2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Frame Fusion with Vehicle Motion Prediction for 3D Object Detection

Xirui Li[1], Feng Wang[2], Naiyan Wang[2], and Chao Ma[1,⋆]

***Abstract*— In LiDAR-based 3D detection, history point clouds contain rich temporal information helpful for future prediction. In the same way, history detections should contribute to future detections. In this paper, we propose a detection enhancement method, namely FrameFusion, which improves 3D object detection results by fusing history detection frames. In FrameFusion, we "forward" history frames to the current frame and apply weighted Non-Maximum-Suppression on dense bounding boxes to obtain a fused frame with merged boxes. To "forward" frames, we use vehicle motion models to estimate the future pose of the bounding boxes. Our method is flexible in motion model selection. We explore three motion models in our work and show how the unicycle model and the bicycle model improve turning cases. On Waymo Open Dataset, our FrameFusion method consistently improves the performance of various 3D detectors by about 2.0 vehicle LEVEL 2 APH with negligible latency and slightly enhances the performance of the temporal fusion method MPPNet. We also conduct extensive experiments on motion model selection.**

I. Introduction

LiDAR-based 3D object detection is a crucial component of modern autonomous driving systems. To overcome the inherent sparsity of LiDAR scans, a common practice is to utilize temporal information by incorporating multiple frame point clouds as input. For example, we typically use 10 sweeps on nuScenes dataset [1] and $2-5$ frames on Waymo Open Dataset [2]. The point cloud sequences are usually transformed to the latest timestamp using accurate sensor poses. Then, all the points from multiple sweeps or frames are concatenated together and fed into the network [3], [4]. This approach results in denser static objects and moving objects with "tails", which provide more prominent features for detectors.

With the temporal information, some work additionally adds a planar velocity regressor (i.e., the speed for x and y axes) for downstream applications such as object tracking [3] and trajectory prediction. But the regressor has not been shown to benefit the detection task itself. However, recent research in 3D object tracking [5] suggests that *a simple motion model in 3D space is sufficient to predict the states of objects in future frames*. While this principle is difficult to apply to 2D object detection due to perspective projection, it is straightforward in 3D space. Meanwhile, in 3D detection, when an object is partially occluded, detectors usually predict inaccurate bounding boxes or miss it entirely, even if its previous observations are reliable. Our work is motivated by this insight and seeks to enhance detection performance by exploiting temporal information with motion models.

With the estimated velocities, we can construct a naive motion model to predict the trajectories of detected objects. For weakly detected objects in the current frame, we can supplement them with results forwarded by the motion model from previous frames, potentially improving the detection performance. Importantly, our method does not explicitly rely on object tracking and is solely based on short-term prediction, making it immune to inaccuracies in tracking. By fusing detected objects from the current frame with history frames, our approach enhances detection performance, and we refer to it as **FrameFusion**.

In addition to the basic velocity motion model, our method can switch to other vehicle motion models for handling various practical scenarios. We delve into three motion models in our work. The constant velocity model performs well in most cases but predicts inaccurate poses for turning vehicles. To address its limitation, we introduce two additional motion models from robotic dynamics: the unicycle model and the bicycle model. These models consider the motion constraints of turning vehicles and produce more robust results for such cases, particularly for the bicycle model. It is important to note that our exploration of different motion models is not intended to improve the performance of our method on benchmarks, as turning vehicles constitute only a negligible portion of the dataset. Rather, the flexibility of our method in motion model selection helps our method adapt to specific cases, such as turning vehicles.

Recently, some temporal fusion methods have shown impressive detection performance, such as MPPNet [6], which employs a carefully-designed three-level hierarchical framework to encode and fuse features from multiple frames. It adopts a first-stage 3D detector to generate 3D proposal trajectories and leverages motion information from the proposal trajectories, which is similar to our method. While MPPNet achieves very high detection performance improvement with its elaborate structure, it requires two-stage training and introduces additional latency during inference. In contrast, our method delivers a considerable enhancement with negligible latency cost as a simple postprocessing step.

We evaluate our method on various existing 3D detectors. On Waymo Open Dataset, our method improves LEVEL 2 vehicle APH of CenterPoint [3] by 2.1 and achieves a slight 0.6 enhancement on MPPNet-4frame [6]. Further experiments demonstrate the effectiveness of both unicycle and bicycle models for turning vehicles.

In brief, our contributions are:

[1] Xirui Li and Chao Ma are with MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, Shanghai, China. {lixirui142, chaoma}@sjtu.edu.cn
[2] Feng Wang and Naiyan Wang are with Tusimple, Beijing, China. {feng.wff, winsty}@gmail.com
⋆ Chao Ma is the corresponding author.

- We propose a detection enhancement method that improves LiDAR-based 3D object detection results by a postprocessing fusion step. Unlike past feature-level or point-level fusion methods, our method directly exploits temporal information in the detection frames by fusing boxes from previous frames forwarded with motion models.
- We introduce vehicle motion models into the 3D object detection task. We explore the unicycle model and the bicycle model to alleviate the motion prediction error in the turning cases.
- Evaluation on benchmarks demonstrates that our method can enhance the detection performance of various 3D detection methods with negligible cost. We also conduct extensive experiments on the selection of motion models.

II. Related Work

a) 3D LiDAR Object Detection: 3D LiDAR object detection methods can be generally divided into two categories by point cloud representation: point-based and voxel-based. Point-based methods [7], [8], [9], [10] directly extract point features on raw irregular 3D point clouds supported by PointNet [11] and then perform 3D detection. Voxel-based methods [12], [13], [14], [15], [16], [3] instead divide the points into fixed-size voxels and apply highly efficient 3D or 2D convolutions for detection. In this work, we base most of our experiments on CenterPoint [3] while including a comparison of our method performance on multiple 3D detectors [13], [17], [6].

b) Temporal Information Exploitation: Exploiting temporal information in point cloud sequences proves to be a practical approach. A simple multi-frame point cloud concatenation adopted in some recent work [18], [19], [20], [3], [1] can lead to an improvement compared to single frame input. There are a series of methods diving deeper into modeling the temporal information interaction at the feature level [21], [22], [23], [24], [25], [6]. 3D-Man [21] utilizes a memory bank to store temporal features and aggregate them through an attention network. Offboard3D [24] significantly improves the detection performance in the off-board scenario by utilizing the whole point cloud sequence. MPPNet [6] recently proposed a sophisticated three-level hierarchical framework that incorporates temporal information from a long sequence using proxy points.

Unlike the feature-level fusion methods mentioned earlier, our FrameFusion technique directly fuses temporal information at the frame level. This approach enables our method to leverage the abundant information present in correlated detection boxes across continuous frames. As a post-processing step, FrameFusion is independent of any specific detection model and is much faster than previous feature-level temporal fusion methods (as efficient as a normal NMS step). Recently, MoDAR [26] uses an early fusion to augment LiDAR points with motion forecasting, while our method works as a late fusion.

III. Preliminaries

a) 3D Detection: In 3D point clouds, an object is represented by a 3D bounding box $b = (x, y, z, w, l, h, \phi)$ where (x, y, z) is the center location, (w, l, h) is 3D size and ϕ is the box heading angle. Given an orderless point cloud $\mathcal{P} = \{(x, y, z)_i\}$, 3D object detection aims to predict a set of 3D object bounding boxes $\mathcal{B} = \{b_k\}$ for all the target objects in the given point cloud. We use a concatenated point cloud sequence as input in our temporal setting. To encode temporal information, we append a time indicator t to the point, and thus the input point cloud is $\mathcal{P}_t = \{(x, y, z, t)_i\}$.

b) Motion Model for 3D Bounding Boxes: Among the 3D bounding box attributes, (w, l, h) is constant for a certain object. We regard the others as object pose $p(t) = (x(t), y(t), z(t), \phi(t))$, or p_t in short. It indicates the location and orientation of an object at a certain moment t.

We define the motion model on 3D bounding boxes as the pose derivative with respect to time. A motion model M makes an assumption about the object dynamics with a set of motion parameters θ_M,

$$M(p; \theta_M) = \frac{\partial p}{\partial t}. \tag{1}$$

Given a motion model M and the object motion parameters θ_M, the pose at time t can be estimated by integration,

$$p_t = p_0 + F_M(p_0, t; \theta_M) = p_0 + \int_0^t M(p_t; \theta_M) dt, \tag{2}$$

where p_0 is the initial object pose. We denote the integration function $F_M(p_0, t; \theta_M)$ as the forward model of motion model M. The specific formulation of motion models is introduced in Sec. IV-B.

In this paper, we only model vehicle objects. For vehicles, the z-axis variation is negligible. So we only keep the object pose as $p = (x, y, \phi)$ in the remaining sections.

IV. Method

In this section, we first elaborate on our proposed FrameFusion algorithm, following a description of the three motion models we explored with our method. We explain our model implementation at last.

A. FrameFusion

The key idea of FrameFusion is to fuse the current frame with history frames leveraging motion models for pose estimation. Fig. 1 outlines the steps involved in frame fusion. For one specific frame, we forward history frame boxes to this frame with vehicle motion models. Then dense overlapped boxes are fused via weighted NMS. The use of weighted NMS ensures that dense boxes are fused rather than simply being selected, thereby facilitating the fusion of temporal information from different frames more effectively than standard NMS (Non-Maximum-Suppression).

For a target frame t, let $\{b_t\}$ be the detected bounding box set. We select N history frames with their detected bounding boxes set $\{b_{t-n}, \ldots, b_{t-1}\}$, and then forward the history bounding boxes to frame T with forward model (Equation 2)

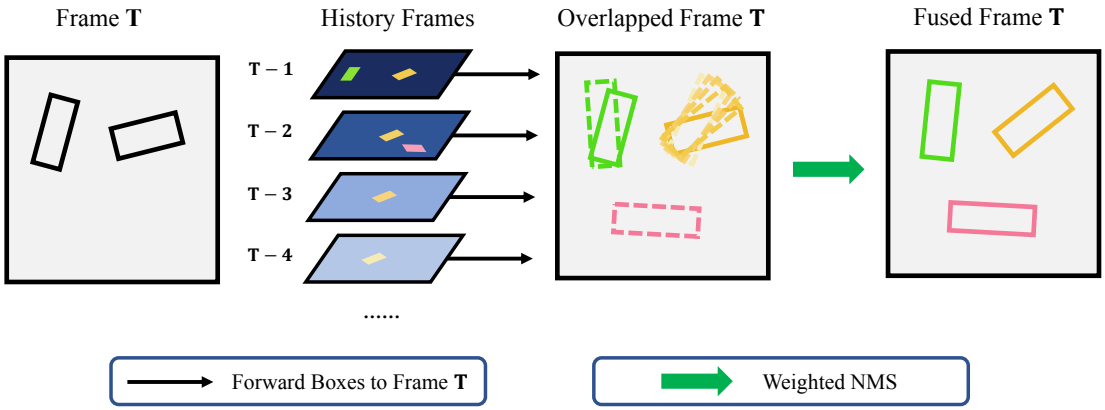


Fig. 1: Frame fusion procedure. To enhance a detection frame, first select a set of history frames. Then, move forward detected boxes from history frames to the latest frame with vehicle motion models (Equ. 2). It generates an overlapped frame with dense boxes. Apply weighted NMS on the dense boxes to get a fused frame with missed detections recovered and box poses refined.

to get $\{b_{t-n}^{for}, \ldots, b_{t-1}^{for}\}$. Since the forwarded pose may not be that accurate, we decay the confidence score by a preset weight decay factor d to reduce their voting weight in the fusion step. The decayed score is computed as $s \cdot d^i$ where s is the original confidence score and i indicates the i_{th} history frame. Intuitively, more distant frames will be less accurate and thus less trusted.

We fuse the dense detection set by applying weighted non-maximum suppression (weighted NMS) [27]. Similar to standard NMS, we first sort the bounding boxes by their scores. For each top-ranked bounding box b, we select a set of boxes overlapped with b with IoU larger than h_{low}. Then bounding boxes whose IoUs with b are larger than h_{high} is voted together as,

$$b_f = \frac{\sum_{b_k \in \mathcal{N}(b;h_{high})} w_k \cdot b_k}{\sum_{b_k \in \mathcal{N}(b;h_{high})} w_k}, \tag{3}$$

where w_k is the (decayed) confidence score of b_k and $\mathcal{N}(b;h_{high}) = \{b_k | IoU(b,b_k) > h_{high}\}$. All the box attributes, as well as the motion parameters and the box scores, are fused as Equation 3. We only reduce the scores of boxes fused by history bounding boxes to avoid affecting the current frame detection. Bounding boxes whose IoUs with b are in the range $[h_{low}, h_{high}]$ are discarded. h_{high} prevents some wrongly estimated forward boxes from affecting current detection and h_{low} allows slightly overlapped boxes to be interpreted as individual proposals. We iterate the above procedure until all bounding boxes are fused together as a set $\mathcal{B}_f$, which is our frame fusion output.

Overall, our FrameFusion technique can be regarded as a post-processing method that can be applied to any continuous detection frames with motion parameters prediction. In the next section, we will further introduce the motion models we used, which determine the forward model and the motion parameters in FrameFusion.

B. Motion Model

In our work, we explore three motion models for vehicle objects. It is important to note that the selection of these models is not aimed at improving benchmark performance. In fact, adopting different motion models achieves similar results on the benchmarks. However, the selection of motion models helps our method adapt to various practical circumstances, such as the turning cases discussed in the following section. We will further demonstrate these models in the following sections.

a) Constant Velocity (CV) Model: Estimating velocities on xy-axes is one of the defined tasks on nuScenes [1] Dataset, and the xy velocities are also provided as ground-truth labels for Waymo Dataset. So it is a common practice to predict the xy velocities if the input is multi-frame or multi-sweep. Its motion model can be simply derived as $V(p,\theta_V) = (v_x, v_y, 0)$, and its forward model is $F_V(p_0,t;\theta_V) = (v_x t, v_y t, 0)$ with $\theta_V = (v_x, v_y)$. As a naive motion model, it performs well with our frame fusion method on benchmarks. However, it fails to describe the motion of turning vehicles. We will introduce the following two vehicle models to fix this issue.

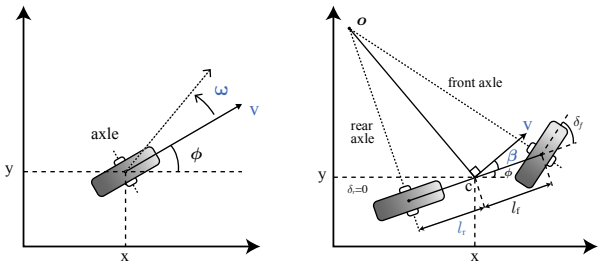
b) Unicycle Model: The unicycle model assumes a single-axle system (Fig. 2a). It can move along the heading angle ϕ with speed V and rotate its heading with angular speed ω. According to Equation 1, we give the unicycle model U,

$$U(p;\theta_U) = \frac{\partial p}{\partial t} = \begin{bmatrix} V\cos\phi \\ V\sin\phi \\ \omega \end{bmatrix}. \tag{4}$$

By integrating over time t, we can estimate pose at time t with the forward model F_U (Equation 2),

$$p_t = p_0 + F_U(p_0,t;\theta_U) = p_0 + \begin{bmatrix} (\sin\phi_t - \sin\phi_0)\frac{V}{\omega} \\ \frac{V}{\omega}(\cos\phi_0 - \cos\phi_t) \\ \omega t \end{bmatrix}, \tag{5}$$

where $\phi_t = \phi_0 + \omega t$ is the heading at time t.



(a) Unicycle model (b) Bicycle model

Fig. 2: Unicycle model and bicycle model diagram. For the unicycle model, the wheel moves with velocity V and rotates with angular velocity ω. Unicycle model parameter $\theta_U = (V, \omega)$. For the bicycle model, the vehicle velocity V deviates from the vehicle heading by a slip angle β. Bicycle model parameter $\theta_\beta = (V, \beta, l_r)$. The lighter part indicates the wheel heading.

The unicycle model provides a compact way to describe a rigid motion with both heading and position variation. However, it is not accurate enough for most vehicles, which usually have two axles instead of a single one. Nonetheless, because of its simplicity, it is also widely used as an approximation in mobile robotics [28], [29].

c) Bicycle Model: Bicycle model represents a system with two axles separated by a distance (Fig. 2b). For simplicity, we assume the rear wheel is parallel with the vehicle heading ϕ. The front wheel is the turning wheel with a rotation angle difference δ_f over ϕ. The distances between the center of mass C and two wheels are l_f, l_r. The velocity V does not align with the heading in the bicycle model. We define the angle between vehicle heading and velocity direction as the tire slip angle β. The bicycle model is

$$B(p;\theta_B) = \frac{\partial p}{\partial t} = \begin{bmatrix} V\cos(\phi+\beta) \\ V\sin(\phi+\beta) \\ \frac{V\sin\beta}{l_r} \end{bmatrix}. \tag{6}$$

Similar to the unicycle model, we have the bicycle forward model by Equation 2,

$$p_t = p_0 + \begin{bmatrix} \frac{l_r}{\sin\beta}(\sin(\phi_t+\beta) - \sin(\phi_0+\beta)) \\ \frac{l_r}{\sin\beta}(\cos(\phi_0+\beta) - \cos(\phi_t+\beta)) \\ \frac{V\sin\beta}{l_r}t \end{bmatrix}, \tag{7}$$

where $\phi_t = \phi_0 + \frac{V\sin\beta}{l_r}t$ is the heading at time t.

Compared with the more advanced Ackermann model [30], the bicycle model trades off between accuracy and simplicity. We will show in experiments that the bicycle model is accurate enough to model the turning vehicles.

d) Discussion: In this section, we briefly discuss the pros and cons of the three aforementioned motion models. A naive observation is that these motion models are the same when the vehicle moves straight. We can set ω and β in the unicycle and bicycle model to zero and get the same forward model as the constant velocity model. So we only discuss the turning cases when comparing the motion models.

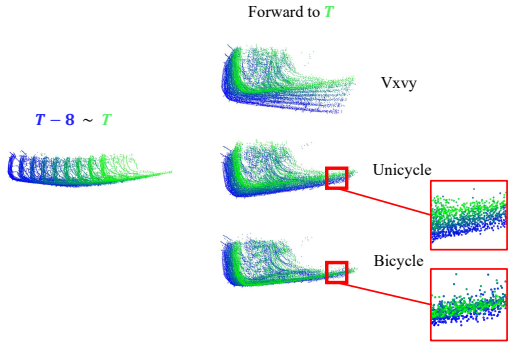


Fig. 3: Forwarding history point cloud into the current frame with motion model.

Fig. 3 shows a turning vehicle point cloud sequence in eight frames. We estimate its motion parameter at each timestamp with adjacent box poses and forward all the point clouds to the latest frame T. We can see the constant velocity model diverges in the heading orientation as it estimates incorrect velocity directions. In comparison, the unicycle model shows significantly better motion estimation performance with a better-aligned point cloud. However, it still shifts a little that can be figured out at the border. The bicycle model generates an almost perfect matching. We can hardly tell points from different frames, presenting its high accuracy in vehicle motion prediction.

Though the bicycle model result is satisfying, we observe a short blue tail at the rear. It actually suggests an acceleration in a short period. As we do not introduce second-order derivatives such as acceleration in our motion models for model simplicity, it may cause inevitable errors in long-term motion estimation. It also indicates the rationality behind the confidence decay mechanism in our frame fusion method.

C. Model Implementation

To extract motion information from point cloud sequences, we augment existing 3D detectors with additional motion parameter regressors. For any models mentioned in our work, multi-frame input fusion [13], [20], [1] is adopted to integrate temporal information into model input for motion parameter prediction.

In common autonomous driving datasets, most vehicle objects have zero or near zero angular velocity, which significantly hinders the learning of angular velocity prediction. We employ a balanced loss function that equalizes the influence of turning and non-turning objects to resolve the issue. The angular velocity regression loss is first averaged in each class and then summed up to achieve class balance.

Before training, we pre-process the datasets to generate ground-truth motion parameter labels with inverse motion models. Given a pair of poses (p_0, p_t), an inverse model estimates the motion parameter θ_M. For frame t, its motion parameters are estimated using object poses in adjacent frames $t-1$ and $t+1$ as (p_0, p_t).

TABLE I: Vehicle detection enhancement over various 3D detectors on Waymo validation set. + means performance after applying frame fusion. All methods take **4-frame** input and constant velocity motion model. Latency is measured on a Nvidia 3090 GPU in milliseconds per frame. $*$: we report the additional parameter number and latency introduced by themselves for our methods and MPPNet. †: improve over the same CenterPoint model.

Method	AP/APH↑ (VEH)		#Params*	Latency*
	Level 1	Level 2		
SECOND [13]	71.9/71.2	63.9/63.3	5.3M	137ms
CenterPoint [3]	76.7/76.1	69.1/68.6	7.8M	74ms
PV-RCNN [17]	78.1/77.6	70.1/69.6	13.5M	1085ms
SECOND+	73.6/73.0	65.6/65.1	-	+3ms
†CenterPoint+	78.7/78.2	71.2/70.7	-	+3ms
†MPPNet-4F [6]	81.5/81.0	74.0/73.6	+7.9M	+301ms
PV-RCNN+	80.0/79.4	72.1/71.6	-	+3ms
MPPNet-4F+	82.0/81.6	74.6/74.2	-	+3ms

TABLE II: 3D vehicle detection improvements on Waymo validation set with different motion models. + means performance after applying frame fusion. All models use **4-frame** input. CenterPoint [3]: backbone model without motion parameter regression. Method names indicate the motion model used.

Method	VEH AP↑		VEH APH↑	
	Level 1	Level 2	Level 1	Level 2
CenterPoint	76.08	68.17	75.55	67.68
Constant Velocity	75.64	67.74	75.11	67.26
Unicycle	75.38	67.46	74.84	66.98
Bicycle	75.56	67.62	75.02	67.13
Constant Velocity+	77.82	70.14	77.33	69.68
Unicycle+	77.65	69.93	77.14	69.46
Bicycle+	77.84	70.11	77.33	69.65

V. Experiments

A. Experimental Settings

a) Waymo Open Dataset: Waymo Open Dataset [2] is a large-scale, diverse 3D detection dataset, including three classes: vehicle, bicycle, and pedestrian. The official 3D detection evaluation metrics are standard 3D bounding box mean average precision (mAP) [31] and mAP weighted by heading accuracy (mAPH). As we solely consider vehicle class in evaluation, tables show AP and APH on vehicle class. The annotated objects are split into two difficulty levels: LEVEL 1 (More than five Lidar points in bbox), and more difficult LEVEL 2 (at least one Lidar point in bbox).

b) Implementation Details: We build our models on the CenterPoint-Voxel model with corresponding motion parameter regressors, referred to as Constant Velocity (CV), Unicycle, and Bicycle model. Our implementation is based on 3D detection codebases MMDetection3D [32] and OpenPCDet [33].

All models use multi-frame point fusion as input ($[-4, 0]$). We train CenterPoint-based models for 12 epochs with AdamW optimizer and a cyclic learning rate scheduler with a maximum learning rate 1e-3 and a weight decay 0.01. Models in Table I are trained with the default configuration. For frame fusion, we set the frame fusion range $N = 4$ by default (fuse 4 previous frames).

B. Main Results

a) 3D Detection Enhancement: FrameFusion improves the 3D detection performance by fusing history detection frames with the current frame. On the Waymo dataset, we demonstrate that our method can enhance the performance of existing 3D detection methods by applying a 4-frame fusion to their results (Table I). When applied to the 3D detectors SECOND [13], CenterPoint [3], and PV-RCNN [17], FrameFusion achieves significant enhancements in the vehicle 3D APH (level 2) by $1.8, 2.1, 2.0$ with negligible latency costs (3ms). To ensure a fair comparison with MPPNet, we use its 4-frame version, which aligns with our 4-frame fusion setting, and take CenterPoint evaluated in the same table as its first-stage model. Our method achieves a 42% $(2.1/5.0)$ enhancement obtained by MPPNet on the same CenterPoint model without the retraining step, model parameters, or high latency introduced by MPPNet. Notably, our method can even improve MPPNet's performance slightly by 0.6 (MPPNet-4F+), suggesting that our method is complementary to other temporal fusion methods to some extent.

We compare the frame fusion performance for 3D detection enhancement using different motion models in (Table II). Note the motion model names in the table represent the CenterPoint base model with corresponding motion parameter regression. On the Waymo validation set, frame fusion shows similar enhancement for Constant Velocity, Unicycle, and Bicycle model, with an increase of $+2.42, +2.48, +2.52$ in vehicle level 2 APH. It indicates that FrameFusion works consistently with different motion models. In the next section, we will analyze the differences between these models in turning cases.

b) Turning Case Evaluation: As discussed in Sec. IV-B.0.d, motion models perform identically when the angular velocity is zero. In Waymo, the ratio of turning vehicles is about 0.05. Thus previous results do not fully reflect the difference among motion models. We further explore the motion model features in the turning case.

Table III shows the evaluation results on the turning subset (velocity $V > 5$ and turning radius $R < 25$). Constant Velocity model decreases the performance on turning vehicles (about -3.78 level 2 APH) after frame fusion, while Unicycle model and Bicycle model still improve. Bicycle model shows its superiority with the best performance on the turning case.

The preceding experiments confirm that our method can improve the hard cases such as turning vehicles by selecting a proper motion model. If we aim to consistently improve detection across various motion states, which is crucial in practical scenarios, the unicycle or bicycle model should be prioritized due to its superior precision in vehicle modeling.

TABLE III: 3D vehicle detection improvements evaluated on turning subset of Waymo validation set with different motion models. The method name indicates the motion model used.

Method	VEH AP↑		VEH APH↑	
	Level 1	Level 2	Level 1	Level 2
Constant Velocity	81.36	80.67	80.80	80.12
Unicycle	80.85	80.35	80.17	79.67
Bicycle	81.62	80.93	81.13	80.44
Constant Velocity+	78.48	77.74	77.07	76.34
Unicycle+	82.39	81.67	81.85	81.14
Bicycle+	**82.96**	**82.29**	**82.34**	**81.67**

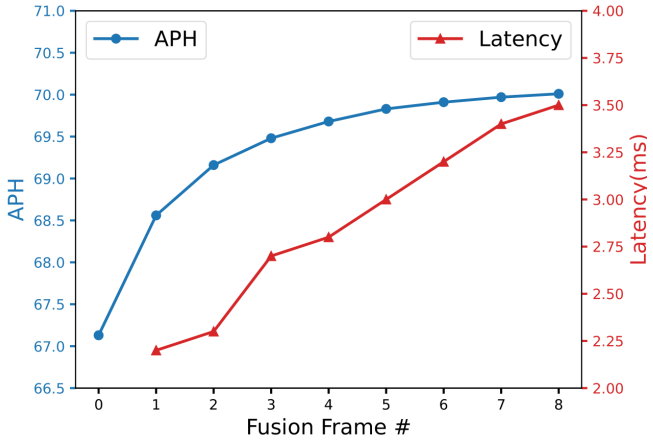


Fig. 4: Detection Performance with various numbers of fusion frames evaluated with CenterPoint-Bicycle model. Blue: Vehicle level 2 APH on Waymo Validation set. Red: Latency in milliseconds per frame.

C. Ablation Studies

a) Analysis of Fusion Frame Number: We analyze the performance and efficiency of our method concerning the fused frame number (Fig. 4). As more frames are fused, the performance increment soon converges, while the latency increases linearly. This can be attributed to the fact that, as the fusion number grows, less additional temporal information is integrated from distant frames. Furthermore, as the time interval grows, the motion prediction becomes less accurate, and the confidence decay strategy causes the voting weight of forwarded bounding boxes to decay rapidly. Therefore, fusing a distant frame may hardly enhance current detection. In practical scenarios, fusing 2 to 4 frames can provide sufficient improvement.

b) Analysis of Vehicle Motion States: To further analyze the performance of the motion models on different motion states, we evaluated the frame fusion enhancements separately for stationary, straight, and turning motion vehicles (Table IV). We found that the motion models performed almost identically when vehicles were stationary or moving straight, indicating that they were equally effective in these scenarios. However, we observed a decreasing trend in performance from stationary to straight to turning motion, which suggests a decrease in the capability of the motion models and the accuracy of the motion parameter predictions as the motion becomes more complex. The turning cases are discussed in more detail in section V-B.b.

TABLE IV: Compare frame fusion enhancement on different motion states. Show vehicle level 2 APH detection increment after frame fusion. Stationary, straight, turning vehicles account for 0.63, 0.31, 0.05 in Waymo validation set.

Method	All	Stationary	Straight	Turning
Constant Velocity	+2.42	+2.95	+1.24	-3.40
Unicycle	+2.48	+2.95	+1.39	+0.83
Bicycle	+2.52	+3.02	+1.20	+1.28

TABLE V: Frame fusion ablation on CenterPoint-Bicycle model. Show vehicle level 2 APH on Waymo validation set. Above: Default frame fusion result. Below: Ablation settings.

Method	APH(Δ APH)	Latency
CenterPoint	67.1	-
CenterPoint+(Default)	69.7(+2.6)	2.8ms
Circle NMS	68.2(+1.1)	3.0ms
Generate new detection	67.7(+0.6)	2.8ms
Refine existing detection	69.1(+2.0)	2.8ms
Refine bounding box	68.6(+1.5)	2.8ms
Refine score	67.7(+0.6)	2.8ms

c) Frame Fusion Ablation: We present the results of frame fusion ablations in Table V. The replacement of weighted NMS with standard circular NMS results in the selection of new boxes instead of fusion, leading to a significant drop in performance. Subsequent ablations aim to quantify the sources of performance improvement, categorizing them based on whether they involve generating a new detection through fusing history boxes or refining an existing detection in the current frame. Refinement is further categorized into refining bounding box attributes or scores. Fine-grained refinement of existing detections emerges as the primary source of improvement. Latency analysis demonstrates that our method maintains efficiency comparable to a GPU-implemented NMS postprocessing step.

VI. Conclusion

In this paper, we propose a 3D object detection enhancement method FrameFusion, fusing a detection frame with a series of history frames forwarded by motion models. We introduce vehicle motion models, including the unicycle model and the bicycle model to better capture the motion of turning vehicles. Our approach is simple and highly efficient, making it suitable for practical applications in 3D detection systems. We hope that our work will inspire further improvements and applications of the method in the future.

Acknowledgments. This work was supported in part by NSFC (62376156, 62322113), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[2] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[3] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.

[4] R. Ge, Z. Ding, Y. Hu, Y. Wang, S. Chen, L. Huang, and Y. Li, "Afdet: Anchor free one stage 3d object detection," *arXiv preprint arXiv:2006.12671*, 2020.

[5] Q. Wang, Y. Chen, Z. Pang, N. Wang, and Z. Zhang, "Immortal tracker: Tracklet never dies," *arXiv preprint arXiv:2111.13672*, 2021.

[6] X. Chen, S. Shi, B. Zhu, K. C. Cheung, H. Xu, and H. Li, "Mppnet: Multi-frame feature intertwining with proxy points for 3d temporal object detection," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII*. Springer, 2022, pp. 680–697.

[7] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[8] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 040–11 048.

[9] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1951–1960.

[10] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9277–9286.

[11] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[12] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[13] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[14] B. Yang, M. Liang, and R. Urtasun, "Hdnet: Exploiting hd maps for 3d object detection," in *Conference on Robot Learning*. PMLR, 2018, pp. 146–155.

[15] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[16] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[17] S. Shi, C. Guo, J. Yang, and H. Li, "Pv-rcnn: The top-performing lidar-only solutions for 3d detection/3d tracking/domain adaptation of waymo open dataset challenges," *arXiv preprint arXiv:2008.12599*, 2020.

[18] Y. Hu, Z. Ding, R. Ge, W. Shao, L. Huang, K. Li, and Q. Liu, "Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 969–979.

[19] P. Sun, W. Wang, Y. Chai, G. Elsayed, A. Bewley, X. Zhang, C. Sminchisescu, and D. Anguelov, "Rsn: Range sparse net for efficient, accurate lidar 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5725–5734.

[20] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova, "4d-net for learned multi-modal alignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 435–15 445.

[21] Z. Yang, Y. Zhou, Z. Chen, and J. Ngiam, "3d-man: 3d multi-frame attention network for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1863–1872.

[22] R. Huang, W. Zhang, A. Kundu, C. Pantofaru, D. A. Ross, T. Funkhouser, and A. Fathi, "An lstm approach to temporal 3d object detection in lidar point clouds," in *European Conference on Computer Vision*, 2020, pp. 266–282.

[23] J. Yin, J. Shen, C. Guan, D. Zhou, and R. Yang, "Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 495–11 504.

[24] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, "Offboard 3d object detection from point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6134–6144.

[25] C. Luo, X. Yang, and A. Yuille, "Exploring simple 3d multi-object tracking for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 488–10 497.

[26] Y. Li, C. R. Qi, Y. Zhou, C. Liu, and D. Anguelov, "Modar: Using motion forecasting for 3d object detection in point cloud sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 9329–9339.

[27] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[28] T.-C. Lee, K.-T. Song, C.-H. Lee, and C.-C. Teng, "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *IEEE transactions on control systems technology*, vol. 9, no. 2, pp. 305–318, 2001.

[29] J. Ghommam, H. Mehrjerdi, M. Saad, and F. Mnif, "Formation path following control of unicycle-type mobile robots," *Robotics and Autonomous Systems*, vol. 58, no. 5, pp. 727–736, 2010.

[30] A. Mueller, "Modern robotics: Mechanics, planning, and control [bookshelf]," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 100–102, 2019.

[31] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[32] M. Contributors, "Mmdetection3d: Openmmlab next-generation platform for general 3d object detection," 2020.

[33] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.