2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Visual Perception System for Autonomous Driving

Qi Zhang†, Siyuan Gou†, and Wenbin Li†

***Abstract*— The recent surge in interest in autonomous driving is fueled by its rapidly developing capacity to enhance safety, efficiency, and convenience. A key component of autonomous driving technology lies in its perceptual systems, where advancements have led to more precise algorithms applicable to autonomous driving, such as vision-based Simultaneous Localization and Mapping (SLAM), object detection, and tracking algorithms. This work introduces a visual-based perception system for autonomous driving that integrates trajectory tracking and prediction of moving objects to prevent collisions while addressing the localization and mapping needs of autonomous driving. The system leverages motion cues from pedestrians to monitor and forecast their movements while simultaneously mapping the environment. This integrated approach resolves camera localization and tracks other moving objects in the scene, ultimately generating a sparse map to facilitate vehicle navigation. The performance, efficiency, and resilience of this approach are demonstrated through comprehensive evaluations of both simulated and real-world datasets.**

I. Introduction

Perception, including object recognition and prediction, plays a crucial role in autonomous driving systems. In particular, reconstructing the map of the surrounding environment, identifying moving objects, and predicting their trajectories, are essential for ensuring safety in real-world autonomous vehicle operation. Although different perception solutions may rely on various sensors, such as LiDAR, GPS, IMU, RGB-D, etc. [1], the visual only method stands out for its cost-effectiveness.

Autonomous driving under unstructured scenarios poses a significant challenge for Visual SLAM (V-SLAM) systems, as they are required to handle dynamic objects on the road, such as pedestrians, vehicles, etc., using only visual images. Traditionally, visual SLAM systems prioritize enhancing odometry accuracy by detecting and tracking moving objects, as demonstrated in DynaSLAM [2], Mid-fusion [3]. Nevertheless, recent studies, such as VDO-SLAM [4] and Cubeslam [5], have attempted to detect and incorporate moving objects into the static environment structure.

While these methods often overlook leveraging semantic clues from moving objects, DynaSLAM2 optimize the camera ego-motion together with the movement of the moving objects in a feature-based framework [6].

The real-world driving environment often presents complexities arising from the frequent presence of numerous moving objects. Despite advancements, there remains a notable gap in the literature concerning the prediction of moving object trajectories and the implementation of a dynamic SLAM solution to enhance collision avoidance during autonomous vehicle navigation. *Addressing this gap, our work introduces a V-SLAM system that seamlessly integrates moving object tracking and trajectory prediction*, which not only enhances localization and mapping accuracy but also significantly improves motion planning and collision avoidance capabilities in autonomous driving scenarios.

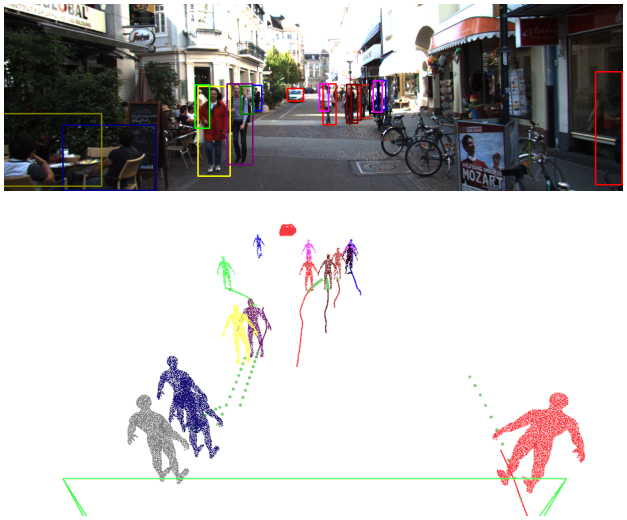


Fig. 1: A brief overview of our perception system's output: the top image shows an RGB frame, while the bottom image depicts the corresponding perception result. It includes reconstructed moving objects, color-coded for classification, with their predicted trajectories marked by green dashes.

Visual cues have gained popularity within the community for extracting rich semantic information from moving objects in dynamic V-SLAM systems. Several data-driven approaches, including QD-3DT [7], TripletTrack [8], and CC-3DT [9], are capable of detecting various moving objects, such as pedestrians, vehicles, and cyclists. This led us to enhance our perception system by incorporating both 2D and 3D clues from moving objects. However, the visual information of moving objects often introduces jitter into the system, especially when a large number of fast-moving objects are present in the view. This occurs because the 3D information of moving objects is discretely learned from frame to frame, leading to a nonlinear nature that can lose information and produce noisy and discontinuous training outcomes. *Our proposed method effectively addresses such issues through a lightweight trajectory prediction approach, enabling reliable and precise predictions.*

To achieve effective collision avoidance, the system requires comprehensive access to scene information and the ability to predict the motion trajectories of moving objects.

†Department of Computer Science, University of Bath, UK, {qz727, sg2655, w.li}@bath.ac.uk

However, current studies utilize models trained on top-view street images with fixed camera views, concentrating solely on predicting pedestrian trajectories in comparable road scenarios [10] [11] [12] [13] [14] [15]. In contrast, autonomous vehicle systems often employ moving cameras and navigate diverse road conditions, posing challenges for existing methods in terms of generalization. Furthermore, some autonomous vehicles rely solely on front-facing camera information, which may not align with the requirements of many existing approaches. Thus, we develop an approach to bridge the research gap between current visual perception systems for autonomous driving, and V-SLAM systems with real-time pedestrian and vehicle trajectory prediction.

In this paper, we introduce a stereo SLAM system that is capable of tracking and recognizing dynamic objects as shown in Fig. 1. Furthermore, we present a method that reduces the noise associated with the 3D information of machine learning-based objects when integrating them into the SLAM system. Our key contributions are summarized as follows:

- A visual-based perception system for autonomous driving scenarios incorporates a dynamic SLAM algorithm (that removes the moving objects as outliers), together with moving object trajectory optimization and prediction. This can take into account any 2D and 3D bounding box detection method.
- A concise architecture for predicting trajectories of multiple moving objects, including both pedestrians and vehicles, simultaneously for each frame. This approach can also re-track object trajectories in situations where the data driven tracking method fails in a short period, leading to more accurate and robust tracking.
- A method solves the jitter problem when applying neural network-based object tracking methods to real-time SLAM systems, enabling more accurate trajectory predictions.

II. Related Work

Typically, Visual SLAM systems operate under the assumption of a static environment, exemplified by the ORB-SLAM series [16] [17]. However, dynamic environments present a formidable challenge for feature association across views or frames. Current research in SLAM systems within dynamic scenes often treats features in dynamic regions as outliers, subsequently removing them to enhance pose estimation accuracy [2] [18]. Alternative approaches have emerged, focusing on neural networks for semantic recognition of moving objects [6] [19], and utilizing 3D object information to optimize camera poses [5]. Additionally, dense optical flow algorithms, as demonstrated in studies such as [4] and [20], differentiate between moving and static objects, attempting to track these objects. However, in autonomous driving systems, not only is localization and mapping in dynamic scenarios crucial, but also predicting object trajectories is of paramount importance, highlighting a notable research gap in this area.

Extensive research has been conducted on object detection and tracking in 2D images and videos, with established methods like the YOLO series [21], Perma-track [22], and Observation-Centric SORT [23], offering robust solutions for tracking complex objects in image sequences. Tracking 3D objects from 2D images presents a challenging task, requiring the recognition of objects in 2D images and the extraction of directional, volumetric, rotational, and translational information from image sequences. To tackle this challenge, [7] incorporates Long-Short Term Memory (LSTM) motion extrapolation, integrating previous 3D information with current data when relying solely on visual images. However, applying the detected position to an online SLAM system often introduces jittering issues in sequential results. Therefore, we propose a lightweight method to eliminate jittering when utilizing our SLAM system.

To predict pedestrian motion trajectories, a common approach involves interleaving an LSTM network with a message-passing mechanism to effectively extract social influence from neighbors, exemplified by SR-LSTM [10] and SRAI-LSTM [11]. Other methods employ alternative network structures, such as those described in [12] and [13], which may also support vehicle trajectory prediction, as demonstrated by Grip [14] and Grip++ [15]. However, these methods are often trained on static top-view cameras and may require substantial computational resources, rendering them cost-prohibitive for autonomous driving systems. To tackle this challenge, CAPOs [24] develops an autonomous driving system within a simulated environment using CARLA [25]. Although it employs deep-learning methods to predict object movement, enabling the autonomous car to efficiently mitigate collision risks within a 3-second prediction window, it lacks localization and mapping capabilities. Consequently, the system exhibits a notable drawback in terms of a high Average Displacement Error (ADE).

III. Method

A. Perception system with SLAM system

As shown in Fig. 2, we propose a perception system that tightly integrates an online SLAM system, object tracking, and trajectory prediction. When processing moving objects (e.g., vehicles, pedestrians), we track their positions upon the arrival of a new frame F. We obtain a set of objects $\{O_N,\ldots,O_M\} = \{O_i\}_{i=N}^{M}$ for each frame, where the subscripts correspond to the object ID $\{N,\ldots,M\}$. The centroid of an object is represented as $\{P_N,\ldots,P_M\} = \{P_i\}_{i=N}^{M}$, where $P_i = (x_i, y_i, z_i)$. The heading directions of the objects, presented as the Euler angle with axis z, are denoted as $\{D_N,\ldots,D_M\} = \{D_i\}_{i=N}^{M}$, where $D_i \in (-\pi,\pi)$.

In addition to tracking moving objects, we concurrently predict their trajectories based on real-time positions. By aggregating the central points of each frame, we derive a set of object trajectories $\{S_N,\ldots,S_M\} = \{S_i\}_{i=N}^{M}$. Assuming the current frame is F_k, an object trajectory S_i is defined as $\{P_i^j\}_{j=a}^{k}$, where a represents the frame in which the object first appeared. Moreover, the prediction of object trajectories within the future time interval T_p is denoted as

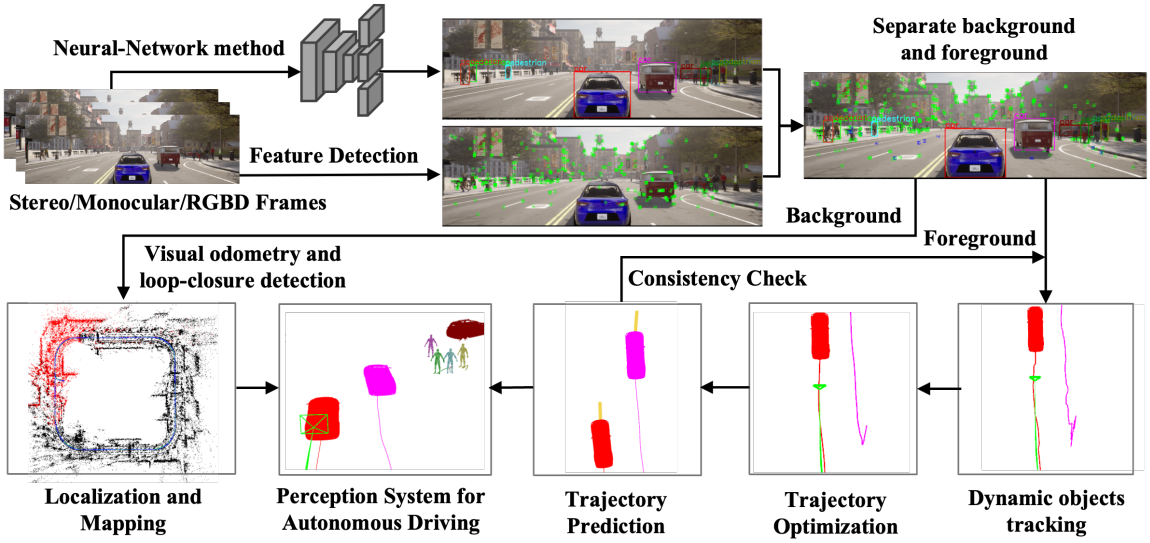


Fig. 2: The overview of our system. We first employ data-driven 3D bounding box detection method on the frame image. Next, we extract background features and match them to create a sparse 3D map using ORB-SLAM2 [17]. Concurrently, we estimate the trajectories of moving objects and predict their future paths over a short time period.

Algorithm 1 Online Trajectory Prediction

Input: $\{S_i\}_{i=N}^{M}$

while Detect new frame F_k **do**
  for S_i in $\{S_i\}_{i=N}^{M}$ **do**
    Consistency check (Sec. III-C.3)
    Kalman filter on S_i (Sec. III-C.2)
    if $|S_i| < \tau_{KF}$ **then**
      Initialization trajectory (Sec. III-C.2)
      Update S_i
    else if $|S_i| > \tau_{KF}$ **then**
      Local trajectory optimization (Sec. III-C.1)
      Update S_i
      Calculate $P_{s,i}$ $P_{c,i}$ $P_{e,i}$ $C_{p,i}$ (Sec. III-B)
    end if
  end for
end while

Output: $\{C_{p,i}\}_{i=N}^{M}$

$C_{p,i} = \{P_i^j\}_{j=k}^{k+b}$, where $b = T_p/\Delta T$ is the number of future trajectory prediction points. We then utilize $\{C_{p,i}\}_{i=N}^{M}$ to assess the consistency of object tracking results.

B. Heading-based Moving Objects Trajectories Prediction

While we can track moving objects individually using neural network-based methods, real-time prediction of their trajectories becomes challenging, particularly when numerous objects are observed in each frame. Therefore, we introduce a lightweight, fast-moving object trajectory prediction algorithm based on the head direction vector. For detailed information, please refer to Algorithm 1.

To streamline the problem and better suit the autonomous driving context, we opt not to consider the z axis (vertical direction) when predicting trajectories. This choice stems from minimal displacement observed along the z axis and other relevant factors specific to autonomous driving.

We assume that real-world motion patterns of moving objects, such as pedestrians and vehicles, follow independent motion curves. While these curves may intersect, our primary focus is on ensuring our vehicle avoids collisions with pedestrians or other vehicles, disregarding collisions between external entities themselves. In this approach, we analyze a sequence of past object positions to identify patterns and predict future trajectories. The object trajectory prediction model, depicted in Fig. 3a, illustrates a section of the trajectory of interest represented by curve C_p.

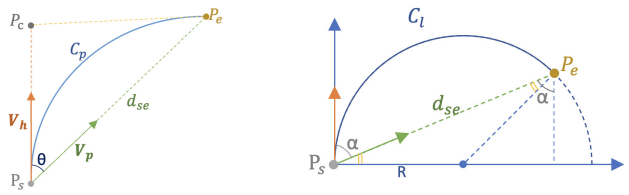
For short time intervals, target trajectories typically exhibit minimal curvature in various directions. To approximate such trajectories, we utilize a quadratic Bezier Curve, requiring three points: P_s, P_e, and P_c (with P_s as a known point).

To determine the position of P_c, we use the head direction vector $\mathbf{V_h}$ and the endpoint P_e. Subsequently, to calculate the position of P_e, we rely on the predicted vector $\mathbf{V_p}$ and the trajectory distance $d_{se} = \overline{P_s P_e}$.

*1) **Head Direction Vector**:* The head direction D_i obtained from per-frame object detection is converted into unit direction (normal) vectors $\hat{\mathbf{d}}_i$. However, directions from previous learning-based detections are often noisy and unreliable. Therefore, we calculate the instantaneous tangent vector $\mathbf{a}_i = \overrightarrow{P_i^{k-1} P_i^k}$ at the current object position with respect to previous trajectories, and convert it into a unit vector $\hat{\mathbf{a}}_i$.

If the angle between $\hat{\mathbf{a}}_i$ and $\hat{\mathbf{d}}_i$, denoted as θ_i, is not greater than a predefined threshold τ_θ, we define the head direction of the i-th object as a weighted sum $\mathbf{V_{h},}_i = w_1 \hat{\mathbf{a}}_i + w_2 \hat{\mathbf{d}}_i$, where w_1 and w_2 represent the weights assigned to the unit vectors. This method yields a more accurate and reliable $\hat{\mathbf{V}}_{\mathbf{h},i}$.

*2) **Prediction Vector**:* To derive the prediction vector $\mathbf{V_{p},}_i$, we assume that the trajectories maintain consistent



(a) Trajectory Model. (b) The method to calculate d_{se}

Fig. 3: (a) The model of trajectory prediction. P_s is the moving object position in the current frame, while P_e is the predicted position. The curve C_p and line d_{se} are the trajectory and distance between them respectively. The unit vector $\mathbf{V_h}$ is the head direction of the object, and $\mathbf{V_p}$ is the prediction direction. (b) The method to calculate d_{se}.

momentum within the time interval ΔT. Three points $P_{i,m} = \left(x_i^{k-mn\Delta T}, y_i^{k-mn\Delta T}, z_i^{k-mn\Delta T}\right)$ are selected, where $m \in \{0,1,2\}$ and n represents the frame rate. Unit vectors between $P_{i,0}$ and $P_{i,1}$, denoted as $\hat{\mathbf{v}}_{i,01}$, and between $P_{i,1}$ and $P_{i,2}$, denoted as $\hat{\mathbf{v}}_{i,12}$, are calculated. The prediction vector is subsequently computed as $\mathbf{V_{p},}_i = w_3\hat{\mathbf{v}}_{i,01} + w_4\hat{\mathbf{v}}_{i,12}$, with w_3 and w_4 representing the weights assigned to each vector. Finally, the prediction vector is normalized to yield $\hat{\mathbf{V}}_{\mathbf{p},i}$.

*3) **Trajectory Distance**:* To maintain the continuity of trajectories, we initially apply a smoothing process as outlined in section III-C. Once we have these smoothed trajectories, we then compute the corresponding smoothed instantaneous velocity for each object in every frame, along with their velocity derivatives.

Next, for each frame, we extract the smoothed instantaneous velocity v_i by considering the past h frames.

$$v_i = \frac{\gamma}{h}\sum_{j=k-h}^{k}\frac{|P_i^{j+1} - P_i^j|}{\Delta T + \varepsilon} \tag{1}$$

where γ represents the smoothing factor, ΔT denotes the time interval with $\Delta T = t^{j+1} - t^j$, and ε acts as a small constant to prevent instability. Consequently, the length of the curve $C_{p,i}$ can be determined as $|C_{p,i}| = v_i \Delta T$. For each frame, we then derive a set of lengths corresponding to the predicted curves $\{|C_N|, \ldots, |C_M|\}$ for the objects $\{O_N, \ldots, O_M\}$.

We further introduce two additional assumptions. Firstly, as previously mentioned, we assume that trajectories remain constant along the z axis. Secondly, if the angle $\alpha_i = \angle(\mathbf{V_{h}},_i, \mathbf{V_{p}},_i)$ is less than the threshold τ_α, we consider the length error $e(|C_{p,i}|, d_{se,i}) = |||C_{p,i}| - d_{se,i}||$ to be negligible. This allows us to approximate the trajectory length $d_{se,i}$ as:

$$d_{se,i} \approx |C_{p,i}| \tag{2}$$

Conversely, we assume that the error $e(|C_{p,i}|, |C_{l,i}|) = |||C_{p,i}| - |C_l||$ can be neglected. Thus, we approximate $|C_{l,i}|$ to be approximately equal to $|C_{p,i}|$, where $C_{l,i}$ corresponds to a segment of the circle depicted in Fig. 3b as $C_{l,i} = \overset{\frown}{P_{s,i}P_{e,i}}$. Given the angle α_i and the length of $C_{l,i}$, we can calculate $d_{se,i}$ using the equation:

$$d_{se,i} = \frac{|C_{l,i}|}{\alpha_i}|sin(\alpha_i)| \tag{3}$$

where $R_i = |C_{l,i}|/(2\alpha_i)$.

After acquiring the head direction vector $\hat{\mathbf{V}}_{\mathbf{h},i}$, prediction vector $\hat{\mathbf{V}}_{\mathbf{p},i}$, and trajectory distance $d_{se,i}$ for each object O_i, we proceed to compute the distance $d_{sc,i}$ between $P_{s,i}$ and $P_{c,i}$, which we set to half of $|C_{p,i}|$. Therefore, the trajectory prediction $C_{p,i}$ for object O_i can be derived as follows:

$$\begin{aligned} P_{s,i} &= P_i^k \\ P_{c,i} &= P_{s,i} + d_{sc,i} \cdot \hat{\mathbf{V}}_{\mathbf{h},i} \\ P_{e,i} &= P_{s,i} + d_{se,i} \cdot \hat{\mathbf{V}}_{\mathbf{p},i} \\ C_{p,i} &= \text{Bezier}(P_{s,i}, P_{c,i}, P_{e,i}) \end{aligned} \tag{4}$$

Consequently, for each frame, we derive a series of predicted trajectories $\{C_N, \ldots, C_M\}$ at the current frame.

C. Moving Objects Trajectories Updating

The accuracy of past trajectories plays a crucial role in forecasting the future paths of objects in motion within a scene. The trajectories of moving 3D objects must demonstrate coherence and retain smooth curves within the 3D environment. A particular challenge arises from the fact that 3D bounding boxes generated by neural networks are constructed frame by frame. This results in a misalignment between the trajectory, which comprises discrete object central points, and the continuous camera movement. Consequently, the motion of these moving objects appears erratic within the 3D environment. This observation motivates us to establish motion priors aimed at refining and smoothing out these trajectories.

*1) **3D Trajectory Motion Priors**:* It has been proved that accurate trajectories within small time intervals can be determined by minimizing costs associated with predefined priors, as shown in [26]. Hence, we utilize the least kinetic motion prior cost to refine prior trajectories. For a moving object O_i within the time interval ΔT, the least kinetic motion prior cost is expressed as $Q_i = \frac{1}{2} r_i \overline{v_i}^2 \Delta T$, where r_i represents the weight associated with object O_i, and $\overline{v_i}$ denotes the velocity of object O_i during the time interval ΔT. Therefore, at the current frame F_k, we define the cost of previous trajectories as:

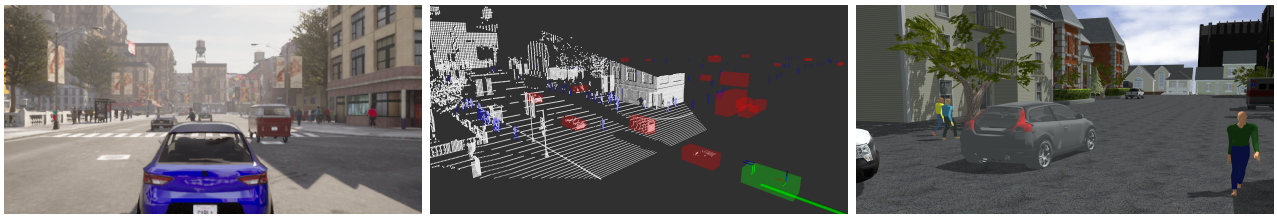
$$Q_{i,k} = \sum_{j=k-h}^{k}\frac{1}{2} r_i \left(\frac{|P_i^{j+1} - P_i^j|}{t^{j+1} - t^j + \varepsilon}\right)^2 \left(t^{j+1} - t^j\right) \tag{5}$$

Next, we obtain the Jacobian matrix of the cost function with respect to points P_i^{j+1} and P_i^j as:

$$\frac{\partial Q_{i,k}}{\partial P_i^{j+1}} = \frac{r_i}{|P_i^{j+1} - P_i^j|\Delta T}\left[x_i^{j+1} - x_i^j,\ y_i^{j+1} - y_i^j,\ z_i^{j+1} - z_i^j\right] \tag{6}$$

$$\frac{\partial Q_{i,k}}{\partial P_i^{j}} = \frac{-r_i}{|P_i^{j+1} - P_i^j|\Delta T}\left[x_i^{j+1} - x_i^j,\ y_i^{j+1} - y_i^j,\ z_i^{j+1} - z_i^j\right] \tag{7}$$

With the Jacobian matrix, we use Levenberg-Marquardt [27] to refine the trajectories between frame F_{k-b} and F_k. This yields the optimized trajectories $\{S_i\}$ along with the updated 3D points $\{P_i^j\}_{j=k-h}^k$, where $i \in \{N, \ldots, M\}$.



(a) CARLA simulator (b) Ground truth visualization (c) GAZEBO simulator

Fig. 4: (a) The CARLA simulator spawns a large number of pedestrians and cars, and generates images with ego vehicle view. (b) The object detection and trajectory ground truth are visualized. The green, red and blue boxes denote the ego vehicle, generated cars and generated pedestrians, respectively. (c) The GAZEBO simulator is also established for evaluation.

*2) **Initialization**:* As previously mentioned, the acquisition of raw moving object trajectories, from each frame generated by the neural network can introduce significant noise, potentially resulting in pronounced jittering in trajectory estimation. To address this issue, a preliminary step is taken before applying the trajectory update method described above. This initial step involves using Kalman Filters (KF) [28] to initialize the generated 3D points. The Kalman Filters are utilized to estimate the temporal state of each object, with one linear filter dedicated to each object, considering a total of 18 states. Additionally, when the length of the trajectory falls below a predefined threshold, denoted as τ_{KF}, we employ n-point moving average (NPMA) filters in conjunction with the Kalman Filters.

*3) **Trajectory Consistency Check**:* The object detection system, relying on neural network methods, may encounter intermittent tracking failures, leading to inconsistent and inaccurate object tracking. To address this issue, we assess the consistency of object positions with previous trajectories and apply a Gaussian filter $G(P) \sim \mathcal{N}(\mu, \sigma^2)$ to smooth the noisy position data P. When a new object is detected in frame F_k, we initialize a new trajectory and merge it with any previously detected trajectories if the trajectories predicted in the previous frame F_{k-1} align with the positions within the probability distribution $G(P)$. Otherwise, we update the existing trajectory. In cases where two trajectories cross, we choose the one closest to the center of the distribution.

IV. Experiments

A. Experimental Setup

Simulation and real-world datasets. We consider two categories of benchmark datasets: 1) simulated outdoor environments implemented by CARLA [25] and GAZEBO [29] integrated with the ROS [30] system, and 2) the real-world KITTI datasets [31]. In CARLA, we have designed diverse scenarios/datasets featuring over 400 moving pedestrians and vehicles, comprising nearly 10,000 frames. These scenarios include 5 highly dynamic sequences labeled as **CARLA 01** to **0**5 and a closed-loop sequence called CARLA Easy Loop (**CARLA EZL**). An example of our CARLA environments is depicted in Fig. 4a. In addition, we also utilized the GAZEBO environment with a loop in the scene, designated as **GAZEBO L**. Furthermore, we incorporated outdoor sequences from the MOTChallenge dataset [32], [33] within the KITTI dataset for multi-object tracking and evaluation (**MOT KITTI**).

All experiments were conducted on a computer equipped with an AMD 5600X CPU, 2080ti GPU, with 12GB memory.

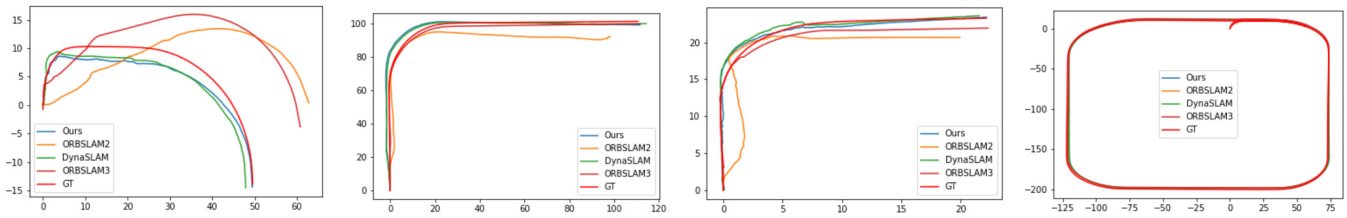
Baseline methods. We build our V-SLAM system upon ORB-SLAM2 [16] and utilize the open-source system QD-3DT [7] for applying neural network-generated 2D and 3D bounding boxes. In our evaluation, we compare our SLAM system with three state-of-the-art techniques: ORB-SLAM2, ORB-SLAM3 [17] and DynaSLAM [2]. ORB-SLAM3 represents an upgraded version of ORB-SLAM2, and we identify its limitations when observing large moving objects within the scene. DynaSLAM is an open-source dynamic vision-based SLAM system.

The evaluation metrics employed include Root Mean Square Error (RMSE), Mean, and Standard Deviation (SD).

For trajectory prediction evaluation, we adopt a 3-second prediction horizon based on the findings of Control-Aware Prediction Objectives (CAPOs) [24] for effective collision avoidance in autonomous driving scenarios. Given the limited prior research exploring the integration of object trajectory prediction within SLAM systems, we establish baselines by predicting trajectories for a 3-second interval and fitting a 3D polynomial curve. In particular, we fit the curve to 4 evenly spaced points on the previous trajectory (with 5 frames in between) and determine the endpoints d_{se} distance away from the start points. The Average Displacement Error (ADE) and Final Displacement Error (FDE) are calculated on both the KITTI and simulated datasets.

For trajectory evaluation, we employ QD-3DT [7] as our baseline and calculate the RMSE of the trajectories. Additionally, leveraging trajectory prediction, we assess the improvement in Precision (P), Recall, and F1-score for moving object detection when considering trajectory consistency.

Implement details. The threshold τ_θ for the head direction vector $\mathbf{V}_{\mathbf{h},i}$ is set to $\pi/2$, and $\mathbf{V}_{\mathbf{h},i} = 0.9\hat{\mathbf{a}}_i + 0.1\hat{\mathbf{d}}_i$. For the prediction vector $\mathbf{V}_{\mathbf{p},i}$, we set $\mathbf{V}_{\mathbf{h},i} = -2\hat{\mathbf{v}}_{i,01} + \hat{\mathbf{v}}_{i,12}$. For the distance threshold, we set $\tau_\alpha = \pi/18$. The Kalman Filter's weights are set to $w_k = 0.5$ and $w_n = 0.5$. Moreover, we set $\tau_{KF} = 7$ in Algorithm 1 and $h = 20$. In Equation 5, we assign $r_{pedestrians} = 80$ as the average weight for pedestrians and $r_{car} = 1900$ for cars.



(a) CARLA 01 (b) CARLA 03 (c) CARLA 05 (d) CARLA EZL

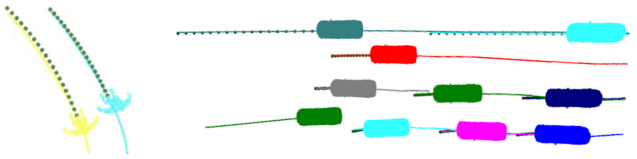
Fig. 5: Examples of camera pose on simulator sequences. The units of x and y axis are meters.

TABLE I: Camera pose of SLAM systems (RMSE, Mean and SD).

	Ours			DynaSLAM			ORBSLAM2			ORBSLAM3 (with IMU)		
	RMSE ↓	Mean ↓	SD ↓	RMSE ↓	Mean ↓	SD ↓	RMSE ↓	Mean ↓	SD ↓	RMSE ↓	Mean ↓	SD ↓
CARLA01	**4.791**	**4.204**	**2.656**	5.041	4.387	2.762	14.679	11.059	7.299	11.959	9.054	5.469
CARLA02	5.062	4.902	2.491	5.365	5.174	2.655	6.391	7.254	2.179	**4.445**	**2.426**	**2.188**
CARLA03	5.804	4.902	3.410	5.972	5.093	3.687	9.606	8.973	3.431	**4.488**	**2.933**	**2.845**
CARLA04	3.145	3.059	0.728	**2.985**	**2.902**	**0.648**	4.573	4.248	1.691	3.223	3.127	0.746
CARLA05	**1.390**	**1.048**	**0.916**	1.573	1.217	1.000	1.972	1.380	1.412	1.396	1.057	0.940
CARLA EZL	3.663	**1.805**	**0.741**	3.689	2.033	0.794	3.844	1.947	0.906	**3.625**	2.634	1.029
GAZEBO L	**0.477**	**0.442**	**0.177**	0.480	**0.442**	**0.177**	0.738	0.646	0.343	0.482	0.444	0.181

TABLE II: Result of trajectory Prediction (ADE, FDE).

	Baseline		Ours - DM		Ours - HD		Ours - SV		Ours	
	ADE ↓	FDE ↓	ADE ↓	FDE ↓	ADE ↓	FDE ↓	ADE ↓	FDE ↓	ADE ↓	FDE ↓
MOT 1-18 AVG	0.415	0.726	0.332	0.548	0.476	0.763	0.437	0.843	**0.316**	**0.531**
MOT19	0.312	0.586	**0.186**	0.423	0.326	0.615	0.276	0.699	**0.186**	**0.391**
MOT20	0.732	1.231	0.435	0.741	1.279	1.429	0.853	2.177	**0.416**	**0.701**
GAZEBO L	0.403	0.702	**0.276**	0.558	1.593	1.769	0.486	1.208	**0.276**	**0.533**
CARLA01	1.336	1.536	1.065	1.201	2.601	2.968	1.576	1.968	**1.036**	**1.167**
CARLA02	1.045	1.264	0.864	**1.063**	2.068	2.631	1.136	1.769	**0.831**	1.103
CARLA03	0.372	0.603	0.196	0.438	0.486	0.772	0.301	0.723	**0.190**	**0.429**
CARLA04	0.698	0.862	0.456	0.791	1.367	1.582	1.191	1.946	**0.436**	**0.787**
CARLA05	0.599	0.841	0.321	0.601	0.936	1.280	0.983	1.096	**0.316**	**0.579**
CARLA EZL	0.863	1.027	0.749	0.923	1.563	1.747	1.639	2.001	**0.721**	**0.893**
CARLA AVG	0.829	1.022	**0.610**	0.861	1.555	1.890	1.131	1.585	0.614	**0.849**



(a) (b)

Fig. 6: Results of trajectory prediction. The green dashes are the *predicted* trajectories, while the ground truth is the lines with the same colors as the *real* trajectories.

B. Camera Pose

Fig. 5 provides visual comparisons of camera pose estimation results among our proposed method (Ours), ORB-SLAM2 [16], DynaSLAM [2], ORB-SLAM3 [17], and the ground truth (GT). Under the same computational resource, our method runs in **9.7fps** while DynaSLAM runs in 0.2fps.

The evaluation results are also shown in Table I. It is worth noting that the MOT KITTI dataset mentioned in Section IV-A lacks ground truth odometry information, making comparisons feasible only on our simulator dataset. In particular, ORBSLAM3 performs well with *an additional IMU sensor* in most sequences, except for CARLA01. In CARLA01, both our method and DynaSLAM outperform other baselines, mainly because of a large number of moving objects at the very beginning. ORBSLAM3, lacking the ability to recover clues from external moving objects, struggles in this scenario despite the IMU sensor. Among the exclusively vision-based methods, our approach consistently achieves top or near-top precision and effectively handles distractions caused by abnormal moving objects in the scene.

C. Trajectory Prediction

The trajectories predicted by our method are displayed in Fig. 6, where the green dashed lines represent the predicted positions/trajectories, and the solid lines with the same color indicate the ground truth trajectories of the objects. As illustrated in Fig. 6a, the predicted future paths of pedestrians closely match the ground truth curve. Additionally, the predicted vehicle trajectories in Fig. 6b also follow the original driving path. These results demonstrate the robustness of our method and its ability to accurately predict both curved and straight trajectories for objects moving at different speeds.

Our trajectory prediction approach, as seen in Table II, significantly outperforms the baseline method in terms of both

TABLE III: Result of trajectory evaluation with RMSE, Precision, Recall and F1 value.

	QD-3DT				Ours + QD-3DT				Increase			
	RMSE ↓	P ↑	Recall↑	F1 ↑	RMSE ↓	P ↑	Recall↑	F1 ↑	RMSE ↑	P ↑	Recall↑	F1 ↑
KITTI MOT1-18	0.916	0.331	**0.926**	0.487	**0.789**	**0.362**	0.920	**0.519**	**13.86%**	**0.94%**	-0.65%	**6.57%**
KITTI MOT19	0.785	0.179	**0.966**	0.302	**0.642**	**0.181**	**0.996**	**0.305**	**18.21%**	**1.11%**	0%	**0.98%**
KITTI MOT20	0.582	0.578	**0.992**	0.729	**0.505**	**0.584**	**0.992**	**0.735**	**13.23%**	**1.3%**	0%	**0.81%**
CARLA 01-05	1.086	0.389	**0.881**	0.538	**0.863**	**0.425**	**0.881**	**0.573**	**20.53%**	**8.47%**	0%	**6.11%**
CARLA EZL	0.842	0.351	**0.882**	0.502	**0.669**	**0.384**	0.876	**0.504**	**20.55%**	**8.59%**	-0.68%	**0.39%**
GAZEBO L	1.105	0.161	**0.926**	0.274	**0.935**	**0.180**	**0.926**	**0.301**	**15.38%**	**10.56%**	0%	**8.97%**

Fig. 7: Comparison between original QD-3DT (top) and our proposed method (bottom). The person in the red shirt is the same person in a different time step.

Average Displacement Error (ADE) and Final Displacement Error (FDE). Due to the page limitation, we only show the average results of KITTI MOT 1-18. The baseline sets the distance d_{se} equal to the curve length C_p, similar to Ours-DM. The effectiveness of our head direction vector-based trajectory modelling is evident when comparing Ours-DM and Ours. In contrast, compared to Ours-HD, our method demonstrates greater robustness, especially in cases where the head direction from QD-3DT [7] may be less reliable. In fact, our velocity smoothing procedure proves necessary, as instantaneous velocity often contains noise. In a similar CARLA simulation environment, the original CAPOs [24] paper reports ADE of predictions over 2m, while we have improved it by an order of magnitude. Notably, our trajectory prediction relies on geometry and requires no training, taking approximately 2.5^{-7} seconds per frame. These findings underscore the strength and potential of our method for trajectory prediction in various real-world applications.

D. Object Trajectory

As illustrated in the pedestrian paths of Fig. 8 (MOT 19), we directly compare QD-3DT [7] to the proposed system and the ground truth. The results highlight the high effectiveness of our method in preserving accuracy while removing noise and smoothing trajectories. We assess the accuracy of object trajectory estimation by measuring the Root Mean Square Error (RMSE) of individual trajectories in each frame. Our results, as shown in Table III, demonstrate a significant improvement in trajectory estimation accuracy, achieving at least a 13% increase in accuracy compared to the QD-3DT.

In addition to trajectory analysis, we employ a consistency check. Table III indicates that the object detection based on QD-3DT exhibits low precision but high recall values, as many initial detections are incorrect and include duplicates. Our proposed method enhances the accuracy of detections by eliminating these duplicates, as visualized in Fig. 7. For instance, QD-3DT detects the same pedestrian three times, each marked with a 2D bounding box in a different color, while our approach consistently detects the same pedestrian just once. Moreover, our proposed method does not substantially affect recall values, as it neither introduces new detections nor rectifies erroneous ones. Although there is a slight reduction in recall for the CARLA EZL sequence, it is noteworthy that such errors are exceedingly rare, occurring only once across all datasets.

V. Conclusion

In this work, we present a vision-based perception system integrated with a SLAM system capable of tracking moving objects, optimizing trajectories, and predicting future paths. We evaluate the proposed system on both real-world and simulated datasets, demonstrating its superior accuracy compared to other methods. Additionally, our system showcases high performance in optimizing and predicting trajectories, while also enhancing object tracking capabilities.

References

[1] H. Cho, Y.-W. Seo, B. V. Kumar, and R. R. Rajkumar, "A multi-sensor fusion system for moving object detection and tracking in urban driving environments," in 2014 IEEE International Conference on Robotics and Automation (ICRA), pp. 1836–1843, IEEE, 2014.

[2] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, "Dynaslam: Tracking, mapping, and inpainting in dynamic scenes," IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4076–4083, 2018.

[3] B. Xu, W. Li, D. Tzoumanikas, M. Bloesch, A. Davison, and S. Leutenegger, "Mid-fusion: Octree-based object-level multi-instance dynamic slam," in 2019 International Conference on Robotics and Automation (ICRA), pp. 5231–5237, IEEE, 2019.

[4] J. Zhang, M. Henein, R. Mahony, and V. Ila, "Vdo-slam: a visual dynamic object-aware slam system," arXiv preprint arXiv:2005.11052, 2020.

[5] S. Yang and S. Scherer, "Cubeslam: Monocular 3-d object slam," IEEE Transactions on Robotics, vol. 35, no. 4, pp. 925–938, 2019.

[6] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, "Dynaslam ii: Tightly-coupled multi-object tracking and slam," IEEE robotics and automation letters, vol. 6, no. 3, pp. 5191–5198, 2021.

[7] 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.

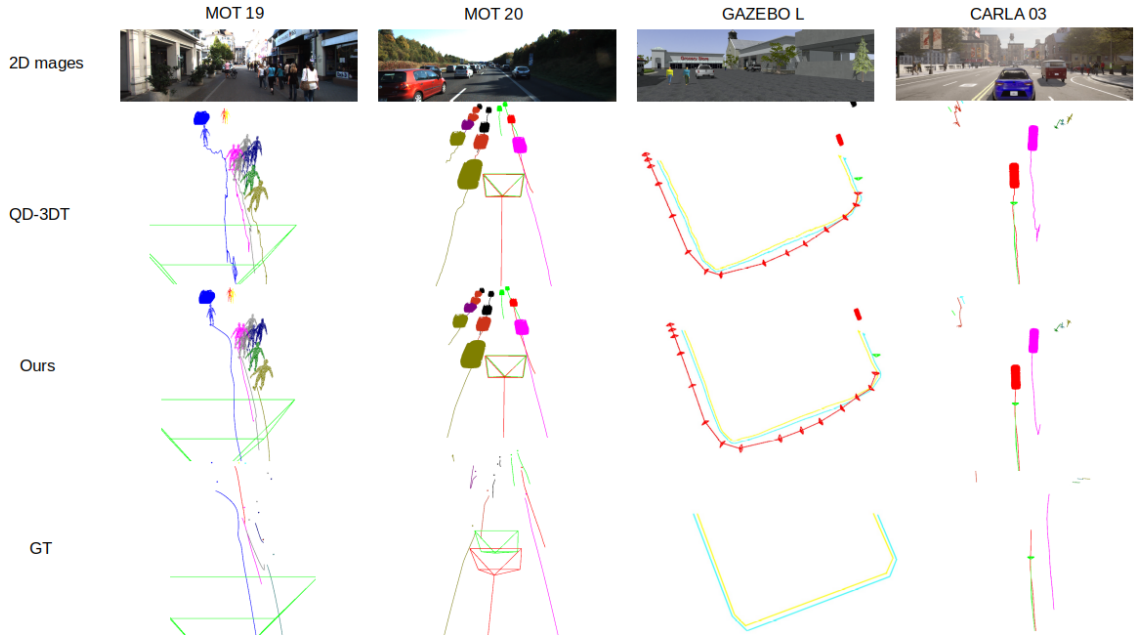


Fig. 8: Trajectories for MOT19, MOT20, GAZEBO L and CARLA03 sequences.

[8] N. Marinello, M. Proesmans, and L. Van Gool, "Triplettrack: 3d object tracking using triplet embeddings and lstm," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4500–4510, 2022.

[9] T. Fischer, Y.-H. Yang, S. Kumar, M. Sun, and F. Yu, "Cc-3dt: Panoramic 3d object tracking via cross-camera fusion," arXiv preprint arXiv:2212.01247, 2022.

[10] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12085–12094, 2019.

[11] Y. Peng, G. Zhang, J. Shi, B. Xu, and L. Zheng, "Srai-lstm: A social relation attention-based interaction-aware lstm for human trajectory prediction," Neurocomputing, vol. 490, pp. 258–268, 2022.

[12] H. Zhou, D. Ren, H. Xia, M. Fan, X. Yang, and H. Huang, "Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction," Neurocomputing, vol. 445, pp. 298–308, 2021.

[13] I. Bae, J.-H. Park, and H.-G. Jeon, "Non-probability sampling network for stochastic human trajectory prediction," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6477–6487, 2022.

[14] X. Li, X. Ying, and M. C. Chuah, "Grip: Graph-based interaction-aware trajectory prediction," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 3960–3966, IEEE, 2019.

[15] X. Li, X. Ying, and M. C. Chuah, "Grip++: Enhanced graph-based interaction-aware trajectory prediction for autonomous driving," arXiv preprint arXiv:1907.07792, 2019.

[16] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," IEEE transactions on robotics, vol. 31, no. 5, pp. 1147–1163, 2015.

[17] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," IEEE transactions on robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[18] L. Xiao, J. Wang, X. Qiu, Z. Rong, and X. Zou, "Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment," Robotics and Autonomous Systems, vol. 117, pp. 1–16, 2019.

[19] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, "Ds-slam: A semantic visual slam towards dynamic environments," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1168–1174, IEEE, 2018.

[20] R. A. Wadud and W. Sun, "Dyob-slam: Dynamic object tracking slam system," arXiv preprint arXiv:2211.01941, 2022.

[21] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," arXiv preprint arXiv:2107.08430, 2021.

[22] P. Tokmakov, J. Li, W. Burgard, and A. Gaidon, "Learning to track with object permanence," in Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10860–10869, 2021.

[23] J. Cao, X. Weng, R. Khirodkar, J. Pang, and K. Kitani, "Observation-centric sort: Rethinking sort for robust multi-object tracking," arXiv preprint arXiv:2203.14360, 2022.

[24] R. McAllister, B. Wulfe, J. Mercat, L. Ellis, S. Levine, and A. Gaidon, "Control-aware prediction objectives for autonomous driving," in 2022 International Conference on Robotics and Automation (ICRA), pp. 01–08, IEEE, 2022.

[25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in Conference on robot learning, pp. 1–16, PMLR, 2017.

[26] M. Vo, S. G. Narasimhan, and Y. Sheikh, "Spatiotemporal bundle adjustment for dynamic 3d reconstruction," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1710–1718, 2016.

[27] J. J. Moré, "The levenberg-marquardt algorithm: implementation and theory," in Numerical analysis: proceedings of the biennial Conference held at Dundee, June 28–July 1, 1977, pp. 105–116, Springer, 2006.

[28] G. Welch, G. Bishop, et al., "An introduction to the kalman filter," 1995.

[29] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3, pp. 2149–2154, IEEE, 2004.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., "Ros: an open-source robot operating system," in ICRA workshop on open source software, vol. 3, p. 5, Kobe, Japan, 2009.

[31] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[32] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[33] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixe, and B. Leibe, "Hota: A higher order metric for evaluating multi-object tracking," International Journal of Computer Vision (IJCV), 2020.