Streaming Motion Forecasting for Autonomous Driving

Ziqi Pang[1], Deva Ramanan[2], Mengtian Li[2], Yu-Xiong Wang[1]

***Abstract*— Trajectory forecasting is a widely-studied problem for autonomous navigation. However, existing benchmarks evaluate forecasting based on independent *snapshots* of trajectories, which are not representative of real-world applications that operate on a *continuous stream* of data. To bridge this gap, we introduce a benchmark that continuously queries future trajectories on streaming data and we refer to it as "streaming forecasting." Our benchmark inherently captures the disappearance and re-appearance of agents, presenting the emergent challenge of *forecasting for occluded agents*, which is a safety-critical problem yet overlooked by snapshot-based benchmarks. Moreover, forecasting in the context of continuous timestamps naturally asks for *temporal coherence* between predictions from adjacent timestamps. Based on this benchmark, we further provide solutions and analysis for streaming forecasting. We propose a plug-and-play meta-algorithm called "*Predictive Streamer*" that can adapt any snapshot-based forecaster into a streaming forecaster. Our algorithm estimates the states of occluded agents by propagating their positions with multi-modal trajectories, and leverages differentiable filters to ensure temporal consistency. Both occlusion reasoning and temporal coherence strategies significantly improve forecasting quality, resulting in 25% smaller endpoint errors for occluded agents and 10-20% smaller fluctuations of trajectories. Our work is intended to generate interest within the community by highlighting the importance of addressing motion forecasting in its *intrinsic* streaming setting. Code is available at https://github.com/ziqipang/StreamingForecasting.**

I. INTRODUCTION

Motion forecasting is *inherently* a *streaming* task for autonomous driving, as it operates on continuous data streams. Imagine agents constantly moving in a dynamic traffic scene, as illustrated in Fig. 1a. Naturally, agents may temporarily disappear due to *occlusions* and then re-appear, such as agent B. Even when an agent is invisible, a forecasting algorithm or forecaster is still supposed to predict its future trajectories. Ignoring occlusions could lead to the sudden appearance of agents, posing critical safety risks. Moreover, in this streaming setting where trajectories predicted on continuous timestamps have overlaps, ensuring *temporal coherence* between the trajectories becomes a key challenge. The forecaster needs to produce stable and smooth trajectories over time to serve downstream planning and control models.

Unfortunately, current motion forecasting benchmarks universally adopt a *snapshot*-based setup [2][5][6], as illustrated in Fig. 1b – it only considers *independent* and *isolated* snapshots of trajectories, without explicitly modeling the spatial-temporal connection among them. Such a setting standardizes but over-simplifies the motion forecasting problem, making it not representative of the streaming nature of the real world.

[1]University of Illinois Urbana-Champaign, [2]Cargenie Mellon University
Corresponding to {ziqip2, yxw}@illinois.edu

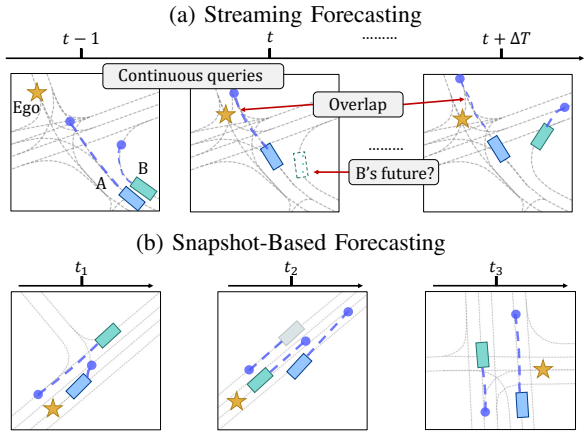

Fig. 1: (**Best viewed in color**) Gray lines: HD-Map; Blue lines: predicted trajectories. **(a)** Our streaming setup queries predictions on *continuous frames* to reflect the streaming property in the real world. Under our setting, the challenge of *forecasting for occluded agents* emerges from the frequent disappearance and re-appearance of agents (*e.g.*, agent B, highlighted by "B's future"), and *temporal coherence* becomes a natural constraint for predictions in adjacent timestamps (highlighted by "Overlap"). **(b)** The snapshot-based setup in existing forecasting benchmarks sample queries *independently* that are isolated in space and time. Such a setting hides away *occlusion* and *temporal coherence* challenges, which are presented in realistic streaming forecasting.

To overcome this limitation and demonstrate the streaming reality of autonomous driving, we propose "*streaming forecasting*" which involves *continuously* querying the future trajectories of agents *on every frame*. Such a new perspective of motion forecasting presents a clear departure from the snapshot-based setting with independent samples. Importantly, our formulation also exhibits two novel real-world challenges. (1) *Forecasting for occluded agents*: as in frame t of Fig. 1a, our streaming setup captures the changing visibility of agents and reveals the occlusion challenge. However, snapshot-based datasets [2][5][6] uniformly assume that the target agents are visible and collect the data accordingly. (2) *Temporal coherence*: as in frames t and $t+\Delta T$ of Fig. 1a, trajectories in neighboring frames have overlaps and our streaming setting reflects such a constraint by modeling motion forecasting continuously, which is impossible for isolated samples in the snapshot-based setting.

Based on this formulation, we introduce a new benchmark for streaming forecasting. Our key insight is to *re-purpose tracking* datasets, which offer realistic observations from the ego vehicle. This strategy mitigates the bias of forecasting datasets in their data collection and processing. Specifically,

Argoverse [5] is chosen for supporting modern vectorized high-definition maps (HD-Maps) on its tracking dataset. We call our benchmark *Argoverse-SF*, which faithfully captures the challenges of frequent occlusions and temporal continuity of predictions. To evaluate the performance of forecasting algorithms in this streaming setting, we design tailored metrics that investigate occlusion reasoning and temporal coherence.

To address both challenges, we further propose a simple yet effective meta-algorithm, called "*Predictive Streamer*," to adapt any existing snapshot-based forecaster into the streaming world. Our predictive streamer estimates the states of occluded agents to enable robust predictions for them. While inferring occluded positions is a long-standing problem in 3D perception [25][26][29], existing strategies of using the Kalman filter or single-modal trajectories are insufficient for the forecasting purpose, because the forecaster depends on the estimated occluded positions and amplifies the errors in occlusion reasoning. We discover that a more advanced strategy of utilizing *multi-modal trajectories* predicted by the forecaster performs better, as it provides higher-quality estimation covering a larger spectrum of motion distributions.

For *temporal coherence*, our predictive streamer introduces *differentiable filters* (DFs) [12] into multi-modal trajectory forecasting, adapting them from the state estimation literature. Our modified DFs represent future trajectories as hidden states and emphasize the temporal coherence of multi-modal trajectories in the process model of DFs. Compared with recurrent models, *e.g.*, long short-term memory networks (LSTMs), our DFs also perform better because of the explicit modeling of temporal continuity. Finally, our approach enhances the temporal coherence of trajectories and improves the accuracy of forecasting results accordingly.

In brief, we have made the following contributions.

1) ***Benchmark.*** **(a)** We introduce a novel *streaming forecasting* formulation and construct an associated "Argoverse-SF" benchmark by re-purposing the tracking dataset. **(b)** Our benchmark reveals and captures the inherent challenges in a streaming world: "*occlusion reasoning*" and "*temporal coherence*," which are previously ignored by the widely-used snapshot-based benchmarks.
2) ***Solution.*** **(a)** We propose a plug-and-play meta-algorithm called "*Predictive Streamer*" that can adapt any snapshot-based forecaster into a streaming forecaster. **(b)** We instantiate our meta-algorithm by applying *multi-modal trajectories* to estimate occluded positions and introducing *differentiable filters* to improve the temporal coherence of multi-modal trajectories. Our solution addresses the two key challenges in streaming forecasting, decreasing the minimum final displacement error (minFDE) by 25% for occluded agents and reducing the fluctuations of trajectories by 10-20%.

II. RELATED WORK

A. Motion Forecasting in Autonomous Driving

Motion forecasting methods aim to predict the trajectories of agents by modeling their dynamics, interactions, and relationship with high-definition maps (HD-Maps). Previous work has developed effective encoders to extract agent features using graph neural networks (GNNs) [7][20], LSTMs [27], and transformers [21][24][31]. Other work has focused on decoding trajectories from agent features using improved sampling strategies and learning objectives [8][9][10][28][33]. Despite rapid progress, they are all developed and evaluated under the contrived snapshot-based forecasting benchmarks [2][5][6]. Although some work [27][34] provides interfaces for making predictions on continuous timestamps, we found that adding recurrent neural networks or relative positional embedding cannot fully address the streaming forecasting challenges, especially occlusion reasoning. Instead, we analyze the limitations of existing models and propose simple yet effective strategies to estimate occluded positions and improve temporal coherence. In addition, our streaming forecasting is broadly relevant to *open-loop planning* [3] which similarly models a streaming world, with the key difference of predicting the motion of the ego-vehicle, while we focus on surrounding agents.

B. Joint Perception and Forecasting

Motion forecasting is often studied jointly with other perception modules. This naturally positions forecasting in data streams. End-to-end perception and prediction connects forecasting with 3D perception [4][11][13][23][26], which mainly explores how to benefit motion forecasting from the 3D perception features instead of improving the capability of forecasting models. By contrast, our work analyzes the setbacks of existing forecasting models under a streaming formulation and proposes an advanced "predictive streamer" to address the challenges of occlusion reasoning and temporal coherence.

C. Differentiable Filters

Differentiable filters (DFs) [1][12] combine the benefits of neural networks with Bayes filters [15], where a neural network handles high-dimensional sensor inputs and enables the filters with the flexibility to adapt to a wide range of scenarios. Recently, DFs have demonstrated their effectiveness in 3D tracking and visual odometry [14][17][18][30], as well as robot manipulation [19][32]. However, they mainly concentrate on state estimation and apply filters to *current* positions of objects. Instead, we extend DFs to enhance the prediction of *long-term future* trajectories of agents in autonomous driving scenarios, which exhibit more significant stochastic fluctuations and multi-modalities.

III. STREAMING FORECASTING BENCHMARK

A. Background: Conventional Motion Forecasting

Motion forecasting. Given historical observations, motion forecasting aims to predict the trajectories of agents in future timestamps. In the conventional formulation (Fig. 2a), historical observations have a fixed length of τ_h, and consist of the center positions of N agents $C_{t-\tau_h+1:t} = \{c^i_{t-\tau_h+1:t}\}^N_{i=1}$ and an HD-Map M. The future trajectories are multi-modal predictions denoted as $P_{t:t+\tau_f} = \{\{p^{i,k}_{t:t+\tau_f}\}^K_{k=1}\}^N_{i=1}$, where K

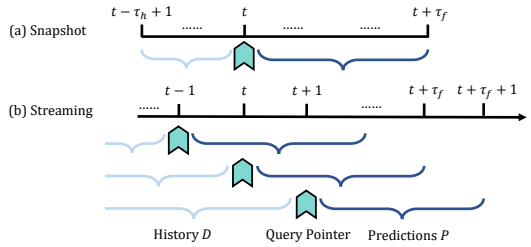

Fig. 2: Formulations of *snapshot*-based and *streaming* forecasting. **(a)** Conventional forecasting handles snapshots with a fixed length and is only intended for single-timestamp predictions. **(b)** Our *streaming* setup queries the future trajectories on *every* timestamp, aligning better with the streaming world. Compared with the snapshot-based setting, the length of history grows without an upper bound.

is the number of predictions and each $p^{i,k}_{t_1:t_2}$ represents the estimated *movement* between timestamps t_1 and t_2.

Snapshot-based benchmarks. As illustrated in Fig. 1b, existing benchmarks, such as [2][5][6], are typically constructed by collecting *independent* and *isolated* "snapshots" of trajectories, without explicitly considering the spatial-temporal connection among them. Within each snapshot, only the predictions for a *single* timestamp are queried.

Forecasters. Most of the current forecasting models are based on encoder-decoder architectures. The encoder takes as input the past locations $C_{t-\tau_h+1:t}$ and the HD-Map M to generate agent features F_t. The decoder then predicts the trajectories $P_{t:t+\tau_f}$ from the agent features F_t. That is,

$$F_t = \texttt{Encoder}(C_{t-\tau_h+1:t}, M), \quad P_{t:t+\tau_f} = \texttt{Decoder}(F_t). \quad (1)$$

We denote a snapshot-based forecasting model as $\texttt{Model} = \{\texttt{Encoder}, \texttt{Decoder}\}$.

B. Our Formulation

Although it is widely used, the snapshot-based formulation is incompatible with the "streaming" nature of the real world. We thus propose a *streaming forecasting* formulation that evaluates a forecasting algorithm on continuous timestamps instead of a single run. As shown in Fig. 2b, streaming forecasting iteratively executes two steps: (1) *Input* the positions of agents; (2) *Query* the corresponding future trajectories.

Input. We define A_t as the set of IDs of all the agents that have ever appeared up to time t (details of agent selection are in Sec. III-C). At time t, the input consists of the 3D coordinates c^a_t and visibility v^a_t for every agent $a \in A_t$. If agent a is visible or present at time t, we set v^a_t as true; otherwise, we set v^a_t as false and $c^a_t \leftarrow \varnothing$. We represent all the input data at time t as $D_t = \{(c^a_t, v^a_t)\}_{a \in A_t}$, and $D_{t-\tau_h+1:t}$ denotes all the input data between frames $t-\tau_h+1$ and t.

Query. Streaming forecasting queries the future trajectories of agents in A_t on every timestamp. We adopt the same multi-modality setting with K futures for each agent as the conventional formulation in Sec. III-A. Thus, the predictions for agent $a \in A_t$ are $P^a_{t:t+\tau_f} = \{p^{a,k}_{t:t+\tau_f}\}^K_{k=1}$. The corresponding ground truth is $G_{t:t+\tau_f} = \{g^a_{t:t+\tau_f}\}_{a \in A_t}$, which comes from the observations in future frames $D_{t:t+\tau_f}$ of the streaming data.

Evaluated agents. Our streaming forecasting queries and evaluates all the agents in A_t, further setting it apart from conventional forecasting benchmarks that only concern a subset of target agents. While selecting specific target agents may help to focus on interesting movements, pre-assuming target agents can be risky in safety-critical applications. For similar considerations, joint perception and forecasting studies [4][11][23][26] often evaluate all the agents.

By evaluating all the agents, our streaming forecasting can capture the challenge of "*forecasting for occluded agents*." By contrast, previous forecasting benchmarks implicitly assume the visibility of target agents and overlook this issue. Note that occluded agents also have available ground truth for future trajectories in streaming forecasting, because *they can re-appear after occlusion*. This makes streaming forecasting a natural scenario for investigating occlusions and addressing the challenges they pose.

C. Argoverse-SF Benchmark

Guided by our formulation, we construct the Argoverse-SF benchmark for streaming forecasting. We make the following important design choices and contributions.

Using tracking instead of forecasting data. Autonomous driving datasets [2][5][6] typically have separate splits for tracking and forecasting. While using the forecasting data seems the obvious choice, such data have been processed for snapshot-based evaluation: predicting on a single timestamp instead of continuous timestamps, and filtering out noisy (*e.g.*, occluded) agents for a cleaner setup. By contrast, tracking data satisfy our purpose, as the input to forecasting components comes from an upstream tracker in autonomous driving. Therefore, tracking data are a more appropriate option for our benchmark to reflect the real world faithfully.

Argoverse vs. other datasets. We prioritize evaluating datasets that support algorithms based on modern vectorized HD-Maps. As Waymo [6] does not include HD-Maps in its tracking data, Argoverse [5] and nuScenes [2] become our main options. We ultimately choose Argoverse, because more forecasters using vectorized HD-Maps are studied on Argoverse than nuScenes.

Agent positions. As for the agent positions in our benchmark, instead of relying on the tracking results from a specific tracker, we leverage high-quality ground truth from Argoverse to create an "oracle" tracker. Specifically, we use the centers of ground truth bounding boxes as the input positions. This design avoids the influence of specific trackers and enables us to concentrate on the difficulties of forecasting. For instance, the occlusions in our Argoverse-SF are faithful in that they reflect the actual invisibility of occluded agents and cannot be filled in by human annotators. We further control the set of agents A_t by using the oracle life-cycle management [25][29]: an agent is removed from the set of agents only if it does not re-appear.

Selection of agents. For a realistic setup and fair assessment of forecasting algorithms, Argoverse-SF carefully selects the agents. First, agents outside the perception range

TABLE I: The number of queried predictions ("Queries") and the total number of agents to which these predictions belong ("Agents") in Argoverse-SF.

(a) Statistics of Argoverse-SF training set.

	Moving-Visible	Moving-Occluded	Static-Visible	Static-Occluded
Queries	90814	8342	103838	20875
Agents	1042	520	2424	1494

(b) Statistics of Argoverse-SF validation set.

	Moving-Visible	Moving-Occluded	Static-Visible	Static-Occluded
Queries	29124	2968	47957	5069
Agents	350	187	742	451

(>100m) or far from the road (outside the regions of interest in Argoverse) are excluded. Second, Argoverse-SF begins querying at $\tau_h = 20$ frame, ensuring that the benchmark does not create short histories artificially. Finally, Argoverse-SF only includes automobile agents such as vehicles, buses, and trailers, aligning with the Argoverse forecasting dataset, and we leave the implementation of additional agent types and datasets as future work.

Dataset splits. Argoverse-SF adopts the same training and validation sets as Argoverse's tracking dataset: 65 and 24 sequences for training and evaluation, respectively. We use all the frames in the logs of Argoverse, which are 10Hz sensor inputs with a duration of 15-30 seconds. The training and validation sets have 9,937 and 3,839 timestamps with valid queries of future trajectories, respectively. Note that Argoverse-SF allows using Argoverse's forecasting dataset for pre-training forecasters. This is useful, because the forecasting dataset contains a curated collection of diversified trajectories that are necessary for learning effective initialization and comparing fairly with existing models.

Evaluation metrics. Argoverse-SF quantitatively evaluates the performance of the predictions $P_{t:t+\tau_f}$ by comparing them with the ground truth $G_{t:t+\tau_f}$, which is the observations of agents' positions in future frames $D_{t:t+\tau_f}$ (Sec. III-B). We adopt the commonly-used evaluation metrics of minimum final displacement error (minFDE), minimum average displacement error (minADE), and miss rate (MR). However, due to the frequent occlusion and de-occlusion of agents in streaming data, some frames and agents may not have ground truth. Therefore, we only compute the metric values for timestamps with available ground truth, *i.e.*, where the visibility mask v_t^a is true. For example, the minFDE for agent a is calculated as:

$$\text{minFDE}(a) = \frac{1}{|T^a|} \sum_{t \in T^a} \text{minFDE}(\{p_{t:t+\tau_f}^{a,k}\}_{k=1}^{K}, g_{t:t+\tau_f}^{a}), \quad (2)$$

where T^a is the set of all the queried timestamps with valid ground truth for agent a: $T^a = \{t | v_{t+\tau_f}^a \text{ is true}\}$. For metrics like minADE, we adopt a "masked" version by performing average only on the frames with v_t^a being true.

Breakdown into subsets for evaluation. Conventional motion forecasting only considers moving and visible agents. By contrast, our streaming forecasting introduces three new types of agents: moving-occluded, static-visible, and static-occluded, and we evaluate all of them. Table I summarizes the statistics with respect to the number of agents and queries, showing that the four types (subsets) of agents have a highly-imbalanced distribution. Here, we intuitively define *moving* agents as those having an overall traveling distance greater than 3.0 meters. To avoid the domination of a single motion pattern or visibility, we introduce the four-subset division of movements and visibility for evaluation. Specifically, we first average the metrics within each subset A of the agents, calculated as "$\text{minFDE}(A) = (1/|A|)\sum_{a \in A}(\text{minFDE}(a))$," and then average the metrics across the four subsets to evaluate overall performance.

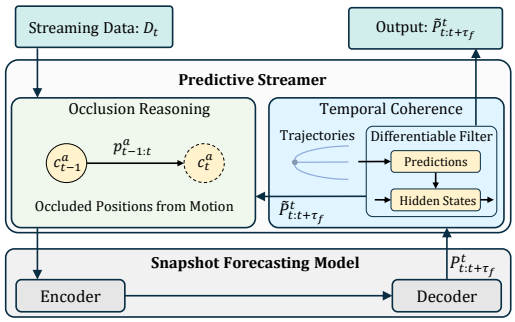

Fig. 3: Predictive streamer. It serves as an intermediate integration between snapshot-based forecasters and streaming data. The occlusion reasoning module estimates the positions of occluded agents from the predicted movement (Sec. IV-B). Differentiable filters (DFs) estimate future trajectories as their hidden states and refine the forecasting results through temporal coherence in adjacent frames (Sec. IV-C).

IV. ALGORITHMS

A. Pipeline of Streaming Forecasting

We propose a general pipeline to adapt a well-learned snapshot-based model seamlessly into streaming forecasting.

Three-stage training. (1) *Pretraining*. We train a standard snapshot-based model on the forecasting dataset of Argoverse to reasonably initialize the forecaster. (2) *Finetuning*. We extract snapshots from the Argoverse-SF training set to finetune the pretrained model. This mitigates the distribution shift between Argoverse's forecasting dataset and Argoverse-SF. (3) *Streaming training*. We design and train *predictive streamer* to better integrate with the streaming setup, detailed in Sec. IV-B and Sec. IV-C.

Streaming inference. To align with the interface of a snapshot-based forecaster, we integrate it into streaming data in a sliding window way. At frame t, the model takes as input the last τ_h frames, $D_{t-\tau_h+1:t}$, and predicts future τ_f frames, $P_{t:t+\tau_f}$. If our predictive streamer is available, it uses predictions and agent features from the forecaster and refines $P_{t:t+\tau_f}$ into $\tilde{P}_{t:t+\tau_f}$ as final predictions for output.

Baseline. Hallucinating the occluded positions is necessary to operate forecasting models on all the agents. Our baseline uses a Kalman filter to predict occluded positions, inspired by how 3D multi-object tracking addresses occlusions [25][29].

Overview of predictive streamers. Occlusion reasoning and temporal coherence are emergent challenges in streaming forecasting. The baseline described above lacks specialized

Algorithm 1 Algorithm for Occlusion Reasoning

Input:

$D_{1:t}$: The input data, $D_t = \{(c_t^a, v_t^a)\}_{a \in A_t}$ are coordinates and visibility masks (see Sec. III-B).

$P_{t-1:t+\tau_f-1}$: Predictions of `Decoder` at frame $t-1$.

M: HD-Maps.

Output:

$\widetilde{P}_{t:t+\tau_f}$: Predictions at frame t.

for arbitrary agent $a \in A_t$ **do**
  if v_t^a is false (a is invisible) **then**
    Choose the most confident $\widehat{p}_{t:t+\tau_f}^a$ from $p_{t:t+\tau_f}^a$
    $c_t^a \leftarrow c_{t-1}^a + \widehat{p}_{t-1:t}^a$
  end if
end for
Encode agent features: $F_t \leftarrow \texttt{Encoder}(D_{t-\tau_h+1:t}, M)$
Multi-future forecasting: $P_{t:t+\tau_f} \leftarrow \texttt{Decoder}(F_t)$
$P_{t:t+\tau_f}$ acts as the final result: $\widetilde{P}_{t:t+\tau_f} \leftarrow P_{t:t+\tau_f}$
return $\widetilde{P}_{t:t+\tau_f}$

solutions to address these issues. Therefore, our *predictive streamer* accommodates these challenges and acts as a general intermediate component as in Fig. 3, better integrating a snapshot-based forecaster into the streaming setup.

B. Forecasting for Occluded Agents

Occlusion reasoning with multi-modal predictions. Estimating the occluded positions is the prerequisite of forecasting for those agents. Its common approach is adding predicted motion to previously visible positions. Although our Kalman filter baseline (Sec. IV-A) follows this intuition, it is insufficient in that the Kalman filter is unable to handle complex motions and contextual information, such as HD-Maps and the movements of surrounding agents. Therefore, we propose to leverage the forecasting results from snapshot-based models to provide more accurate estimation. In practice, a special branch $\widehat{p}_{t:t+\tau_f}^t$ from multi-modal trajectories is selected according to the highest confidence scores. The movement in $\widehat{p}_{t:t+\tau_f}^t$ propagates the agents' positions through the occluded timestamp as in Algorithm 1. Specifically, if an agent a is occluded, we replace its invisible location with previous motion $c_{t-1}^a + \widehat{p}_{t-1:t}^a$. We highlight that the selection process connects the multi-modal trajectories with the single-modal input of forecasters. In this way, our approach retains the advantage of multi-modal trajectories from learning-based forecasters, including their diverse movements, HD-Map context, and cross-agent interaction.

Discussion. Our occlusion reasoning is different from prior 3D perception work [26][25][29] under a forecasting context. First, we leverage *multi-modal* trajectories, which cover a larger spectrum of distributions and thus outperform single-modal trajectories (empirically validated in Sec. V-C). Second, while previous studies avoid updating predictions on occluded frames due to unreliable information, our streaming module updates $p_{t:t+\tau_f}$ on every timestamp, providing a solution for handling long occlusions ($>\tau_f$ frames) and incorporating the latest information for instant reactions. The remaining unreliability challenge is addressed by leveraging temporal coherence, which is discussed in Sec. IV-C.

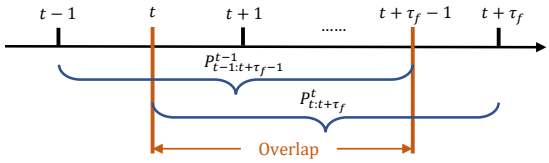

Fig. 4: Illustration of temporal coherence. The predictions on adjacent frames ($t-1$ and t) have a large overlap, and such consistency is useful for improving the forecasting quality.

C. Differentiable Filters for Temporal Coherence

Temporal coherence. As streaming forecasting operates on continuous frames, adjacent timestamps have strong temporal consistency because of their large overlaps, as in Fig. 4. Suppose we use the superscript in $\widetilde{p}_{t:t+\tau_f}^t$ to denote predictions originating from t. The trajectories on $t-1$, denoted as $\widetilde{p}_{t-1:t+\tau_f-1}^{t-1}$, and the trajectories on t, denoted as $\widetilde{p}_{t:t+\tau_f}^t$, should be aligned between frames t and $t+\tau_f-1$, which cover most of their prediction horizons. To account for the predictions in nearby frames, *our key insight* is that differentiable filters (DFs) can effectively enhance temporal coherence by recursively fusing predictions into their hidden states.

Background of DFs. The basis of DFs is Bayes filtering, *e.g.*, Kalman filters [15]. It considers a *process model* and an *observation model* as Eqn. 3, where x_t denotes hidden states, z_t denotes observations and the assumptions are linear dynamics, no control signals, and Gaussian distribution. Additionally, w and δ are noises following distributions $N(0,Q)$ and $N(0,R)$, where Q and R are covariance matrices.

$$x_t = Ax_{t-1} + w, \text{ and } z_t = Cx_t + \delta. \tag{3}$$

The filter recursively approximates the posterior distribution of x_t in the form of $N(\mu_{x_t}, \Sigma_{x_t})$. During this process, DF [12] proposes using a neural network ϕ to infer low-dimensional z_t from high-dimensional sensor inputs or generate adaptive covariance R. As the Kalman filter is a differentiable process, the neural network ϕ can be learned jointly with it.

DFs for motion forecasting. In the case of motion forecasting, we treat future trajectories $p_{t:t+\tau_f}^t$ as both the hidden states x_t and observation z_t in DF. Then DF represents the hidden states with a Gaussian distribution $N(\mu_{x_t}, \Sigma_{x_t})$ and recursively estimates the mean μ_{x_t} of the hidden states x_t as refined forecasting results, denoted as $\widetilde{p}_{t:t+\tau_f}^t$. We illustrate this process as follows:

$$\mu_{x_t} \leftarrow \texttt{DF}(p_{t:t+\tau_f}^t, \mu_{x_{t-1}}, \Sigma_{x_{t-1}}), \text{ then } \widetilde{p}_{t:t+\tau_f}^t \leftarrow \mu_{x_t}. \tag{4}$$

In DF, we first use a learnable neural network ϕ_R to predict adaptive covariance R_t for varied agents and timestamps based on their features F_t, which enables DF to fuse the highly stochastic trajectories properly. Then we feed the

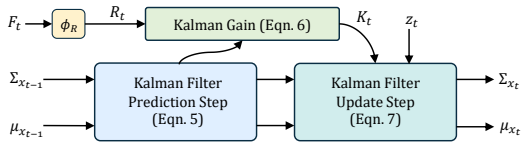

Fig. 5: Computation of differentiable filters. A neural network ϕ_R predicts the observation covariance R_t from agent features. Then a Kalman filter fuses the latest predictions into the hidden states accordingly.

predicted R_t into a standard Kalman filter to update $\mu_{x_{t-1}}$ as in Fig. 5, according to the following filtering steps.

$$\widetilde{\mu}_{x_t} \leftarrow A\mu_{x_{t-1}},\ \widetilde{\Sigma}_{x_t} \leftarrow A\Sigma_{x_{t-1}}A^\top + Q_t \text{ [Prediction]} \quad (5)$$

$$K_t \leftarrow \widetilde{\Sigma}_{x_t}C^\top(C\widetilde{\Sigma}_{x_t}C^\top + R_t)^{-1} \text{ [Kalman gain]} \quad (6)$$

$$\mu_{x_t} \leftarrow \widetilde{\mu}_{x_t} + K_t(z_t - C\widetilde{\mu}_{x_t}), \Sigma_{x_t} \leftarrow (\mathbb{I} - K_tC)\widetilde{\Sigma}_{x_t} \text{ [Update]} \quad (7)$$

We further discover that *specifying inter-frame dynamics* is the key to capturing temporal coherence. Taking the movements on X-axis for example, denoted as $\widetilde{p}^{t,x}_{t:t+\tau_f}$, we define the matrix A in the process model of DF (Eqn. 3) as:

$$A \leftarrow \left[\begin{array}{c|c} 0_{(\tau_f-1)\times 1} & \mathbb{I}_{(\tau_f-1)\times(\tau_f-1)} \\ \hline 0_{1\times 1} & 1_{1\times 1} \end{array}\right],\ \widetilde{p}^{t,x}_{t:t+\tau_f} \leftarrow A\widetilde{p}^{t-1,x}_{t-1:t+\tau_f-1}. \quad (8)$$

The above matrix A uses the identity matrix $\mathbb{I}_{(\tau_f-1)\times(\tau_f-1)}$ to indicate that the overlapped forecasting frames, *i.e.*, $\widetilde{p}^{t,x}_{t:t+\tau_f-1}$ and $\widetilde{p}^{t-1,x}_{t:t+\tau_f-1}$, should be consistent. The bottom-right $1_{1\times 1}$ means that the process model pads the additional timestamp $t+\tau_f$ with the last movement on the previous timestamp, where we assume that motion cannot change drastically. As for C in the observation model, we define it as an identity matrix $\mathbb{I}$, because the hidden states x_t and observations z_t both represent future movements.

For implementation, we follow prior work [12][18] in treating Q_t as a fixed hyper-parameter. Moreover, as our future trajectories have larger dimensions than conventional state estimation tasks, we assume R_t is diagonal and predict it by $(\phi_R(F_t))^2$ to avoid invalid covariance matrices.

Training and inference. We train DF using the same forecasting loss penalizing the distance between predictions and ground truth (details in Sec. V-A). During inference, we recursively apply DF as in Fig. 5 to refine every trajectory.

Discussion. Our DF in predictive streamer not only improves trajectory prediction but also highlights the temporal coherence property of streaming forecasting. In response to the question from the discussion of Sec. IV-B, our DF-based approach allows the occluded frames to aggregate the information from previous visible frames, which addresses the unreliability issue of hallucinated occluded positions.

V. EXPERIMENTS

A. Implementation Details

Snapshot forecasting models. We use two encoder/decoder architectures for generalizability: VectorNet [7] and mmTransformer [21]. To ensure a fair comparison, we *double the number of layers* of the original VectorNet, as it was proposed earlier and relatively light-weight. Our loss functions follow the *winner-takes-all* strategy, where the best branch is selected from the multi-modal trajectories based on the smallest distance to ground truth. Then a cross-entropy loss supervises its confidence score to approach 1, and a smooth L1 loss penalizes its distance to the ground truth. Our re-implementation of VectorNet and mmTransformer outperforms their reported results in [7][21].

Differentiable filters. ϕ_R adopts the same multi-layer perceptron (MLP) architecture as the decoder in VectorNet. During streaming training, we freeze the encoder and decoder of the snapshot-based forecaster and supervise DFs with the same smooth L1 loss for multi-modal trajectories.

Model training. We introduce the details of three-stage training. (1) *Pretraining*: we use AdamW [22], 5e-4 as learning rate, 1e-4 as weight decay, and train for 24 epochs. (2) *Finetuning*: we use AdamW, 1e-4 as learning rate, 1e-4 as weight decay, and train for 8 epochs. The training samples come from Argoverse-SF, and every iteration involves *forecasting on 5 adjacent frames*. (3) *Streaming training*: we freeze the snapshot encoder and decoder and adopt the same configuration as the finetuning stage.

B. Ablation Studies

Predictive streamer. We analyze *predictive streamer* in Table II and Table III based on the encoders and decoders from VectorNet and mmTransformer, respectively. As illustrated, *every streaming strategy significantly improves the forecasting quality*, especially for the minFDE and minADE. We highlight that DF is vital for better predictions. It refines occlusion reasoning by fusing the results from previous reliable frames. Furthermore, better temporal coherence also improves the trajectories of visible agents without modifying the underlying encoders and decoders, which is plug-and-play for deploying snapshot forecasters.

Properties of streaming datasets. Before further analysis, we clarify the differences between a snapshot-based dataset and our streaming datasets, which are reflected in Table II and Table III and subtle to notice at first look. (1) The minADE for occluded cases can be larger than minFDE. This is because minADE computes more occlusion cases than minFDE in Argoverse-SF. Concretely, we adopt a "masked" minADE that computes the metrics once there is one frame with ground truth (as in Sec. III-C, evaluation metrics); thus, any prediction accounted by minFDE is also considered by minADE, but not the reverse. (2) The static agents (excluded in the table due to space limit) have much smaller minADE and minFDE values and decrease the scales of changes in overall metrics. (3) We compare the performance between VectorNet and mmTransformer, where mmTransformer is better for visible agents and K=1 cases, but fails in the K=6 occlusion scenarios. This is because the queries of mmTransformer are tailored to distinct movement patterns, and their less confident directions have a weaker ability to adapt to noisy occlusions. Even so, our predictive streamer consistently improves for all scenarios.

TABLE II: Analysis of predictive streamer with VectorNet. We analyze the overall metrics (primary) and the breakdown metrics specifically for moving agents (secondary). Beginning from the baseline of directly adapting snapshot models with a Kalman filter (Sec. IV-A), we use the most confident trajectory for occlusion reasoning ("OCC") and learn a differentiable filter to improve the temporal coherence ("DF"). Our strategies significantly improve the performance, indicating the benefits of viewing motion forecasting from a streaming perspective. More analysis is in Sec. V-B.

(a) Evaluation metrics (K=6) with VectorNet.

OCC	DF	Overall			Moving-Occluded			Moving-Visible		
		minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
		1.67	1.64	0.20	4.05	4.59	0.50	1.70	1.09	0.25
✓		1.43	1.24	0.18	3.22	3.17	0.44	1.70	1.09	0.25
✓	✓	**1.37**	**1.20**	**0.17**	**3.03**	**3.07**	**0.41**	**1.67**	**1.07**	**0.24**

(b) Evaluation metrics (K=1) with VectorNet.

OCC	DF	Overall			Moving-Occluded			Moving-Visible		
		minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
		3.90	2.57	0.37	9.39	7.23	**0.75**	4.63	1.94	0.63
✓		3.42	2.05	**0.36**	7.84	5.42	**0.75**	4.63	1.94	0.63
✓	✓	**3.32**	**1.98**	**0.36**	**7.42**	**5.16**	0.76	**4.57**	**1.89**	**0.62**

TABLE III: Analysis of predictive streamer with mmTransformer. Abbreviations are the same as Table II. Our algorithm improves the forecasting quality on mmTransformer, further supporting the significance of exploring streaming forecasting.

(a) Evaluation metrics (K=6) with mmTransformer.

OCC	DF	Overall			Moving-Occluded			Moving-Visible		
		minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
		1.74	1.61	0.19	4.27	4.48	0.49	1.70	1.07	0.23
✓		1.61	1.37	0.18	3.89	3.67	0.43	1.70	1.07	0.23
✓	✓	**1.54**	**1.30**	**0.17**	**3.60**	**3.40**	**0.41**	**1.66**	**1.05**	0.23

(b) Evaluation metrics (K=1) with mmTransformer.

OCC	DF	Overall			Moving-Occluded			Moving-Visible		
		minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
		3.78	2.44	0.38	9.25	6.83	0.80	4.25	1.81	0.59
✓		3.30	1.98	**0.35**	7.42	5.20	**0.77**	4.24	1.80	0.58
✓	✓	**3.25**	**1.93**	**0.35**	**7.21**	**4.94**	**0.77**	**4.22**	**1.78**	**0.57**

TABLE IV: Comparing DF with Kalman filter (KF) and LSTM for temporal coherence. Our DF outperforms both KF and LSTM, especially for occlusion cases.

(a) Overall metrics.

Model	Overall (K=6)			Overall (K=1)		
	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
Snapshot	1.67	1.64	0.20	3.90	2.57	0.37
KF	1.42	1.24	0.18	3.39	2.03	0.37
LSTM	1.56	1.39	0.18	3.75	2.30	**0.36**
DF (Ours)	**1.37**	**1.20**	**0.17**	**3.32**	**1.98**	**0.36**

(b) Breakdown metrics.

Model	Moving-Occluded (K=6)			Moving-Visible (K=6)		
	minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
Snapshot	4.05	4.59	0.50	1.70	1.09	0.25
KF	3.13	3.15	0.42	1.70	1.09	0.25
LSTM	3.74	3.78	0.46	1.68	**1.07**	0.25
DF (Ours)	**3.03**	**3.07**	**0.41**	**1.67**	**1.07**	**0.24**

Three-stage training. The three-stage procedure of adapting forecasters, which is proposed in Sec. IV-A, is necessary: finetuning improves the performance, where the overall minFDE decreases from 3.07m to 1.67m and the overall MR drops from 25% to 20%.

C. Quantitative Analysis

We conduct several analyses for our design choices and provide further insights for streaming forecasting. By default, we use the VectorNet encoder and decoder.

Comparing DF with KF and LSTM. We compare the effects of DF and LSTM, because LSTM is a common approach for temporal coherence [12][18]. Specifically, we replace the DF in our predictive streamer with a 4-layer LSTM to enhance F_t, and jointly train it with the decoder. For a fair comparison, both DF and LSTM *adopt the occlusion reasoning* described in Sec. IV-B. As shown in Table IVa, LSTM cannot outperform our DF. After more detailed analysis in Table IVb, we find that LSTM can slightly improve over the snapshot models on visible cases, but not as large as DFs, because DF explicitly specifies temporal coherence by a process model (Eqn. 8). The main disadvantage of LSTMs is on the occluded cases because the problematic features for occluded agents can corrupt the hidden states of LSTM and influence the features in subsequent frames. We also compare DF to the baseline of Kalman filters, where DF has a better performance by learning adaptive covariance matrices.

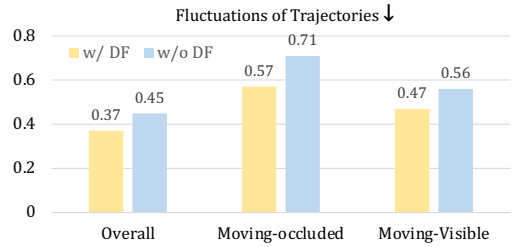

Fig. 6: DF decreases the fluctuations of trajectories by 10%-20%. Y-axis is measured by meters/frame.

DF improves temporal coherence. We mimic [16] in defining a second-order metric "fluctuation" to analyze the temporal coherence. Specifically, we convert predictions on $t-1$ and t into their absolute coordinates $\widehat{c}^{t-1}_{t-1:t+\tau_f-1}$ and $\widehat{c}^{t}_{t:t+\tau_f}$ and compute their distance in overlapped frames $\frac{1}{\tau_f-1}\|\widehat{c}^{t-1}_{t:t+\tau_f-1} - \widehat{c}^{t}_{t:t+\tau_f-1}\|_2$. As in Fig. 6, DF decreases the fluctuation by a large margin.

Occlusion reasoning is better with multi-modality. We compare our estimation of occluded positions with previous 3D perception strategies that capitalize occlusion reasoning [25][26][29]. The main distinction is that we utilize multi-modal predictions. To validate the benefits of multi-modality, we integrate an additional single-modal decoder $\texttt{Decoder}_O$ and a DF into our predictive streamer for occlusion reasoning. As in Table V, *multi-modal predictions are beneficial to estimating invisible positions*. The reason is that multi-modal trajectories can cover a larger spectrum of futures. We also notice that the improvement of DF generalizes to both single-modal and multi-modal trajectories.

D. Qualitative Analysis

We first illustrate the intuition of streaming forecasting in Fig. 7a, where our predictions gradually adapt to an agent's incoming observations. For example, the trajectories have smaller divergence on $t+3$ than on t. Furthermore,

TABLE V: Multi-modality trajectories are better for occlusion reasoning. The metrics are evaluated under K=6. Using DF also enhances the performance for both single-modal and multi-modal predictions.

Modality	DF	Overall			Moving-Occluded		
		minFDE↓	minADE↓	MR↓	minFDE↓	minADE↓	MR↓
Single	✗	1.53	1.36	**0.17**	3.54	3.59	0.42
	✓	1.40	1.25	**0.17**	3.14	3.20	**0.41**
Multiple	✗	1.43	1.24	0.18	3.22	3.17	0.44
	✓	**1.37**	**1.20**	**0.17**	**3.03**	**3.07**	**0.41**

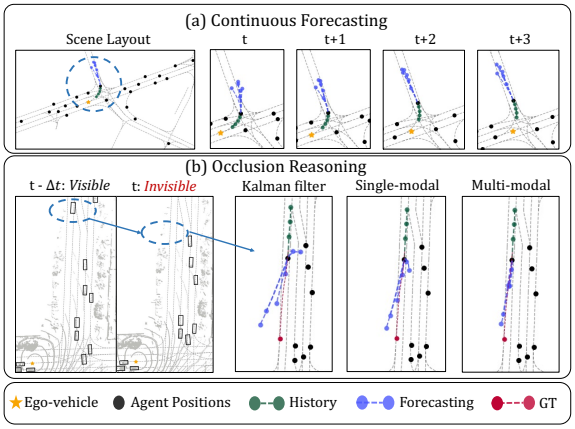

Fig. 7: (**Best viewed in color**) We visualize the streaming forecasting results. Only the interesting agent is visualized for clarity. (a) When forecasting proceeds in continuous frames, the trajectories (blue dashed lines) gradually adapt and change according to the latest agent positions. (b) To deal with an occluded agent, using multi-modal trajectories is better than the conventional Kalman filters or single-modal trajectories, because the predicted trajectories (blue dashed lines) are closer to the ground truth (red lines). Please note that point clouds are only for visualization purposes.

we demonstrate the challenge of occlusion reasoning in Fig. 7b. Although the errors for occluded positions are subtle, the Kalman filter and single-modal methods have seriously wrong predictions, while our multi-modal method has relatively larger fidelity. Please check our demo for more visualizations.

VI. CONCLUSIONS AND FUTURE WORK

This paper proposes a novel perspective to study motion forecasting: *streaming forecasting*. By repurposing tracking data, we capture the intrinsic challenges of *forecasting for occluded agents* and *temporal coherence* in real-world traffic, which are overlooked by previous snapshot-based setups. In addition, we propose general *predictive streamers* that leverage multi-modal forecasting to address occlusions and introduce differentiable filters to enhance temporal continuity. These solutions and analyses intend to generate interest in studying motion forecasting in a realistic streaming setting.

As the first study on streaming forecasting, we expect future works to improve predictive streamers for miss rates. In addition, the benchmark is also extensible to more types of agents and larger datasets.

Acknowledgement. This work was supported in part by NSF Grant 2106825, NIFA Award 2020-67021-32799, the Jump ARCHES endowment, the NCSA Fellows program, the IBM-Illinois Discovery Accelerator Institute, the Illinois-Insper Partnership, and the Amazon Research Award.

REFERENCES

[1] P. Abbeel *et al.*, "Discriminative training of Kalman filters." in *RSS*, 2005.
[2] H. Caesar *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.
[3] ——, "NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles," *arXiv preprint arXiv:2106.11810*, 2021.
[4] S. Casas *et al.*, "MP3: A unified model to map, perceive, predict and plan," in *CVPR*, 2021.
[5] M.-F. Chang *et al.*, "Argoverse: 3D tracking and forecasting with rich maps," in *CVPR*, 2019.
[6] S. Ettinger *et al.*, "Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset," in *ICCV*, 2021.
[7] J. Gao *et al.*, "VectorNet: Encoding HD maps and agent dynamics from vectorized representation," in *CVPR*, 2020.
[8] T. Gilles *et al.*, "Home: Heatmap output for future motion estimation," in *ITSC*, 2021.
[9] ——, "Gohome: Graph-oriented heatmap output for future motion estimation," in *ICRA*, 2022.
[10] J. Gu *et al.*, "DenseTNT: End-to-end trajectory prediction from dense goal sets," in *ICCV*, 2021.
[11] ——, "ViP3D: End-to-end visual trajectory prediction via 3D agent queries," *CVPR*, 2023.
[12] T. Haarnoja *et al.*, "Backprop KF: Learning discriminative deterministic state estimators," *NeurIPS*, 2016.
[13] A. Hu *et al.*, "FIERY: Future instance prediction in bird's-eye view from surround monocular cameras," in *ICCV*, 2021.
[14] R. Jonschkowski *et al.*, "Differentiable particle filters: End-to-end learning with algorithmic priors," *RSS*, 2018.
[15] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.
[16] A. Kanazawa *et al.*, "Learning 3D human dynamics from video," in *CVPR*, 2019.
[17] P. Karkus *et al.*, "Particle filter networks with application to visual localization," in *CoRL*, 2018.
[18] A. Kloss *et al.*, "How to train your differentiable filter," *Autonomous Robots*, 2021.
[19] M. A. Lee *et al.*, "Multimodal sensor fusion with differentiable filters," in *IROS*, 2020.
[20] M. Liang *et al.*, "Learning lane graph representations for motion forecasting," in *ECCV*, 2020.
[21] Y. Liu *et al.*, "Multimodal motion prediction with stacked transformers," in *CVPR*, 2021.
[22] I. Loshchilov *et al.*, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
[23] W. Luo *et al.*, "Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net," in *CVPR*, 2018.
[24] J. Ngiam *et al.*, "Scene transformer: A unified architecture for predicting multiple agent trajectories," *ICLR*, 2021.
[25] Z. Pang *et al.*, "SimpleTrack: Understanding and rethinking 3D multi-object tracking," *ECCVW*, 2021.
[26] ——, "Standing between past and future: Spatio-temporal modeling for multi-camera 3D multi-object tracking," *CVPR*, 2023.
[27] T. Salzmann *et al.*, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *ECCV*, 2020.
[28] B. Varadarajan *et al.*, "Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction," in *ICRA*, 2022.
[29] Q. Wang *et al.*, "Immortal tracker: Tracklet never dies," *arXiv preprint arXiv:2111.13672*, 2021.
[30] B. Yi *et al.*, "Differentiable factor graph optimization for learning smoothers," in *IROS*, 2021.
[31] Y. Yuan *et al.*, "AgentFormer: Agent-aware transformers for socio-temporal multi-agent forecasting," in *ICCV*, 2021.
[32] P. A. Zachares *et al.*, "Interpreting contact interactions to overcome failure in robot assembly tasks," in *ICRA*, 2021.
[33] H. Zhao *et al.*, "TNT: Target-driven trajectory prediction," in *CoRL*, 2021.
[34] Z. Zhou *et al.*, "Query-centric trajectory prediction," in *CVPR*, 2023.