# TEMPORAL LIDAR-CAMERA FUSION FOR TARGET VEHICLE VELOCITY ESTIMATION

**Aayush Anand, Jack Baude, Lilian Coelho de Freitas, and Srijan Kumar Pal**
Department of Computer Science & Engineering
University of Minnesota
{anand239, baude022, ldefreit, pal00036}@umn.edu

## ABSTRACT

Reliable velocity estimates are critical for autonomous driving because planning depends not only on where nearby vehicles are, but also on how they are moving. However, LiDAR provides geometry rather than velocity, so motion is often estimated from changes in detected box centers across frames, which can amplify box jitter, sparse returns, occlusion, and detector localization errors. We study target-vehicle velocity refinement on nuScenes using short temporal windows of LiDAR bird's-eye-view (BEV) tensors and noisy 3D box trajectories. Our model fuses full-scene BEV context, a target-centered BEV crop, and box-level kinematic features over $T = 4$ keyframes with a Transformer encoder. Instead of predicting velocity from scratch, the best model predicts a residual correction to a finite-difference prior computed from noisy boxes, preserving coarse motion while learning to correct systematic errors. On 1,562 held-out validation windows, the residual model achieves 1.145 m/s mean vector error and 0.782 m/s median error, reducing mean error by 46.1% relative to the noisy finite-difference baseline. A BEVFormer diagnostic shows that synthetic box noise does not fully match real camera-detector errors, but lightweight detector-specific fine-tuning reduces the gap. Overall, the results show that a small temporal LiDAR-BEV refinement module can improve target-vehicle velocity estimates on top of existing detections.

## 1 INTRODUCTION

Autonomous vehicles rely on object velocity to predict traffic motion and make safe planning decisions. Knowing where an object is located is not sufficient: the same nearby vehicle can imply very different actions depending on whether it is stopped, moving with traffic, braking, or entering the ego vehicle's path. Relative speed affects prediction, tracking, merging, lane changes, and emergency braking. A small 3D localization error may be tolerable for detection, but a poor velocity estimate can produce an incorrect future trajectory and lead to unsafe planning behavior. LiDAR provides accurate 3D geometry of the surrounding scene, but it does not directly measure object velocity. Instead, velocity must be inferred from how an object's estimated position changes across frames. This makes velocity estimation sensitive to noise in the underlying 3D boxes. Small errors in box center, heading, or association can become large velocity errors when divided by the short time interval between keyframes. This is especially problematic for slow-moving vehicles, partially occluded objects, or objects with sparse LiDAR returns, where apparent motion may be dominated by box jitter rather than true physical movement.

Classical approaches estimate velocity using finite differences over object box centers, sometimes followed by temporal smoothing or Kalman filtering. These methods are simple and effective in many cases, but they treat the box trajectory as the primary source of motion information. As a result, they inherit box-position noise and rely on hand-designed assumptions such as constant velocity. Modern 3D detectors often include velocity as an auxiliary prediction head, but velocity is only one part of a much larger detection objective that also includes object classification, center localization, size estimation, and heading prediction. In this work, we focus on a narrower question: given an already detected vehicle and its recent temporal context, can we refine its velocity using LiDAR evidence more accurately than simple kinematic estimates?

We formulate this as a targeted velocity-refinement problem. Given one target `vehicle.car` instance observed over $T$ consecutive keyframes, the goal is to estimate its planar velocity $(v_x, v_y)$ at the final frame. At each keyframe, the model receives a full-scene LiDAR BEV tensor, a target-centered BEV crop, and the target 3D box parameters. The full-scene BEV captures road geometry and surrounding traffic context, while the target-centered crop preserves a higher-resolution view of the object's local LiDAR footprint. The box sequence provides the object's recent position, heading, dimensions, and coarse motion history.

Our approach treats finite-difference velocity as a useful but noisy prior rather than as the final estimate. The model predicts a residual correction to this kinematic prior. This keeps the learning problem focused: the prior already captures the rough direction and speed of the target, while the network learns to correct systematic errors caused by localization noise, sparse returns, partial occlusion, and short-term box jitter. Instead of learning velocity from scratch, the model learns when the kinematic estimate should be trusted and when the surrounding temporal LiDAR evidence suggests a correction.The model combines three complementary signals. The full BEV provides global scene context and nearby traffic structure. The target-centered crop captures object-specific evidence at higher effective resolution. The box trajectory provides compact geometric and kinematic information over time. A temporal Transformer fuses these inputs across the keyframe window, allowing the model to smooth noisy detections, recognize changes in motion, and produce a refined velocity estimate for the final frame.

## 2 Related work

Recent advances in autonomous driving have highlighted the importance of robust multi-object tracking and trajectory prediction in complex urban environments. The nuScenes dataset has emerged as one of the most widely used benchmarks for these tasks because it provides large-scale multimodal data collected using synchronized LiDAR, cameras, radar, GPS, and IMU sensors with full 360° coverage and box-level velocity annotations (Caesar et al., 2020). Its 2,Hz annotated keyframes make it particularly suitable for supervising velocity estimation and short-horizon motion forecasting tasks.

LiDAR-based bird's-eye-view (BEV) methods such as PointPillars (Lang et al., 2019) project point clouds into vertical pillars or BEV grids, enabling efficient 2D convolutional processing of 3D spatial information. Building upon this idea, CenterPoint (Yin et al., 2021) introduced center-based detection and integrated velocity regression for moving objects. Unlike these approaches, which jointly learn object detection and motion estimation, our work assumes the target object track is already available and focuses specifically on refining target velocity estimation.

Temporal BEV approaches for multi-camera perception have also shown strong performance. Methods such as BEVFormer (Li et al., 2022), BEVDet4D (Huang & Huang, 2022), BEVerse (Zhang et al., 2022), and PowerBEV (Li et al., 2023) exploit temporal fusion across sequential frames to improve scene understanding and motion reasoning. In particular, BEVFormer maintains temporal BEV queries derived from multi-view image features and serves as an important baseline for our camera-based detector analysis. Our temporal aggregation strategy is also inspired by the standard Transformer encoder architecture introduced in Attention Is All You Need (Vaswani et al., 2017), which effectively models temporal dependencies across $T{=}4$ frame tokens.

Additionally, recent monocular and multi-camera 3D detectors such as FCOS3D (Wang et al., 2021) and DETR3D (Wang et al., 2022) have demonstrated that camera-only systems often exhibit depth, lateral position, and yaw estimation noise compared to LiDAR-based annotations. Motivated by these observations, our work incorporates a synthetic box-noise model to better study the robustness of velocity refinement under realistic camera-detector errors.

## 3 Goals

This project addresses target-vehicle velocity estimation from short temporal windows of autonomous-driving sensor data. Given consecutive keyframes containing LiDAR point clouds and noisy 3D detection boxes for a target vehicle instance, our objective is to estimate the target vehicle's planar velocity at the final frame of the window.

We formulate the task as a residual velocity prediction problem. A finite-difference estimate $\hat{\mathbf{v}}_{\mathrm{FD}}$ is first computed from the noisy box centers, providing an explicit kinematic prior. The model then

predicts the correction

$$\Delta \mathbf{v} = \mathbf{v}_{\mathrm{gt}} - \hat{\mathbf{v}}_{\mathrm{FD}},$$

rather than the full velocity directly. This design allows the network to refine a noisy but informative motion estimate by using LiDAR-based BEV representations, vehicle-centered crops for scene geometry and target shape, occupancy cues, and temporal box features for motion context.

We evaluate our proposed model against finite-difference velocity estimation and Kalman-filter-based smoothing on the same validation windows. Performance is measured using vector velocity error statistics, including breakdowns by target distance. In addition, we conduct evaluation using detection boxes from a pretrained BEVFormer model to examine how well a model trained with synthetic box noise transfers to real camera-detector outputs.

## 4 Dataset Construction and BEV Representation

### 4.1 Dataset and Instance Filtering

We use the nuScenes v1.0 trainval split (Caesar et al., 2020), which contains 1,000 driving scenes of approximately 20 seconds each with annotated keyframes at 2 Hz. Following the official scene-level split, we use 700 scenes for training and 150 scenes for validation; the held-out test split is not used. Because the split is performed at the scene level, validation samples come from distinct driving sequences rather than from different windows of the same scene.

We restrict the dataset to only moving car instances that are sufficiently observed across consecutive keyframes. Specifically, each retained annotation must have speed between 0.5 and 30.0 m/s, be within 50 m of the ego vehicle, be visible in at least one forward-facing or backward-facing camera, and belong to a track with enough consecutive annotations to form a temporal window. After filtering the trainval split, we obtain 314 scenes with qualifying vehicle instances. Using overlapping sliding windows of length $T = 4$, the validation split contains 1,562 samples.

This filtering is intentionally conservative. Stationary vehicles are excluded because near-zero velocity prediction is comparatively easy and would otherwise dominate the training objective. Extremely fast tracks, short tracks, and poorly observed instances are removed to reduce the effect of annotation outliers and to ensure that each retained instance contributes meaningful temporal context. The camera-visibility constraint aligns the data distribution with the detector-noise setting considered later, although the proposed model itself uses LiDAR-derived BEVs.

### 4.2 LiDAR BEV Tensor Construction

For each keyframe, LiDAR sweeps are transformed into the keyframe ego coordinate system using ego-motion compensation and voxelized into a BEV grid. The BEV covers a $100\,\mathrm{m} \times 100\,\mathrm{m}$ region centered on the ego vehicle at a resolution of 0.2 m per pixel, producing a $500 \times 500$ spatial tensor. We accumulate 10 LiDAR sweeps for the global BEV representation in order to increase point density, especially for distant objects.

Each BEV cell stores 22 channels summarizing the vertical column of LiDAR points: log point density, mean height, maximum height, minimum height, height spread, mean intensity, and 16 height-bin occupancy indicators. These channels provide complementary geometric cues. Density distinguishes occupied from empty cells, while the logarithmic scaling limits the dominance of dense nearby surfaces. Height statistics help separate road surfaces, vehicles, poles, and buildings, while the height-bin occupancy channels retain coarse vertical structure such as low wheel returns and higher roof returns. Ego-motion compensation keeps static scene elements aligned across accumulated sweeps.

### 4.3 Target-Centered Crops

In addition to the full-scene BEV tensor, we extract a target-centered BEV crop for each frame in the temporal window. The crop has spatial size $64 \times 64$ and covers approximately $16\,\mathrm{m} \times 16\,\mathrm{m}$ around the target vehicle. It is rotation-aligned using the target yaw, so that the crop encoder observes vehicles in a consistent orientation across samples.

The crop representation addresses a limitation of the global BEV: a target vehicle occupies only a small fraction of the $500 \times 500$ grid, particularly at moderate or long range. The crop provides a

Table 1: Model architecture summary for the final residual velocity predictor; $T = 4$ keyframes.

| Component | Input shape | Module | Output |
|---|---|---|---|
| Full-scene BEV | $T \times 22 \times 500 \times 500$ | 4-stage depthwise-separable CNN with residual skips and adaptive pooling | $T \times 256$ |
| Target crop | $T \times 22 \times 64 \times 64$ | 3-layer CNN with BatchNorm and adaptive pooling | $T \times 128$ |
| Box/kinematic | $T \times 10$ | Linear projection from box, $\Delta t$, and velocity-prior features | $T \times 64$ |
| Fusion | $T \times (256+128+64)$ | Per-frame concatenation | $T \times 448$ |
| Temporal encoder | $T \times 448$ | 2-layer Transformer, 4 heads, FFN width 512, learned positions | 448 |
| Prediction head | 448 | Linear$(448, 318)$, ReLU, Dropout$(0.35)$, Linear$(318, 2)$ | 2D residual velocity |

higher-focus representation of target occupancy, local shape, and nearby context. Fewer accumulated sweeps are used for the crop than for the global BEV to reduce motion blur around the moving vehicle.

### 4.4 Box Features and Noise Injection

For each frame, the box feature vector describes the target vehicle's 3D state and recent motion. The center coordinates $(x, y, z)$ locate the target in the ego frame, $(l, w, h)$ give the box length, width, and height, $\psi$ gives the target heading, and $\Delta t$ gives the elapsed time from the previous keyframe. We also include a planar velocity prior $\hat{v} = (\hat{v}_x, \hat{v}_y)$, whose components estimate motion in the ego-frame $x$ and $y$ directions. These state, timing, and velocity-prior terms form the model input. The raw finite-difference components $v_x^{FD}$ and $v_y^{FD}$ are also stored, but only for diagnostics and visualization.

During training, synthetic camera-detector noise is applied to the center coordinates $(x, y, z)$, dimensions $(l, w, h)$, and yaw $\psi$ before computing the velocity prior. Thus, the noisy box sequence both supplies the model input and defines the finite-difference prior that the residual model learns to refine.

## 5 Model Architecture and Training Objective

### 5.1 Architecture Overview

Our model, `TemporalVelocityPredictor`, estimates target vehicle velocity from a temporal window of $T = 4$ keyframes. For each frame, the model processes three complementary inputs: the full-scene LiDAR BEV tensor, a target-centered BEV crop, and the corresponding box and kinematic features. These inputs are encoded by separate branches and fused into one frame-level token.

The full-scene BEV branch encodes the $22 \times 500 \times 500$ BEV tensor using a 4-stage depthwise-separable CNN, producing a 256-dimensional scene-context feature. The target crop branch applies a lightweight 3-layer CNN to the $22 \times 64 \times 64$ crop and produces a 128-dimensional target-appearance feature. The box branch projects the 10-dimensional box and kinematic feature vector into a 64-dimensional trajectory feature. Concatenating the three branches yields a 448-dimensional token for each frame.

The BEV encoder uses depthwise-separable blocks to reduce computation on large BEV inputs. Each block applies a depthwise $3 \times 3$ convolution to capture spatial structure within each channel, followed by a pointwise $1 \times 1$ convolution to mix information across channels. Batch normalization, ReLU activations, and residual connections are used to stabilize optimization. We also evaluated larger image-style encoders, including ResNet18 for the full BEV and EfficientNet-B0 for the crop branch, but they did not improve validation performance enough to justify their additional computational cost. This is consistent with the fact that LiDAR BEV tensors differ substantially from RGB images, making ImageNet-style feature hierarchies less directly useful. Table 1 summarizes the final architecture, including the input shapes, encoder modules, and output dimensions for each stage.

### 5.2 Temporal Aggregation

The sequence of frame tokens is passed to a 2-layer Transformer encoder with 4 attention heads, feed-forward width 512, and learned positional embeddings. The positional embeddings are important because the Transformer is otherwise permutation-invariant and cannot distinguish earlier frames from later ones. For the short fixed window length used here, learned embeddings are sufficient and avoid imposing a sinusoidal structure designed for much longer sequences.

The Transformer output tokens are mean-pooled across time to obtain a single window-level representation. We found mean pooling preferable to using only the final token because all four frames can contain useful motion evidence, and pooling reduces sensitivity to a single noisy box, sparse LiDAR return, or partial occlusion. The pooled feature is passed through a prediction head consisting of a linear layer to a hidden size of 318, ReLU, and a final 2D output layer. The complete network has about 4.5 million parameters and runs in 5.82 ms per sample in our inference measurement.

### 5.3 Residual Velocity Prediction

Rather than predicting the absolute velocity directly, the final model predicts a residual correction to a finite-difference velocity prior. Let $\hat{\mathbf{v}}_{\mathrm{FD}}$ denote the finite-difference estimate computed from the noisy box centers. The regression target is

$$\mathbf{r} = \mathbf{v}_{\mathrm{gt}} - \hat{\mathbf{v}}_{\mathrm{FD}},$$

where $\mathbf{v}_{\mathrm{gt}}$ is the ground-truth planar velocity at the final frame of the window.

During training, this residual is normalized using statistics computed from the training set. At inference time, the model output is unnormalized and added back to the finite-difference prior:

$$\hat{\mathbf{v}} = \hat{\mathbf{r}} + \hat{\mathbf{v}}_{\mathrm{FD}}.$$

This formulation gives the network an explicit motion prior while still allowing it to correct systematic errors using LiDAR scene context, target occupancy, and temporal box features. It also reduces the variance of the supervised target compared with full-velocity regression, making the learning problem better conditioned.

### 5.4 Synthetic Detector Noise

Training directly on ground-truth boxes would create a train–test mismatch, since a deployed velocity model would receive boxes from an object detector rather than perfect annotations. To reduce this domain gap, we inject synthetic camera-detector noise into the box parameters during training. Noise is applied to the center position, yaw, dimensions, and height, after which the finite-difference velocity prior is recomputed from the noisy box sequence.

The noise model is designed to mimic common camera-based 3D detection errors. It includes multiplicative log-normal depth noise, heavy-tailed lateral and vertical perturbations, yaw ambiguity including front–back and side-facing flips, and dimension shrinkage toward class-level priors. We use a conservative global noise scale of 0.5.

The synthetic noise model is designed to mimic common camera-based 3D detection errors. We perturb depth using multiplicative log-normal noise with rare Laplace outliers, capturing range-dependent depth uncertainty and occasional projection failures. Lateral position is perturbed with Laplace noise to model heavy-tailed angular localization error, while height uses Laplace noise with a positive bias to reflect ground-plane uncertainty and vertical outliers. Yaw noise is sampled from a mixture concentrated near $0$, $\pi$, and $\pm\pi/2$, with an additional uniform component, capturing front–back ambiguity, side-facing ambiguity, and severe heading errors. Box dimensions are perturbed using a shared log-normal scale with per-axis Laplace shrinkage, modeling detector size priors and regression toward class-level mean dimensions. We use a conservative global noise scale of 0.5 in the final training configuration. Noise is sampled independently for each frame, rather than as a fixed bias shared across the track. This approximates a detector without an explicit temporal tracking prior and deliberately makes the finite-difference velocity estimate noisy. Under this augmentation, the noisy finite-difference baseline obtains a validation mean vector error of 2.123 m/s, giving the residual model a meaningful correction problem.


Figure 1: Training and validation Huber loss for the final residual velocity model over 30 epochs. Both curves decrease and stabilize, indicating stable convergence.

### 5.5 Optimization

We train for 30 epochs using AdamW with learning rate $2 \times 10^{-4}$, cosine learning-rate decay, and gradient clipping at 1.0. The physical batch size is 10, with 4 gradient-accumulation steps, giving an effective batch size of 40. The training loss is Huber loss with $\delta = 0.5$ applied to the normalized residual velocity. Figure 1 shows the training and validation Huber loss over the 30 training epochs. The validation curve decreases along with the training curve and remains stable near the end of training, indicating that the final residual model converges without severe overfitting under the chosen augmentation and regularization settings.

Huber loss is used because rare frames with large localization errors or unusual maneuvers can otherwise dominate the gradient under mean-squared error. For small residual errors, the loss behaves quadratically and encourages precise corrections; for larger errors, it transitions to a linear penalty and is less sensitive to outliers. Bfloat16 automatic mixed precision and `torch.compile` are supported to accelerate convolutional training.

Each checkpoint stores the model weights, optimizer, and scheduler state, epoch, loss history, and all normalization statistics. As a result, inference can be run directly from a checkpoint without requiring a separate training configuration file. All training and inference-time measurements were performed on a single NVIDIA RTX 5000 Ada Generation GPU; reported runtimes therefore reflect this hardware configuration.

## 6 Experimental Results

### 6.1 Validation Results

We report the vector velocity error

$$\|\hat{\mathbf{v}} - \mathbf{v}_{\mathrm{gt}}\|_2$$

in m/s, where $\hat{\mathbf{v}}$ is the predicted planar velocity and $\mathbf{v}_{\mathrm{gt}}$ is the nuScenes ground-truth velocity label. All main validation results use the same 1,562 validation windows constructed from the official nuScenes validation scenes. Table 2 compares the proposed residual model against finite-difference and Kalman-smoothing baselines under the synthetic detector-noise setting. The residual model achieves a mean vector error of 1.145 m/s, a 46.1% reduction relative to finite difference. Its median error is 0.782 m/s, with P90 and P95 errors of 2.295 m/s and 3.236 m/s, respectively.

The higher mean relative to the median indicates a right-skewed error distribution: most predictions are within about 0.8 m/s of the nuScenes velocity label, while a smaller number of difficult cases at range, under occlusion, or during unusual motion dominate the tail.

Figure 2 shows representative validation velocity curves under the synthetic detector-noise setting. The residual model generally follows the ground-truth speed profile across time, while the remaining errors are more pronounced for distant or fast-moving targets, where LiDAR point density and box localization are less reliable.

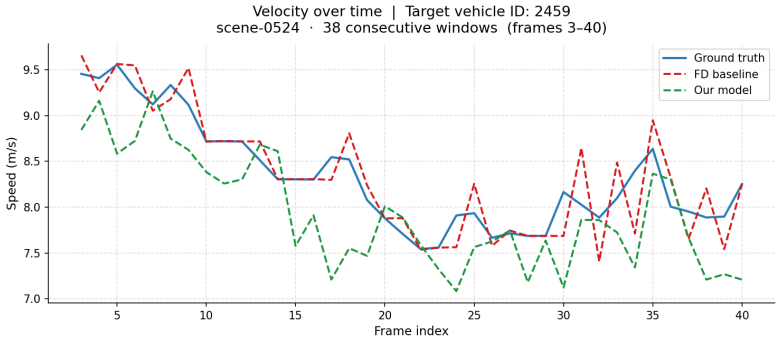


Figure 2: Representative validation velocity curves under the synthetic detector-noise. The residual model tracks the ground-truth velocity trend, with larger deviations occurring in more difficult range or motion regimes.

### 6.2 BEVFormer Transfer Diagnostic

At the strict 2 m matching threshold, 260 of 2,133 target frames matched, yielding 188 complete $T = 4$ windows; unmatched targets often had no nearby predicted car box, suggesting real detector misses for many fast or distant vehicles. Looser 5 m and 10 m thresholds provide more samples but increase the risk of wrong-car matches, so we treat the 2 m setting as the cleanest comparison and use the looser thresholds mainly to examine trends.

Without adaptation, the pretrained residual model does not transfer well to BEVFormer boxes: at the 2 m threshold, it reaches 3.19 m/s mean error on the complete matched windows, compared with 2.25 m/s for BEVFormer's own velocity head. This shows that synthetic noise alone does not fully capture BEVFormer's detector-specific box and velocity distribution. We therefore fine-tune the residual model on matched BEVFormer windows while freezing the LiDAR BEV and crop encoders. Only the box projection, temporal model, and prediction head are updated. This isolates adaptation to BEVFormer's box and velocity statistics while preserving the LiDAR feature extractors learned from the original training setup.

Table 3 reports the held-out fine-tuning diagnostic on five validation instances. On the cleanest 2 m matched-window subset, fine-tuning reduces the residual model's mean error from 2.42 m/s to 1.11 m/s, compared with 1.50 m/s for BEVFormer's velocity head on the same windows. The same trend appears at looser thresholds, although those results should be interpreted more cautiously because wrong-car matches become more likely.

Figure 3 shows representative held-out velocity curves after fine-tuning. The curves illustrate that detector-specific adaptation can help the residual model correct BEVFormer velocity estimates.

## 7 Discussion

The main result is that residual velocity prediction is a better fit for this task than direct velocity regression. The finite-difference prior is noisy, but it usually provides a reasonable estimate of motion direction. The network can therefore focus on correcting systematic errors caused by noisy box centers, sparse LiDAR returns, and target-specific appearance cues rather than learning the full velocity from scratch. The target-centered BEV crop is also important because it preserves local LiDAR evidence around the object that can be diluted in the full-scene BEV representation.

Table 2: Main validation results on noisy boxes generated with synthetic detector noise. Lower mean vector error is better.

| Method | Mean vector error | Notes |
|---|---|---|
| Finite difference | 2.123 m/s | No training; uses the last two noisy box centers |
| Kalman + RTS smoother | 1.336 m/s | Constant-velocity smoothing over the $T$-frame sequence |
| Residual model on FD | **1.145 m/s** | BEV, crop, and box features with a $T = 4$ Transformer |

Table 3: Held-out BEVFormer fine-tuning diagnostic on five instances. Errors are mean vector errors in m/s; lower is better.

| Threshold | Windows | Pretrained | Fine-tuned | BEVFormer |
|---|---|---|---|---|
| 2 m | 46 | 2.42 | **1.11** | 1.50 |
| 5 m | 89 | 3.94 | **1.48** | 3.19 |
| 10 m | 130 | 4.51 | **1.61** | 3.80 |

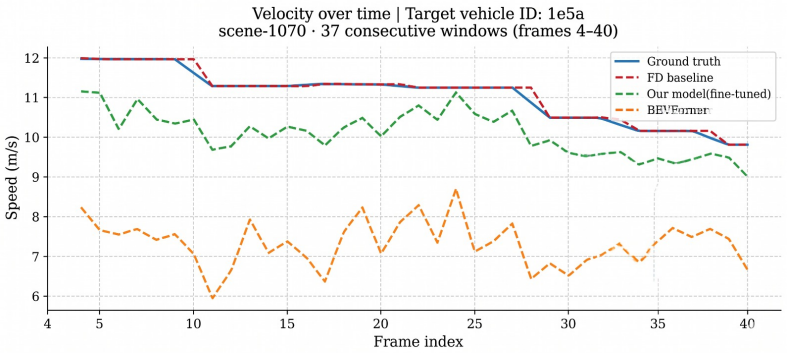


Figure 3: Representative held-out velocity curves using BEVFormer boxes after fine-tuning. The figure illustrates detector-specific adaptation and should be interpreted as a diagnostic rather than a full BEVFormer benchmark.

The approach still has several limitations. First, the model depends on LiDAR input and is trained only on `vehicle.car` instances. Extending the same pipeline to trucks, buses, cyclists, and pedestrians may require different motion priors and different noise models. Second, the method assumes at least $T = 4$ consecutive valid frames for each target. This excludes short tracks and can make the model sensitive to detector dropouts or association failures. Third, the synthetic detector noise reduces the train–test gap, but it may not fully match the error distribution of a real detector. In particular, large depth outliers, wrong-object associations, and temporally correlated detector errors can make the finite-difference prior unreliable.

The dataset filtering choices also narrow the deployment setting. The speed threshold removes parked and very slow vehicles, which improves training stability, but excludes cases such as vehicles creeping into intersections. The front-camera visibility requirement aligns the training data with the camera style detector noise, but it excludes side-only and rear-only traffic. Finally, nuScenes velocity labels are derived from annotated box positions, so the annotation noise can propagate into the target velocity, especially during sharp turns or high-speed motion.

Several extensions remain open. Future work should evaluate Kalman or RTS-smoothed priors, temporal-window length, detector-specific fine-tuning, explicit BEV motion channels, and broader object classes. These directions would clarify how much performance comes from the kinematic prior, how much temporal context is useful, and how well the method transfers beyond the filtered `vehicle.car` setting.

## 8 Conclusion

We implemented a temporal LiDAR-BEV velocity-refinement model for target vehicles on nuScenes. By fusing global BEV, target crops, and noisy box kinematics over $T=4$ frames, and by predicting a residual to finite-difference velocity, the model reaches 1.145 m/s mean vector error on 1,562 validation windows, improving 46.1% over the finite-difference baseline. The BEVFormer diagnostic shows both the value and the limits of synthetic detector noise: direct transfer is poor, but small detector-specific fine-tuning can close much of the gap. Overall, the project supports the central claim that a small dedicated temporal module can substantially improve target-vehicle velocity estimates without requiring a full end-to-end detector. Project repository available at GitHub.

## References

Holger Caesar, Varun Bankiti, Alex H Lang, Sid Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous

driving. In *CVPR*, 2020.

Junjie Huang and Guan Huang. BEVDet4D: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

Peizheng Li, Shuxiao Ding, Xieyuanli Chen, Niklas Hanselmann, Marius Cordts, and Juergen Gall. PowerBEV: A powerful yet lightweight framework for instance prediction in bird's-eye view. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1080–1088, 2023.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European Conference on Computer Vision (ECCV)*, pp. 1–18, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: Fully convolutional one-stage monocular 3d object detection. In *ICCV Workshops*, 2021.

Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. DETR3D: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, 2022.

Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

Yunpeng Zhang, Zheng Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. BEVerse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving. *arXiv preprint arXiv:2205.09743*, 2022.