Real-time Stereo-based 3D Object Detection for Streaming Perception

Changcai Li[1,2] **Zonghua Gu**[3] **Gang Chen**[1,2,*] **Libo Huang**[4]
Wei Zhang[2] **Huihui Zhou**[2]
[1]Sun Yat-sen University [2]Pengcheng Laboratory
[3]Hofstra University [4]National University of Defense Technology
lichc5@mail2.sysu.edu.cn zonghua.gu@hofstra.edu cheng83@mail.sysu.edu.cn
libohuang@nudt.edu.cn zhangwei1213052@126.com zhouhh@pcl.ac.cn

Abstract

The ability to promptly respond to environmental changes is crucial for the perception system of autonomous driving. Recently, a new task called streaming perception was proposed. It jointly evaluate the latency and accuracy into a single metric for video online perception. In this work, we introduce StreamDSGN, the first real-time stereo-based 3D object detection framework designed for streaming perception. StreamDSGN is an end-to-end framework that directly predicts the 3D properties of objects in the next moment by leveraging historical information, thereby alleviating the accuracy degradation of streaming perception. Further, StreamDSGN applies three strategies to enhance the perception accuracy: (1) A feature-flow-based fusion method, which generates a pseudo-next feature at the current moment to address the misalignment issue between feature and ground truth. (2) An extra regression loss for explicit supervision of object motion consistency in consecutive frames. (3) A large kernel backbone with a large receptive field for effectively capturing long-range spatial contextual features caused by changes in object positions. Experiments on the KITTI Tracking dataset show that, compared with the strong baseline, StreamDSGN significantly improves the streaming average precision by up to 4.33%. Our code is available at https://github.com/weiyangdaren/streamDSGN-pytorch.

1 Introduction

Stereo-based 3D object detection [53, 6, 7] presents a notable advantage of low-cost in the deployment compared to LiDAR-based methods. However, existing stereo-based methods encounter challenges in delivering timely responses due to the intensive computations. Although there have been efforts towards lightweight approaches [35, 38], the latency of the inference process is still non-negligible. This is because the latency will lead to the misalignment between the prediction of the latest frame and the real-time changing scene, as shown in Figure 1.

Recently, a new metric [30, 52] named streaming accuracy was proposed for real-time online perception. Different from previous offline metrics [10, 3, 50] which only emphasize the detection accuracy, it simultaneously considers both accuracy and latency in the performance evaluation. With this metric, the model is forced to evaluate the processed world state against the prediction of the received frame. Hence, many high-accuracy but low-efficiency detectors [33, 17] will result in significant performance degradation for real-time perception applications.

*Corresponding author

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

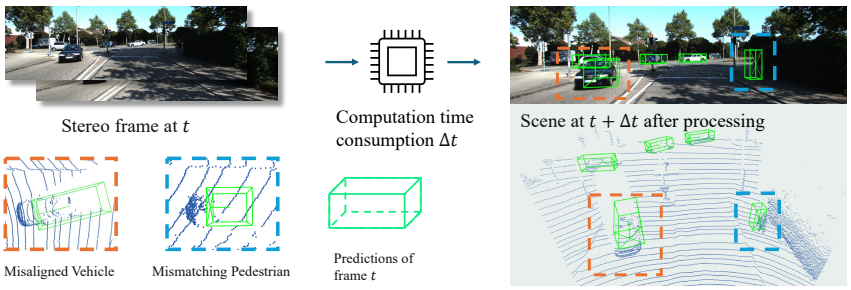


Figure 1: In the context of online streaming perception, the environment changes during inference.

Challenges. A widely recognized solution for addressing streaming perception is to combine historical features and directly predict future outcomes at the current moment within a real-time framework [59, 28, 15]. This is because when the model's inference speed exceeds the input frame rate, the metric of streaming accuracy consistently matches and evaluates the results of the current frame with the ground truth of the next frame. However, several challenges arise in the implementation of this solution. As delineated in Figure 2, these challenges include the following three specific aspects:

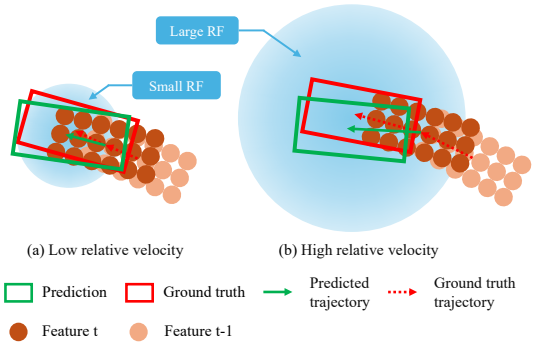


Figure 2: Illustration of the challenges.

The **first challenge** is the misalignment between the future supervisory signals and the current features. It is observed that for moving objects, their ground truth positions in the next frame at time step $t + 1$ (depicted by the red bounding box) consistently differ from their current positions at time step t. In such cases, the model needs to learn geometric information about the foreground from distant historical features. Furthermore, this misalignment will be exacerbated as the increasing disparity in relative velocities between the ego vehicle and the surrounding vehicles.

The **second challenge** lies in the implicit supervision when only using the ground truth of a single future frame. It is observed that motion objects with diverse relative velocities exhibit distinct trajectory lengths (depicted by the red arrow). Previous works [59, 28, 15] lack explicit trajectory supervision, requiring models to spontaneously learn various object motion offsets within latent space. Further, due to the complexity of 3D object detection tasks, a single regression supervision method may not be sufficient to ensure accurate detection of objects moving at different speeds.[2]

The **third challenge** arises from the effective utilization of context information embedded in the combined features. Due to the generally lower frame rates of 3D datasets compared to 2D ones, for example, the KITTI Tracking dataset [10] has a frame rate of only 10Hz while Argoverse-HD [30] has 30Hz. Therefore, in 3D datasets, larger intervals will result in larger spatial distances between the same objects in adjacent frames. At this point, feature extractor with small receptive field (RF) (depicted by the cyan areas) is insufficient to capture information from distant historical features.

Our solution. In this paper, we propose StreamDSGN, a streaming stereo-based 3D detector based on the advanced DSGN++ [7] architecture. It directly predicts the 3D properties of the **next** frame by leveraging the fusion of bird's-eye view (BEV) features from both the current and historical frames. Further, StreamDSGN combines three strategies to tackle the above-mentioned challenges as follows:

- *Feature-Flow Fusion (FFF).* To address the **first challenge** of misalignment between features and ground truth, we introduce a novel fusion method based on feature flow. It computes the feature flow between the current frame and the previous frame through similarity matching, and subsequently warps the current feature into the BEV coordinates of the next frame according to the flow map. This yields pseudo-next features aligned with the ground truth of the next frame.

[2]In offline perception, this issue does not arise, since the ground truth is synchronized with each input frame.

- *Motion Consistency Loss function (MCL).* To address the **second challenge** regarding implicit supervision, we establish additional explicit supervision based on motion consistency between adjacent frames. Specifically, we utilize the historical ground truth trajectories to supervise the predicted trajectory, guiding the model to recognize the offset magnitude and direction of object positions in the next frame.
- *Large Kernel BEV Backbone (LKBB).* To address the **third challenge** involving large-span contextual information, we adopt a large receptive field backbone to extract fused features. This structure follows existing research on large kernel convolutions [12, 25], aiming to enhance the model's capacity to capture long-range dependencies.

We perform streaming simulations [30, 52] and conduct comprehensive experiments on the KITTI Tracking dataset [10], showing significant improvements in the streaming perception task. To the best of our knowledge, our work represents the **first** endeavor to explore the implementation of 3D object detection for streaming perception.

2 Related Works

Stereo-based 3D object detection. The methods in this field can be broadly classified into three categories as follows:

1) *2D-detection-based methods.* These methods [31, 49, 56, 44, 42] typically begin by feeding stereo image pairs into a 2D detector or an instance segmentation task to generate prior information before performing 3D object detection. The works in [35, 43] implement a single-stage stereo detection framework. Furthermore, Wu et al. [55] employ a semi-supervised method to enhance the foundational model by adopting plentiful unannotated images.

2) *Pseudo-LiDAR-based methods.* These methods [53, 62, 45, 9] employ the stereo matching task to obtain depth information from the scene and then transform it into a data structure resembling a LiDAR point cloud. Further, Li et al. [27] leverage the confidence derived from semantic segmentation to enhance the pseudo-LiDAR. The works in [26, 38] adopt a binary neural network (BNN) to quantize the disparity estimator for significantly reducing the computational overhead.

3) *Geometric-volume-based methods.* The methods in [6, 54, 13, 36, 7] adopt a representation that encodes stereo features derived from Siamese networks into differentiable geometric volumes. Further, Wang et al. [54] employ 3D occupancy to directly supervise the depth estimation model. The works in [13, 36] improve the performance by leveraging knowledge distillation. Chen et al. [7] perform depth-wise reconstruction of the volume in geometric modeling to alleviate the bottleneck of 2D to 3D information propagation. These methods represent state-of-the-art (SOTA) for stereo-based 3D object detection, and our work explores the application of the work in [7] for streaming perception.

Streaming perception. The concept of streaming perception was initially introduced in [30], which evaluates streaming average precision (sAP) while accounting for latency considerations. With this metric, non-real-time detectors [33, 17] will lead to great performance degradation since they unavoidably miss some intermediate frames. Therefore, Li et al. [30] further propose to address this issue through decision-theoretic scheduling, asynchronous tracking [1], and future prediction [22].

The work in [46] corroborates the findings of [30] and extends their analysis to object tracking. Ghosh et al. [11] propose a learned approximate execution framework to balance accuracy and latency implicitly. Instead of seeking better trade-offs or enhancing base detectors, StreamYOLO [59] simplifies streaming perception to an end-to-end task of "predicting the next frame" with a real-time detector. Based on this principle [59], LongShortNet [28] and DAMO-StreamNet [15] improve the streaming accuracy by leveraging longer temporal fusion and knowledge distillation respectively. The work in [52] expands the nuScenes dataset [3] and introduces a 3D benchmark tailored for streaming perception assignments. To the best of our knowledge, there are currently no existing applications of 3D perception algorithms for this task.

Temporal 3D object detection. These methods can be broadly classified into three categories: (1) The *LiDAR sequences-based methods* [60, 61, 63, 39] complement the 3D shape within a single frame by integrating historical features through temporal modeling. (2) The *streaming data-based methods* [14, 8, 5] treat each LiDAR packet as an independent sample without requiring a complete

sweeping by the LiDAR. (3) The *Video-based methods* [18, 2] extend the image-based methods by tracking and fusing the same objects across different frames. Note that the essential difference between streaming perception and these methods is the misalignment of ground truth and features.

3 Our Approaches

3.1 Base detector

We choose the recently proposed DSGN++ [7] as the base detector in our study. Due to its high latency, our first goal is to optimize its pipeline to ensure the completion of inference for the current frame before the arrival of the next frame.

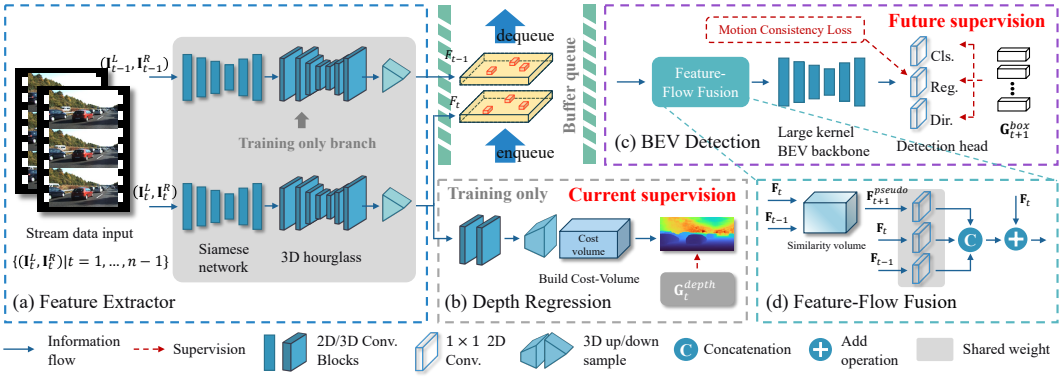


Figure 3: The architecture of StreamDSGN pipeline. (a) The feature extractor retrieves features from streaming stereo image pairs $\{(\mathbf{I}_t^L, \mathbf{I}_t^R)|t = 1, ..., n-1\}$ and flattens them into BEV features. (b) The depth regression component utilizes $\mathbf{G}_t^{depth}$ as the supervision. (c) The BEV detector predicts the object state of the next moment by merging features from the current and previous frames. (d) The Feature-Flow Fusion generates a pseudo-next feature $\mathbf{F}_{t+1}^{pseudo}$ by extrapolating from past features and then concatenates it with the existing historical feature set $\{\mathbf{F}_t, \mathbf{F}_{t-1}\}$.

The optimization strategies are detailed in Appendix A.1. These strategies aim to ensure real-time processing of the detector without significantly sacrificing the offline accuracy. Experiments in Appendix A.2 show that the optimized model incurs only minor losses in offline accuracy while achieving a significant reduction in latency, decreasing from 263.33ms to 80.71ms (with a frame rate of 10Hz for the KITTI Tracking dataset [10]).

Next, we extend the optimized model into a streaming data detection framework, as illustrated in Figure 3. During the training phase, we take the consecutive stereo images $\left(\mathbf{I}_{t-1}^L, \mathbf{I}_{t-1}^R, \mathbf{I}_t^L, \mathbf{I}_t^R\right)$ as input and directly predict the result $\mathbf{P}_{t+1}^{box}$ of the next frame. This data organization allows for random shuffling of training samples. Note that in streaming perception tasks, the input data cannot include time step $t+1$ since we cannot access future frames at the current moment in real-world scenarios. During inference, as the input frames are continuous, we only need to input the current frame and store its intermediate feature in a buffer queue for fusion with the next input.

Different from previous 2D approaches [59, 28, 15], our temporal fusion is executed before the BEV backbone instead of preceding the detection head. This is because we believe that the shallow convolutional layers in the detection head lack an effective receptive field to extract large-span contextual information from the fused features. We validate this conjecture in the experiments presented in Appendix A.4.

3.2 Feature-Flow Fusion (FFF)

Different from fusion methods for temporal 3D object detection [19, 32, 34, 58], which aim to complement the geometric shape of objects, our Feature-Flow Fusion (FFF) is designed to warp current feature to align with the next ground truth. The pipeline of FFF is shown in Figure 4, it utilizes historical changes to accomplish the warping based on motion consistency.

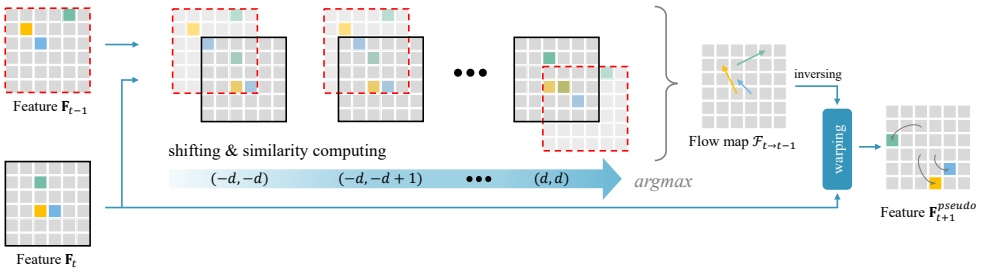


Figure 4: A toy example of pseudo-next feature generation.

Calculation of Feature Flow. Recall that our input features are $\mathbf{F}_t, \mathbf{F}_{t-1} \in \mathbb{R}^{H\times W\times C}$. The first step of FFF involves computing the feature flow between the $\mathbf{F}_t$ and $\mathbf{F}_{t-1}$. This process is similar to the computation of similarity volume in optical flow [20, 48]. We assume that the search space for the computing is discrete and rectangular. Specially, we define a shift indices set $\mathcal{S}$ as:

$$\mathcal{S} = \{(-d,-d),(-d,-d+1),...,(0,0),...,(d,d-1),(d,d)\} \tag{1}$$

where d is the maximal displacement. Then we shift the elements $\{\mathbf{c}_{t-1}(i,j)|i=0,...,H-1;j=0,...,W-1\}$ in $\mathbf{F}_{t-1}$ according to each shift index in $\mathcal{S}$, and match the shifted $\mathbf{F}_{t-1}$ with $\mathbf{F}_t$ to compute the cosine similarity $\mathbf{s}(i,j)$ of each matched elements. When considering a single match with a shift index of $(\mathbf{u}_s,\mathbf{v}_s)$, the similarity at pixel index $(\mathbf{u},\mathbf{v})$ can be defined as:

$$\mathbf{s}(\mathbf{u},\mathbf{v}) = \frac{1}{\|\mathbf{c}_{t-1}(\mathbf{u}+\mathbf{u}_s,\mathbf{v}+\mathbf{v}_s)\|\|\mathbf{c}_t(\mathbf{u},\mathbf{v})\|}\left((\mathbf{c}_{t-1}(\mathbf{u}+\mathbf{u}_s,\mathbf{v}+\mathbf{v}_s))^{\intercal}\,\mathbf{c}_t(\mathbf{u},\mathbf{v})\right) \tag{2}$$

where $\intercal$ represents the transpose operator. Stacking the whole similarity matrix $\mathcal{C} = \{\mathbf{s}(i,j)|i=0,...,H-1;j=0,...,W-1\}$ obtained from different shift indices along a new dimension, we can yield the similarity volume $\mathcal{V} \in \mathbb{R}^{H\times W\times D}$, where $D=(2d+1)^2$.

Next, we ascertain the feature flow $\mathcal{F}_{t\to t-1} \in \mathbb{R}^{H\times W\times 2}$ between $\mathbf{F}_t$ and $\mathbf{F}_{t-1}$ by identifying the index with the maximum similarity. This process can be formulated as:

$$\mathcal{F}_{t\to t-1} = gather\left(\mathcal{S}, \arg\max_D \mathcal{V}\right) \tag{3}$$

Note that the acquisition of $\mathcal{F}_{t\to t-1}$ does not require any learnable parameters. In our implementation, we accelerate the above process through parallelization, and reduce computation by downsampling $\mathbf{F}_{t-1}$ and $\mathbf{F}_t$ via max poooling, then restore the resolution of $\mathcal{F}_{t\to t-1}$ through bilinear interpolation.

Pseudo-next Feature Generation. According to the motion consistency, with a sufficiently high frame rate, the magnitude of displacement of an object from t to $t-1$ is consistent with that from t to $t+1$, but in opposite directions, *i.e.*, $\mathcal{F}_{t\to t+1} \approx -\mathcal{F}_{t\to t-1}$. Thus, we can warp $\mathbf{F}_t$ to the $t+1$ grid by using inverted $\mathcal{F}_{t\to t-1}$. Let $\mathbf{w}_{t\to t-1}(\mathbf{u},\mathbf{v}) = (\mathbf{x}(\mathbf{u},\mathbf{v}),\mathbf{y}(\mathbf{u},\mathbf{v}))$ denotes the flow at pixel index $(\mathbf{u},\mathbf{v})$ in $\mathcal{F}_{t\to t-1}$, where $\mathbf{x}$ and $\mathbf{y}$ represent values in the vertical and horizontal directions, respectively. Let $\mathbf{F}_{t+1}^{pseudo} = \{\mathbf{c}_{t+1}^{pseudo}(i,j)|i=0,...,H-1;j=0,...,W-1\}$ represents the warped feature in $t+1$ grid. The warped element at $(\mathbf{u},\mathbf{v})$ can be formulated as [21, 20, 48]:

$$\mathbf{c}_{t+1}^{pseudo}(\mathbf{u},\mathbf{v}) = \mathbf{c}_t\left(\mathbf{u}-\mathbf{x}(\mathbf{u},\mathbf{v}),\mathbf{v}-\mathbf{y}(\mathbf{u},\mathbf{v})\right) \tag{4}$$

We refer to the $\mathbf{F}_{t+1}^{pseudo}$ obtained through this pixel-level backward warping as pseudo-next feature. In theory, with a sufficiently fast frame rate and accurate matching, it can be aligned well with real $\mathbf{F}_{t+1}$. As this process does not require the real $\mathbf{F}_{t+1}$, it satisfies the constraints of streaming perception.

Note that $\mathbf{F}_{t+1}^{pseudo}$ has limitations when dealing with occluded or truncated objects, and the warping operation may introduce additional edge noise into the scene. Thus, we fuse it with historical features to complement the geometric shape information of the objects. Similar to [59], we initially utilize weight-shared convolutions to reduce the channels of $\mathbf{F}_{t-1}$, $\mathbf{F}_t$, and $\mathbf{F}_{t+1}^{pseudo}$, and then concatenate them together. The concatenated features are connected with $\mathbf{F}_t$ via add operator to enhance the representation of static information, as illustrated in part (d) of Figure 3. The latency of our FFF process is only 7.67ms, please refer to Appendix A.5 for more latency and hyperparameter reports.

3.3 Motion Consistency Loss (MCL)

To explicitly guide the model in learning the offset magnitude, we propose a Motion Consistency Loss (MCL) as supplementary regression supervision, which consists of velocity loss and acceleration loss. This loss is also grounded on the principle of motion consistency. It leverages historical motion trajectories to guide the prediction of the next frame, as illustrated in Figure 5.

The initial step in calculating the MCL involves establishing correspondences between bounding boxes across different time steps. We establish these correspondences between the ground truth of $\{\mathbf{G}_{t-2}^{box}, \mathbf{G}_{t-1}^{box}, \mathbf{G}_{t}^{box}\}$ by utilizing target IDs. For the prediction $\mathbf{P}_{t+1}^{box}$, we calculate an IoU (Intersection over Union) matrix with respect to $\mathbf{G}_{t}^{box}$ and employ a $\max$ operation to identify the index corresponding to the highest matching value, thus establishing their correspondence.

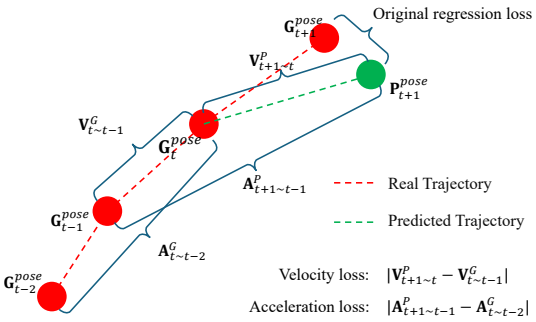


Figure 5: Illustration of MCL.

Let $\mathbf{G}_{i}^{pose}$ denotes the ground truth center position and rotation angle of the objects at time step i, where $i = t-2, t-1, t$, such that $\mathbf{G}_{i}^{pose} = (x_i^g, y_i^g, z_i^g, \theta_i^g)$. Similarly, $\mathbf{P}_{t+1}^{pose} = (x_{t+1}^p, y_{t+1}^p, z_{t+1}^p, \theta_{t+1}^p)$ represents the prediction for the next frame. Base on the correspondence, now we can calculate the displacement vector and the sine difference of the rotation angles between $\mathbf{P}_{t+1}^{pose}$ and $\mathbf{G}_{t}^{pose}$ for the predicted offset $\mathbf{V}_{t+1\to t}^{p} = (\Delta x^p, \Delta y^p, \Delta z^p, \Delta\theta^p)$:

$$\begin{aligned} \Delta x^p &= x_{t+1}^p - x_t^g, \qquad \Delta y^p = y_{t+1}^p - y_t^g, \\ \Delta z^p &= z_{t+1}^p - z_t^g, \qquad \Delta\theta^p = \sin(\theta_{t+1}^p - \theta_t^g) \end{aligned} \tag{5}$$

We can similarly calculate the ground truth offset $\mathbf{V}_{t\to t-1}^{g}$ between $\mathbf{G}_{t}^{pose}$ and $\mathbf{G}_{t-1}^{pose}$ as supervision for the predicted offset $\mathbf{V}_{t+1\to t}^{p}$. Thus, the regression for velocity loss can be formulated as:

$$\mathcal{L}_{\mathbf{V}} = SmoothL1(\mathbf{V}_{t+1\to t}^{p} - \mathbf{V}_{t\to t-1}^{g}) \tag{6}$$

Further consideration is given to velocity change. We calculate the ground truth offset $\mathbf{V}_{t-1\to t-2}^{g}$ between $\mathbf{G}_{t-1}^{pose}$ and $\mathbf{G}_{t-2}^{pose}$ and then utilize the ground truth $\mathbf{A}_{t\to t-2}^{g} = \mathbf{V}_{t\to t-1}^{g} - \mathbf{V}_{t-1\to t-2}^{g}$ of velocity change to supervise the prediction $\mathbf{A}_{t+1\to t-1}^{p} = \mathbf{V}_{t+1\to t}^{p} - \mathbf{V}_{t\to t-1}^{g}$. We also employ the Smooth L1 loss [57] to regress the acceleration consistency:

$$\mathcal{L}_{\mathbf{A}} = SmoothL1(\mathbf{A}_{t+1\to t-1}^{p} - \mathbf{A}_{t\to t-2}^{g}) \tag{7}$$

The MCL can then be defined as:

$$\mathcal{L}_{MCL} = \mathcal{L}_{\mathbf{V}} + \tau\mathcal{L}_{\mathbf{A}} \tag{8}$$

where $\tau = 0.8$. The grid search of τ can be seen in Appendix A.6. Let $\mathcal{L}_{ori}$ denotes the original loss of DSGN++ [7] with the supervision of the next frame, our total loss is finally therefore:

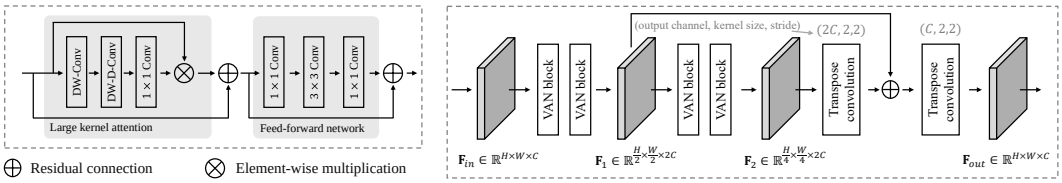
$$\mathcal{L} = \frac{1}{N_{pos}}(\mathcal{L}_{ori} + \lambda\mathcal{L}_{MCL}) \tag{9}$$

where N_{pos} is the number of positive anchors and $\lambda = 0.5$.

3.4 Large Kernel BEV Backbone (LKBB)

Our LKBB is built upon Visual Attention Network (VAN) [12], which comprises a Large Kernel Attention (LKA) [12] module and a Feed-Forward Network (FFN) [51], as shown in the Figure 6a. The LKA module is composed of cascaded depth-wise convolution (DW-Conv), depth-wise dilation convolution (DW-D-Conv), and 1×1 convolution.

We migrate VAN into our BEV backbone network to increase its receptive field, as illustrated in Figure 6b. For fair comparison, we adjust the number of VAN blocks and the multiplier for channel



(a) Architecture of a VAN block.

(b) Architecture of LKBB.

Figure 6: Illustration of LKBB.

transformations to keep the parameter count of LKBB similar to that of the original structure. Further, to maintain the sensitivity of the model to fine-grained geometric details, we incorporate multi-scale features via residual connections.

Let $f_{\mathsf{T}}^{c,k,s}$ denotes the transpose convolution operation, where c, k and s respectively represent the number of filters, kernel size and stride. The multi-scale fusion process can be formulated as:

$$\mathbf{F}_{out} = f_{\mathsf{T}}^{C,2,2}\left(f_{\mathsf{T}}^{2C,2,2}\left(\mathbf{F}_2\right) + \mathbf{F}_1\right) \tag{10}$$

where $\mathbf{F}_1$, $\mathbf{F}_2$ and $\mathbf{F}_{out}$ represent the downsampled features from two stages and the final output feature, respectively, with dimensions $\frac{H}{2} \times \frac{W}{2} \times 2C$, $\frac{H}{4} \times \frac{W}{4} \times 2C$ and $H \times W \times C$.

We report the comparison of parameter count and computational complexity between our LKBB and the original hourglass backbone [41] in Appendix A.7. Specifically, our parameter count is similar to the original structure while the computational complexity is reduced by approximately 6GFLOPs.

4 Experiments

4.1 Experimental setup

Dataset. The prevalent autonomous driving datasets, such as nuScenes [3], nuScenes-H [52] and Waymo Open [50], lack stereo image provision. Further, the stereo frame rate (only 5Hz) within Argoverse [4] is insufficient for streaming simulation. Consequently, we conduct our experiments on KITTI tracking dataset [10], which provides stereo images and a higher frame rate (10Hz). We partition the training set with 20 scenes into numerous frame sequences, each comprising 40 frames. Among them, the even-numbered sequences are used for training with a total of 4,291 frames, and the odd-numbered sequences are used for testing with a total of 3,672 frames. We analyze the domain gap of this partitioning method in Appendix A.3.

Evaluation metrics. We follow [30, 52] to conduct the streaming simulation to obtain the streaming average precision (sAP) for both BEV and 3D perspectives. Consistent with KITTI [10], objects are categorized into three levels based on their recognition complexity: Easy, Moderate, and Hard. All of our precision measurements for experiments and ablation studies are calculated at $IoU = 0.7$ and with 40 recall positions. If not specified, all results are for the "Car" category.

Implementation details. Our experiments are conducted on the NVIDIA TITAN RTX platform with a total of 20 epochs. During the training phase, the Adam optimizer [23] is employed in conjunction with the OneCycle learning rate decay strategy [47]. The initial learning rate is set to $1e-3$, progressively increased to $1e-2$, and finally decayed to $1e-8$. Further, for a fair comparison with offline perception, we extend the copy-paste data augmentation [57, 7] to streaming data. It cascades moving foreground objects into consecutive training frames to enrich the training samples.

4.2 Comparison with Meta-detector

We re-implement the meta-detector named Streamer for streaming benchmark in the stereo-based 3D object detection domain. Following the works in [30, 52], Streamer respectively employs DSGN++ [7] and DSGN++$_{opt}$ (equipped with real-time optimization strategies) as the base detector to predict the current frame, and then utilizes a Kalman filter [22] to combine historical outputs for forecasting the future state. Note that Streamer [30, 52] is a non-end-to-end solution. In Table 1, we compare our

Table 1: Comparison with the meta-detector named Streamer. DSGN++$_{opt}$ represents the method equipped with our real-time optimization strategies. The symbol "†" denotes the basic framework of StreamYOLO [59], which is built upon the DSGN++$_{opt}$ architecture.

Methods	Base detector	Latency (ms)	sAP_{BEV} (IoU=0.5)			sAP_{3D} (IoU=0.5)			sAP_{BEV} (IoU=0.7)			sAP_{3D} (IoU=0.7)		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Car														
Streamer	DSGN++	263.33	14.76	11.56	10.62	12.51	8.86	7.74	5.26	3.42	3.11	2.82	1.92	1.51
Streamer	DSGN++$_{opt}$	80.71	70.57	61.04	56.31	64.47	55.18	50.36	30.89	22.50	19.96	25.50	17.68	14.75
StreamYOLO†	DSGN++$_{opt}$	81.32	92.22	85.75	82.45	89.40	82.89	78.10	80.72	68.51	63.14	73.02	58.37	51.86
StreamDSGN (ours)	DSGN++$_{opt}$	91.45	93.10	87.46	84.34	92.53	84.55	81.34	85.40	72.47	68.66	77.47	63.76	57.42
Pedestrian														
Streamer	DSGN++	263.33	7.38	7.22	7.42	6.58	6.85	6.59	1.87	1.68	1.51	1.55	1.13	1.12
Streamer	DSGN++$_{opt}$	80.71	58.23	54.57	50.25	56.56	53.97	49.70	31.26	28.92	26.55	25.18	23.50	21.49
StreamYOLO†	DSGN++$_{opt}$	81.32	74.54	69.10	63.48	73.44	68.68	63.08	53.27	49.35	44.67	45.53	42.15	38.01
StreamDSGN (ours)	DSGN++$_{opt}$	91.45	80.70	75.38	69.23	80.41	75.12	68.96	62.33	57.51	51.34	54.12	50.05	44.89
Cyclist														
Streamer	DSGN++	263.33	3.73	4.29	4.31	3.74	4.15	3.74	1.13	0.83	0.82	1.13	0.60	0.60
Streamer	DSGN++$_{opt}$	80.71	41.11	40.73	40.07	40.78	40.37	39.24	7.56	15.52	14.82	6.77	14.35	14.13
StreamYOLO†	DSGN++$_{opt}$	81.32	39.34	41.31	40.59	38.72	40.20	39.59	31.37	34.03	33.43	30.63	33.14	32.36
StreamDSGN (ours)	DSGN++$_{opt}$	91.45	44.62	48.92	48.74	43.52	48.03	47.46	37.42	41.10	40.64	35.01	37.37	37.17

Table 2: Ablation studies of our methods. Setting ***a*** denotes the DSGN++$_{opt}$. Setting ***b*** represents directly predicting future results. Setting ***c*** incorporates historical feature based on ***b*** and serves as our baseline method (highlighted in gray). Settings ***d***, ***e***, and ***f*** respectively incorporate our enhancement strategies. Setting ***g*** is our final proposed solution (highlighted in green).

Setting ID	Pipeline		FFF	MCL	LKBB	sAP_{BEV}			sAP_{3D}		
	predict $t+1$	fuse $t-1$				Easy	Mod.	Hard	Easy	Mod.	Hard
a	–	–	–	–	–	30.89	22.50	19.96	25.50	17.68	14.75
b	✓	–	–	–	–	53.73	47.46	43.36	38.62	32.55	29.39
c	✓	✓	–	–	–	83.20	71.15	65.74	75.38	59.55	53.11
d	✓	✓	✓	–	–	85.08	71.98	66.49	75.88	61.57	54.77
e	✓	✓	–	✓	–	84.63	71.01	66.18	78.15	62.28	56.80
f	✓	✓	–	–	✓	84.97	71.96	66.53	76.74	60.31	55.01
g	✓	✓	✓	✓	✓	85.40	72.47	68.66	77.47	63.76	57.42

method with these non-real-time and real-time settings. From these comparisons, we can make the following observations.

First, the Streamer [30, 52] combined with DSGN++ [7] setting results in a noticeable inferior in streaming accuracy. In our streaming simulation, we find that this is due to the large-span temporal interval, making it challenging for the Kalman filter [22] to accurately track each target in multi-object scenarios. Second, the Streamer [30, 52] combined with the DSGN++$_{opt}$ setting shows higher performance compared to the DSGN++ combined one due to the latency reduction. However, with a high matching threshold ($IoU = 0.7$), the improvement in streaming accuracy is quite limited. Third, the StreamYOLO [59] framework built upon DSGN++$_{opt}$ achieves significant improvements across all metrics. These results demonstrate the competitiveness of the end-to-end framework that directly predicts future states compared to non-end-to-end solutions in the benchmark. Finally, our StreamDSGN reaches the highest level of performance, with approximately a 5% improvement in each metric compared to the extended version of StreamYOLO [59]. Despite the latency in this setting reaching 91.45ms, the inference speed remains faster than the input frame rate, thus meeting real-time requirements.

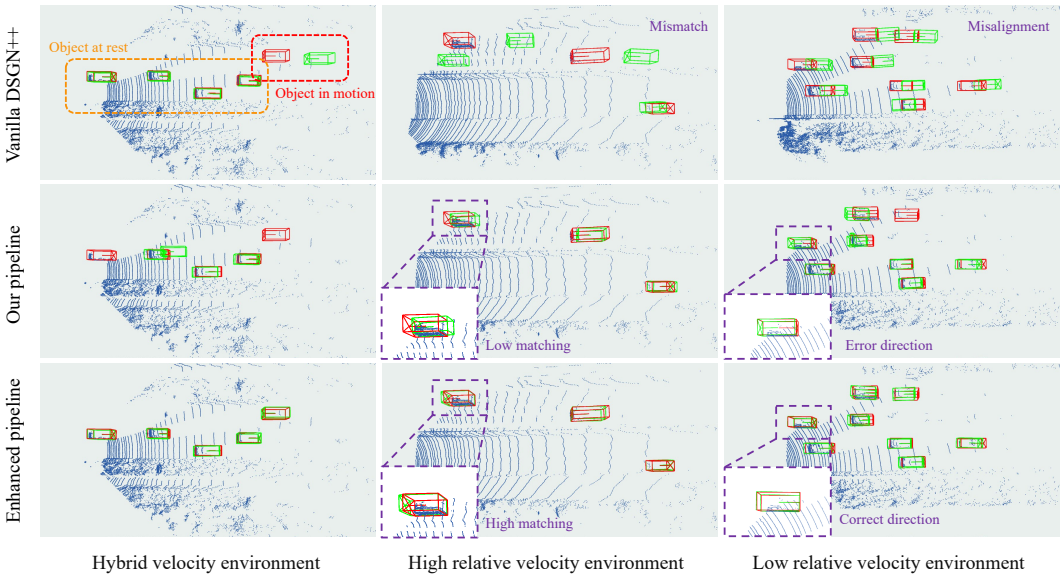


Figure 7: Qualitative analysis of different relative velocity scenarios. We visualize the predictions in point clouds for a clearer comparison. The first row describes DSGN++ [7] without any modifications. The second row integrates real-time optimization and fusion of historical frames to predict the next frame. The third row showcases our three enhancement strategies. Ground truth and prediction instances are respectively delineated by red and green bounding boxes, with lines inside the boxes indicating the orientation of objects.

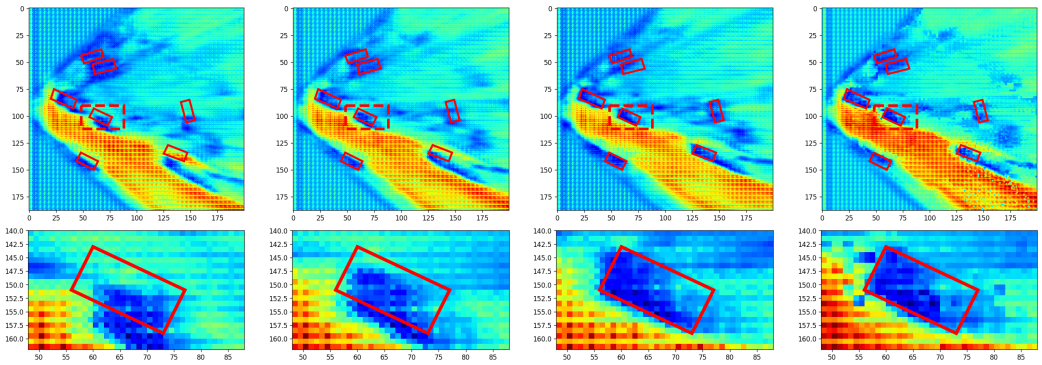
4.3 Ablation Studies

To validate the effectiveness of each strategy, we conduct ablation experiments by toggling them on and off. Experimental results is illustrated in Table 2. In the following, we provide a more detailed description of the ablation experiments.

Effectiveness of the Pipeline. The comparison between setting ***b*** and setting ***a*** reveals that directly predicting future states by using a real-time detector can alleviate the accuracy degradation caused by streaming perception constraints. However, the improvement in accuracy at this time is limited due to the lack of information on each object's displacement magnitude. Furthermore, by incorporating historical features, streaming accuracy can be significantly improved (see setting ***c***). This improvement occurs because the fused features implicitly embed motion cues, enabling the model to learn different offsets for each object. Thus, setting ***c*** serves as our baseline and is utilized for subsequent ablation studies to assess each enhancement strategy's effectiveness.

Effectiveness of Enhanced Strategies. The comparison of settings ***d***, ***e***, and ***f*** with setting ***c*** demonstrates the effectiveness of each enhancement strategy, respectively. From these comparisons, we can make the following observations.

First, since the fusion scheme used in setting ***c*** is derived from the Dual-Flow module of StreamYOLO [59], the comparison between setting ***d*** and setting ***c*** can be viewed as a comparison between our FFF and the SOTA streaming perception fusion method. The results show that our FFF achieves improvements across all metrics. Second, setting ***e*** reveals the effectiveness of MCL, as the additional supervision of trajectory consistency yields significant improvements. Compared to the baseline, improvements across the three levels reach 2.77%, 2.73%, and 3.69% in sAP_{3D}, respectively. Third, due to the consistency between the downsampling dimensions of the feature maps in LKBB and the baseline, as well as the similar parameter counts between the two structures, it is evident that the improvements in setting ***f*** are primarily attributed to the advantage of the large receptive field feature extractor in capturing long-range contextual features. Finally, when we combine all strategies together (highlighted in green), the improvement in streaming accuracy reaches its peak, with a 4.33% increase in sAP_{3D} compared to the baseline at the hard level.



(a) Previous feature (b) Current feature (c) Next feature (d) Pseudo-next feature

Figure 8: Qualitative analysis of the pseudo-next feature. The first row displays complete feature maps from different time steps, while the second row shows corresponding local regions of the feature maps. The **solid** red box and **dashed** box respectively represent the ground truth of the next frame used for supervision and the locally zoomed-in area.

4.4 Qualitative Analysis

Visualization of Streaming Detection. Figure 7 presents the qualitative results of our proposed pipeline compared to the vanilla DSGN++ [7]. In scenarios where surrounding vehicles remain relatively stationary, DSGN++ [7] demonstrates a robust capacity for accurately aligning with ground truth. However, as these vehicles initiate motion relative to the ego vehicle, DSGN++ [7] exhibits misalignments or mismatches in predictions due to the latency. In contrast, our baseline pipeline adeptly addresses this issue. With the enhancement strategy incorporating FFF, MCL and LKBB, the alignment between predicted boxes and ground truth boxes is further improved.

Visualization of Pseudo-next Feature. We visualize the pseudo-next feature in Figure 8 for qualitative analysis. We can observe that due to the constraints of streaming perception, there is varying misalignment between the previous feature and the current feature with respect to the next ground truth used for supervision. However, after similarity matching and reverse warping, our pseudo-next feature aligns well with the ground truth.

5 Conclusions

For the first time, we propose StreamDSGN, a real-time stereo-based 3D object detection framework for streaming perception. It is further equipped with Feature Flow Fusion, Motion Consistency Loss, and a Large Kernel BEV Backbone to enhance performance.

Limitations and Future Work. When objects are occluded or truncated, FFF may produce incorrect pseudo-next features due to erroneous similarity matching. Currently, we mitigate this issue by simply integrating historical features. In the future, we plan to leverage neural networks to directly predict the flow of dynamic foreground objects, aiming to be more robust in addressing this challenge by exploiting the adaptability of neural networks. The second limitation of our approach is that it has only been validated using stereo-based methods. In contrast, currently mainstream autonomous driving perception tasks typically employ surrounding multi-view camera systems, which differ from stereo-based methods in their construction of BEV representations. Consequently, our future research will focus on validating the effectiveness of this approach within the context of multi-view methods.

6 Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant 62027804, the Fund of National Key Laboratory of Multispectral Information Intelligent Processing Technology (No. 202410487201), and the Major Key Project of PCL (PCL2021A13).

References

[1] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019.

[2] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3d object detection in monocular video. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pages 135–152. Springer, 2020.

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[4] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019.

[5] Qi Chen, Sourabh Vora, and Oscar Beijbom. Polarstream: Streaming object detection and segmentation with polar pillars. *Advances in Neural Information Processing Systems*, 34: 26871–26883, 2021.

[6] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Dsgn: Deep stereo geometry network for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12536–12545, 2020.

[7] Yilun Chen, Shijia Huang, Shu Liu, Bei Yu, and Jiaya Jia. Dsgn++: Exploiting visual-spatial relation for stereo-based 3d detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4416–4429, 2022.

[8] Davi Frossard, Shun Da Suo, Sergio Casas, James Tu, and Raquel Urtasun. Strobe: Streaming object detection from lidar packets. In *Conference on Robot Learning*, pages 1174–1183. PMLR, 2021.

[9] Divyansh Garg, Yan Wang, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Wasserstein distances for stereo disparity estimation. *Advances in Neural Information Processing Systems*, 33:22517–22529, 2020.

[10] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[11] Anurag Ghosh, Vaibhav Balloli, Akshay Nambi, Aditya Singh, and Tanuja Ganu. Chanakya: Learning runtime decisions for adaptive real-time perception. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Meng-Hao Guo, Cheng-Ze Lu, Zheng-Ning Liu, Ming-Ming Cheng, and Shi-Min Hu. Visual attention network. *Computational Visual Media*, 9(4):733–752, 2023.

[13] Xiaoyang Guo, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Liga-stereo: Learning lidar geometry aware representations for stereo-based 3d detector. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3153–3163, 2021.

[14] Wei Han, Zhengdong Zhang, Benjamin Caine, Brandon Yang, Christoph Sprunk, Ouais Alsharif, Jiquan Ngiam, Vijay Vasudevan, Jonathon Shlens, and Zhifeng Chen. Streaming object detection for 3-d point clouds. In *European Conference on Computer Vision*, pages 423–441. Springer, 2020.

[15] Jun-Yan He, Zhi-Qi Cheng, Chenyang Li, Wangmeng Xiang, Binghui Chen, Bin Luo, Yifeng Geng, and Xuansong Xie. Damo-streamnet: Optimizing streaming perception in autonomous driving. *arXiv preprint arXiv:2303.17144*, 2023.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399, 2019.

[19] Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022.

[20] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017.

[21] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[22] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[24] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[25] Kin Wai Lau, Lai-Man Po, and Yasar Abbas Ur Rehman. Large separable kernel attention: Rethinking the large kernel attention design in cnn. *Expert Systems with Applications*, 236: 121352, 2024.

[26] Changcai Li, Haitao Meng, Gang Chen, and Long Chen. Real-time pseudo-lidar 3d object detection with geometric constraints. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3298–3303. IEEE, 2022.

[27] Chengyao Li, Jason Ku, and Steven L Waslander. Confidence guided stereo 3d object detection with split depth estimation. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5776–5783. IEEE, 2020.

[28] Chenyang Li, Zhi-Qi Cheng, Jun-Yan He, Pengyu Li, Bin Luo, Hanyuan Chen, Yifeng Geng, Jin-Peng Lan, and Xuansong Xie. Longshortnet: Exploring temporal and semantic features fusion in streaming perception. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[29] Jinyu Li, Chenxu Luo, and Xiaodong Yang. Pillarnext: Rethinking network designs for 3d object detection in lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17567–17576, 2023.

[30] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 473–488. Springer, 2020.

[31] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019.

[32] 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*, pages 1–18. Springer, 2022.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[34] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, and Xiangyu Zhang. Petrv2: A unified framework for 3d perception from multi-camera images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3262–3272, 2023.

[35] Yuxuan Liu, Lujia Wang, and Ming Liu. Yolostereo3d: A step back to 2d for efficient stereo 3d detection. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13018–13024. IEEE, 2021.

[36] Zhe Liu, Xiaoqing Ye, Xiao Tan, Errui Ding, and Xiang Bai. Stereodistill: Pick the cream from lidar for distilling stereo-based 3d object detection. *arXiv preprint arXiv:2301.01615*, 2023.

[37] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4721–4730, 2021.

[38] Haitao Meng, Changcai Li, Gang Chen, Long Chen, et al. Efficient 3d object detection based on pseudo-lidar representation. *IEEE Transactions on Intelligent Vehicles*, 2023.

[39] Gregory P Meyer, Jake Charland, Shreyash Pandey, Ankit Laddha, Shivam Gautam, Carlos Vallespi-Gonzalez, and Carl K Wellington. Laserflow: Efficient and probabilistic object detection and motion forecasting. *IEEE Robotics and Automation Letters*, 6(2):526–533, 2020.

[40] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. In *International Conference on Learning Representations*, 2018.

[41] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, pages 483–499. Springer, 2016.

[42] Wanli Peng, Hao Pan, He Liu, and Yi Sun. Ida-3d: Instance-depth-aware 3d object detection from stereo vision for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13015–13024, 2020.

[43] Xidong Peng, Xinge Zhu, Tai Wang, and Yuexin Ma. Side: center-based stereo 3d detector with structure-aware instance depth estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 119–128, 2022.

[44] Alex D Pon, Jason Ku, Chengyao Li, and Steven L Waslander. Object-centric stereo matching for 3d object detection. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8383–8389. IEEE, 2020.

[45] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[46] Gur-Eyal Sela, Ionel Gog, Justin Wong, Kumar Krishna Agrawal, Xiangxi Mo, Sukrit Kalra, Peter Schafhalter, Eric Leong, Xin Wang, Bharathan Balaji, et al. Context-aware streaming perception in dynamic environments. In *European Conference on Computer Vision*, pages 621–638. Springer, 2022.

[47] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, pages 369–386. SPIE, 2019.

[48] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018.

[49] Jiaming Sun, Linghao Chen, Yiming Xie, Siyu Zhang, Qinhong Jiang, Xiaowei Zhou, and Hujun Bao. Disp r-cnn: Stereo 3d object detection via shape prior guided instance disparity estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10548–10557, 2020.

[50] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin 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*, pages 2446–2454, 2020.

[51] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022.

[52] Xiaofeng Wang, Zheng Zhu, Yunpeng Zhang, Guan Huang, Yun Ye, Wenbo Xu, Ziwei Chen, and Xingang Wang. Are we ready for vision-centric driving streaming perception? the asap benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9600–9610, 2023.

[53] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[54] Yan Wang, Bin Yang, Rui Hu, Ming Liang, and Raquel Urtasun. Plumenet: Efficient 3d object detection from stereo images. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3383–3390. IEEE, 2021.

[55] Wenhao Wu, Hau San Wong, and Si Wu. Semi-supervised stereo-based 3d object detection via cross-view consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17471–17481, 2023.

[56] Zhenbo Xu, Wei Zhang, Xiaoqing Ye, Xiao Tan, Wei Yang, Shilei Wen, Errui Ding, Ajin Meng, and Liusheng Huang. Zoomnet: Part-aware adaptive zooming neural network for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12557–12564, 2020.

[57] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[58] Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, et al. Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17830–17839, 2023.

[59] Jinrong Yang, Songtao Liu, Zeming Li, Xiaoping Li, and Jian Sun. Real-time object detection for streaming perception. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5385–5395, 2022.

[60] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3d-man: 3d multi-frame attention network for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1863–1872, 2021.

[61] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang 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*, pages 11495–11504, 2020.

[62] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *International Conference on Learning Representations (ICLR)*, 2020.

[63] Zhenxun Yuan, Xiao Song, Lei Bai, Zhe Wang, and Wanli Ouyang. Temporal-channel transformer for 3d lidar-based video object detection for autonomous driving. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(4):2068–2078, 2021.

A Appendix / supplemental material

A.1 Real-time Optimization

Our real-time optimization strategies are outlined as follows:

(1) We replace the original ResNet34 with ResNet18 [16] as the backbone for stereo images.

(2) We abandon the Dual-view Stereo Volume representation in DSGN++ [7] and instead retained only one of PSV or 3DGV. See more detail in [7].

(3) Inspired by [24, 29], we extend the grid size of the feature volume from $(0.2m, 0.2m, 0.2m)$ to $(0.2m, 0.2m, 0.4m)$ along the vertical axis.

(4) We perform edge cropping for detection range [37]. The new range is set to $[2m, 53.2m]$ for the Z-axis (depth), $[-28.8m, 28.8m]$ for the X-axis, and $[-1m, 3m]$ for the Y-axis respectively (in the camera coordinate system).

(5) Referring to [40], we incorporate Automatic Mixed Precision (AMP) techniques during both the training and inference stages.

A.2 Effectiveness of Real-Time Optimization

Table 3: Comparison of real-time optimization strategies. The term "r18" denotes the use of ResNet18 as the image feature extractor. "Only 3DGV" and "Only PSV" represent the exclusive adoption of 3DGV and PSV as stereo features, respectively. "Low-res. (h)" indicates low-resolution representation in the vertical dimension. "EC" signifies edge cropping, and "AMP" stands for the utilization of automatic mixed precision [40] during both training and inference stages.

Ablation method						Latency (ms)	offAP_{3D}			sAP_{3D}		
r18	Only 3DGV	only PSV	Lo-res. (h)	EC	AMP		Easy	Mod.	Hard	Easy	Mod.	Hard
Non-real-time (KF Forecasting [22, 30, 52])												
–	–	–	–	–	–	263.33	91.79	75.35	69.79	2.82	1.92	1.51
✓	✓	–	–	–	–	180.96	90.60	74.08	68.35	6.46	4.26	3.58
✓	✓	–	✓	–	–	150.55	90.77	74.45	68.73	8.94	5.80	4.74
✓	–	✓	✓	–	–	147.81	**91.98**	75.91	68.96	7.51	5.15	4.41
✓	–	✓	✓	✓	–	134.21	91.81	**77.22**	**69.98**	9.52	6.08	5.15
Real-time (End-to-end)												
✓	–	✓	✓	–	✓	86.68	91.67	76.52	69.34	19.62	14.60	12.79
✓	–	✓	✓	✓	✓	**80.71**	91.22	74.95	69.14	**25.50**	**17.68**	**14.75**

We compare the effectiveness of various real-time optimization strategies, experimental results are described in Table 3. For real-time settings (<100ms), each ground truth frame has a corresponding predicted frame, making it an end-to-end approach. For non-real-time settings (>100ms), we employ Kalman filter [22, 30, 52] to forecast and interpolate unmatched intermediate frames. Hence, Table 3 can also be viewed as a comparison between end-to-end methods and traditional SOTA methods. The table reveals that all optimization strategies mentioned in Appendix A.1 effectively reduce model latency with minimal impact on offline accuracy, decreasing from the initial 263.33ms to 80.71ms. Concurrently, the sAP_{3D} gradually increases due to the reduced latency, rising from 2.53% to 23.73% in the easy level, achieving an improvement of over 20%. Therefore, we choose the configuration highlighted in green row in Table 3 as the final optimization strategy.

A.3 Comparison of the Domain Gap

We compare the domain gap between our split tracking dataset, which consists of 4,291 training samples and 3,672 validation samples, and the widely recognized KITTI 3D Object Detection dataset [10], comprising 3,712 training samples and 3,769 validation samples. Specifically, we train and test PointPillar [24] and DSGN++ [7] on both datasets, and then compare the AP_{3D} for the Car category at $IoU = 0.7$.

Table 4: Comparison of the domain gap between our split Tracking dataset and the 3D Object Detection dataset.

Method	Sensor	Dataset	AP_{3D}		
			Easy	Mod.	Hard
PointPillars [24]	LiDAR	Object Detection	87.75	78.38	75.18
PointPillars [24]	LiDAR	our split Tracking	94.57	88.35	84.85
DSGN++ [7]	Stereo	Object Detection	83.63	66.41	61.38
DSGN++ [7]	Stereo	our split Tracking	91.79	78.35	69.79

The experimental results are shown in the Table 4. From the table, we observe that both methods perform better on our split Tracking dataset, indicating a smaller domain gap compared to the Object Detection dataset. However, given that we have 579 additional training samples and that accuracy may further decrease under streaming perception constraints, this difference is deemed reasonable.

A.4 Impact of Fusion Stage

Table 5: Ablation studies of fusion stage.

Fusion Stage	sAP_{BEV}			sAP_{3D}		
	Easy	Mod.	Hard	Easy	Mod.	Hard
before head [59]	80.72	68.51	63.14	73.02	58.37	51.86
before BEV backbone	83.20	71.15	65.74	75.38	59.55	53.11

To validate the hypothesis about insufficient receptive field of shallow detection heads, we conducted comparative experiments by setting the fusion stage before the detection heads and before the BEV backbone, respectively. The comparison results are described in Table 5. The table demonstrates that fusing historical features before the BEV backbone, rather than preceding the detection head as proposed in StreamYOLO [59], leads to competitive improvements. Specifically, the improvements at the three difficulty levels exceed 2%, 1%, and 1%, respectively.

Table 6: Comparison with SOTA fusion method and ablation studies on FFF. The term "r_d" and "d" represent the downsample ratio and maximum displacement, respectively.

r_d	d	Latency (ms)	sAP_{BEV}			sAP_{3D}		
			Easy	Mod.	Hard	Easy	Mod.	Hard
Dual-Flow [59]								
–	–	2.67	83.20	71.15	65.74	75.38	59.55	53.11
Feature-Flow Fusion (Ours)								
2	3	7.67	85.08	71.98	66.49	75.88	61.57	54.77
2	4	12.19	85.50	71.68	66.13	76.82	60.73	54.02
2	5	14.63	85.48	71.93	66.22	75.91	59.83	54.46
4	3	6.26	83.27	71.35	64.24	76.11	59.40	53.82

Table 7: Grid search of τ in Equation 8 for MCL.

τ	sAP_{BEV}			sAP_{3D}		
	Easy	Mod.	Hard	Easy	Mod.	Hard
w/o MCL						
–	83.20	71.15	65.74	75.38	59.55	53.11
w/ MCL						
0.0	84.99	71.71	66.04	76.13	61.26	54.40
0.2	83.43	71.82	66.10	76.72	61.88	55.08
0.4	83.35	71.17	64.35	76.30	61.58	54.82
0.6	84.68	71.63	65.85	76.30	61.68	54.74
0.8	84.63	71.01	66.18	78.15	62.28	56.80
1.0	85.09	71.49	66.03	75.74	59.62	54.24

A.5 Latency and Hyperparameters of FFF

We report the latency and accuracy of FFF under different feature downsampling ratios and maximum search displacement in Table 6. The Table shows that with a downsampling ratio of 2 and maximum displacement of 3, FFF achieves the best performance while only taking 7.67ms (highlighted in green). At this setting, sAP_{3D} shows a 2.01% improvement at the moderate level compared to the SOTA Dual-Flow [59].

A.6 Value of τ

We get the value of hyperparameter τ in Equation 8 through grid search. Experimental results are shown in Table 7. We observe that when only utilizing the velocity loss setting in MCL ($\tau = 0$), the improvements at the three difficulty levels compared to the scheme without MCL are about 1%, 1.5%, and 1%, respectively. Further, when introducing the acceleration loss, the performance reaches its maximum improvement when the balancing parameter $\tau = 0.8$ (highlighted in green). At this setting, the improvements at the three difficulty levels are 2.77%, 2.73%, and 3.69%, respectively.

A.7 Complexity of LKBB

Table 8: Comparison of the Complexity of BEV Backbones.

Method	Params (M)	FLOPs (G)	Latency (ms)	sAP_{BEV}			sAP_{3D}		
				Easy	Mod.	Hard	Easy	Mod.	Hard
Hourglass (original)	0.794	17.737	3.66	83.20	71.15	65.74	75.38	59.55	53.11
LKBB (ours)	0.810	11.695	6.73	84.97	71.96	66.53	76.74	60.31	55.01

We present in Table 8 a comparison between LKBB and the original hourglass structure [41] in terms of parameter count, computational complexity, and latency. Our LKBB has a similar parameter count to the original structure, yet it reduces computational complexity by nearly 6 GFLOPs. At this juncture, the overhead incurred is merely an additional approximate 3ms, while achieving a competitively enhanced accuracy.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We delineate our motivations, claims, and contributions in the abstract and introduction sections. Please see Section 1

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Please refer to Section 5 for a more detailed discussion on limitations.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: Our work does not encompass theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided detailed descriptions of the experimental setup for reproducibility, including dataset configurations and model hyperparameters. For further details, please refer to Section 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code is temporarily hosted on an anonymous platform: `https://anonymous.4open.science/r/streamDSGN-FD29`.. If the paper is accepted, we will release the source code on GitHub.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please refer to Section 4 for a more detailed discussion of the experimental setup.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Our experiments do not include error bars or confidence intervals, but they support the main claims of the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We introduce the computational resources in the experimental setup. Please refer to Section 4 for details.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research adheres to the ethical guidelines of NeurIPS.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work has no societal impact.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work does not entail such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our code is based on several released works. All have been cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have released an anonymous version of the code.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our work does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.