4. Adversarial Tracking

In this chapter, we present adversarial attacks against object tracking systems. Unlike object detection systems that only make predictions based on the current input image from the camera, object tracking models combine prior predictions with current inputs to generate trajectories of moving objects, making them more robust to adversarial attacks.

4.1 Introduction

As introduced in Chapter 2, deep learning models are widely used in autonomous driving for perception tasks, such as object-tracking, to generate trajectories of surrounding vehicles. In the context of autonomous driving, we lack access to gradients, which makes white-box attacks nearly impossible. Additionally, the difficulty in obtaining intermediate query results within the internal data path makes it impractical for black-box attacks to query the model.

However, this does not necessarily mean that current autonomous driving systems are robust against adversarial attacks. Many object tracking systems still rely on pre-trained object detection models (as outlined in Table 4.1), making them susceptible to transferable perturbations generated using well-known object detection models. Consequently, adversarial attacks targeting object detection could also impact object tracking systems.

Rank	Model	Popularity
1	SSD	605
2	YOLO-v3	251
3	Faster R-CNN	236
4	R-CNN	178
5	YOLO-v2	164
6	RetinaNet	75
7	Yolo-v1	52

Table 4.1: Most popular object detection models on GitHub (Wang et al., 2021a)

4.1.1 Object Tracking

The first tracking system used radar and sonar to track objects for military applications, and the algorithm consists of four parts: data association, state update, state prediction, and track management. Later, the employment of deep learning yields significant performance increases for visual tracking tasks.

Before the widespread adoption of deep learning, visual tracking problems were commonly addressed using low-level features and statistical learning techniques. These included methods such as the Joint Probabilistic Data Association Filter (JPDAF), Multi-Hypothesis Tracking (MHT), and Random Finite Sets (RFS). These techniques were often combined with nonlinear filters such as Bayesian Estimation, the Kalman Filter, the Particle Filter, and the Gaussian Sum Filter to establish correspondences between detections and tracklets (Vo et al., 2015).

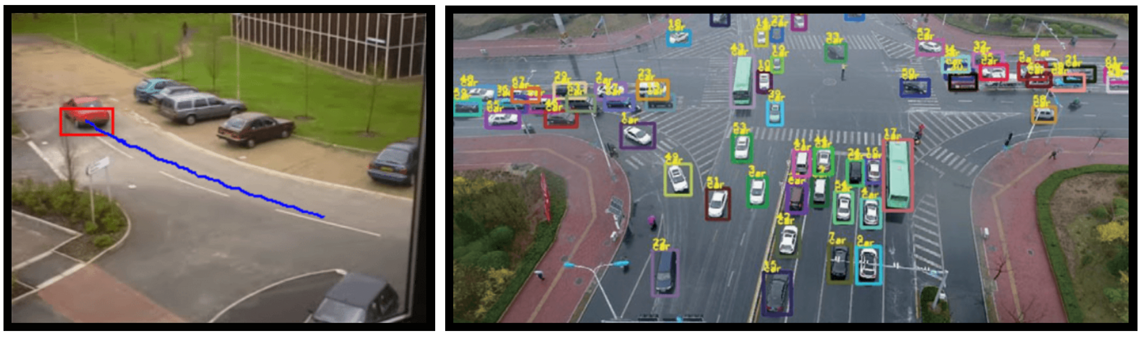
As research in deep learning advances, traditional low-level features and statistical learning techniques for object tracking are being replaced by deep neural networks. Deep neural networks are widely utilized for both Single-Object Tracking (SOT) and Multi-Object Tracking (MOT) applications (Krebs, Duraisamy and Flohr, 2017).

Single-Object Tracking (SOT) focuses on tracking one specified target within a sequence of input image frames. The target can be specified using a template image or be chosen from the first frame (Soleimanitaleb and Keyvanrad, 2022) (see Fig. 4.1a). Similar to face recognition, the image of a target person may not appear in the training set, so the SOT model must be capable of tracking any given object with only one image. Besides, the class label may not be available in the training set, requiring the model to handle a variety of potential targets.

SOT commonly uses a Siamese network to learn a similarity function that differentiates objects and then utilizes statistical methods to update and maintain a single tracklet effectively (He et al., 2018; Guo et al., 2017b; Bertinetto et al., 2016; Dong and Shen, 2018; Zhang and Peng, 2019).

Multi-Object Tracking (MOT), in contrast, involves tracking multiple objects simultaneously and maintaining multiple tracklets. In MOT, the number of classes is typically predefined, and the tracking model assigns a unique ID to each detected object. The primary challenge in MOT is differentiating distinct objects and associating the same object across consecutive frames to maintain consistent tracklets (see Fig. 4.1b).

Recent research interests are shifting from SOT to MOT, especially in the context of autonomous driving, which demands a comprehensive perception of surrounding vehicles. Table 4.2 highlights the top-performing methods on the KITTI multi-object tracking leaderboard for autonomous driving (Geiger, Lenz and Urtasun, 2012).



(a) SOT (b) MOT

Figure 4.1: The difference between Single-Object Tracking (SOT) (Jansari, Parmar and Saha, 2013) and Multi-Object Tracking (MOT) (Liu et al., 2022b).

Method	Sensor	Type	Real-Time	HOTA
BiTrack (Not publications yet)	Lidar	-	N	82.69%
LEGO (Zhang et al., 2023b)	Lidar	-	Y	80.75%
CollabMOT (Ninh and Kim, 2024)	Stereo	TBD	N	80.02%
RAM (Tokmakov et al., 2022)	Mono	JDT	Y	79.53%
PermaTrack (Tokmakov et al., 2021)	Mono	JDT	Y	78.03%
OC-SORT (Cao et al., 2023)	**Mono**	**TBD**	**Y**	**76.54%**

Table 4.2: Top Multi-Object Tracking (MOT) methods on the KITTI leaderboard.

Object Tracking Sensors

Camera and Lidar are the two most popular sensors used in autonomous vehicles. On the KITTI leaderboard, the method that utilizes point clouds from a Velodyne laser scanner achieves the highest Higher Order Tracking Accuracy (HOTA) (Luiten et al., 2020). We introduce the HOTA evaluation metric in Sec. 4.3.

Monocular Tracker: Vision-only methods rely solely on input images from a single camera (Wu et al., 2021) (Hu et al., 2022) are accurate for 2D multi-object tracking. However, they cannot reliably predict 3D locations because depth information is absent from single-camera input.

Binocular Tracker: This approach combines 2D object detection results with depth information obtained from stereo vision to achieve 3D object detection and tracking. However, using a stereo camera setup requires extensive calibration and complex matching algorithms (Ninh and Kim, 2024).

Multi-Modality Tracker: Accurately estimating vehicles' pose in 3D space is crucial for autonomous driving. By combining 2D detection results with 3D Lidar data, this approach achieves the highest tracking accuracy on the KITTI leaderboard, though at the cost of using more expensive 3D laser scanners than CMOS cameras.

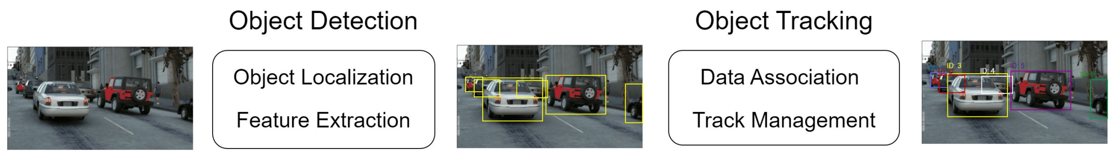
Object Tracking Frameworks

Tracking-By-Detection (TBD) and Joint Detection and Tracking (JDT) are the two most popular frameworks for object tracking applications.

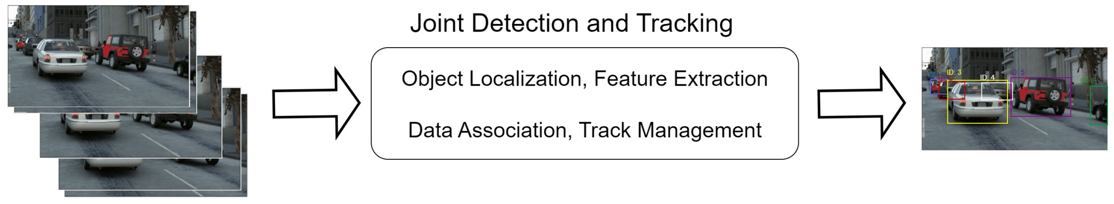
Tracking-By-Detection (TBD) is a modular object tracking framework that consists of object localization, feature extraction, data association, and track management (see Fig. 4.2a). Object detection models handle object localization and feature extraction to generate bounding boxes, while traditional methods such as the Hungarian Algorithm manage data association, and the Kalman Filter oversees track management. The overall accuracy of the TBD framework depends significantly on the precision of the detector, such as SiamRPN++ (Li et al., 2019) for SOT and YOLO, Faster R-CNN, and SSD for MOT applications (Sun et al., 2020).

Joint Detection and Tracking (JDT): As more accurate and efficient deep neural networks, such as Vision Transformer (Dosovitskiy et al., 2020), are being developed, there is also a growing trend of employing end-to-end tracking models (Pal et al., 2021; Guo et al., 2022). PermaTrack is a JDT method that takes multiple frames of images as input and outputs the position and velocity of each vehicle (see Fig. 4.2b). As can be seen from Tab. 4.2, JDT methods achieve slightly higher accuracy than TBD methods.

To explore whether traditional statistical methods, such as the Kalman Filter, can help mitigate the impact of adversarial attacks, we focus on modular TBD methods (highlighted in Tab 4.2) instead of end-to-end JDT models. Additionally, due to the lower cost and ease of setup of monocular cameras compared to 3D Lidar and stereo cameras, our primary focus is on vision-only monocular 2D object detection.



(a) Tracking-By-Detection (Modular System)


Joint Detection and Tracking
Object Localization, Feature Extraction
Data Association, Track Management

(b) Joint Detection and Tracking (End-to-End Model)

Figure 4.2: The difference between Tracking-By-Detection (TBD) and Joint Detection and Tracking (JDT).

4.1.2 Adversarial Attacks

Adversarial attacks have progressed from image classification to object detection and then object tracking models. Notably, Joint Detection and Tracking (JDT) methods that utilize end-to-end deep learning models, such as GOTURN (Wiyatno and Xu, 2019) and FairMOT (Lin et al., 2021b), are vulnerable to adversarial attack. For Tracking-By-Detection (TBD) methods, adversaries focus on vulnerabilities in various components within the tracking system, including the object detector, the association method, and the tracklet management module.

Attacking Object Detectors: The accuracy of TBD methods heavily relies on the object detector's ability to produce precise bounding boxes of target objects. If the object detector produces bounding boxes at incorrect positions or does not predict bounding boxes at all, the subsequent modules will struggle to generate precise tracklets.

Table 4.3 presents studies focusing on attacking object detectors, and most work attacks SiameseRPN-based trackers (Wu et al., 2019). Siamese neural networks employ triplet loss or contrastive loss to train a similarity function to discern objects, such as distinguishing facial features in face recognition systems.

In 2019, the first adversarial attack against MOT models attacked a TBD method that used YOLOv3 as the object detector (Jia et al., 2020). Moreover, YOLOX (Ge et al., 2021) and CenterNet (Duan et al., 2019) stand out as two widely-used object detectors for autonomous driving applications, yet they both exhibit vulnerability to adversarial attacks (Zhou et al., 2023; Pang et al., 2024).

Attacking Association Methods: Another crucial factor for MOT system is to assign a unique ID to each detected object. It is possible to perturb the association module so that the same object may be given different IDs in different frames, thus breaking the continuity of tracklets. Liu et al. introduced the Blind Attack, which assigns the background label to the target object, and the Blur Attack, which assigns other labels to the target object (Liu et al., 2022c).

Attacking Tracklets: Tracklets, commonly managed by the Kalman Filter, store the position and velocity of each vehicle. By perturbing objects' speed and direction, incorrect states are predicted for each object, enabling the switching of tracklets between adjacent objects. For instance, Lin et al. perturbed the appearance and distance similarity of two adjacent pedestrians, causing FairMOT and ByteTrack models to fail to track the targets in the subsequent frames (Lin et al., 2021a).

While it is acknowledged that MOT systems utilizing 3D Lidar Point Clouds are also susceptible to adversarial attacks (Cheng et al., 2021; Cheng et al., 2022; Wang et al., 2022d), our research focuses on attacks targeting vision-only MOT systems.

Year	Method	Tracking	Model	Attack Type	Patch / Filter
2019	Hijacking Attack (Jia et al., 2020)	MOT	Yolov3	Whitebox	Digital Patch
2019	EOT Attack (Wiyatno and Xu, 2019)	SOT	GOTURN	Whitebox	Physical Patch
2020	SPARK Attack (Guo et al., 2020)	SOT	SiamRPN++	Whitebox	Digital Filter
2020	One-Shot Attack (Chen et al., 2020b)	SOT	Siamese Tracker	Whitebox	Digital Filter
2020	FAN Attack (Liang et al., 2020)	SOT	Siamese Network	Whitebox	Digital Filter
2020	Hijacking Tracker (Yan et al., 2020c)	SOT	SiamRPN	Whitebox	Digital Filter
2020	Cool-Shrink Attack (Yan et al., 2020a)	SOT	SiamRPN++	Whitebox	Digital Filter
2021	Tracklet-Switch Attack (Lin et al., 2021a)	MOT	FairMOT & ByteTrack	Whitebox	Digital Filter
2021	MTD Attack (Ding et al., 2021)	SOT	SiamRPN++	Whitebox	Physical Patch
2021	ABA Attack (Guo et al., 2021)	SOT	SiamRPN++	Whitebox	Digital Blurry
2021	IoU Attack (Jia et al., 2021)	SOT	SiamRPN++	Blackbox	Digital Filter
2022	Few-Shot Attack (Li et al., 2022b)	SOT	SiamFC++	Whitebox	Physical Patch
2022	Ad2 Attack (Fu et al., 2022)	SOT	SiamAPN	Whitebox	Digital Filter
2022	Shuffle Attack (Liu et al., 2022c)	SOT	Yolov3 + Kalman	Whitebox	Digital Filter
2022	Diminishing (Suttapak, Zhang and Zhang, 2022)	SOT	SiamRPN++	Whitebox	Digital Filter
2023	F&F Attack (Zhou et al., 2023)	MOT	YOLOX & CenterNet	Whitebox	Digital Filter
2024	Blinding & Blurring Attack (Pang et al., 2024)	MOT	YOLOX & CenterNet	Whitebox	Digital Filter

Table 4.3: Adversarial Attacks against Object Tracking Models.

4.2 Methodology

4.2.1 Problem Formulation

For autonomous driving applications, there is no access to future frames, making it an online tracking problem, which means the model can only use past and present information to make hypothesis for the present. As introduced in Sec. 1.4, we aim to answer the question: Can traditional methods mitigate the impact of adversarial attacks? To explore this, we choose the TBD framework as our target model, which consists of object detection, state estimation, and data association, incorporating both deep neural networks and traditional algorithms (see Fig. 4.3).

Object Detection: We use similar notations as in Chapter 3 to formulate the detection problem. To differentiate the model's hypotheses from ground truths, we use $\{h_1, h_2, h_3, ..., h_m\}$ to represent hypotheses and $\{o_1, o_2, o_3, ..., o_n\}$ as the ground truths. Given an input image x at time t, the object detection model outputs a hypothesis of N candidate bounding boxes $\{h_1^t, h_2^t, h_3^t, ..., h_N^t\}$. Each hypothesis $h_i^t = (b_i, c_i, p_i)$ includes a bounding box $b_i = (b_i^x, b_i^y, b_i^w, b_i^h)$, the confidence value $c_i \in [0, 1]$ and the softmax probability vector, $p_i = (p_i^1, p_i^2, ..., K_i)$ for K classes.

State Estimation: To assign a consistent ID to the same object across frames, the Kalman Filter is commonly used to track the state of each object. The state of each object, modeled using a linear constant velocity model, is stored as a vector:

$$\boldsymbol{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{4.1}$$

where (u, v) are the coordinates in the image, s represents the scale, r is the aspect ratio of the object, and $[\dot{u}, \dot{v}, \dot{s}]$ are the corresponding velocities. The state vector is updated if a predicted bounding box is associated with the object.

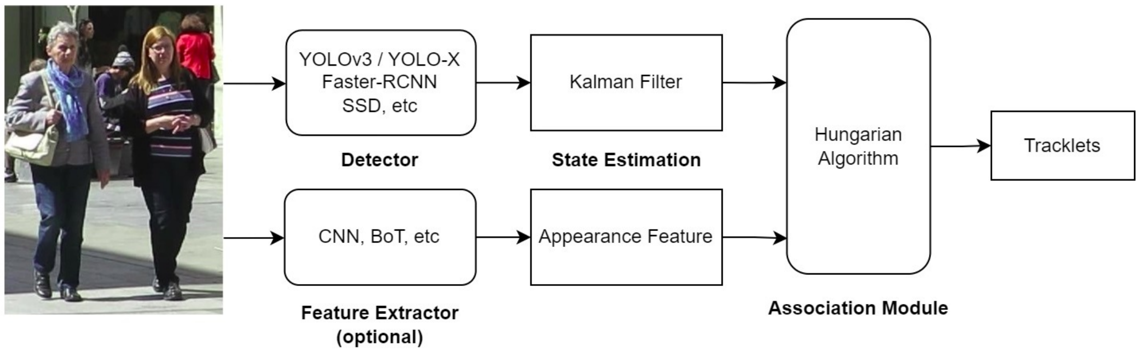


Figure 4.3: Common modules in online tracking methods that adopt the TBD framework.

Feature Extractor (Optional): The state of each object may not be sufficient to differentiate objects when occlusion occurs. Some detection frameworks use another deep neural network, such as a Siamese Network (Li et al., 2019), to learn a similarity function that produces a feature vector for each object and can be used to differentiate objects. However, using another deep neural network requires more computational power to achieve real-time performance, and thus, it is not utilized in all online tracking models.

Data Association: Besides object detection, the tracking model assigns unique IDs to each object. The association module combines the state vector and feature vector to measure the similarity between the predicted objects and ground truths. For instance, the Mahalanobis distance is often used to measure the similarity between state vectors. Then, similar objects can be associated with the same IDs by solving a linear assignment problem using the Hungarian Algorithm.

4.2.2 Adversarial Tracking

Although object detection models are vulnerable to adversarial attacks, the impact of such attacks can be mitigated if false positive bounding boxes generated by adversarial perturbations are inconsistent with the linear velocity model maintained by the Kalman Filter or cannot be associated with previous frames. In other words, the state estimation and data association methods that use traditional algorithms can filter out inconsistent bounding boxes crafted by adversarial attacks.

The accuracy of a tracking method can be easily improved with a better detection model, even if the Kalman Filter (KF) and Hungarian Algorithm remain unchanged (see Tab. 4.4). For instance, Strong-SORT outperforms Deep-SORT partly because it adopts a more accurate detector and feature extractor that were not available when Deep-SORT was proposed. Therefore, it is crucial to use the same detection model and feature extractor across all methods for a fair comparison of state estimation and data association methods. Consequently, we use a less accurate detection model, YOLOv3, along with the same attack methods described in Chapter 3, to test the robustness of state estimation and assignment modules.

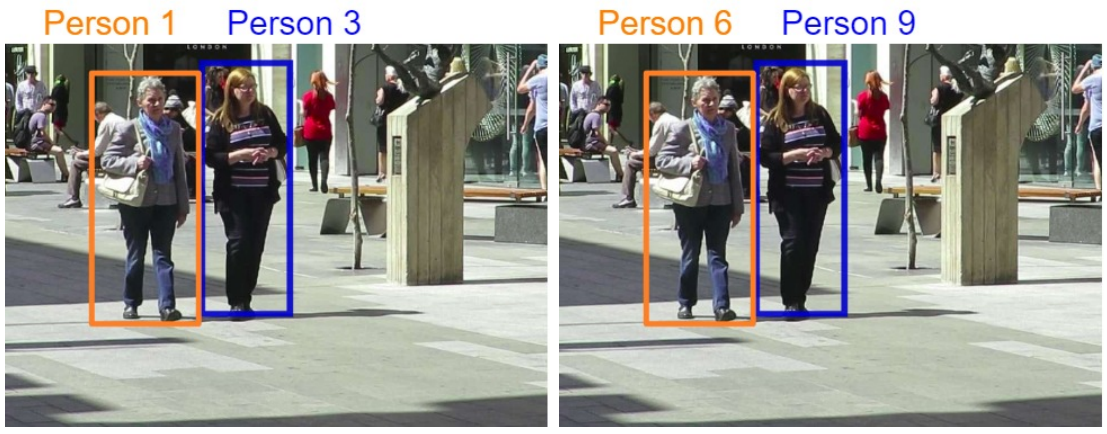
Method	Detector	State Estimation
SORT (Bewley et al., 2016)	FRCNN	KF
Deep-SORT (Wojke, Bewley and Paulus, 2017)	FRCNN	KF
Strong-SORT (Du et al., 2023)	YOLOX	NSA KF
OC-SORT (Cao et al., 2023)	YOLOX	KF & Re-Update

Table 4.4: A comparison of different real-time online trackers.

4.3 Evaluation metrics

Object tracking models should distinguish between different objects and consistently assign the same ID to the same object across frames. However, the IDs assigned by the tracking model do not need to match the ground truth IDs exactly. For example, a person labeled as ”person 6” in the ground truth might be assigned ”person 1” by the tracking model (see Fog. 4.4). As long as the ID remains consistent across frames, the specific numbers do not need to match.

Consequently, to evaluate an object tracking model, we first need to find a one-to-one match that maps the IDs assigned by the tracking model to the IDs in the ground truths. The matching should minimize the total distance between the detected objects and their corresponding ground truth objects in all frame (Ciaparrone et al., 2020). Finding the match can be formulated as a minimum weight assignment problem, which can be solved using the Hungarian algorithm.



(a) Hypotheses (b) Ground Truths

Figure 4.4: The same person may be assigned a different ID by the tracking model compared to the ground truth.

After finding the match, the tracking results can be evaluated using the following widely adopted evaluation metrics MT, ML (Wu and Nevatia, 2006), MOTA (Bernardin and Stiefelhagen, 2008), IDF1 (Ristani et al., 2016), and HOTA (Luiten et al., 2021).

4.3.1 Classical Metrics (MT, ML)

Mostly Tracked (MT) and Mostly Lost (ML) are the two most popular evaluation metrics proposed in (Wu and Nevatia, 2006) and are widely reported in MOTChallenge (Dendorfer et al., 2020), KITTI (Geiger, Lenz and Urtasun, 2012) and nuScenes (Caesar et al., 2020) dataset.

For the video frame at time t, the object tracking model assigns a unique ID j to a detected object and produces a hypothesis h_j^t. After solving the linear assignment problem using the Hungarian Algorithm, if the hypothesis h_j^t is mapped to o_i^t (see 4.5), the ground truth object is considered tracked at time step t.

Ground truth objects with the same ID at different time steps are combined into a tracklet. A tracklet is classified as mostly tracked if correctly tracked in at least 80% of the frames. Conversely, a tracklet is classified as mostly lost if correctly tracked in at most 20% of the frames. The evaluation metric MT counts the number of mostly tracked tracklets, while ML counts the number of mostly lost tracklets (Wu and Nevatia, 2006).

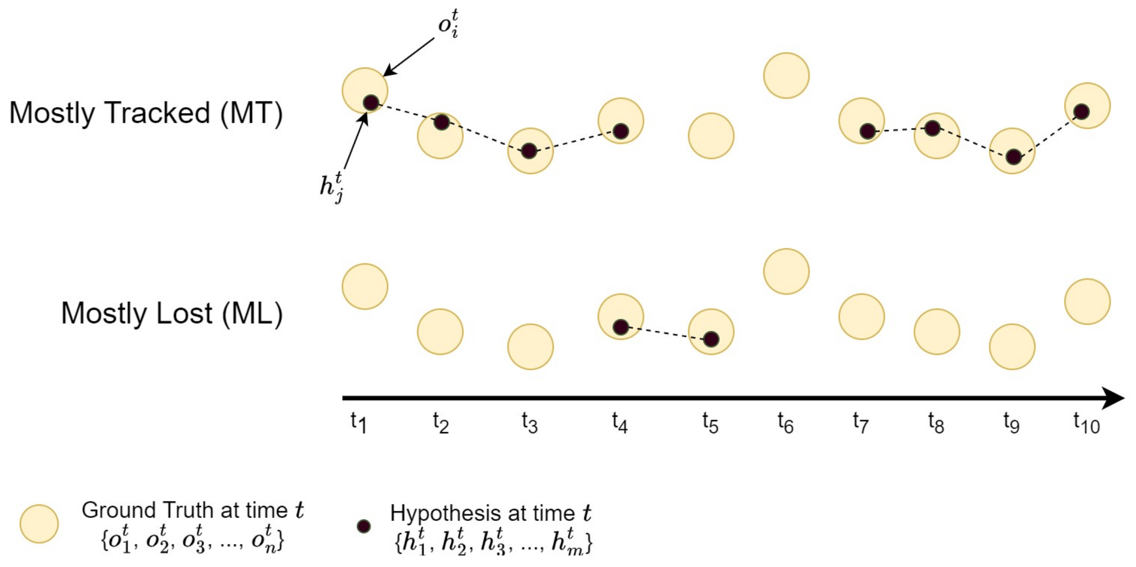

o_i^t
Mostly Tracked (MT)
h_j^t
Mostly Lost (ML)
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10}
Ground Truth at time t $\{o_1^t, o_2^t, o_3^t, ..., o_n^t\}$
Hypothesis at time t $\{h_1^t, h_2^t, h_3^t, ..., h_m^t\}$

Figure 4.5: Examples of mostly tracked and mostly lost tracklets.

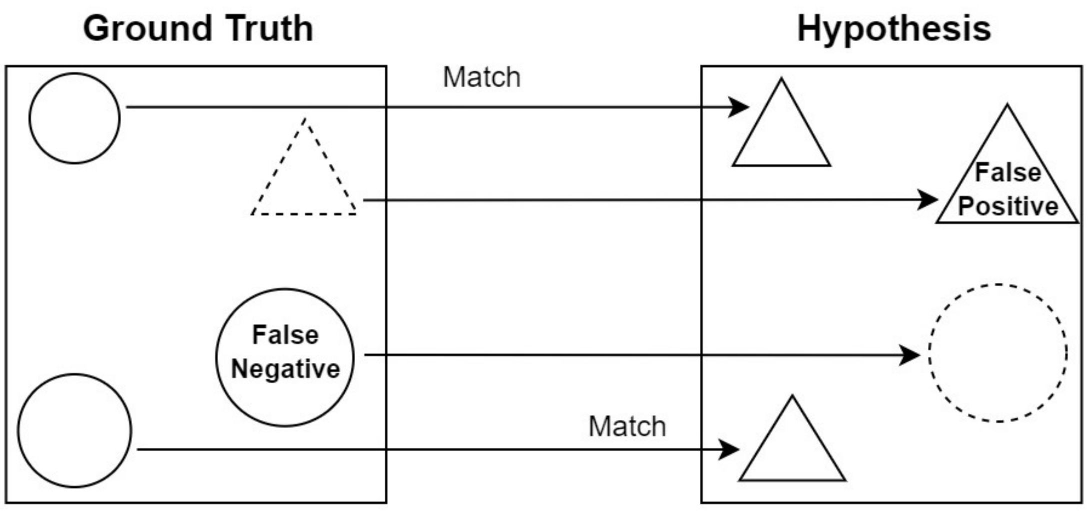
4.3.2 CLEAR MOT Metrics (MOTA)

Although MT and ML count the total number of good (mostly tracked) tracklets and bad (mostly lost) tracklets, they do not provide information about the bottleneck of the tracker or how to improve it. These metrics do not indicate whether improvements should be focused on the detector or the association algorithm.

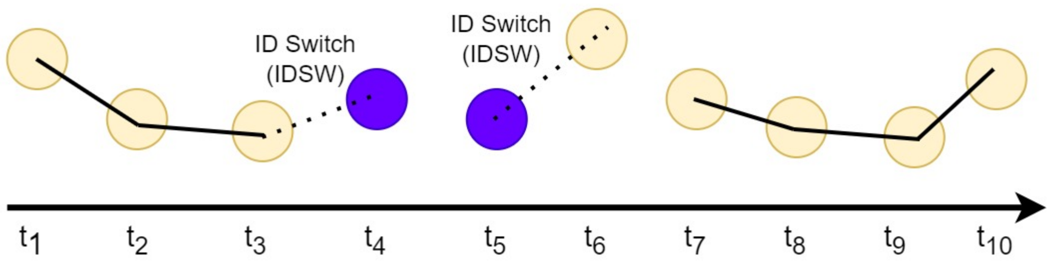
To address this issue, the CLEAR MOT Metrics use the number of False Positives and False Negatives to measure the accuracy of the detector, and ID Switches to measure the quality of the association algorithm. An overall metric, MOTA, is then calculated to provide a comprehensive evaluation.

$$MOTA = 1 - \frac{\sum_t FN_t + \sum_t FP_t + \sum_t IDSW_t}{\sum_t GT_t} \in (-\infty, 1] \tag{4.2}$$

As illustrated in Fig. 4.6a, the tracking result is accurate if a ground truth is matched with a hypothesis. A false negative (FN) is counted when a ground truth bounding box cannot be associated with a hypothesis (the left side), and a false positive (FP) is counted when a hypothesis cannot be associated with a ground truth bounding box (the right side). Besides, the ID switch happens when the same ground truth object is assigned different IDs in consecutive frames (see Fig. 4.6b).



(a) False Negative and False Positive.



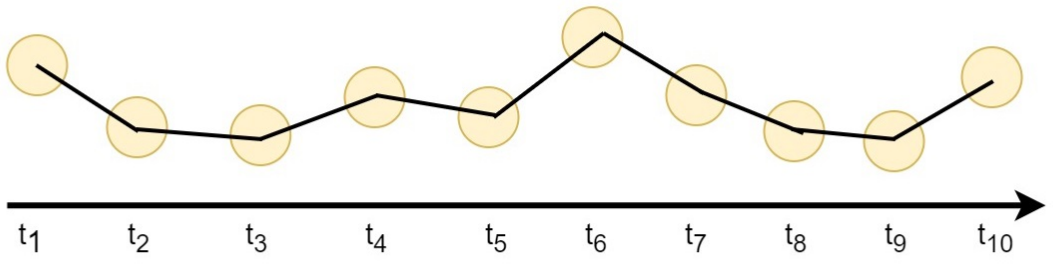
(b) ID Switch (Different colors represent different IDs assigned by the tracking model).

Figure 4.6: The False Negatives (FN), False Positives (FP) and ID Switches (IDSW) for Multiple Object Tracking Accuracy (MOTA).

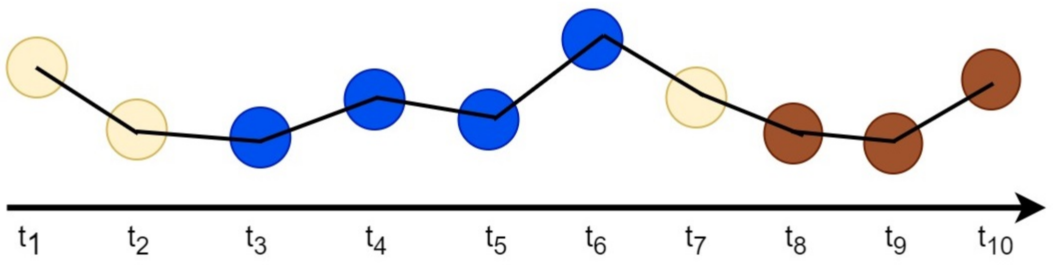
4.3.3 ID Scores (IDF1)

The MOTA metric provides insight into detector performance by considering false negatives (FN) and false positives (FP) and uses ID switches (IDSW) to evaluate the association algorithm. However, it is biased towards detection performance. For instance, both Tracklet 1 (4.7b) and Tracklet 2 (4.7c) achieve a MOTA of 70%, as all ground truth objects in Fig. 4.7a are matched with hypotheses, indicating an effective detector. Despite this, Tracklet 2 is preferable in some scenarios because it maintains consistent IDs for the same object over a longer period than Tracklet 1.

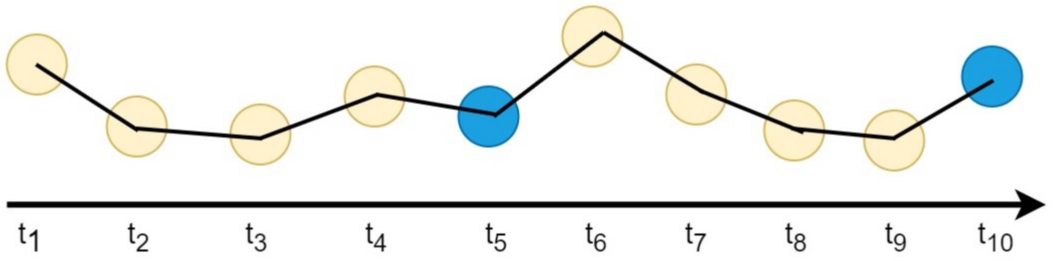
Another evaluation metric is needed to measure how long the tracker correctly identifies targets. The IDF1 score addresses this issue by first matching each ground-truth object to a specific tracklet with a consistent ID. A hypothesis is considered a match only when both the position and IDs align. Consequently, Tracklet 2, which maintains consistent IDs for the same object over a longer period, results in fewer FPs and FNs than Tracklet 1, indicating it is a better tracker.



(a) The ground truth tracklet.



(b) Tracklet 1: **MOTA = 70%** (FN=0, FP=0, IDSW=3), **IDF1 = 30%** (IDTP = 3, IDFP = 7, IDFN = 7)



(c) Tracklet 2: **MOTA = 70%** ((FN=0, FP=0, IDSW=3), **IDF1 = 80%** (IDTP = 8, IDFP = 2, IDFN = 2)

Figure 4.7: A comparison of MOTA and IDF1. Different colors indicate different IDs assigned by the tracking model.

Formally, the ID score uses a Bipartite graph $G = (V_T, V_C, E)$ to connect nodes in ground truths and hypothesis at time t. V_T includes regular nodes τ (ground truths) and false positives f^+_γ, while V_C includes regular nodes γ (hypotheses) and false negatives f^-_τ (see Fig. 4.8). An edge represents a true positive ID (IDTP) if both connected nodes are regular nodes, a false positive ID (IDFP) if the connection is (f^+_γ) and a false negative ID (IDFN) if the connection is (τ, f^-_τ).

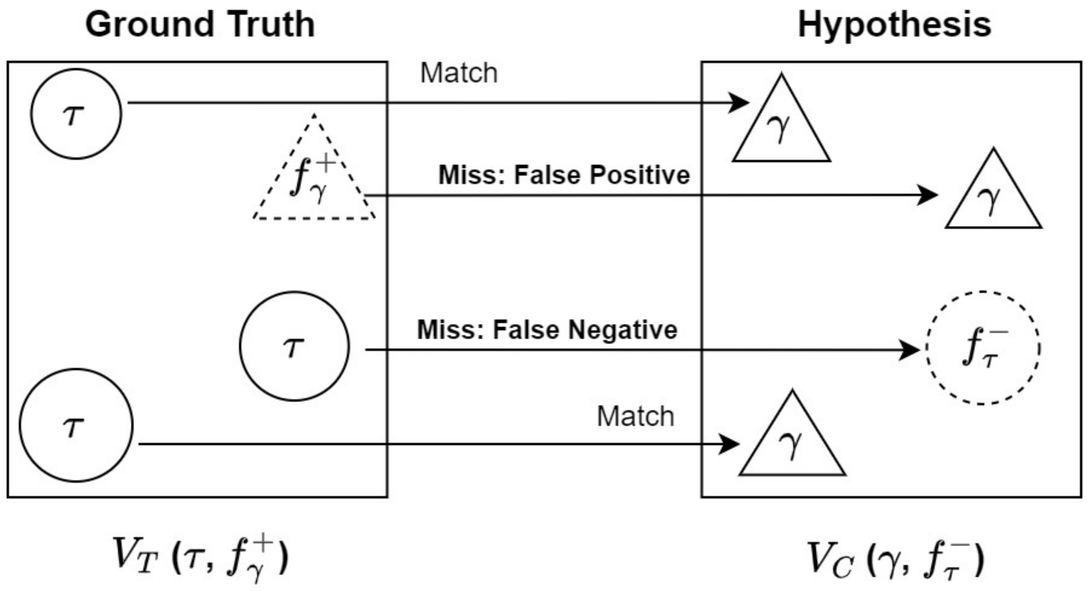


Figure 4.8: The IDF1 score defines IDFP and IDFN using a Bipartite Graph.

Lastly, the IDF1 score can be computed from Identification Precision (IDP) and Identification Recall (IDR) using the following equations:

$$\mathrm{IDP} = \frac{\mathrm{IDTP}}{\mathrm{IDTP} + \mathrm{IDFP}} \quad \mathrm{IDR} = \frac{\mathrm{IDTP}}{\mathrm{IDTP} + \mathrm{IDFN}} \tag{4.3}$$

$$\mathrm{IDF}_1 = \frac{2\mathrm{IDTP}}{2\mathrm{IDTP} + \mathrm{IDFP} + \mathrm{IDFN}} \tag{4.4}$$

4.3.4 Higher Order Tracking Accuracy (HOTA)

The MOTA is biased towards detection, while IDF1 is biased towards association. The HOTA metric tries to find a balance between the detection accuracy (DetA) and association accuracy (AssA) using the following equation:

$$\mathrm{HOTA}_\alpha = \sqrt{\mathrm{DetA}_\alpha \cdot \mathrm{AssA}_\alpha} = \sqrt{\frac{\sum_{c\in \mathrm{TP}_\alpha} \text{Ass-IoU}(c)}{|\mathrm{TP}_\alpha| + |FN_\alpha| + |\mathrm{FP}_\alpha|}} \tag{4.5}$$

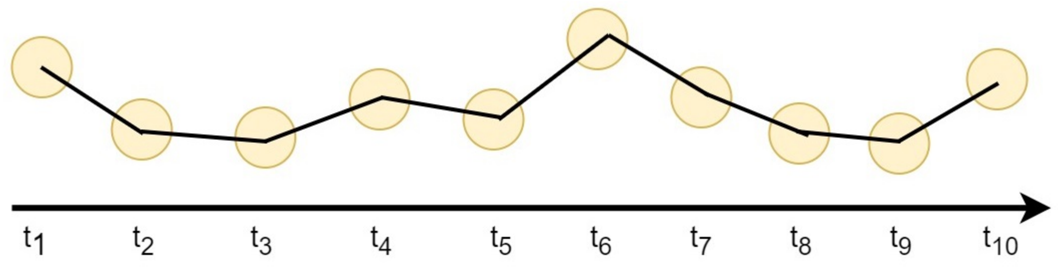
First, the HOTA metric runs the Hungarian algorithm to determine a one-to-one matching between hypotheses and ground truths with IoU $> \alpha$. Then, detection accuracy (DetA) is calculated for hypotheses that match the ground truth detections without considering IDs. Association accuracy (AssA), on the other hand, is calculated for hypotheses that match both detection and IDs (Luiten et al., 2021).

$$\mathrm{DetA}_\alpha = \frac{|\mathrm{TP}_\alpha|}{|\mathrm{TP}_\alpha| + |\mathrm{FN}_\alpha| + |\mathrm{FP}_\alpha|} \tag{4.6}$$

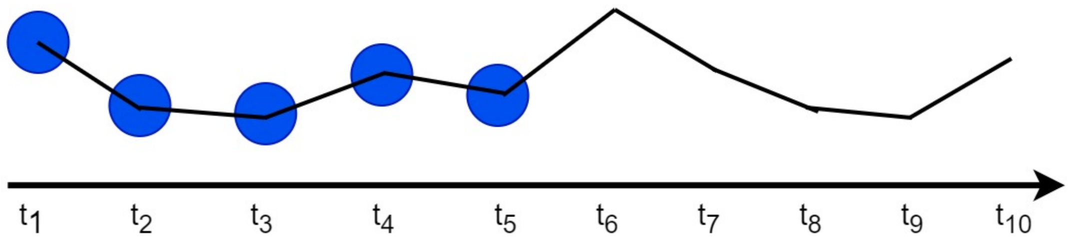
$$\mathrm{AssA}_\alpha = \frac{1}{|\mathrm{TP}_\alpha|} \sum_{c \in \mathrm{TP}_\alpha} \text{Ass-IoU}(c) \tag{4.7}$$

$$\text{Ass-IoU(c)} = |\mathrm{TP} \sum_{c \in \mathrm{TP}} \frac{|\mathrm{TPA}|}{|\mathrm{TPA}| + |\mathrm{FNA}| + |\mathrm{FPA}|} \tag{4.8}$$

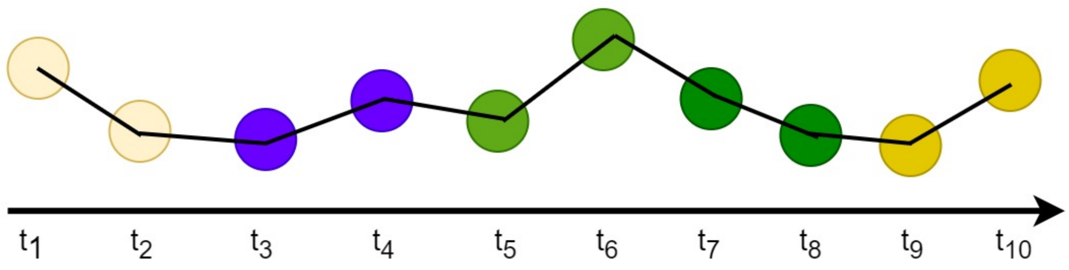
For example, Tracklet 1 in Fig. 4.9b only detects the first 5 objects out of 10, while Tracklet 2 in Fig. 4.9c detects all 10 objects but with less consistency in ID assignment. As a result, MOTA prefers Tracklet 2 due to its better detection accuracy, while IDF1 favors Tracklet 1 for its superior association accuracy. On the other hand, HOTA assigns a score of 50% to both tracklets, attempting to balance detection and association performance.



(a) The ground truth tracklet.



(b) Tracklet 1: DetA = 50%, MOTA = 50%, **HOTA = 50%**, IDF1 = 67%, AssA = 50%.



(c) Tracklet 2: DetA = 100%, MOTA = 97%, **HOTA = 50%**, IDF1 = 25%, AssA = 25%.

Figure 4.9: A comparison of MOTA, IDF1, and HOTA. Different colors indicate different IDs assigned by the tracking model.

4.4 Experimental Results

In this section, we apply the white-box PCB attack introduced in Chapter 3 to the SORT tracking framework (Bewley et al., 2016), a widely used TBD framework recognized for its real-time performance. We also attack several subsequent tracking methods built upon SORT, including Deep SORT (Wojke, Bewley and Paulus, 2017), Strong SORT (Du et al., 2023), and OC SORT (Cao et al., 2023).

To investigate whether traditional algorithms, such as the Kalman Filter and the Hungarian Algorithm, can alleviate the effects of adversarial attacks, we first set up a baseline by evaluating SORT, Deep SORT, Strong SORT, and OC SORT combined with different detectors (YOLOv3, YOLOv4, YOLOX) on the KITTI and CARLA datasets without applying adversarial attacks (see Fig. 4.10). The experimental results are presented in Table 4.5 and Table 4.6 [1], and we have the following findings:

1. Even with a perfect detector (using ground truth as detections), the tracking methods could not achieve 100% HOTA because the same object may be assigned different IDs when occlusion occurs.

2. However, if the accuracy of the detector decreases, the the tracking performance significantly deteriorates. Since YOLOX and YOLOv4 achieve comparable performance, we focus on YOLOv4 and YOLOv3 in our further experiments.

3. The tracking module can be improved without using neural networks. Although both OC SORT and Strong SORT improve the tracking accuracy, OC SORT does not use an additional neural network for ID association, demonstrating that substantial improvements can be achieved with traditional algorithms.

(a) KITTI Autonomous Vehicle (b) CARLA Simulator

Figure 4.10: Sample images from the KITTI and CARLA dataset.

[1] Our source code and dataset are available on GitHub: `https://github.com/wuhanstudio/adversarial-tracking`

Detector	Tracker	HOTA ↑	MOTA ↑	IDF1 ↑	MT ↑	ML ↓
Ground Truth	OC SORT	**93.84**	**93.70**	**96.41**	473	**11**
Ground Truth	Strong SORT	93.72	93.05	96.38	**506**	14
Ground Truth	Deep SORT	85.43	90.43	94.83	480	14
Ground Truth	SORT	84.95	92.80	95.12	467	**11**
YOLOX	OC SORT	53.78	52.24	**68.29**	143	113
YOLOX	Strong SORT	**54.30**	**54.31**	66.93	**187**	**89**
YOLOX	Deep SORT	52.82	50.83	68.11	173	106
YOLOX	SORT	51.32	54.14	66.96	163	105
YOLOv4	OC SORT	53.38	56.76	68.97	160	81
YOLOv4	Strong SORT	**55.17**	60.03	**69.32**	**220**	**65**
YOLOv4	Deep SORT	51.99	**64.95**	67.45	190	81
YOLOv4	SORT	50.63	59.25	66.33	188	72
YOLOv3	OC SORT	40.60	43.98	55.94	89	130
YOLOv3	Strong SORT	**43.25**	**47.64**	57.47	**122**	**96**
YOLOv3	Deep SORT	41.69	45.55	**57.65**	112	111
YOLOv3	SORT	39.27	46.83	53.72	108	113

Table 4.5: A comparison of different real-time online trackers (KITTI dataset).

Detector	Tracker	HOTA ↑	MOTA ↑	IDF1 ↑	MT ↑	ML ↓
Ground Truth	OC SORT	**92.93**	**95.03**	**93.55**	173	**9**
Ground Truth	Strong SORT	91.17	94.20	91.79	**181**	**9**
Ground Truth	Deep SORT	82.41	91.21	90.16	165	10
Ground Truth	SORT	82.11	93.89	88.98	171	**9**
YOLOX	OC SORT	**60.40**	64.89	**77.10**	103	30
YOLOX	Strong SORT	59.60	63.08	73.64	**121**	**26**
YOLOX	Deep SORT	56.97	59.59	73.61	101	28
YOLOX	SORT	57.01	**64.91**	72.96	118	27
YOLOv4	OC SORT	57.99	63.25	**75.78**	82	36
YOLOv4	Strong SORT	**59.49**	**65.36**	74.81	**107**	**28**
YOLOv4	Deep SORT	56.65	60.43	74.07	89	31
YOLOv4	SORT	56.31	65.27	73.37	103	**28**
YOLOv3	OC SORT	55.90	61.10	**74.31**	86	34
YOLOv3	Strong SORT	**56.19**	61.43	71.43	**109**	**27**
YOLOv3	Deep SORT	53.80	57.24	70.92	91	**27**
YOLOv3	SORT	53.45	**62.63**	69.30	107	28

Table 4.6: A comparison of different real-time online trackers (CARLA dataset).

4.4.1 Image-Specific PCB Attack

The image-specific attack was tested with hyperparameters of attack strength $\epsilon = 8$ and learning rate decay $k = 0.99$. We updated the perturbation frame-by-frame, iterating one step per frame for 300 consecutive frames on each of the 7 maps.

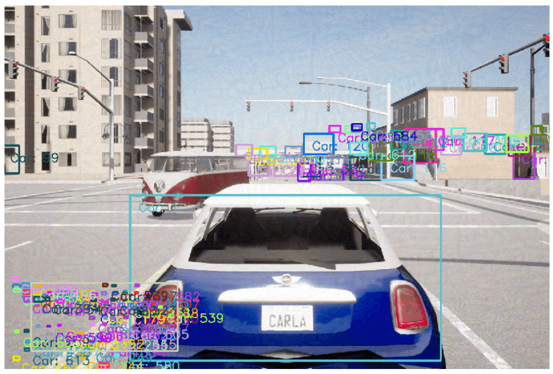
After applying adversarial perturbations, the tracking accuracy drops significantly for both YOLOv4 and YOLOv3 (see Table 4.7). Notably, the MOTAs are negative due to a large number of false positives. As shown in Fig. 4.11, the attack against tracking methods using YOLOv3 as detectors generates a large number of dense, small bounding boxes detected as cars, while YOLOv4 produces fewer false positives because the bounding boxes are larger. As a result, YOLOv4 is more robust than YOLOv3 against the image-specific attack.

The effectiveness of the image-specific attack can be further enhanced by increasing the number of iterations per frame, because in the first few frames, the occurrence of false positives is minimal and the tracking method remains largely unperturbed.

In conclusion, perturbing the object detection model alone proves to be sufficient for attacking tracking models.

Detector	Tracker	HOTA ↑	MOTA ↑	IDF1 ↑	MT ↑	ML ↓
YOLOv4	OC SORT	25.34	17.07	34.46	21	120
YOLOv4	Strong SORT	26.86	0.94	33.20	24	105
YOLOv4	Deep SORT	25.06	-6.30	32.95	23	104
YOLOv4	SORT	26.32	19.19	34.78	27	115
YOLOv3	OC SORT	17.03	-482.66	14.09	56	52
YOLOv3	Strong SORT	14.85	-772.60	9.95	65	42
YOLOv3	Deep SORT	15.32	-692.26	11.29	59	44
YOLOv3	SORT	15.80	-525.81	12.00	66	46

Table 4.7: Evaluation results of the Image-Specific PCB attack (CARLA dataset).


(a) YOLOv3

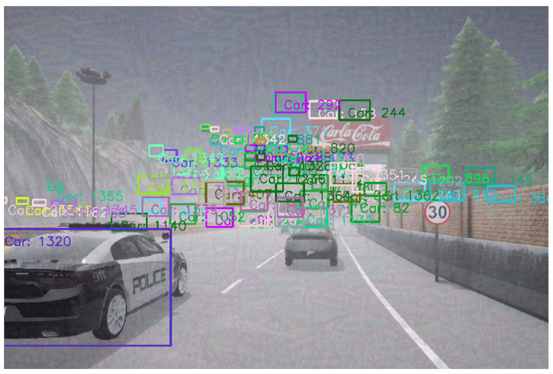

(b) YOLOv4

Figure 4.11: The image-specific attack against object tracking methods.

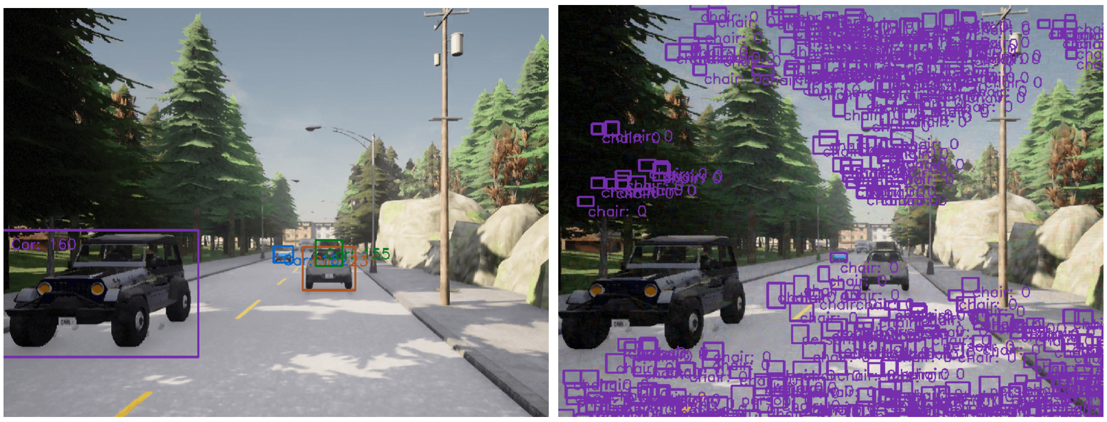
4.4.2 Image-Agnostic PCB Attack

The image-agnostic attack generates an Universal Adversarial Perturbation (UAP) to attack all video frames. We trained the UAP on Map 02 for 100 iterations, as Map 02 generalizes best across all maps according to Fig. 3.18 in Chapter 3. The training was conducted with a learning rate of $\alpha = 0.001$, a learning rate decay of $k = 0.98$, and an attack strength of $\epsilon = 8$.

For YOLOv4, the image-agnostic attack is slightly more effective than the image-specific attack because the UAP iterates for more steps than the image-specific attack (see Tab. 4.8). A significant difference in outcomes between image-specific and image-agnostic attacks is that false positives generated by UAPs are predominantly recognized as chairs and persons rather than cars (see Fig. 4.12b). However, autonomous driving systems primarily track cars, and thus most small, dense false positives bounding boxes (recognized as chairs, persons) are eliminated, rendering attacks against YOLOv3 less effective than YOLOv4. In other words, the effectiveness of UAPs can be improved by incorporating targeted attacks.

Detector	Tracker	HOTA ↑	MOTA ↑	IDF1 ↑	MT ↑	ML ↓
YOLOv4	OC SORT	22.05	14.26	33.07	10	136
YOLOv4	Strong SORT	24.11	16.95	34.37	9	122
YOLOv4	Deep SORT	24.87	10.98	33.54	7	131
YOLOv4	SORT	21.96	14.70	30.20	10	129
YOLOv3	OC SORT	43.45	45.45	62.43	39	61
YOLOv3	Strong SORT	43.01	47.18	57.42	61	49
YOLOv3	Deep SORT	43.34	43.28	60.17	49	52
YOLOv3	SORT	40.58	46.85	55.20	55	51

Table 4.8: Evaluation results of the Image-Agnostic PCB attack (CARLA dataset).



(a) Original image (b) YOLOv3

Figure 4.12: The image-agnostic attack against object tracking methods.

4.5 Summary

This chapter introduced the two most popular tracking frameworks, Tracking-By-Detection (TBD) and Joint Detection and Tracking (JDT), and evaluation metrics such as HOTA, MOTA, IDF1, MT, and ML. We evaluated four real-time online tracking methods—SORT, Deep SORT, Strong SORT, and OC SORT—on the KITTI and CARLA datasets. With the same detector, OC SORT achieved comparable tracking accuracy to Strong SORT but without additional neural networks for feature extraction, suggesting that tuning traditional methods (e.g., the Kalman Filter) could improve tracking accuracy without adding complexities to deep learning models that increase vulnerability to adversarial attacks. For TBD methods, tracking accuracy heavily relies on the precision of the detector. Our experimental results revealed that both image-specific and image-agnostic PCB attacks significantly decreased tracking accuracy, indicating that attacking the detector itself is sufficient to compromise the tracking methods.