Sensor-Agnostic Multimodal Fusion for Multiple Object Tracking from Camera, Radar, Lidar and V2X

Marc Perez* [1,2] **and Antonio Agudo** [2]

[1]*Applus+ IDIADA*
PO Box 20, Santa Oliva, 43710, L'Albornar, Tarragona, Spain
[2]*Institut de Robòtica i Informàtica Industrial, CSIC-UPC*
Carrer de Llorens i Artigas, 4, 08028 Barcelona, Spain

ABSTRACT: Automated vehicles rely on different sensors to detect and track other vehicles and road users over time, to then be able to plan and execute safe trajectories. The characteristics of sensors are quite diverse and will probably be even more so in the future, so in this work we present a sensor-agnostic multimodal fusion framework for multiple object tracking that can seamlessly integrate information coming from different object detectors, sensors, and vehicle-to-everything messages, either from other road users or from the infrastructure. All the information received is converted to a standardized set of detections that are then combined using a Kalman Filter with a constant velocity model. To ensure robustness, we propose methods to handle errors in classification and incorrect bounding box reconstruction, a couple of problems that are often ignored in academic literature, although they are very relevant in practice. To evaluate our framework, we use three diverse and challenging scenarios. First, we present results for a perception system based on camera and radar that was integrated into a prototype traffic jam chauffeur function. Second, we show qualitative results for a traffic monitoring application in highways, with multiple cameras, lidars and a radar. And finally, we show how our framework can integrate vehicle-to-everything messages to improve the safety of vulnerable road users, such as pedestrians and cyclists, with an autonomous emergency braking function in proving grounds.

KEY WORDS: Multimodal, Detection, Tracking, Sensor Fusion, Cooperative perception.

1. Introduction

In 3D Multi-Object Tracking (MOT) the goal is to infer the trajectories of all the relevant objects in an unknown and dynamic scene. Frequently, the input data can be composed of raw data from a set of sensors such as cameras, radars or lidars; or 3D detections in the case of tracking-by-detection strategies. In the tracking-by-detection context, a common approach is online tracking [1]-[4], where the MOT module keeps a list of tracks. As new detections are received at every time, the track list is updated by modifying the state of some tracks, creating or removing others.

The performance of MOT approaches depends on the effectiveness of the object detector as well as on the handling of errors from the detector. In this work, we propose the introduction of a detection and tracking pipeline that works with a configurable set of sensors and provides robust solutions under different scenarios. Our solution can naturally manage misclassification errors and incomplete detections (smaller bounding box detected than the real one) from the object detector in the MOT module.

Object detectors can provide accurate estimations but sometimes fail to classify them, being rarely taken into account in current metrics. For most of the metrics, a car detected as a truck will be reported as both a false positive and a false negative, although in the case of self-driving a misclassification error is preferable as it is reported in planning-based metrics. Due to recent MOT methods [1]–[4] focus on associating detections from each class independently, they cannot handle the misclassification errors from the object detector. To alleviate this limitation, in this work we improve the robustness of a MOT module, by allowing detection-to-track association for similar classes.

A common assumption in MOT methods is that the detector can detect the whole bounding box. Unfortunately, some approaches and sensors are only able to reconstruct the visible part of the object, providing incomplete detections. As a consequence, it is one of our goals to propose a method to handle incomplete detections in a MOT module and demonstrate that it can be used to improve the robustness of state-of-the-art MOT methods.

Next, we review related works in MOT and Cooperative Perception.

Multi-object tracking: A MOT method takes as input a sequence of detections and it obtains a list of tracks with a sequence of states. The state of each track usually contains the position, velocity, orientation, and size. Only a variant of CBMOT [2] considers the acceleration as well. This is expected since common datasets do not include the acceleration in the annotations and do not consider it in the evaluation. However, being able to calculate the acceleration is a key factor when deploying these systems in real applications, e.g., to be able to react to a vehicle in front braking suddenly. There are three main tasks that a MOT method needs to handle: Associate detections to tracks, update the tracks based on those associations, and track life cycle management. Next, we introduce them.

In terms of data association, the methods can be classified into offline and online tracking. For offline tracking, the goal is to associate the global set of detections from different time stamps to a global set of tracks using, for example, network flows [5]. For online tracking, the goal is to associate the detections to the tracks for each time step independently. For real-time applications, only online tracking is possible as the detections from the future are not available. Most methods assume that one detection can be assigned only to one track and that one track can only be assigned to one detection, creating a bipartite matching problem that can be solved using a Hungarian algorithm [6], or others [7]. An alternative is to associate each detection sequentially to the closest track by order of confidence in a greedy way. This approach normally provides better results when there are a lot of detections and large differences in confidence [2], [3]. The key factor for data association is selecting a good distance, some examples include the use of the Intersection over Union (IoU), the Mahalanobis distance, the Euclidean distance, or even learned cost functions based on data. Detections can also be combined from different modalities to create a merged detection that is then associated to tracks [8]. These fusion mechanisms assume that the two modalities are synchronized, a requirement that could be hard to achieve in practice. In contrast, our method can work without this assumption. To the best of our knowledge, no previous MOT method considers association between different classes with no synchronization as we do in this work. This is especially relevant when a trained model is used on a different domain, since the model might detect some objects but fail to classify them properly, as we will see in the experimental section. Thanks to our method, this can be considered in the MOT system.

Regarding filtering, to update the tracks based on the associated detections, we need to define a state, how this state evolves over time (prediction), and how this state is modified when a detection is assigned to it (update). Most methods [1], [3], [4], use some kind of Bayesian filter to do that, such as Kalman Filters (KF) [9]. When low-frequency accurate detections with velocity are available, we can use a simpler approach [2], and predict the position of the detection in the last frame instead of predicting the position of the tracks in the current one. We can then set the state of the track to be equal to the state of the detection associated. All recent methods for 3D MOT assume that the complete bounding box of the object is detected, making it very complex to handle incomplete detections and achieve a good generalization when only the visible part of the object is detected. To improve the robustness of state-of-the-art MOT, we propose a method to cope with incomplete detections.

Finally, we discuss track management. Every MOT method needs some form of track management system to decide which tracks are considered as an output, which ones are inactive in the list, and which ones need to be removed from the list. A simple solution for that is to use count-based thresholds and output tracks that have a minimum number of successive detections, and delete tracks after a successive number of time steps without getting any detections associated [1]. A better alternative could be to use a score or confidence for each track. [2] shows that confidence-based methods outperform count-based methods when a proper score update function is used to combine the score of the detection as well as the track.

Cooperative Perception: Works proposing cooperative perception systems can be classified in terms of the information shared (raw sensor data, feature maps, or detections), in terms of the source of information (vehicles, other actors, or infrastructure), in terms of the environment for testing (simulation, proving grounds, or open road), in terms of real-time/offline processing, and depending on the scenarios tested. There is a lack of cooperative perception works analysing their performance in real-time real-world safety-critical scenarios with vulnerable road users such as pedestrians. Most works using real-world data focus on collected datasets and evaluate only the performance of the perception system when the vehicle-to-everything (V2X) communications are available [10], [11], but do not analyse how these communications can help actuate critical ADAS functions such as an AEB in real-time.

In this work we present a method to generate robust 3D detections from raw sensor data, including sensor-specific detection methods and a sensor-agnostic multi-object tracking module. This allows us to work with a large set of sensors and detection methods, without the need to synchronize them, and to generalize well to different real-world scenarios, thanks to the capability to handle incomplete detections and misclassifications. Moreover, our method can integrate V2X messages seamlessly. We test our method on three real-world challenging scenarios. A traffic jam chauffeur, a traffic monitoring application, and an AEB function being applied on safety-critical scenarios with Vulnerable Road Users (VRUs) such as pedestrians and cyclists. In these scenarios, we show the effectiveness of our sensor fusion solution.

2. Sensor-agnostic Multi-Object Tracking

Our key contribution is to present a versatile and configurable technique for multi-object tracking that can easily combine a wide variety of sensors and detection methods including RGB cameras, lidars, radars, and V2X messages either from other road users or from infrastructure. As the type of information we obtain with every sensor is different, our approach includes a perception module where all data are transformed to 3D detections. Finally, all 3D detections are fused by means of a KF [9] and a track management logic module, increasing the robustness of the detections. Figure 1 shows a diagram of the process we follow to merge these different modalities and obtain robust 3D detections.

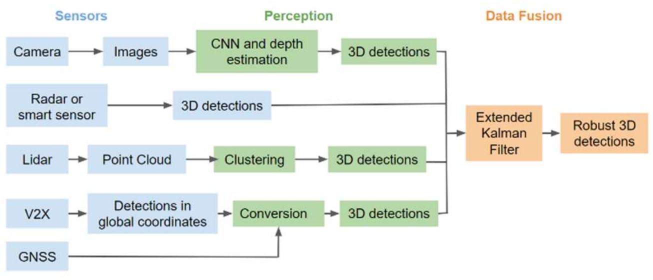

Figure 1 Detection and tracking pipeline.

To obtain 3D detections from visual information, we first use YOLO [12] trained on the COCO dataset [13] to obtain 2D bounding boxes, and then apply [14] for depth estimation. The resulting 3D detections have a larger positioning error in the direction of the principal axis of the camera, and as the object gets further away this effect is increased. but this is handled by the MOT as it considers the covariance matrices for the detections.

To process lidar data, we use a height-based filter to remove the ground so that different objects are not connected, and then use a clustering algorithm to obtain bounding boxes of the visible parts of all the objects of the scene. These detections do not have an associated class, but this will be obtained in the MOT module along with the reconstruction of the complete bounding box.

In terms of V2X integration, the vehicle receives via V2X the positions of the other objects in the scene in global coordinates. To transform from global coordinates to the relative position in the ego vehicle frame of reference, we get the position of the vehicle at the time from the localization module on the vehicle, based on a GNSS system with a Kalman Filter with a constant velocity model. Once we have the two latitudes and longitudes at that time, we convert them to UTM coordinates and then calculate the relative position. The confidence α is set to 1 and the covariance is set according to the error of the localization KF on the ego vehicle and the positioning error of the V2X message. Then we pass this information to the data fusion module that processes the information as it would process detections from a sensor.

In the sensor fusion module, the input corresponds to a sequence of 3D detections from different sensors, composed by a timestamp and a vector $\boldsymbol{d} = [r_x, r_y, r_z, w, h, l, \theta, c, \alpha, \sigma_{rx,rx}, \sigma_{rx,ry}, \sigma_{ry,ry}]^\top$, that includes the center of the object (r_x, r_y, r_z), the width w, height h, and length l of the bounding box, the yaw rotation θ in the axis perpendicular to the road plane –pitch and roll angles can be ignored in this context–, the class of the object c, the corresponding confidence score α, and the 2D-location covariance values ($\sigma_{rx,rx}$, $\sigma_{rx,ry}$, $\sigma_{ry,ry}$). For radar sensors, we also consider the velocity $\boldsymbol{v} = [v_x, v_y]^\top$ and its covariance entries $\boldsymbol{\sigma_v} = [\sigma_{vx,vx}, \sigma_{vx,vy}, \sigma_{vy,vy}]^\top$. Since all these detections are encoded into the same representation, the MOT module can process them in the same way without needing to take into account the modality of origin.

We keep a track list of all the objects that have been detected and are being tracked in the scene, and this list is shared for all sensors. Every track in the list stores a KF [13] with a constant velocity model. At time t_k, a given track stores the state $\boldsymbol{s}_k = [r_x, r_y, v_x, v_y]^\top$ and its covariance $\boldsymbol{P}_{k,i}$. Then, we can update this track list asynchronously every time that we receive a new set of detections from any sensor. When we receive a detection, we consider the time difference $\Delta t = t_k - t_{k-1}$, between the time of detection and the last time of update for each track, then all the tracks are updated to the time of the detection following the KF prediction step, and associated to tracks by minimizing the global cost of the associations. We consider the Mahalanobis distance to define these costs, as it considers the covariance of the detections and tracks. Once a detection has been associated to a track, that detection is used to update the track that has been associated with by following the KF update step. If the detection is not associated, it can be used to create a new track. Then a track remover periodically checks the track list and removes tracks that have dropped the confidence below a certain threshold.

Some terms are not kept in the state of the KF, and they are instead computed separately. This helps to generalize better to a larger set of sensors that cannot estimate these terms. For instance, we use the orientation of the velocity vector included in the state to calculate the yaw angle or heading, we can also choose to calculate it from the orientation of the bounding boxes when complete and accurate detections are available.

We also infer the acceleration by filtering the local differences in velocity from the KF state. Given the velocity of the track $\boldsymbol{v}_k$ at the current time step t_k, and the acceleration $\boldsymbol{a}_{k-1}$ and velocity $\boldsymbol{v}_{k-1}$, at the previous time step t_{k-1}, the instant acceleration $\hat{\boldsymbol{a}}_k$ is calculated as $\hat{\boldsymbol{a}}_k = \max(\min((\boldsymbol{v}_k - \boldsymbol{v}_{k-1})/(t_k - t_{k-1}), b\cdot\boldsymbol{1}_2), -b\cdot\boldsymbol{1}_2)$, where max and min indicate the enterwise max/min operators respectively, $\boldsymbol{1}_2$ represents a 2-dimensional vector of ones, and b is a reasonable maximum for acceleration which for driving scenarios can be set to 6m/s^2. Then we filter the acceleration considering previous accelerations as $\boldsymbol{a}_k = \gamma \cdot \boldsymbol{a}_{k-1} + (1-\gamma)\hat{\boldsymbol{a}}_k$ for $\gamma \in [0,1]$. We use $\gamma = 0.8$.

To update the confidence score, we use the multiplication score update function from [2]. Next, we introduce our approach to calculate the width, height, and length of the bounding boxes, and the class of the track in a way that is robust to errors in these values from the object detectors.

2.1. Handling misclassifications

Learning-based methods tend to have a lower performance when they are applied to sensors and scenarios different from the dataset they have been trained with [15]. We have seen that misclassification errors between similar classes (e.g., trucks and cars) are very common, and although the method might classify the object correctly in most of the frames, it sometimes misclassifies it. To handle this in the MOT module, we first associate tracks and detections of the same class, and then we associate tracks and detections from similar classes. Which classes are similar can be configured based on a confusion matrix when available or empirically. In this way we can still associate detections with tracks when a misclassification occurs. We keep a vector with the number of times a detection from each class has been associated with the track and output the class with more associations.

2.2. Handling incomplete detections

Some sensors, like some radars, can only detect a part of the object or even only one point of it. To be able to associate these partial detections with complete bounding box tracks, we select a minimum bounding box size for each class, and considering that the closest part of the complete bounding box is detected, we resize and reorient the box if necessary, in a way that the closest point remains the same. First, we calculate the intersection of the bounding box with the segment from the center of the bounding box to the origin of the sensor, this is the closest point of the bounding box. If the detection is a point, we can skip this step. Then we set the width, length, and height to the maximum between the detected values and the minimum bounding box for the class. When the orientation is unreliable or not available, we can set it to the orientation of the candidate track we are considering for association.

3. Experiments

3.1. MOT for a traffic jam chauffeur

We use our sensor fusion approach as a perception module for a challenging Traffic Jam Chauffeur scenario. The vehicle is equipped with a front radar (Continental ARS 408-21) and a front camera with integrated detections (Mobileye 630). The detections from the camera have a much higher frontal error than lateral one, and they will be fused with those of the radar. Our vehicle and the corresponding target one in front are equipped with high-precision dGNSS and they communicate with each other so that we can get their relative position at any time with an accuracy of 1cm. These data are used as ground truth. The camera's main contribution is to create and classify the tracks, given that the radar produces too many false positives to be used for track initialization. We can see that the performance of the radar is much better than the camera, and our sensor fusion is able to outperform the radar by leveraging radar and camera detections over time. Figure 2 shows the evolution of the error in longitudinal position and velocity of each sensor over time along with the reference position and velocity, and the estimations of the sensor fusion for a scenario where the ego vehicle is driving at 60km/h and approaching a vehicle driving at 20km/h in the same lane. The ego vehicle reacts and adjusts its speed accordingly.

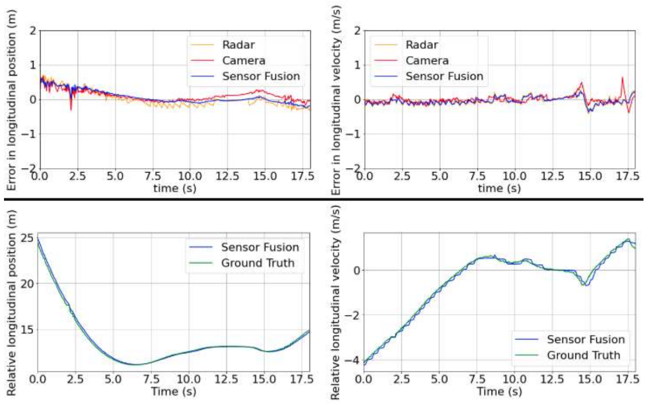

Figure 2 Traffic Jam Chauffer results: Longitudinal position and velocity for each sensor, the sensor fusion, and the ground truth, along with errors.

3.2. MOT for traffic monitoring

To show how our system can be used to monitor the traffic on highways, we propose a real experiment where we equip two vehicles with cameras, radars, and lidars, in order to calculate the trajectories of vehicles around. We then use our perception system to determine the position, size, class, velocity, and acceleration of every vehicle over time. Unfortunately, for this experiment we do not have a ground truth for quantitative evaluation and, therefore, we provide qualitative results. Our results are displayed in Figs. 3-5. Figure 3 shows the whole scenario, while in Fig. 4 we zoom in to the detections in front for an interesting time instance ignoring the radar and low-resolution frontal lidar. As it can be seen in Fig. 4 the lidar clustering algorithm is capable of detecting the truck in front but partially. The camera is also detecting the truck, using YOLOv4 [12] and [14], but misclassifying the truck as a bus. These errors are handled by the MOT using our method and the output is the expected bounding boxes. In Fig. 3 we can see that the camera detections have a larger covariance as the distance to the object increases, but the detections from lidars and radar have a fixed covariance. The covariance of the tracks is lower than those of the detections whether it gets enough detections associated constantly and increases rapidly when no detections are available. In Fig. 5 we can see the results of our method over time, we plot the evolution of the relative frontal position of multiple objects around the ego vehicle along with the detections for a given time.

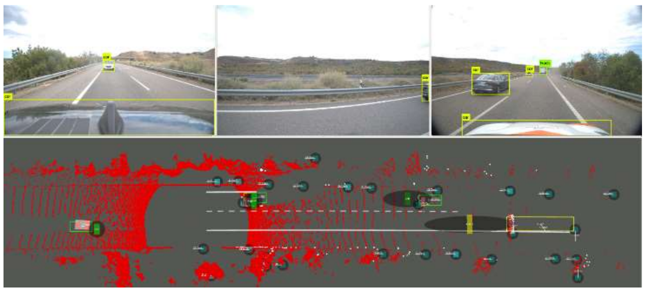

Figure 3 Traffic Monitoring on highways. 360º lidar in red, lidar detections in grey, frontal lidar in white, camera detections color-coded as filled boxes, radar in cyan, and sensor fusion as box edges color-coded.

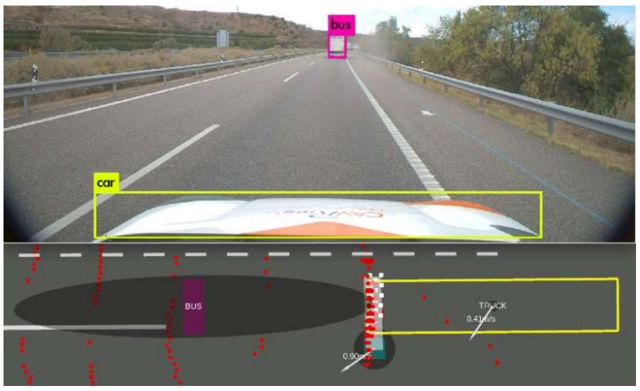

Figure 4 Example of handling of misclassifications.

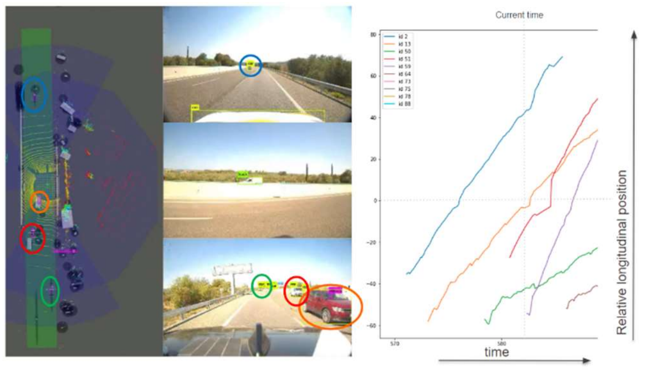

Figure 5 Relative longitudinal trajectories (right) along with sensor data and detections for an instant (left).

3.3. MOT with V2X messages

Finally, we use our system to evaluate whether the addition of V2X communications can improve the safety of pedestrians and cyclist in challenging urban scenarios with occlusions. We consider different scenarios, similar to the one in Fig. 7, where the vehicle is driving at constant speed and a pedestrian comes out of an occlusion perpendicularly to the vehicle at a speed that would generate a collision. The ego vehicle has an AEB function that will brake the vehicle to a full stop if the collision can be predicted successfully. Clearance and crashing statistics are shown in Fig. 8 for scenarios with different speeds of the ego vehicle V_S (25,30,35,40,45km/h for tests 1-5 respectively), and the pedestrian at $V_T = 8$km/h. We can see that the performance improves with V2X messages.

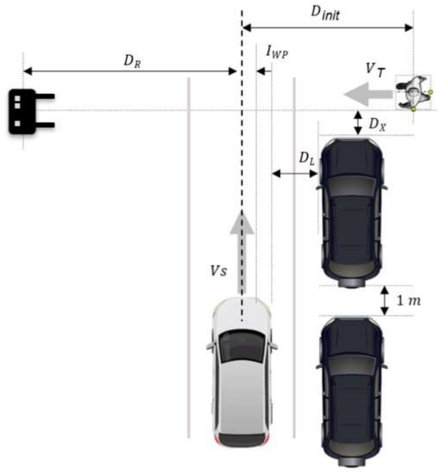

Figure 7 V2X Scenario 1 with a pedestrian.

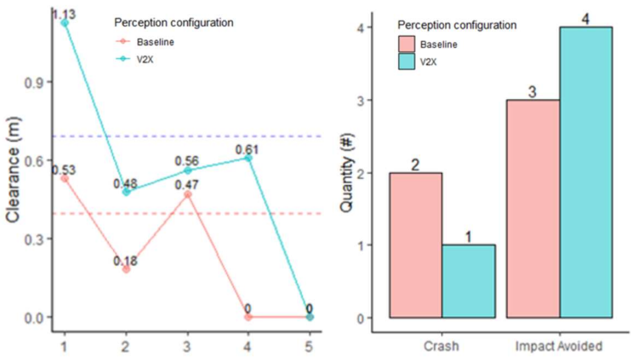

Figure 8 Clearance and total crashes in V2X Scenario 1.

4. Conclusion

We presented a versatile and configurable technique for MOT that can combine easily a wide variety of sensors and be robustly used in different scenarios. We discussed errors in classification and bounding box size of current object detectors and proposed a method to handle these errors in the MOT module. We argued the importance of acceleration information to deploy MOT systems in real applications, and we tested our approach integrating it with a traffic jam chauffeur function. We showcased the general nature of our system by mounting cameras, lidars, and radars on two vehicles to monitor the traffic. And we used our system to show that V2X messages can improve the safety of pedestrians in safety-critical scenarios. Since the MOT module is sensor agnostic, we believe that other works can benefit from our contributions.

Acknowledgments: This work has been supported by the project MoHuCo PID2020-120049RB-I00 funded by MCIN/ AEI/10.13039/501100011033, and by the Government of Catalonia under 2020 DI 105.

References

[1] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani, "3D Multi-Object Tracking: A Baseline and New Evaluation Metrics," in IROS, 2020.
[2] Nuri Benbarka, Jona Schr¨oder, and Andreas Zell, "Score refinement for confidence-based 3D multi-object tracking," in IROS, 2021.
[3] Ziqi Pang, Zhichao Li, and Naiyan Wang, "Simpletrack: Understanding and rethinking 3D multi-object tracking," arXiv preprint arXiv:2111.09621, 2021.
[4] Hsu-kuang Chiu, Jie Li, Rares¸ Ambrus, and Jeannette Bohg, "Probabilistic 3D multi-modal, multi-object tracking for autonomous driving," in ICRA, 2021.
[5] Li Zhang, Yuan Li, and Ramakant Nevatia, "Global data association for multi-object tracking using network flows," in CVPR, 2008.
[6] Harold W Kuhn, "The hungarian method for the assignment problem," Naval research logistics quarterly, vol. 2, no. 1-2, pp. 83–97, 1955.
[7] David F Crouse, "On implementing 2D rectangular assignment algorithms," IEEE Transactions on Aerospace and Electronic Systems, vol. 52, no. 4, pp. 1679–1696, 2016.
[8] Aleksandr Kim, Aljoˇsa Oˇsep, and Laura Leal-Taix´e, "Eagermot: 3D multi-object tracking via sensor fusion," in ICRA, 2021.
[9] Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.
[10] Qi Chen, Sihai Tang, Qing Yang, and Song Fu, "Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds," in 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2019, pp. 514–524
[11] Yanghui Mo, Peilin Zhang, Zhijun Chen, and Bin Ran, "A method of vehicle- infrastructure cooperative perception based vehicle state information fusion using improved kalman filter," Multimedia Tools and Applications, vol. 81, no. 4, pp. 4603–4620, 2022
[12] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll´ar, and Lawrence Zitnick, "Microsoft COCO: Common objects in context," in ECCV, 2014.
[14] Apoorva Joglekar, Devika Joshi, Richa Khemani, Smita Nair, and Shashikant Sahare, "Depth estimation using monocular camera," International journal of computer science and information technologies, vol. 2, no. 4, pp. 1758–1763, 2011.
[15] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao, "Train in germany, test in the usa: Making 3d object detectors generalize," in CVPR, 2020.