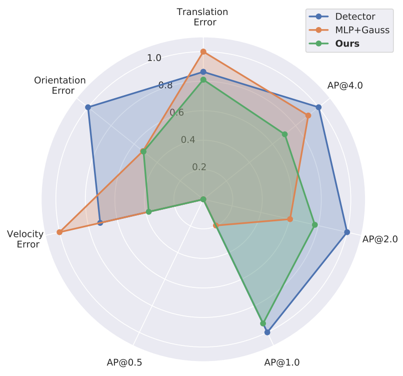
(b) Car



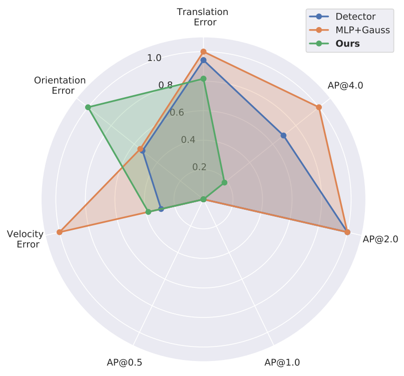
(c) Truck



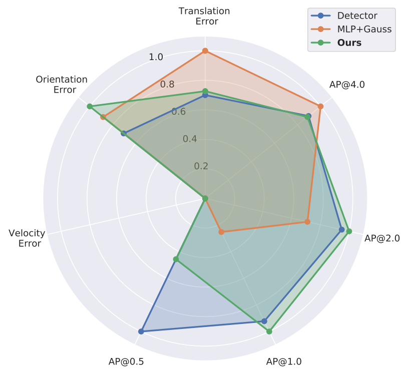
(d) Constr. Veh.



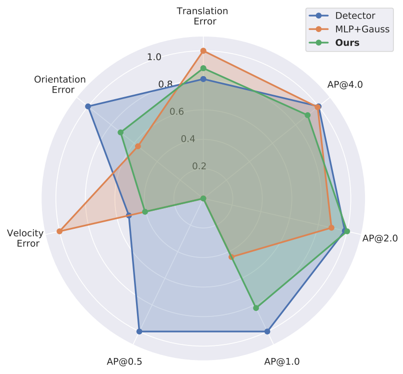
(e) Bus



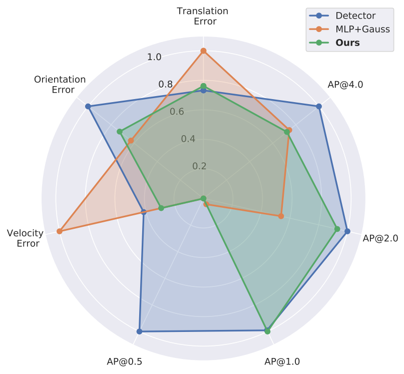
(f) Trailer



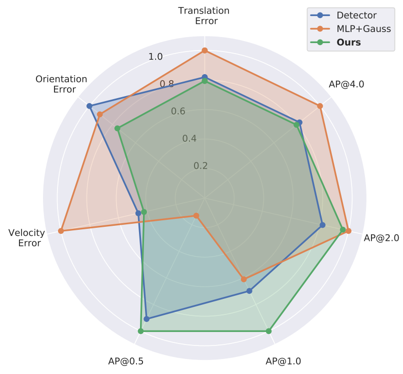
(g) Barrier



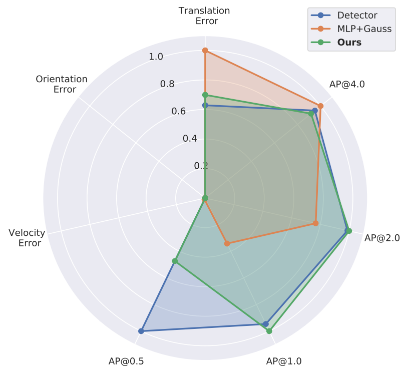
(h) Motorcycle



(i) Bicycle

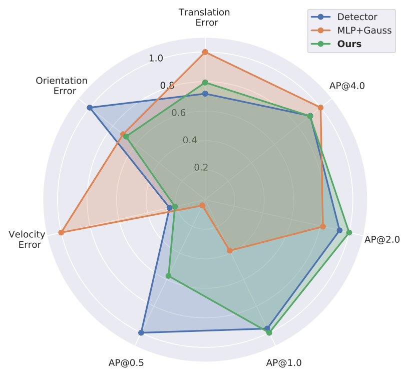


(j) Pedestrian

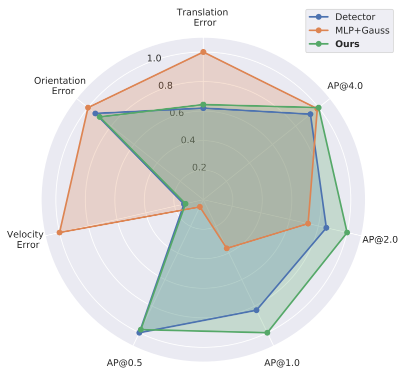


(k) Traffic Cone

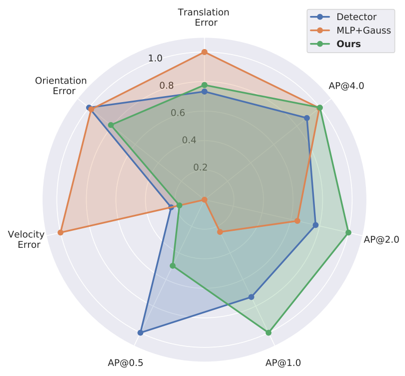
Fig. 3. **PEM Characteristics for Detr3d.**



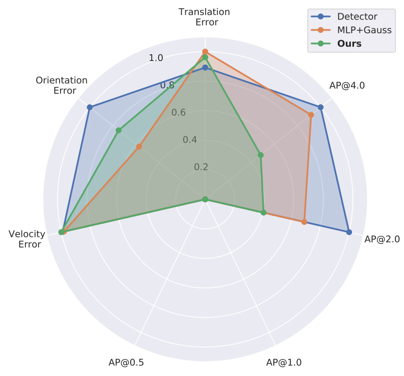
(a) All Classes



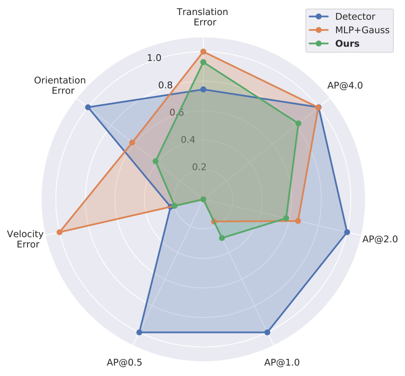
(b) Car



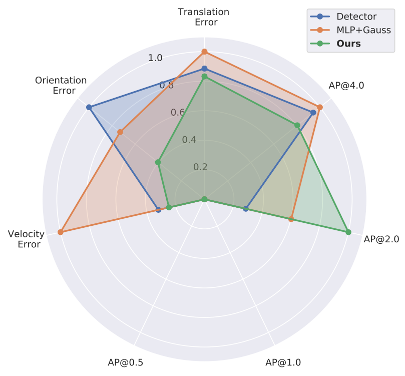
(c) Truck



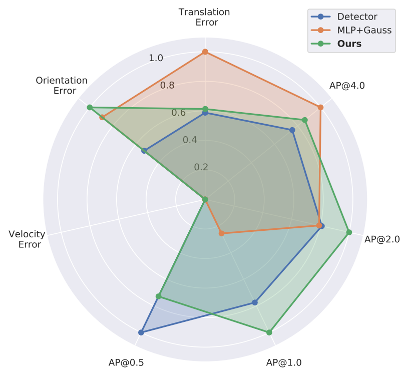
(d) Constr. Veh.



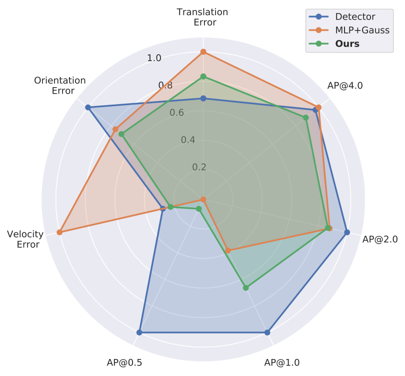
(e) Bus



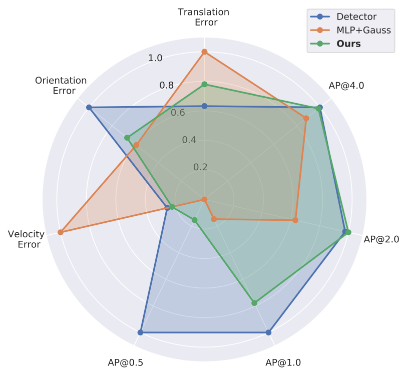
(f) Trailer



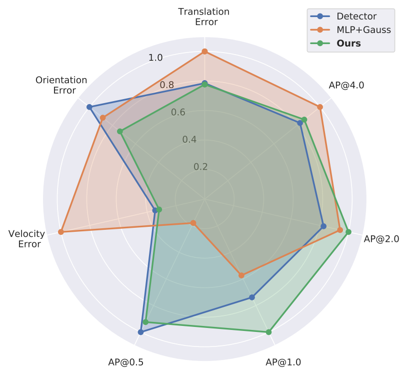
(g) Barrier



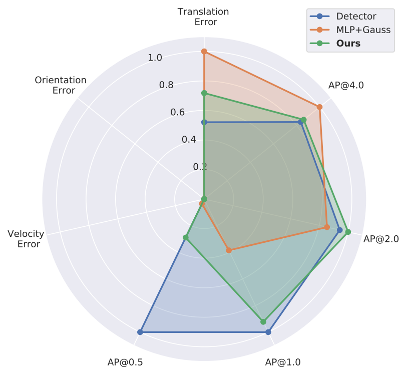
(h) Motorcycle



(i) Bicycle

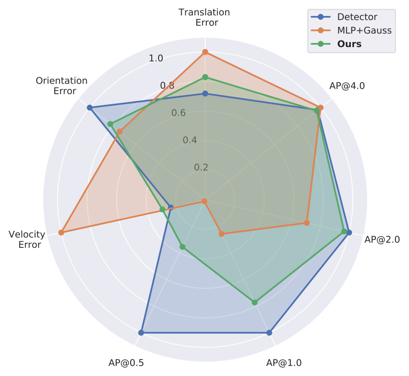


(j) Pedestrian

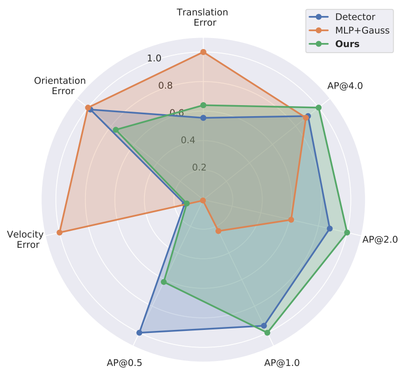


(k) Traffic Cone

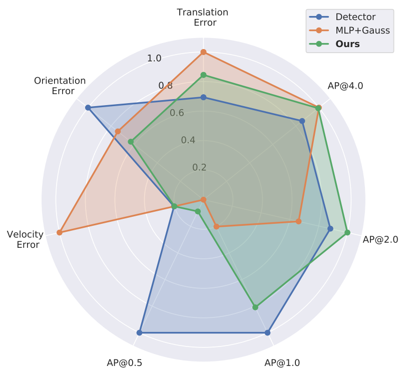
Fig. 4. **PEM Characteristics for BEVFormer.**



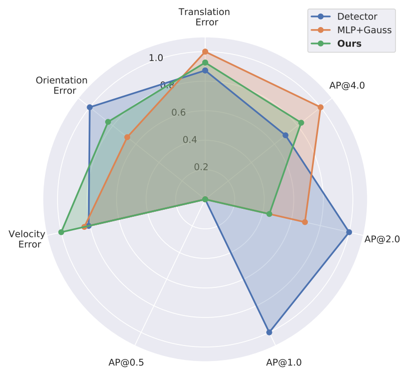
(a) All Classes



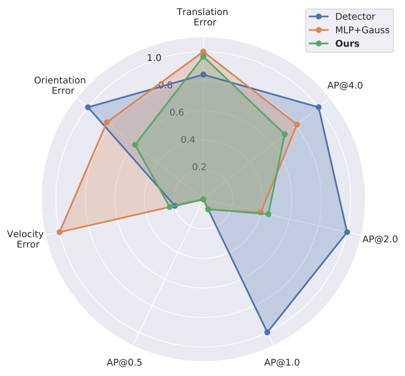
(b) Car



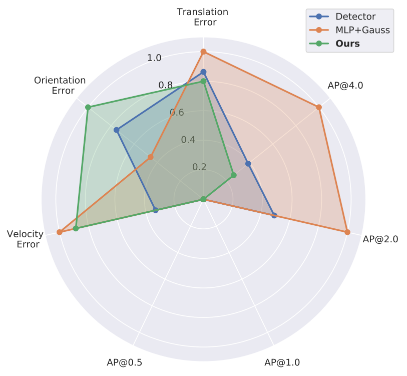
(c) Truck



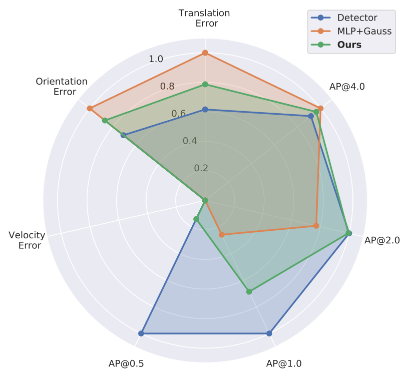
(d) Constr. Veh.



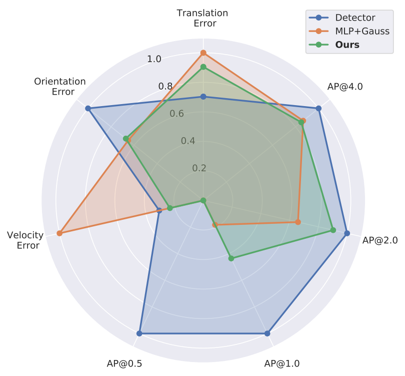
(e) Bus



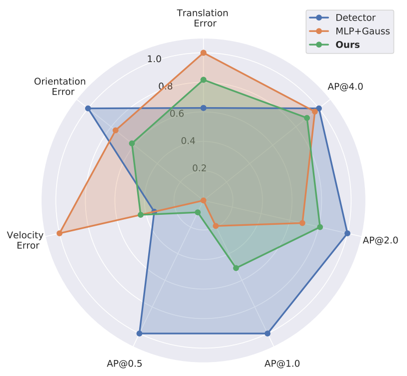
(f) Trailer



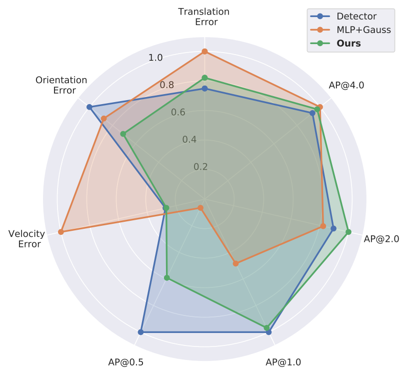
(g) Barrier



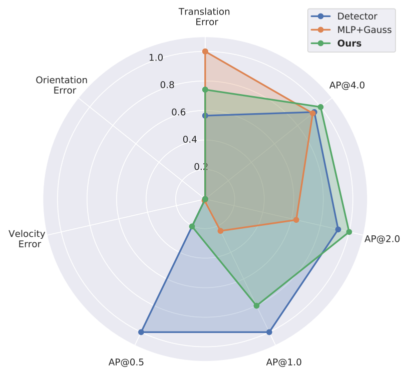
(h) Motorcycle



(i) Bicycle



(j) Pedestrian



(k) Traffic Cone

Fig. 5. **PEM Characteristics for StreamPETR.**