3D Multi-Object Tracking with Semi-Supervised GRU-Kalman Filter

Xiaoxiang Wang¹, Jiaxin Liu², Miaojie Feng³, Zhaoxing Zhang⁴ and Xin Yang*

Abstract-3D Multi-Object Tracking (MOT), a fundamental component of environmental perception, is essential for intelligent systems like autonomous driving and robotic sensing. Although Tracking-by-Detection frameworks have demonstrated excellent performance in recent years, their application in realworld scenarios faces significant challenges. Object movement in complex environments is often highly nonlinear, while existing methods typically rely on linear approximations of motion. Furthermore, system noise is frequently modeled as a Gaussian distribution, which fails to capture the true complexity of the noise dynamics. These oversimplified modeling assumptions can lead to significant reductions in tracking precision. To address this, we propose a GRU-based MOT method, which introduces a learnable Kalman filter into the motion module. This approach is able to learn object motion characteristics through data-driven learning, thereby avoiding the need for manual model design and model error. At the same time, to avoid abnormal supervision caused by the wrong association between annotations and trajectories, we design a semi-supervised learning strategy to accelerate the convergence speed and improve the robustness of the model. Evaluation experiment on the nuScenes and Argoverse2 datasets demonstrates that our system exhibits superior performance and significant potential compared to traditional TBD methods. The code is available at https://github.com/xiang-1208/GRUTrack.

I. INTRODUCTION

Multi-Object Tracking (MOT) [1] is a crucial research topic within the field of computer vision and serves as a foundational technology in numerous intelligent applications, such as autonomous driving, traffic flow analysis, security surveillance, robotics, and action recognition.

At present, with the increasing performance of Multi-Object Detection (MOD) [2]–[5], Multi-Object Tracking (MOT) methods based on the "Tracking-by-Detection" (TBD) [1], [6], [7] have demonstrated superior accuracy and robustness. These TBD methods follow the motion process of the tracked object, contrasting with "Joint Detection and Tracking" (JDT) approaches [2], [8], [9] which do not perform as well. In general, TBD methods update the state of tracked objects incrementally by constructing motion model and employing recursive Bayesian filter estimator. However, due to the varied motion characteristics of different objects within the scene, a single state space (SS) and estimator parameter cannot well match the different motion characteristics between various categories, which reduces the consistency between the motion state update and the actual,

leading to false matching and inaccurate state update. Some approaches [7], [10], [11] take note of this and design motion parameters or association strategies for each class to be more relevant to the different characteristics of different classes.

However, these methods fail to fundamentally solve the problem of multi-category differences. On the one hand, with the continuous addition or refinement of categories, it is not only tedious to design motion models for each category. but also too dependent on the designer's experience. On the other hand, although it is possible to capture different types of motion characteristics by continuously refining the categories, most methods still use model-based state filters, such as Kalman Filter (KF) [12] and Extended Kalman Filter (EKF) [13], [14]. The effectiveness of these models depends on the accuracy of the state model and the validity of the motion hypothesis. In the actual MOT, the latent state of the system is nonlinear and complex, and it is even difficult to be accurately described as a tractable state model. In this case, model-based state estimators typically simplify the motion dynamics by linearizing the process and assuming the system noise follows a Gaussian distribution. This assumption does not match the actual situation, and these modeling inaccuracies often bring the loss of tracking precision to the system.

To this end, we propose a partially learnable MOT method by introducing a Gated Recurrent Unit (GRU)-based Kalman filter into the motion module of TBD. This method can replace the traditional manual model design in a data-driven manner, thus eliminating the need to design a unique SS and estimator for each class. Specifically, we use multiple GRUs to simulate the loops in Recursive Bayesian Filtering. The model automatically learns the noise distribution, state transition matrix and observation matrix. It avoids the mismatch of noise modeling and the loss of precision caused by linearizing the state transition and observation function. This is doable in theory because neural network-based state estimation has been shown to capture the motion characteristics of complex processes [15], [16], which is also applicable to state transitions in MOT.

However, it is not feasible to directly use the learnable Kalman Filter in MOT. On the one hand, due to the partial annotation of the dataset, the amount of trainable data is small, which is easy to overfitting. On the other hand, since the annotations and trajectories are associated by a hand-designed association strategy, the errors of association will introduce error supervision into the system. Therefore, we propose to parallel a Kalman filter during the training process to generate pseudo-labels for those unlabeled data for semi-supervised training. The experimental results prove

^{*}Corresponding author

¹Xiaoxiang Wang, ²Jiaxin Liu, ³Miaojie Feng, ⁴Zhaoxing Zhang and *Xin Yang are with School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China. {xiaoxiang0012,1jx123,fmj,zzx,xinyang2014}@hust.edu.cn

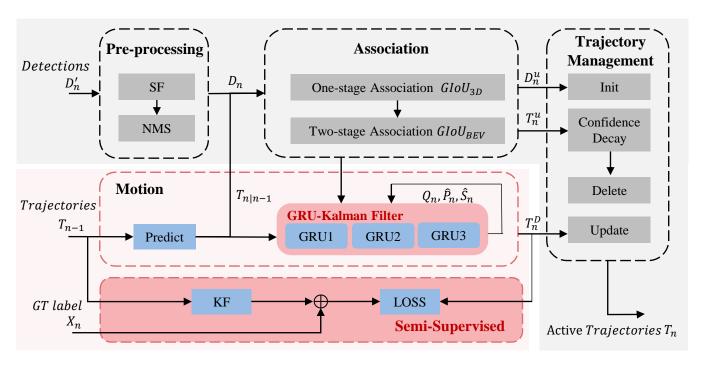


Fig. 1. The pipeline of our proposed method at frame n. T_n^D is the trajectories updated by associating upper observations D_n and using the motion module. Our design focuses on two parts, one is **GRU-Kalman Filter**: it uses three GRUs to simulate the Kalman filtering process. The second is **Semi-Supervised learning**, which uses dataset annotations and pseudo-labels generated by a parallel Kalman filter for joint training.

the effectiveness of our strategy.

Specifically, our contributions are as follows:

- We propose a data-driven MOT method by using a GRU-based motion module to avoid the precision loss of traditional methods for noise mismatch modeling and motion process linearization.
- We design a pseudo-label-based semi-supervised method, which greatly expands the amount of available training data and label robustness, so that the system can converge in fewer training cycles.
- We evaluate our method on the nuScenes [17] and Argoverse2 [18] datasets. Our system demonstrates performance comparable to traditional MOT systems and strong level of generalization, while obviating the need for manually designing a model for each object category.

II. RELATED WORK

Lidar-based 3D MOT has a similar form to image-based MOT. However, the lidar point cloud provides more precise spatial and depth information, significantly enhancing tracking precision. This improvement has led to its widespread adoption in robotic sensing and other applications. Most Lidar-based 3D MOT algorithms can be categorized into two primary paradigms: "Tracking-by-Detection" and "Joint Detection and Tracking".

A. TBD

Leveraging increasingly advanced detectors, AB3DMOT [1] establishes a baseline method for 3D MOT based on

filters and 3D Intersection over Union (3D IoU), which provides a foundation for the design of TBD methods.

SimpleTrack [6] offers a detailed analysis of the strengths and limitations of various models, including the widely used Kalman filter and the constant velocity model, as well as different data association metrics. It proposes specific improvements for each module and integrates them into a streamlined baseline method, achieving competitive results on the Waymo Open Dataset and nuScenes.

Additionally, Poly-MOT [7] incorporates geometric constraints into the motion model and develops multiple motion models tailored to the distinct characteristics of different object categories. It introduces three custom similarity measures and a novel two-stage data association strategy, enabling the identification of the most suitable similarity measure for each object category and reducing mismatches. This approach further optimizes TBD trackers, resulting in superior tracking performance on the NuScenes dataset.

Some methods aim to improve tracking performance by integrating image features. For instance, EagerMOT [10] combines image detection results with Lidar detection data to achieve comprehensive scene perception. CAMO-MOT [11] also leverages both camera and Lidar data to significantly mitigate tracking failures caused by occlusions and false detections. Additionally, CAMO-MOT introduces an occlusion head to effectively select optimal object appearance features multiple times, further reducing the impact of occlusions.

B. JDT

Motiontrack [19] proposed an end-to-end Transformerbased [20] JDT algorithm, which was based on the previous end-to-end detection work Transfusion [21], and further proposed a Transformer-based data association module and query enhancement module. In addition, following Guillem et al. [22], OGR3MOT [9] uses graph neural networks to solve the 3D MOT problem, and achieves the best IDS metrics so far.

C. Recursive Bayesian Filtering

In MOT, predicting and updating existing trajectories is a crucial aspect. KF [12] is used for linear Gaussian state models. However, in practice, many problems do not perfectly adhere to the linear Gaussian model. Consequently, nonlinear filters, such as the EKF [13], are employed for approximate processing. It performs recursive computation by linearizing the state forward function and the observation function.

The model-based algorithms aforementioned, which rely on accurate knowledge of the SS model, often experience significant performance degradation when there is a mismatch between the actual motion model and the modeled one [3]. Recently, there has been an increasing focus on integrating machine learning with SS models. DNN-based algorithms typically encode observations to fit a simplified SS model, and then track the parameters of these implicit SS, as exemplified by KFNet [23]. Additionally, some approaches [24] incorporate graph neural network (GNN) alongside Kalman filters to enhance filter accuracy through neural augmentation. However, such algorithms are often designed for unknown or highly complex SS models that lack mathematical interpretability and are generally not suited for real-time estimation due to their computational demands.

In contrast, KalmanNet [25] offers a novel approach by combining model-based Kalman filtering with RNN to address model mismatch and nonlinearity. Our work builds upon KalmanNet, applying it to practical MOT scenarios. This approach eliminates the need for manual design of multiple SS models and the selection of various filters for different classes, while avoiding additional computational complexity.

III. METHOD

A. 3D MOT Pipeline

Our system can be divided into four parts: the preprocessing module, motion module, association module, and trajectory management module, as shown in Fig. 1.

1) Pre-processing Module: 3D MOD typically generates multiple bounding boxes for the same detection to minimize the risk of missed and false detections. To prevent these detections from causing redundant ID switches, we preprocess the original detections D_n' to reduce false matches. This preprocessing generally involves applying score filtering and non-maximum suppression (NMS) to each frame's detections, retaining only the bounding box with the highest confidence for each object. After preprocessing, our detection boxes $D_n = [x, y, z, w, l, h, v, \theta]$ include the center position of the bounding box, its dimensions, its velocity, and the heading angle.

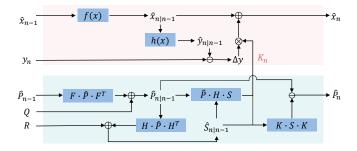


Fig. 2. EKF Block Diagram.

2) Motion Module: The motion module is primarily responsible for predicting the state $\hat{X}_{n-1} = [\hat{x}_n^1, \dots, \hat{x}_n^{num.tra}]$ of the tracked trajectories and updating the state based on observations $Y_n = [y_n^1, \dots, y_n^{num.det}]$. Our design focuses on this module.

The process of traditional motion and prediction, taking EKF as an example, is shown in Fig. 2. The system continuously updates the state through the prior state \hat{x}_{n-1} , the observation y_n , the process noise covariance matrix Q, the measurement noise covariance matrix R, and the continuously maintained state covariance P. Specifically, there are two steps:

a) Prediction Step: In this step, the system predicts the current prior state based on the posterior state from the previous time step, following the predefined SS model:

$$\hat{x}_{n|n-1} = f(\hat{x}_{n-1}). \tag{1}$$

Simultaneously, the uncertainty in the state, represented by the covariance, must also be predicted:

$$\hat{P}_{n|n-1} = F \cdot \hat{P}_{n-1} \cdot F^T + Q. \tag{2}$$

b) Update Step: The first step involves calculating the Kalman gain K_n , which balances the weights between the prediction and the observation:

$$K_n = \hat{P}_{n|n-1} \cdot H^T \cdot (H \cdot \hat{P}_{n|n-1} \cdot H^T + R)^{-1},$$
 (3)

$$\hat{S}_{n|n-1} = H \cdot \hat{P}_{n|n-1} \cdot H^T + R, \tag{4}$$

where $\hat{S}_{n|n-1}$ is the observation error covariance.

Again according to the current observation, update the current state of posterior:

$$\hat{x}_n = \hat{x}_{n|n-1} + K_n \cdot (y_n - h(\hat{x}_{n|n-1})). \tag{5}$$

Meanwhile, the error covariance is updated as follows:

$$\hat{P}_n = \hat{P}_{n|n-1} - K_n \cdot \hat{S}_{n|n-1} \cdot K_n^T.$$
 (6)

Here, the EKF uses the Jacobian matrix F and H to linearize the differentiable functions f(x) and h(x) in a time-dependent manner.

This approach relies heavily on the accuracy of the SS model setup, which often depends on the designer's experience and is difficult to transfer. Moreover, for nonlinear motion, the linearization of the state transition and observation equations can introduce additional errors. Additionally, in MOT, the Kalman filter requires manual adjustment of the observation noise and process noise, assuming they

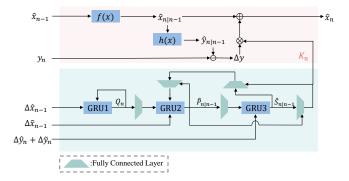


Fig. 3. The GRU-Kalman Filter Block Diagram. Here, GRU simulates the loop iteration of process noise Q_n , state error covariance $\hat{P}_{n|n-1}$ and observation error covariance $\hat{S}_{n|n-1}$ by inputting the state difference and observation difference, so as to reason about the Kalman gain K_n .

follow a multi-dimensional Gaussian distribution. However, in MOT, observations are derived from upstream object detection results, which do not necessarily conform to a Gaussian distribution. Thus, explicitly modeled noise may not adequately address the needs of MOT.

It is reasonable to believe that the differences in motion for different classes are often also reflected in the state. If the model can adaptively obtain different Kalman gain results based on the observations and state, it can then be applied to all classes and adjust the noise accordingly.

GRU-Kalman Filter: We introduce the learnable Kalman filter [25] into our system, using a uniform SS model across all classes. The specific neural network architecture is illustrated in Fig. 3. Following the design of KalmanNet, we represent each second-order statistical moment of the Kalman filter using separate GRU, with fully connected (FC) layers interspersed between the GRU, and dedicated input and output layers.

Specifically, we use the first GRU to track the process noise covariance matrix, which combines the previous process noise covariance matrix Q_{n-1} and the state forward update difference $\Delta \hat{x}_n = \hat{x}_n - \hat{x}_{n|n-1}$ to infer the current process noise covariance matrix Q_n .

The second GRU is employed to simulate the reasoning process of Eq. 2, thereby circumventing the need to linearize f(x) in order to design F. It combines the process noise covariance matrix and the forward evolution difference $\Delta \tilde{x}_n = \hat{x}_n - \hat{x}_{n-1}$, along with the previous state error covariance $\hat{P}_{n-1|n-2}$, to infer the current state error covariance $\hat{P}_{n|n-1}$.

The third GRU is designed to simulate the reasoning process of the observation formula in Eq. 4, avoiding the linearization of h(x) required to design H. It integrates the state covariance $\hat{P}_{n|n-1}$, the observation difference $\Delta \tilde{y}_n = y_n - y_{n-1}$, the innovation difference $\Delta y_n = y_n - \hat{y}_{n|n-1}$, and the previous observation error covariance $\hat{S}_{n-1|n-2}$ to derive the current observation error covariance $\hat{S}_{n|n-1}$.

Finally, simulating the traditional Kalman filter process, we learn the H matrix through an output layer, which takes the state error covariance and observation error covariance as inputs and ultimately outputs the Kalman gain K_n . Compared with end-to-end DNN, this network structure emulates

part of the model-based Kalman filter process, making it more parameter-efficient and easier to train.

3) Association Module: Following the baseline Poly-MOT, we use a two-stage association strategy to reduce false negative associations. In the One-stage Association, we use the 3D Generalized Intersection over Union $(GIoU_{3D})$ to as the match metric between the tracked trajectories and the detections. After One-stage Association, we perform a wider threshold Two-stage Association between mismatched trajectories and mismatched detections. This association will be performed in Bird's Eye View.

$$GIoU_{3D}(T_{n-1}, D_n) = \frac{T_{n-1} \cap D_n}{T_{n-1} \cup D_n} + \frac{T_{n-1} \cup D_n}{Hull(T_{n-1}, D_n)} - 1.$$
(7)

By combining the two-stage association, we effectively obtain three results: updated trajectories T_n^D , mismatched trajectories T_n^u , and mismatched detections D_n^u .

4) Trajectory Management Module: Like most 3D-MOT methods, our system employs a confidence-based trajectory lifecycle management approach [26]. Specifically, when the tracker receives a detection D_n^u that is not associated with an existing trajectory, it is initialized as a new trajectory. During subsequent tracking, if a observation is associated with this trajectory, corresponding to T_n^D , the state of the trajectory is updated based on the new observation. However, if the trajectory is not observed in consecutive frames T_{n-1}^u , its confidence decays following an exponential function until it is ultimately deleted.

B. Training methods

Our design also extends to the training methodology. The prototype system, KalmanNet, is trained in a supervised manner using labeled datasets. The squared error is computed between the estimated value \hat{X}_n and the true value X_n , as shown below:

$$\mathcal{L} = \sum_{n=0}^{seq} \left\| X_n - \hat{X}_n \right\|^2. \tag{8}$$

Although the system output is the Kalman gain, supervising with respect to the state variable X_n is not direct supervision. However, it is still effective because:

$$\frac{\partial \mathcal{L}_n}{\partial K_n} = \frac{\partial \|K_n \Delta \mathbf{y}_n - \Delta X_n\|^2}{\partial K_n},$$

$$= 2 \cdot (K_n \cdot \Delta y_n - \Delta X_n) \cdot \Delta y_n^{\top}$$
(9)

where $\Delta X_n = X_n - \hat{X}_{n|n-1}$. The loss is differentiable with respect to the Kalman gain, meaning that KalmanNet can be trained end-to-end to compute the Kalman gain by minimizing the squared error.

However, in KalmanNet, the training data is automatically generated and can be collected in unlimited quantities. Automatically generated labels ensure a strict correspondence between labels and samples. However, in MOT, the annotations is provided by the labeled bounding boxes in the dataset. On the one hand, the number of annotations labels is typically smaller than the number of objects being

TABLE I
A COMPARISON OF EXISTING METHODS APPLIED TO THE NUSCENES TEST SET.

Method	Detector	Input Data	AMOTA↑	AMOTP↓	IDS↓
CBMOT [26]	CenterPoint [2] & CenterTrack [27]	2D + 3D	68.1	52.8	761
EagerMOT [10]	CenterPoint & Cascade R-CNN [28]	2D + 3D	67.7	55.0	1156
SimpleTrack [6]	CenterPoint	3D	66.8	55.0	575
OGR3MOT [9]	CenterPoint	3D	65.6	62.0	288
CenterPoint	CenterPoint	3D	65.0	53.5	684
Poly-MOT [7]	CenterPoint	3D	70.0	50.9	331
Poly-MOT ¹	LargeKernel3D [29]	2D + 3D	75.4(74.2)	42.2(46.5)	292(276)
Ours	CenterPoint	3D	70.0	50.4	353
Ours	LargeKernel3D	2D + 3D	74.4	42.9	352

1 "()" indicates the result we reproduced.

tracked. On the other hand, associating annotations bounding boxes with trajectories often relies on methods such as Euclidean distance or IoU. Regardless of the association method used, there is no guarantee that the annotations strictly correspond to the system's trajectories, which brings anomalous supervision to the system.

To address these issues, we adopt a semi-supervised training method based on pseudo-labels. Specifically, in addition to using the annotations to provide the true state values for supervised training, we also parallelize the training process with an EKF. For trajectories that are not associated with annotated boxes, we use the system state obtained by the EKF as a pseudo-label for training. Consequently, our final loss function is as follows:

$$\mathcal{L} = \sum \left\| X_n - \hat{X}_n \right\|^2 + \sum \left\| \tilde{X}_n - \hat{X}_n \right\|^2, \tag{10}$$

where X_n is the true annotation of the dataset, \tilde{X}_n is the update output of traditional Kalman filter, and \hat{X}_n is the update output of GRU-Kalman filter.

IV. EXPERIMENTS

A. The dataset

1) NuScenes: The nuSceness [17] dataset comprises 850 training sequences and 150 validation sequences, each consisting of approximately 40 frames. Keyframes are sampled at a rate of 2Hz, with annotations provided for each keyframe. These annotations include geometric details of the object bounding boxes and their unique identifiers within the scene. The official evaluation primarily employs accuracy AMOTA as the key performance metric, precision AMOTP, ID Switch (IDS) as the secondary performance metric.

2) Argoverse2: Argoverse2 [18] expands on Argoverse1 [31] by collecting 1000 scene clips in six US cities. Each sequence lasted 15 seconds, sampled and annotated at 10Hz, with an average of 75 annotated objects per frame. The dataset features over 30 object classes and encompasses multiple complex urban environments. The official evaluation uses HOTA as the key metric.

B. Implementation Details

The system input consists of the detection results from the 3D MOD. For dataset splitting, we follow the official

division of training, validation, and test sets. Regarding training parameters, we employ the AdamW optimizer with a maximum learning rate 1e-5 and weight decay 1e-5. The CosineAnnealingLR adjustment method and smooth-loss supervision are applied to help the model avoid local minima. Gradient accumulation is performed across each object pair per frame in the sequence, followed by backpropagation at the end of the sequence.

C. Experimental Results

1) Comparative evaluation: : We compare our approach with several methods on the validation and test sets of the NuScenes dataset, as well as on the validation set of the Argoverse2 dataset.

NuScenes Test Set: We compared our system with other tracking algorithms, such as the TBD baselines AB3DMOT, CenterTrack, SimpleTrack, Poly-MOT, and JDT method OGR3MOT. The selected algorithms include both traditional tracking methods and machine learning-based approaches.

As shown in TABLE I, according to the AMOTA results, our proposed GRU-based semi-supervised method achieves better performance than most mainstream hand-designed methods using the same detector (CenterPoint [2]). Compared to OGR3MOT with GNN, it also demonstrates superior performance. Furthermore, we obtain the best AMOTP results, indicating that our system provides more accurate final trajectory box information than the traditional Kalman model and constant velocity model. Overall, compared with most TBDs and JDTs, we exhibit excellent performance without the need for manually designing motion models, highlighting great potential for future development.

NuScenes Val Set: As shown in Table III, we evaluated our system on the validation set using the same model configuration, and once again, it demonstrated stable and consistent performance.

Argoverse2 Val Set: We evaluated our method on the Argoverse2 validation set in Table II, which includes ground-truth labels for 30 object classes. Given that most MOT algorithms require modification before being applied to Argoverse2, we compared our method with the Argoverse2 baseline, LT3D, using tracking method such as greedy and ab3dmot tracker. We present results for the seven most prevalent dynamic object categories in Argoverse2 and report the

 $\label{thm:table II} \mbox{A Comparison Of On The Argoverse2 Val Set.(HOTA)}$

	Detector	REGULAR_VEHICLE	PEDESTRIAN	BICYCLE	LARGE_VEHICLE	BUS	BOX_TRUCK	TRUCK	Average
Greedy	LT3D [30]	58.9	59.1	51.6	28.5	48.2	44.0	31.3	46.0
AB3DMOT	LT3D	59.2	54.6	48.7	26.7	47.0	43.3	34.2	42.7
Ours	LT3D	73.3	71.9	53.8	29.3	59.2	55.3	36.0	47.3

TABLE III

A COMPARISON OF EXISTING METHODS APPLIED TO THE NUSCENES $\mbox{Val Set.}$

Method	Input Data	AMOTA↑	AMOTP↓	IDS↓
CBMOT	2D + 3D	72.0	48.7	479
EagerMOT	2D + 3D	71.2	56.9	899
SimpleTrack	3D	69.6	54.7	405
OGR3MOT	3D	69.3	62.7	262
CenterPoint	3D	66.5	56.7	562
Poly-MOT	3D	73.1	52.1	281
Ours	3D	73.2	51.1	267

TABLE IV

THE EFFECTIVENESS ABLATION EXPERIMENTS WERE CONDUCTED ON THE NUSCENES VALIDATION SET. HERE, THE CTRA, BICYCLE, AND CA STATE MODELS FOLLOW THE POLYMOT SPECIFICATION.

Category	Motion Module	AMOTA↑	$\mathbf{AMOTP}{\downarrow}$	
	Bicycle + EKF	54.5	50.7	
	CTRA + EKF	55.0	46.2	
Bicycle	CA + KF	54.5	46.3	
	CTRA + Ours	55.6	45.9	
	Bicycle + Ours	55.6	45.9	
	Bicycle + EKF	86.3	34.2	
	CTRA + EKF	86.3	34.0	
Car	CA + KF	86.0	33.6	
	CTRA + Ours	86.3	33.8	
	Bicycle + Ours	86.3	33.8	

overall average HOTA score. Importantly, we used the model trained on the NuScenes training set, and its performance on Argoverse2 demonstrates a strong level of generalization.

2) Ablation Study: In this part, we conduct extensive ablation experiments to evaluate the performance of the proposed module. We used the same preprocessing parameters and association parameters, and then replaced the motion module as well as different training strategies for a series of experiments.

GRU-based Motion Module: We select the two most representative categories to investigate the tracking performance performance of different SS and optimizers on these two categories separately. As can be seen from the Table IV, the performance of the traditional methods represented by Poly-MOT will vary under different SS models and optimizers, which is precisely caused by the high dependence of the traditional motion module on accurate modeling. However, our data-driven based method, achieves the best tracking results in both categories. And changing SS does not affect

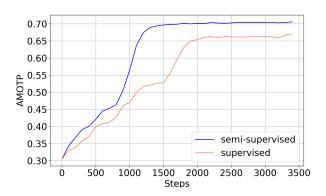


Fig. 4. Experiments comparing the training convergence speed and accuracy of supervised and semi-supervised training.

the tracking accuracy, which reflects the excellent robustness of our motion module.

Semi-Supervised: To verify the effectiveness of our proposed semi-supervised method, we compared the fully supervised and semi-supervised training processes and results on the validation set. As shown in Figure 4, our semi-supervised approach achieved convergence within 1,700 steps, or roughly 2 epochs, reaching an AMOTA score close to 0.7. In contrast, when training solely with only annotations from dataset, the system took 3 epochs to converge, and the final performance was lower. This discrepancy is attributed to the limited amount of training data and errors introduced during the association between annotations and the tracked samples, which semi-supervised learning helps to mitigate.

V. CONCLUSIONS

In this work, we propose a partially learnable MOT method by introducing a GRU-based Kalman filter [25] into the motion module of the TBD framework. This method eliminates the adverse effects of inaccurate noise parameterization and reduces the error of linearization. Our findings demonstrate that the GRU-based motion module is well-suited for MOT in autonomous driving and robotic sensing environments. Additionally, we introduce a semi-supervised training strategy that leverages pseudo-labels, which accelerates training by increasing data volume and minimizing association errors. We believe that combining deep learning methods with interpretable mathematical models can enhance 3D MOT performance, and foresee extending this approach to other modules, potentially leading to the development of parameter-free trackers.

REFERENCES

- [1] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 10359–10366.
- [2] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11784–11793.
- [3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE* conference on computer vision and pattern recognition, 2016, pp. 779– 788
- [4] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in European conference on computer vision. Springer, 2020, pp. 213– 229.
- [5] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," pp. 21674–21683, 2023.
- [6] Z. Pang, Z. Li, and N. Wang, "Simpletrack: Understanding and rethinking 3d multi-object tracking," in *European Conference on Computer Vision*. Springer, 2022, pp. 680–696.
- [7] X. Li, T. Xie, D. Liu, J. Gao, K. Dai, Z. Jiang, L. Zhao, and K. Wang, "Poly-mot: A polyhedral framework for 3d multi-object tracking," in 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2023, pp. 9391–9398.
- [8] K. Huang and Q. Hao, "Joint multi-object detection and tracking with camera-lidar fusion for autonomous driving," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 6983–6989.
- [9] J.-N. Zaech, A. Liniger, D. Dai, M. Danelljan, and L. Van Gool, "Learnable online graph representations for 3d multi-object tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5103–5110, 2022
- [10] A. Kim, A. Ošep, and L. Leal-Taixé, "Eagermot: 3d multi-object tracking via sensor fusion," in 2021 IEEE International conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 11315–11321.
- [11] L. Wang, X. Zhang, W. Qin, X. Li, J. Gao, L. Yang, Z. Li, J. Li, L. Zhu, H. Wang, et al., "Camo-mot: Combined appearance-motion optimization for 3d multi-object tracking with camera-lidar fusion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 11981–11996, 2023.
- [12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, p. 35–45. [Online]. Available: http://dx.doi.org/10.1115/1.3662552
- [13] M. Gruber, AN APPROACH TO TARGET TRACKING. [Online]. Available: http://dx.doi.org/10.21236/ad0654272
- [14] R. E. Larson, R. M. Dressler, and R. S. Ratner, Application of the extended kalman filter to ballistic trajectory estimation. Stanford Research Institute Menlo Park, 1967.
- [15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555, 2014.
- [16] A. Graves and A. Graves, "Long short-term memory," Supervised sequence labelling with recurrent neural networks, pp. 37–45, 2012.
- [17] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.
- [18] B. Wilson, W. Qi, T. Agarwal, J. Lambert, J. Singh, S. Khandelwal, B. Pan, R. Kumar, A. Hartnett, J. K. Pontes, et al., "Argoverse 2: Next generation datasets for self-driving perception and forecasting," arXiv preprint arXiv:2301.00493, 2023.
- [19] C. Zhang, C. Zhang, Y. Guo, L. Chen, and M. Happold, "Motiontrack: end-to-end transformer-based multi-object tracking with lidar-camera fusion," in *Proceedings of the IEEE/CVF Conference on Computer* Vision and Pattern Recognition, 2023, pp. 151–160.
- [20] A. Vaswani, "Attention is all you need," Advances in Neural Information Processing Systems, 2017.
- [21] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1090–1099.

- [22] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," in *Proceedings of the IEEE/CVF conference on* computer vision and pattern recognition, 2020, pp. 6247–6257.
- [23] L. Zhou, Z. Luo, T. Shen, J. Zhang, M. Zhen, Y. Yao, T. Fang, and L. Quan, "Kfnet: Learning temporal camera relocalization using kalman filtering," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4919–4928.
- [24] V. Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," *Neural Information Pro*cessing Systems, Neural Information Processing Systems, Jun 2019.
- [25] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. Van Sloun, and Y. C. Eldar, "Kalmannet: Neural network aided kalman filtering for partially known dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.
- [26] N. Benbarka, J. Schröder, and A. Zell, "Score refinement for confidence-based 3d multi-object tracking," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2021, pp. 8083–8090.
- [27] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European conference on computer vision*. Springer, 2020, pp. 474–490.
- [28] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proceedings of the IEEE conference on computer* vision and pattern recognition, 2018, pp. 6154–6162.
- [29] Y. Chen, J. Liu, X. Qi, X. Zhang, J. Sun, and J. Jia, "Scaling up kernels in 3d cnns," arXiv preprint arXiv:2206.10555, vol. 1, no. 2, p. 5, 2022.
- [30] N. Peri, A. Dave, D. Ramanan, and S. Kong, "Towards long-tailed 3d detection," in *Conference on Robot Learning*. PMLR, 2023, pp. 1904–1915.
- [31] M.-F. Chang, J. W. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays, "Argoverse: 3d tracking and forecasting with rich maps," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.