DeepKalPose: An Enhanced Deep-Learning Kalman Filter for Temporally Consistent Monocular Vehicle Pose Estimation

Leandro Di Bella,[1,2] Yangxintong Lyu,[1,2] and Adrian Munteanu[1,2]
[1]*Department of Electronics and Informatics, Vrije Universiteit Brussel, B-1050 Brussels, Belgium*
[2]*imec, Kapeldreef 75, B-3001 Leuven, Belgium*
Email: yangxintong.lyu@vub.be

In this paper, we introduce an innovative temporal consistency enhancement approach, which enables image-based models on video data by leveraging a deep-learning-based Kalman Filter. More specifically, we propose a novel Bi-direction Kalman filter strategy, utilizing forward and backward processing to capitalize on higher-quality pose estimations near the camera, enhancing the robustness and precision of vehicle tracking across varying distances and conditions. Then, rather than using the conventional mathematical motion model, we propose a learnable motion model, dubbed Future State Predictor, to represent the complex, non-linear motion patterns observed in vehicles. The experimental results demonstrate that our approach enhances pose accuracy and temporal consistency, which allows us to handle the challenging occluded/distant vehicles.

Introduction: In recent years, the importance of scene understanding has become increasingly important, particularly in the development of technologies for smart mobility and intelligent transportation systems. The need to comprehend dynamic urban environments underscores the growing need for accurate monocular vehicle 6D pose estimation in video. Traditional methods, predominantly image-based [1–4], often lead to temporal pose inconsistencies in successive video frames. More precisely, there can be irregular or unrealistic changes in the estimated poses of vehicles across consecutive video frames. To address this challenge, a range of innovative methods have been developed in the field of vehicle pose estimation. The authors in [5] and [6] have proposed frameworks capable of effectively associating moving objects over time and estimating pose information from a sequence of 2D images by leveraging LSTM-based modules for motion learning. [7] employs a Markov Random Field (MRF) to ensure the selection of the optimal pose over time. This approach infers the pose for a current frame by considering past and future poses in a batch process. Additionally, shape completion and tracking techniques are also used, where candidate shapes are encoded into latent vectors and compared against a model shape, as seen in [8, 9]. These methods contribute significantly to maintaining temporal consistency. In contrast, particle filter-based methods remain a primary approach for pose estimation adapted to video. For example, [10] improves vehicle detection by feeding them into a Poisson multi-Bernoulli mixture (PMBM) tracking filter. Furthermore, [11, 12] integrate a 3D Kalman Filter into their frameworks, leveraging the kinematic motion of vehicles. This final approach involves a model-based particle filter, specifically the Kalman filter, which has demonstrated effectiveness. However, challenges arise when dealing with systems where the underlying dynamics is unknown or has highly nonlinear behavior. Under these conditions, creating a mathematical model for motion becomes very complex.

In this letter, we propose an enhanced deep-learning (EDL) Kalman Filter-based method for temporally consistent vehicle pose estimation, dubbed *DeepKalPose* which is adept at reinforcing temporal consistency. It can effectively address flickering artifacts in vehicle pose estimation which refer to the jittery or unstable pose of a vehicle in sequential frames. *DeepKalPose* aims to overcome this by providing more stable and consistent tracking of the vehicle's pose. Moreover, the design tackles the challenges of far-object detection and occlusion by leveraging the predictive capabilities of the Kalman filter. Our method advances beyond the standard Kalman filter by incorporating a bi-branch network. Each of the branches represents a forward and a backward pass for a time-series, respectively. Additionally, unlike standard Kalman filters that rely on typical mathematical motion models, our filter integrates deep learning techniques to effectively handle the nonlinear and complex motion patterns of vehicles. By employing an encoder-decoder architecture motion model, our method aims to design a more nuanced representation of vehicular dynamics.

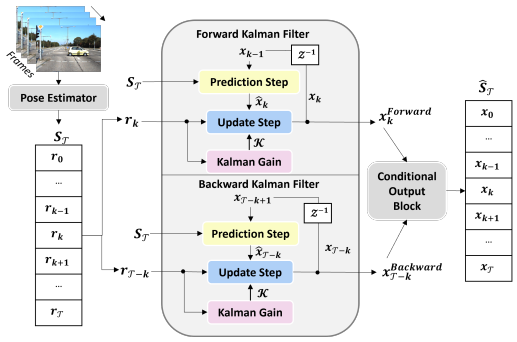

Fig 1 *Schematic Overview of DeepKalPose.*

Notations: DeepKalPose estimates and adjusts vehicle pose estimates over sequential frames given the predictions of an existing pose estimator. Given the importance of the yaw angle in vehicle pose estimation, this letter focuses exclusively on the yaw angle within the rotation components. Our dataset, denoted as $\mathcal{D}$, comprises a series of samples $\boldsymbol{S_{i,\mathcal{T}}}$, each representing a time-series of vehicle poses. $\boldsymbol{S_{i,\mathcal{T}}}$ can be formally expressed as:

$$\mathcal{S}_{i,\mathcal{T}} = \{\boldsymbol{r}_{i,k}\}_{k=1}^{\mathcal{T}} \tag{1}$$

Here, $\mathcal{D} = \{\mathcal{S}_{i,\mathcal{T}}\}_{i=1}^{\mathcal{N}}$ where $i \in \{1, ..., \mathcal{N}\}$ with $\mathcal{N}$ representing the total number of samples. The variable $k \in \{1, ..., \mathcal{T}\}$ denotes the discrete time steps with $\mathcal{T}$ being a fixed time length of the time-series $\mathcal{S}_{i,\mathcal{T}}$. A vehicle pose is represented as $\boldsymbol{r}_{i,k} = [x_{i,k}, y_{i,k}, z_{i,k}, \theta_{i,k}] \in \mathbb{R}^4$ where $x_{i,k}, y_{i,k}, z_{i,k}$ are the 3D translation components of the i^{th} vehicle sample at timestep k and $\theta_{i,k}$ denotes the yaw or heading angle component.

DeepKalPose employs a particle filter approach, specifically using the Kalman Filter (KF). KF is a state space model for real-time system that describes the evolution of the system's state variables over time. The state space model is described by Eq. 2a and Eq. 2b :

$$\boldsymbol{x}_k = F\boldsymbol{x}_{k-1} + \boldsymbol{w}_k \quad \text{(2a)} \qquad \boldsymbol{r}_k = H\boldsymbol{x}_k + \boldsymbol{v}_k \quad \text{(2b)}$$

where: $\boldsymbol{x}_k$ is the state vector, $\boldsymbol{r}_k$ is the measurements vector, F is the motion model, H is the measurements matrix, $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$ are the noise covariances. The Kalman filter operates in two main steps, namely prediction and update steps. The prediction step estimates an *a priori* state vector $\hat{\boldsymbol{x}}_k$ and a prediction error $\hat{P}_k$.

$$\textit{a priori} \text{ state estimate} : \hat{\boldsymbol{x}}_k = F\boldsymbol{x}_{k-1} + \boldsymbol{w}_k \tag{3}$$

$$\textit{a priori} \text{ prediction error} : \hat{P}_k = FP_{k-1}F^T + Q_k \tag{4}$$

Where $\boldsymbol{x}_{k-1}$ is a *a posteriori* state vector at timestep $k-1$. During the update step, we compute the Kalman Gain $\mathcal{K}$ following:

$$\text{Optimal Kalman Gain} : \mathcal{K}_k = \hat{P}_k H_k^T (H_k \hat{P}_k H_k^T + R_k)^{-1} \tag{5}$$

$$\text{Updated State Estimate} : \boldsymbol{x}_k = \hat{\boldsymbol{x}}_k + \mathcal{K}_k(\boldsymbol{r}_k - H\hat{\boldsymbol{x}}_k) \tag{6}$$

$$\text{Updated Estimate Covariance} : P_k = (I - \mathcal{K}_k H)\hat{P}_k \tag{7}$$

Where the measurements vector $\boldsymbol{r}_k$ refines prediction $\hat{\boldsymbol{x}}_k$ as in Eq. (6) and I is the identity matrix. By doing so, we can obtain an improved estimate $\boldsymbol{x}_k$.

Proposed Method: An overview of the methodology is illustrated in Figure 1. This method will refine the pose predictions of an existing pose estimator to produce a more accurate and temporally consistent pose time-series output $\hat{\mathcal{S}}_{\mathcal{T}}$. To achieve this, the filter takes the inconsistent pose prediction sequence $\mathcal{S}_{\mathcal{T}}$ as input. At each iteration k, the measurement vector is updated as $\boldsymbol{r}_k = [\bar{x}_k, \bar{y}_k, \bar{z}_k, \bar{\theta}_k] \in \mathcal{S}_{\mathcal{T}}$. For clarity, the symbol $\bar{}$ is used to denote measurements belonging to $\mathcal{S}_{\mathcal{T}}$, differentiating them from the state vector used in the KF. Given the

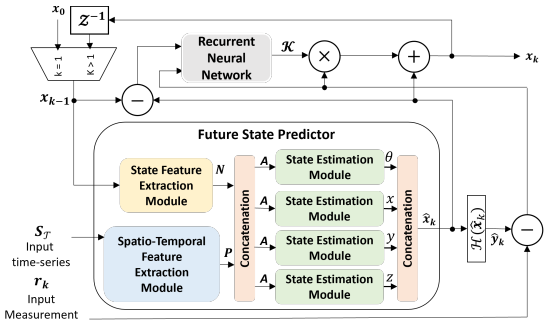

Fig 2 *Schematic of the proposed EDL Kalman Filter module Architecture: This figure presents an integrated learnable motion model with a directional KF for state estimation. $\mathcal{Z}^{-1}$ is a unit delay, $\otimes$ is multiplication operation, $\oplus$ is a sum operation and $\ominus$ is a substraction operation.*

objective of tracking and correcting pose measurements, the state vector is defined as $\boldsymbol{x}_k = [x_k, y_k, z_k, \theta_k]$, thereby directing the Kalman Filter to track the vehicle's pose.

However, existing mathematical model based KF methods [11–14] have difficulties solving the temporal consistency since most time-series data from the autonomous driving task dataset typically depict vehicles approaching or receding from the camera's viewpoint. Thus, the precision of pose estimation is significantly influenced by the vehicle's position within the image. Specifically, the accuracy of the estimates is higher for vehicles positioned closer to the camera than for vehicles farther away. This variability in accuracy directly impacts the effectiveness of Kalman Filter (KF) tracking as the initial measurement and the corresponding precision play a crucial role in setting the starting point of the KF.

To address this challenge, we propose a bi-branch Kalman Filter approach. The first branch, the forward Kalman Filter, processes the time-series from timestep $k = 0$ to $\mathcal{T}$ where $\mathcal{T}$ is the context length of the input time sequence. Conversely, the backward Kalman Filter branch processes the time-series in reverse, from the timestep $k = \mathcal{T}$ to 0. We note that the proposed method acts as an offline smoother and processes the input data on chunks of $\mathcal{T}$ frames. While this approach does not render the method online, selecting sufficiently small $\mathcal{T}$ can enable near real time processing. This method capitalizes on the higher quality of pose estimations near the camera, regardless of their position within the time-series. By feeding each time-series through both the forward and the backward EDL Kalman filter, our methodology aims to enhance the robustness and accuracy of pose estimations for distant and partially observed vehicles, as both scenarios present challenges for baseline estimations.

The Kalman Filter necessitates a thorough understanding of the system's dynamics and noise characteristics to function effectively. However, for the pose estimation task, our limited knowledge of the vehicle's systems presents a significant challenge. Therefore, instead of using a traditional model-based Kalman Filter, inspired by KalmanNet [15], our *DeepKalPose* replaces the computation the Kalman gain $\mathcal{K}$ in Eq. 5 and both of the covariances $\boldsymbol{v}_k$ (Eq. 2a) and $\boldsymbol{w}_k$ (Eq. 2b) by a Recurrent Neural Network (RNN). Moreover, we propose a novel module, named Future State Predictor (FSP), which is able to learn a predictive motion model, as illustrated in Figure 2. The FSP block follows an encoder-decoder architecture where the encoder takes as input the state $\boldsymbol{x}_{k-1}$ and the sequence that is being processed $S_{\mathcal{T}}$ with $\mathcal{T}$ the context length of the sequence and $k \in \{0, ..., \mathcal{T}\}$ representing the timestep of the KF iterative process. More precisely, the top branch of the encoder, namely the State Feature Extractor Module (SFEM), processes the current state $\boldsymbol{x}_{k-1}$ to derive a feature vector **N**. The SFEM captures the relevant aspects of the current state that may influence the future state. The lower branch of the encoder, namely Spatio-Temporal Feature Extraction Module (STFEM) is dedicated to extracting features **P** from the entire sequence $S_{\mathcal{T}}$. This module is designed to capture the local spatial and temporal patterns within the full sequence of 3D position vector to give an indication on the next state. Spatial patterns represent the relationship between the different features at a given timestep, here translation and rotation components. On the other hand, temporal patterns represent how values in the time-series change over time. In contract, the SFEM focus on immediate characteristic of the current state. The feature vectors from the SFEM and the STFEM are concatenated into a combined feature vector **A**. The decoder consists of four State Estimation Modules (SEM) which will transform individually the feature vector **A** into the three translation components and the rotation component of the predicted state vector $\hat{\boldsymbol{x}}_k$. To train this model, we have used a L1-loss for the translation components as follows : $\mathcal{L}_\alpha = |\hat{\alpha} - \alpha|$ where $\alpha \in \{x, y, z\}$ and $\hat{\alpha}$ denotes the predicted translation values. The rotation loss is defined as: $\mathcal{L}_\theta = 1 - \cos(\hat{\theta}, \theta)$ where cos is the cosine similarity between the predicted heading angle $\hat{\theta}$ and the ground truth θ. The loss per mini-batch $\mathcal{B}$ with the mini-batch size $M < N$ is denoted as :

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{\mathcal{T}}\sum_{k=0}^{\mathcal{T}}(\mathcal{L}_x + \mathcal{L}_y + \mathcal{L}_z + \mathcal{L}_\theta)_{j,k} \tag{8}$$

Network Details: For each input image in the sequence, we use both 4D object detection, namely D4LCN [4] and vehicle 6D pose estimation, Mono6D [1], as pose estimator. For Mono6D, the measurement vector is defined as $\boldsymbol{x}_k = [x_k, y_k, z_k, \theta_k]$. In contrast, for D4LCN, due to the inconsistency of the heading angle prediction, our focus is primarily on the translational components with $\boldsymbol{x}_k = [x_k, y_k, z_k]$. In the FSP module, the STFEM consists of two 1-dimensional convolution layers (1D-CNN) with stride and padding equal to 1. The number of filters is set to 64 for the low-level features and to 128 for the high-level ones. Each 1D-CNN is followed by a ReLU activation function and a batch normalization (Batch Norm). The feature map is then flattened and fed into two dense blocks with output dimensions of 256 and 128 filters. In the SFEM, the features **N** from the state at timestep k are extracted with three dense blocks with output dimensions corresponding, respectively, to 512 - 256 and 128 filters. Both branches' output **N** and **P** are concatenated and fed into the decoder. This concatenated vector **A** contains comprehensive information about both the current state and the evolution of the states in the sequence. The decoder is composed of four identical parallel linear blocks SEM for Mono6D and three for D4LCN, each of which will produce a component of the state $\hat{\boldsymbol{x}}_k$. In the training process, the pose estimator firstly learns image by image. Then *DeepKalPose* learns by taking the predictions $S_{\mathcal{T}}$ as inputs.

Experimental setup: The KITTI RAW dataset [16] is a widely used dataset for monocular object pose estimation [4, 17, 18] and tracking [19–21]. The dataset comprises 51 videos, divided into 39 for training and 12 for validation. It includes a total of 9903 images for training and 2977 images for validation. We segment each sequence into fixed length with 20 timesteps using a stride of 1 for continuity. In total, we have 674 vehicle trajectory patches for training and 375 for validation. To train *DeepKalPose*, we extract the pose predictions from an existing vehicle pose estimator [1, 4] as $\boldsymbol{r}_k$. To handle missing data from a non-detected vehicle by the vehicle pose estimator, we applied mean substitution. The network is optimized by Adam Optimizer [22] with a learning rate of 0.001 and a weight decay of 0.00001. We take a batch size of 128 on 1 Nvidia GeForce RTX 2080 (12G). The iteration number for the training process is set to 4,000. Evaluation metrics used to compare against D4LCN [4] include 3D precision-recall curves with a 3D bounding box IoU threshold of 0.7 and 0.5 for cars as this method performs an object detection step. Against Mono6D [1], evaluation is performed using, for translation, the Average Relative Euclidean Distance (**ARED**). For rotation, we use the accuracy with threshold δ, denoted **Acc**(δ) and the median error, **Mederr**, in degrees. Following the evaluation of D4LCN in [4], we only consider the detected vehicle with a 2D bounding box IoU threshold of 0.5.

Experimental results: In Table 1, we compared our proposed method, *DeepKalPose*, with the state-of-the-art (SOTA) method D4LCN [4]. One can note that with *DeepKalPose*, we can significantly outperform D4LCN [4]. The average precision @70 of D4LCN improves to 31.12% for Easy, 24.82% for Moderate, and 16.70% for Hard scenarios, compared to 28.07%, 21.56%, and 14.13% respectively without *DeepKalPose*. The same behavior is observed for the AP@50.

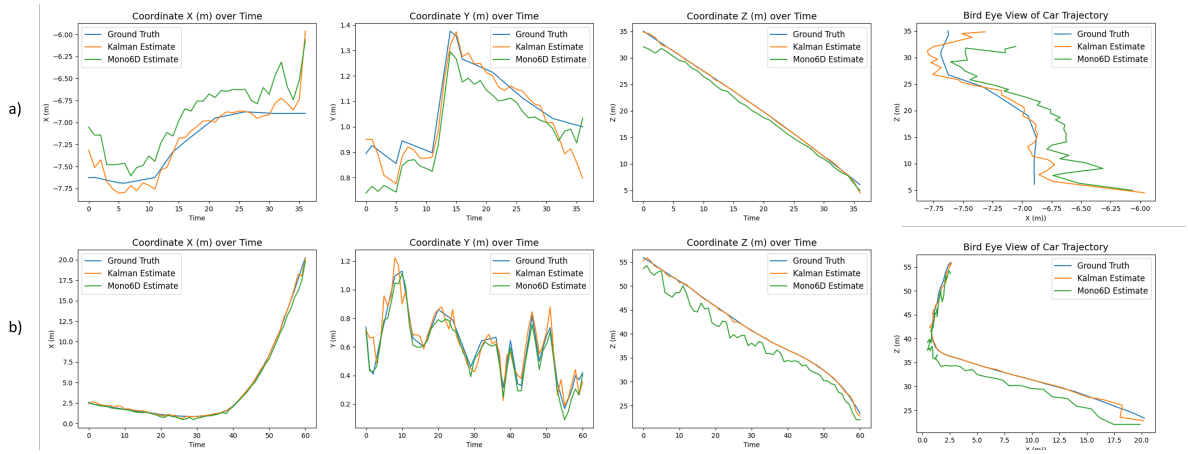

Fig 3 *Qualitative results demonstrating improved car trajectory estimation on the KITTI validation set. Row (a) and (b) depicts car trajectories plotted over time. From left to right, the columns represent the Variation of X, Variation of Y, Variation of Z over time, and a Bird's Eye View of the X-Z position of the car.*

Table 1. Method comparison in terms of average precision.

Method	Average Precision @70			Average Precision @50		
	Easy	Moderate	Hard	Easy	Moderate	Hard
D4LCN [4]	28.07	21.56	14.13	67.08	52.55	35.74
Our Method	**31.12**	**24.82**	**16.70**	**68.67**	**53.74**	**36.89**

Table 2. Method comparison in terms of translation and rotation errors. $z <$ and > 40m denote all detected vehicles with depth coordinate higher or smaller than 40 meters. Occlusion denotes evaluation results on (highly or fully) occluded vehicles.

Method	ARED	Acc ($\frac{\pi}{6}$)	Acc ($\frac{\pi}{18}$)	Mederr
Mono6D [1]	5.34%	84.66%	65.41%	**4.94°**
Our Method	**3.90%**	**89.18%**	**67.47%**	5.46°
Mono6D [1] (*Occlusion*)	6.97%	73.86%	49.07%	10.19°
Our Method (*Occlusion*)	**4.21%**	**82.50%**	**51.53%**	**9.30°**
Mono6D [1] (z<40m)	5.18%	84.96%	65.86%	**4.85°**
Our Method (z<40m)	**3.96%**	**89.53%**	**68.29%**	5.24°
Mono6D [1] (z>40m)	6.28%	82.94%	**62.81%**	**5.51°**
Our Method (z>40m)	**3.55%**	**87.12%**	62.67%	7.02°

In addition, we re-trained Mono6D [1] by adding our proposed module to its original framework. As shown in Table 2, one could observe that this integration led to notable improvements in vehicle 6D pose estimation. As such, the ARED saw an improvement from 5.34% to 3.90% as described in lines 1 and 2. In addition, our method improves the heading angle orientation accuracy by 4.52% for $\delta = \frac{\pi}{6}$ and 2.08% for $\delta = \frac{\pi}{18}$ although the median error slightly increased. This latter result suggests that the Kalman filter's smoothing operation leads to more stable rotations but at the expense of smaller, more frequent errors. Furthermore, in scenarios of partial or complete vehicle occlusion (denoted *Occlusion* in Table 2), this experiment reveals that our *DeepKalPose* method is robust against occlusion demonstrating a reduction of ARED error from 6.97% to 4.21%. Thanks to the predictive capabilities of the Kalman filter and the use of the temporal information from past measurements, *DeepKalPose* is able to adjust the initial noisy measurements from the pose estimator due to occlusion and put more emphasis to the predicted state vector learned by the FSP. Moreover, we evaluated both methods for various vehicle distances from the camera. Specifically, for distances exceeding 40 meters (considered as far-object detection), our proposed method demonstrated a notable decrease in ARED from 6.28% to 3.55%, indicating a 2.73% improvement. This suggests that *DeepKalPose* utilizes past detections and temporal informations effectively to mitigate the impact of distance on model performance, in contrast to the standard pose estimator whose performance diminishes with increasing distance. Figure 4 confirms the efficacy of *DeepKalPose* against far-vehicles pose estimation errors as illustrated by a bigger gap in ARED between the two methods when the distance is increasing. While the translation errors of the pose estimator increase, *DeepKalPose* maintains a constant translation error regardless of distance. Additionally, the reduction of the variance confirms the stability of the KF pose estimations across varying distances. Furthermore, in Figure 3, we plot qualitative results on the KITTI validation set for two different car trajectories. In both trajectories (a) and (b), Kalman filter estimates maintain closer alignment with the ground truth. The overall trajectory described in both Bird's Eye View suggest a higher accuracy in estimating the car's actual path.

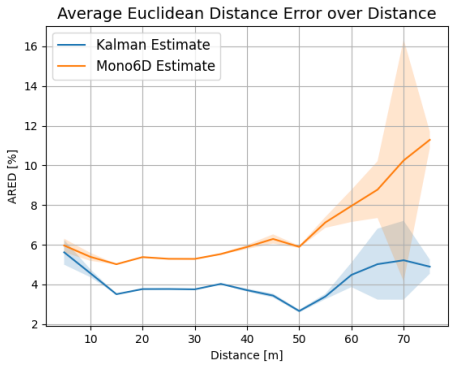

Fig 4 *The ARED of Mono6D and proposed method in function of distance. Solid lines represent the mean values, while the shaded areas indicate the variance.*

Conclusion: In conclusion, our research introduces an innovative approach to monocular vehicle 6D pose estimation applied on video. This approach integrates a deep learning-based Kalman Filter, specifically designed to address the challenges of temporal consistency in autonomous driving scenarios. Our method, employing a learnable motion model, effectively captures the complex, nonlinear motion patterns of vehicles. The experimental results indicate that our proposed approach is superior to the existing techniques on both 4D object detection and vehicle 6D pose estimation tasks. Particularly, our DeepKalPose achieves notable improvements in AP and ARED across varying difficult levels when detecting 3D objects from single-view images while more precise and consistent vehicle pose predictions can be predicted with the integration of our method. These results affirm the efficacy of our deep-learning-based Kalman Filter in video-based pose estimation and suggest its potential in enhancing intelligent transport systems.

Acknowledgments: The authors would like to thank for the financial support provided by Innoviris (TORRES).

Received: DD MMMM YYYY *Accepted:* DD MMMM YYYY
doi: 10.1049/ell.10001

References

1. Y. Lyu, R. Royen, and A. Munteanu, "Mono6d: Monocular vehicle 6d pose estimation with 3d priors," in *2022 IEEE International Conference on Image Processing (ICIP)*, pp. 2187–2191, IEEE, 2022.
2. L. Ke, S. Li, Y. Sun, Y.-W. Tai, and C.-K. Tang, "Gsnet: Joint vehicle pose and shape reconstruction with geometrical and scene-aware supervision," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pp. 515–532, Springer, 2020.
3. D. Wu, Z. Zhuang, C. Xiang, W. Zou, and X. Li, "6d-vnet: End-to-end 6-dof vehicle pose estimation from monocular rgb images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.
4. M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo, "Learning depth-guided convolutions for monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition workshops*, pp. 1000–1001, 2020.
5. H.-N. Hu, Y.-H. Yang, T. Fischer, T. Darrell, F. Yu, and M. Sun, "Monocular quasi-dense 3d object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 2, pp. 1992–2008, 2022.
6. H.-N. Hu, Q.-Z. Cai, D. Wang, J. Lin, M. Sun, P. Krahenbuhl, T. Darrell, and F. Yu, "Joint monocular 3d vehicle detection and tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5390–5399, 2019.
7. M. Hödlmoser, B. Micusik, M. Pollefeys, M.-Y. Liu, and M. Kampel, "Model-based vehicle pose estimation and tracking in videos using random forests," in *2013 International Conference on 3D Vision-3DV 2013*, pp. 430–437, IEEE, 2013.
8. S. Giancola, J. Zarzar, and B. Ghanem, "Leveraging shape completion for 3d siamese tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1359–1368, 2019.
9. S. Sharma, J. A. Ansari, J. K. Murthy, and K. M. Krishna, "Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3508–3515, IEEE, 2018.
10. S. Scheidegger, J. Benjaminsson, E. Rosenberg, A. Krishnan, and K. Granström, "Mono-camera 3d multi-object tracking using deep learning detections and pmbm filtering," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 433–440, IEEE, 2018.
11. G. Brazil, G. Pons-Moll, X. Liu, and B. 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*, pp. 135–152, Springer, 2020.
12. A. Reich and H.-J. Wuensche, "Monocular 3d multi-object tracking with an ekf approach for long-term stable tracks," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*, pp. 1–7, IEEE, 2021.
13. J. Pak, "Visual odometry particle filter for improving accuracy of visual object trackers," *Electronics Letters*, vol. 56, no. 17, pp. 884–887, 2020.
14. Y. Wu, P. Vandewalle, P. Slaets, and E. Demeester, "An improved approach to 6d object pose tracking in fast motion scenarios," in *2022 Sixth IEEE International Conference on Robotic Computing (IRC)*, pp. 229–237, IEEE, 2022.
15. 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.
16. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361, IEEE, 2012.
17. Y. Zhang, J. Lu, and J. Zhou, "Objects are different: Flexible monocular 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3289–3298, 2021.
18. L. Peng, X. Wu, Z. Yang, H. Liu, and D. Cai, "Did-m3d: Decoupling instance depth for monocular 3d object detection," in *European Conference on Computer Vision*, pp. 71–88, Springer, 2022.
19. J. Cao, J. Pang, X. Weng, R. Khirodkar, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9686–9696, 2023.
20. A. Kim, G. Brasó, A. Ošep, and L. Leal-Taixé, "Polarmot: How far can geometric relations take us in 3d multi-object tracking?," in *European Conference on Computer Vision*, pp. 41–58, Springer, 2022.
21. H. Wu, W. Han, C. Wen, X. Li, and C. Wang, "3d multi-object tracking in point clouds based on prediction confidence-guided data association," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5668–5677, 2021.
22. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.