TraSw: Tracklet-Switch Adversarial Attacks against Multi-Object Tracking

Delv Lin*, Qi Chen*, Chengyu Zhou*, and Kun He†

School of Computer Science and Technology, Huazhong University of Science and Technology, China
{derrylin,bloom24,hust_zcy,brooklet60}@hust.edu.cn

Abstract. Multi-Object Tracking (MOT) has achieved aggressive progress and derives many excellent deep learning models. However, the robustness of the trackers is rarely studied, and it is challenging to attack the MOT system since its mature association algorithms are designed to be robust against errors during the tracking. In this work, we analyze the vulnerability of popular pedestrian MOT trackers and propose a novel adversarial attack method called Tracklet-Switch (TraSw) against the complete tracking pipeline of MOT. TraSw can fool the advanced deep trackers (i.e., FairMOT and ByteTrack) to fail to track the targets in the subsequent frames by attacking very few frames. Experiments on the MOT-Challenge datasets (i.e., 2DMOT15, MOT17, and MOT20) show that TraSw can achieve an extraordinarily high success rate of over 95% by attacking only four frames on average. To our knowledge, this is the first work on the adversarial attack against pedestrian MOT trackers. The code is available at FairMOT-attack.

Keywords: Vulnerability-oriented attacks, adversarial attack, multi-object tracking

1 Introduction

Owing to the rapid development of Deep Neural Networks (DNNs), Visual Object Tracking (VOT) [31,4] has been dramatically boosted in recent years with a wide range of applications, such as autonomous driving [21], intelligent monitoring [33], human-computer interaction [3], etc. Researches on VOT fall into two categories, namely Single-Object Tracking (SOT) and Multi-Object Tracking (MOT). The goal of SOT is to track the same target given in the first frame in the video, while MOT aims to track all the targets of interest in an unsupervised setting and link objects in different frames to form their trajectories.

On the other hand, deep learning models of various computer vision tasks are found to be vulnerable to adversarial examples [28,11,17,9,38], which are crafted with imperceptible perturbations but lead the models to wrong predictions. Studying the adversarial attacks can improve understanding of DNNs, dig
Fig. 1: By attacking only two frames in the example video, we can exchange the 19th ID and the 24th ID completely. From frame 592, the exchange state can last until the end without noise. First row: the original video clips. Second row: the adversarial video clips. The blue boxes represent the 19th ID, the red boxes represent the 24th ID, and the gray boxes indicate the 16th ID not participating in the attack.

out their potential risks, and help design robust deep learning systems. Therefore, adversarial attacks have been extensively studied in many computer vision tasks, such as image classification [26,29,25,17,9], object detection [22,1,5,14], semantic segmentation [32,13,10], etc. Current adversarial attack methods of VOT is mainly concentrated in Single-Object Tracking [6,34,15], but there is little work on the more complicated Multi-Object Tracking. To our knowledge, there exists only one work that does MOT adversarial attack on autonomous driving [16].

A typical MOT tracker addresses the tracking in two steps [30,39,23]. Firstly, the tracker locates all objects in each frame. Then, according to the similarity metric, each detected object is associated with a trajectory. Considering most of the current MOT research focuses on pedestrian tracking and a large number of pedestrian tracking datasets have emerged in recent years, our work mainly focuses on attacking the pedestrian MOT trackers. We find that interleaving is an inevitable scene in pedestrian multi-object tracking, which is also an important indicator for evaluating the quality of the model.

In this work, we focus the adversarial attacks on pedestrian MOT trackers and investigate the robustness of MOT systems during interleaving of pedestrians. Attacking these trackers poses new challenges due to the memory characteristics of the MOT trackers. Specifically, the continuous tracking process allows the tracker to save the moving states of the trajectories for a long period (e.g., 30 frames). As a result, if an object disappears for a few frames, it actually has little impact on the final trajectory. Moreover, the corresponding trajectory will be removed only if the tracking object disappears for sufficient frames (e.g., 30 frames). Therefore, simply removing or tampering with the track is quite inefficient. Our experiments in Tab. 1 show that the existing attack methods on object detection that hide the tracking objects require 60% more frames to be
attacked on average. Yet, the average attack success rate is still low, especially in crowded scenarios (e.g., MOT20 [7]).

To address the above challenges, we propose a novel attack method called the **Tracklet-Switch** (TraSw). In a nutshell, our method learns an effective perturbation generator to make the tracker confuse intersecting trajectories by attacking as few as one frame, as illustrated in Fig. 1, yet the error state can transfer across frames until the last frame. In our method, we propose two novel losses, **PushPull** and **CenterLeaping**. **PushPull** works on the re-ID branch, while **CenterLeaping** works on the detection branch. They can work together or independently to accommodate different kinds of MOT trackers.

To better illustrate the effectiveness and efficiency of TraSw, we choose three pedestrian MOT-Challenge datasets, 2DMOT15, MOT17 and MOT20 [18,24,7], for evaluation, and compare TraSw with representative object detection adversarial attack methods [22,34,12,6], as well as the MOT adversarial attack method called **tracker hijacking** [16]. Experiments show that our method achieves significantly higher success rate with fewer attacked frames and smaller perturbations.

## 2 Related Work

### 2.1 Multi-Object Tracking

Multi-Object Tracking aims to locate and identify the targets of interest in the video and estimate their movements in the subsequent frames [2,23,30,36], such as pedestrians on the street, vehicles on the road, or animals on the ground. The mainstream MOT trackers break the tracking into two steps: 1) the detection stage, where objects are located in the images, 2) the association stage, where the objects are linked to the existing trajectories.

The MOT tracker uses tracklets to maintain the moving states of existing trajectories (e.g., position, velocity, and appearance). For a coming frame, the tracker compares the detected objects with tracklets to determine whether it belongs to an existing trajectory or is a new trajectory. The matching process between the tracklets and the detected objects is regarded as a bipartite matching problem based on the pair-wise similarity affinity matrix between all the tracklets and detected objects [16]. The commonly used similarity evaluation metric is Intersection-over-Union (IoU), which measures the spatial overlapping between two bounding boxes (bboxes). Some trackers also calculate the appearance affinity matrix to measure their appearance similarity.

After that, the matched tracklets will be updated (e.g., appearance state, motion state). The unmatched objects will be initialized as new trajectories, and the unmatched tracklets will be moved to the lost pool. If the tracking target of the lost tracklets reappears, the corresponding tracklets will be updated and put back into the tracking pool. Otherwise, if the missing tracklet reaches the maximum cache period (e.g., 30 frames), the unmatched tracklet will be completely removed. Even if the target reappears later on, it will be regarded as an entirely new trajectory.
2.2 Adversarial Attacks on Visual Object Tracking

**Adversarial Attacks on Single-Object Tracking.** The SOT tracker is informed of the tracking target in the first frame, and the goal is to predict the position and size of the tracking target in subsequent frames [20,19,37].

Since the tracking template is given in the first frame, if the tracking template is wrong, the tracker is unable to track the target. The early SOT attack method, one-shot attack [6], aims to disturb the template by adding perturbations on the template patch in the initial frame. To get the perturbation, the attacker needs to iterate over every frame in the video to compute the perturbation. However, most VOT trackers are real-time, so traversing all the frames is inefficient and not practical.

Subsequently, Yan et al. [34] propose a *cooling-shrinking* attack method to fool the SiamPRN-based trackers [19,20] by training a perturbation generator to craft noise to interfere the search regions so as to make the target invisible to the trackers. More recently, Jia et al. [15] present a decision-based black-box attack, named IoU attack, that aims to gradually decrease the IoU score between the bbox of the clean image and the bbox of the adversarial sample, which leads the prediction deviating from its original trajectory. These methods need to attack all the video frames. As the attacked target is determined in SOT, the number of attacks is relatively small, and valid attack samples can be obtained quickly. However, in MOT, the tracking quantity and tracking template vary along the frames. It is difficult to obtain effective adversarial examples by simply transplanting the SOT attack methods.

**Adversarial Attacks on Multi-Object Tracking.** Compared with SOT, there is little work on attacking MOT. There is only one MOT attack method proposed for vehicle tracking, called *tracker hijacking*, that crafts adversarial examples only on the object detection stage [16]. By fabricating the bbox toward the expected attacker-specified direction and erasing the original bbox, the attacked object deviates from its original trajectory. As a result, the successful adversaries can move an object out of (into) the view of an autonomous vehicle. The main differences between the *tracker hijacking* and our TraSw are as follows: 1) *tracker hijacking* focuses on attacking the vehicle tracking of autonomous driving while TraSw focuses on attacking the pedestrian tracking. Generally, the vehicle tracking scenarios of autonomous driving are mainly on highways, while the pedestrian tracking scenarios are much more complicated, including parks, shopping malls, roads, etc. And the tracking objects of pedestrian tracking are much more than that of vehicle tracking. 2) The attack targets of *tracker hijacking* and TraSw are different. *Tracker hijacking* attacks the detection based tracking tracker [23,16], that is, the attack is carried out in the object detection stage. Our method not only attacks on object detection but also on feature extraction, which can adapt to most of the current MOT trackers. For reference, we also provide comparison of *tracker hijacking* and TraSw in experiments.
3 The TraSw Adversarial Attack

In this section, we propose a novel method called the Tracklet-Switch (TraSw) adversarial attack against the pedestrian trackers. Our method aims to switch the tracklets of two intersecting trajectories. We choose FairMOT [36] as our main target model. In the following, we first introduce the overview of FairMOT, then provide the problem definition. In Sec. 3.3 and Sec. 3.4, we propose the PullPush loss and the CenterLeaping technique. In the end, we present the overall pipeline of TraSw in Sec. 3.5.

3.1 Overview of FairMOT

Achieving a good trade-off between accuracy and efficiency, FairMOT stands out among many trackers. As shown in Fig. 2, FairMOT consists of two homogeneous branches for object detection and feature extraction, and follows the standard online association algorithm.

Detection Branch. The anchor-free detection branch of FairMOT is built on CenterNet [8], consisting of the heatmap head, center-offset head and box-size head. Denote the ground-truth (GT) bbox of the i-th object in the t-th frame as $bbox^i_t = (x_{1}^i, y_{1}^i, x_{2}^i, y_{2}^i)$. The i-th object’s center $(c_x^{i,t}, c_y^{i,t})$ is computed by $c_x^{i,t} = \frac{x_{1}^i + x_{2}^i}{2}$ and $c_y^{i,t} = \frac{y_{1}^i + y_{2}^i}{2}$. The response location of the center on the heatmap can be obtained by dividing the stride (which is 4 in FairMOT) $([\frac{c_x^{i,t}}{4}, [\frac{c_y^{i,t}}{4}])$. The heatmap value indicates the probability of presence of an object centering at this point. The GT box size and center offset are computed by $(x_{2}^i - x_{1}^i, y_{2}^i - y_{1}^i)$ and $([\frac{c_x^{i,t} - c_x^{i,t}}{4}, [\frac{c_y^{i,t} - c_y^{i,t}}{4}])$.

Re-ID Branch. The re-ID branch generates the re-ID features of the objects. Denote the feature map as $feat^i_t \in \mathbb{R}^{512 \times H \times W}$. The re-ID feature $feat^i_t \in \mathbb{R}^{512}$ represents the feature vector of the i-th object, whose $L_2$ norm equals 1.

Association. FairMOT follows the association strategy in [30]. The tracker uses tracklet to describe the trajectory’s appearance state $a^t_i$ and motion state $m^t_i = (x^{i,t}, y^{i,t}, \gamma^{i,t}, h^{i,t}, x^{i,t}, y^{i,t}, \gamma^{i,t}, h^{i,t})$ in the t-th frame. The initial appearance state $a^0_i$ is initialized with the first observation’s appearance embedding...
feat_i^t of object i, and a_i^t is updated by:

\[ a_i^t = \alpha \cdot a_i^{t-1} + (1 - \alpha) \cdot feat_i^t, \]

where feat_i^t is the appearance embedding of the matched object in the t-th frame. The bbox information in the motion state m_i^t is updated by the predicted center (\(x_i^{t,i}, y_i^{t,i}\)), height h_i^{t,i} and aspect ratio \(\gamma_i^{t,i}\) in the t-th frame, and velocity information (\(\dot{x}_i^{t,i}, \dot{y}_i^{t,i}, \dot{\gamma}_i^{t,i}, \dot{h}_i^{t,i}\)) is updated by the Kalman filter.

For a coming frame, we compute the pair-wise similarity matrix between the observed objects in the t-th frame and the tracklets maintained by the tracklet pool \(TrP_{t-1}\). Then the association problem is solved by the Hungarian algorithm using the final cost matrix:

\[ d_t = \lambda \cdot d_{box}(K(m_{t-1}), box_t) + (1 - \lambda) \cdot d_{feat}(a_{t-1}, feat_t), \]

where box_t and feat_t denote the detected bboxes and features in the t-th frame, \(K(\cdot)\) represents the Kalman filter that uses motion state \(m_{t-1}\) to predict the expected positions of the trajectories in the t-th frame, and \(d_{box}(\cdot)\) stands for a certain measurement of the spatial distance, which is the Mahalanobis distance in FairMOT, and \(d_{feat}(\cdot)\) represents the cosine similarity.

### 3.2 Problem Definition

Let \(V = \{I_1, \ldots, I_t, \ldots, I_N\}\) denote the sequence frames of a video. Consider a scenario where the tracker detects two trajectories interleaved at frame t. Denote the two trajectories as \(T_i = \{O_i^1, \ldots, O_i^t, \ldots, O_i^{N}\}\) and \(T_j = \{O_j^1, \ldots, O_j^t, \ldots, O_j^{N}\}\). Their bboxes and features are \(B_k = \{box_{sk}^k, box_{tk}^k, \ldots, box_{ek}^k\}\) and \(F_k = \{feat_{sk}^k, feat_{tk}^k, \ldots, feat_{ek}^k\}\) where \(k \in \{i, j\}\), box_k^t \(\in \mathbb{R}^4\) and feat_k^t \(\in \mathbb{R}^{512}\). Then we define the adversarial video as:

\[ \hat{V} = \{I_1, \ldots, I_{t-1}, \hat{I}_t, \ldots, \hat{I}_{t+n-1}, I_{t+n}, \ldots, I_N\}, \]

where \(I, \hat{I}\) indicate the original frame and adversarial frame, respectively. For the attack trajectory \(T_i\), we call \(T_j\), that overlaps with \(T_i\) in the t-th frame, the screener trajectory. The adversarial video \(\hat{V}\) misleads the tracker to estimate trajectory \(i\) as \(\hat{T}_i = \{O_i^1, \ldots, O_i^{t-1}, O_i^t, \ldots, O_i^{t+n-1}, O_i^{t+n}, \ldots, O_i^{N}\}\). The goal is to attack the frame sequences to make the tracking of trajectory \(T_i\) change to that of \(T_j\) since the t-th frame.

### 3.3 Feature Attack with Push-Pull Loss

The tracker distinguishes the objects through a combination of motion and appearance similarity. When objects are close to each other, the tracker relies more on the re-ID features to distinguish the objects. Therefore, we can reform the re-ID features of the objects to make them similar to another tracklet. Inspired
Fig. 3: Key factors in TraSw. (a) The illustration of PushPull loss. The blue and red points represent different IDs to be attacked. The goal of the PushPull loss is to push the feature $feat_i^t$ of object $i$ in frame $t$ away from that of tracklet $i$, $a_{i-1}^t$, and to pull $feat_i^t$ to be close to another tracklet $a_{i-1}'$; and vice versa to $feat_i^t$. (b) The nine-block box ($B$). The surrounding features are also used to calculate the loss. (c) The center leaping for the detection branch. The dotted boxes indicate the original heat points, which need to be cooled down; and the solid boxes in the $t$-th frame indicate the adversarial heat points, which need to be heated up. In this way, the $cet(box_i')$ and $cet(box_j')$ leaps to $cet(box_i)$ and $cet(box_j)$ along the direction of $cet(K(m_{i-1}^t))$ and $cet(K(m_{j-1}^t))$.

by the triplet loss [27], we design the PushPull loss to push away the attack (screener) feature and pull the screener (attack) feature, as follows:

$$
\mathcal{L}_{pullpush}(a_{i-1}^t, a_{j-1}^t, feat_i^t, feat_j^t) = \sum_{k \in \{i, j\}} d_{feat}(a_{i-1}^t, feat_k^t) - d_{feat}(a_{j-1}^t, feat_k^t),
$$

(4)

where $d_{feat}()$ denotes the cosine similarity, $a_{i-1}^t$ and $a_{j-1}^t$ represent the appearance feature of the attack and screener tracklets, $feat_i^t$ and $feat_j^t$ represent the feature of the attack and screener objects, and $k$ represents the attack (screener) ID while $\tilde{k}$ represents the screener (attack) ID, whose object overlaps most with object $k$ (i.e., $i = j$ and $j = i$). The loss will make $feat_i^t$ dissimilar to tracklet $k$ and make $feat_k^t$ similar to tracklet $k$ (see Fig. 3a).

Specifically, in FairMOT, the object’s feature is extracted from the feature map $feat_i \in \mathbb{R}^{512 \times H \times W}$ according to the predicted object center $(x, y)$. Considering that the surrounding locations of the center may be activated, we calculate the appearance cost within a nine-block box location (as illustrated in Fig. 3b) for a more stable attack. So the final PushPull loss for FairMOT is as follows:

$$
\mathcal{L}_{pp} = \sum_{(dx, dy) \in B} \mathcal{L}_{pullpush}(a_{i-1}^t, a_{j-1}^t, feat_i^{t,(dx, dy)}, feat_j^{t,(dx, dy)}),
$$

(5)

where $B$ indicates a set of offsets in the nine-block box location, $feat_i^{t,(dx, dy)}$ and $feat_j^{t,(dx, dy)}$ represent the feature extracted around the center of the attack and screener object, respectively.
3.4 Detection Attack with Center Leaping

Attacking the features of intersecting trajectories can generally fool the tracker. But it may be insufficient when the spatial distance is too large to switch, and some trackers only use the detection result for the association.

In order to adjust the distance between objects and tracklets, we propose a novel and simple method, called the CenterLeaping. The optimization objective can be summarized as reducing the distance between the tracklet and the target object. As the bboxes are computed with discrete locations of heat points in the heatmap, we cannot directly optimize the loss of bboxes to make them close to each other. So the goal is achieved by reducing the distance between the centers of bboxes, as well as the differences of their sizes and offsets. Hence, the detection optimization function for the attack trajectory \( k \) can be expressed as follows:

\[
\min \sum_{k \in \{i, j\}} \sum_{m=1}^{\tilde{m}_k} d_{box}(K(m_{k-1}), box_k^i) = \min \sum_{k \in \{i, j\}} \left( d(\text{cet}(K(m_{k-1})), \text{cet}(box_k^i)) + d(\text{size}(K(m_{k-1})), \text{size}(box_k^i)) + d(\text{off}(K(m_{k-1})), \text{off}(box_k^i)) \right),
\]

where \( m_{k-1} \) represents the motion state of the attack or screener tracklet, \( box_k^i \) represents the bbox of the attack or screener object, \( d(\cdot) \) denotes the \( L_1 \) distance, \( \text{cet}(\cdot), \text{size}(\cdot) \) and \( \text{off}(\cdot) \) compute the bbox’s center, size and offset, respectively.

To make the object to be close to the center of the target tracklet, based on the focal loss of FairMOT, we design the CenterLeaping loss to let the center of the attack (screener) bbox to move close to the screener (attack) trajectory (see Fig. 3c):

\[
L_{cl} = \sum_{k \in \{i, j\}} \left( \sum_{(x,y) \in B_{c \rightarrow \tilde{c}_k}} -(1 - M_{x,y})^\gamma \log(M_{x,y}) + \sum_{(x,y) \in B_{c_k}} -M_{x,y}^\gamma \log(1 - M_{x,y}) \right),
\]

where \( M_{x,y} \) represents the value of heatmap at location \((x,y)\), \( c_k \) represents the center of the attack or screener object, and \( c_{\rightarrow \tilde{c}_k} \) represents the point in the direction from \( c_k \) to \( \text{cet}(K(m_{k-1})) \). During the optimization iteration, point \( c_{\rightarrow \tilde{c}_k} \) will leap to the next grid along the direction. As a result, the heatmap values around the original object centers get cooled down, while the points close to \( \text{cet}(K(m_{k-1})) \) are warmed up.

We also restrain the sizes and offsets of the objects by a regression loss:

\[
L_{reg} = L_{\text{size}} + L_{\text{offset}}
= \sum_{k \in \{i, j\}} L_{1}^{\text{smooth}}(\text{size}(K(m_{k-1})), \text{size}(box_k^i)) +
\sum_{k \in \{i, j\}} L_{1}^{\text{smooth}}(\text{off}(K(m_{k-1})), \text{off}(box_k^i)),
\]

Algorithm 1: The TraSw Attack

**Input:** Video image sequence $V = \{I_1, \ldots, I_N\}$; MOT Tracker($\cdot$); attack ID $ID_{att}$; attack IoU threshold $Thr_{IoU}$; start attack frame $Thr_{frame}$; maximum iteration $Thr_{iter}$

**Output:** Sequence of adversarial video $\hat{V}$; original tracklet pool $TrP_N$; adversarial tracklet pool $\hat{TrP}_N$

1. **Init** $\hat{V} \leftarrow \emptyset$; $TrP_0 \leftarrow \text{None}$; $\hat{TrP}_0 \leftarrow \text{None}$
2. for $t=1$ to $N$ do
   3. $TrP_t \leftarrow \text{Tracker}(I_t, TrP_{t-1})$;
   4. $\hat{TrP}_t \leftarrow \text{Tracker}(I_t, \hat{TrP}_{t-1})$;
   5. $\hat{I}_t \leftarrow I_t$; \hspace{1cm} $\triangleright$ initialize the outputs
   6. $\triangleright$ check if the attack tracklet has been existed for at least $Thr_{frame}$ frames and the object is tracked as $ID_{att}$
   7. if $\text{Exist}(ID_{att}, TrP_t) > Thr_{frame}$ and $\text{CheckFit}(TrP_t|ID_{att}, \hat{TrP}_t|ID_{att})$ then
      8. $\triangleright$ find the screener object that overlaps most with the attack object
      9. $ID_{scr} \leftarrow \text{FindMaxIoU}(TrP_t, ID_{att})$;
     10. if $\text{IoU}(TrP_t|ID_{att}, \hat{TrP}_t|ID_{scr}) > Thr_{IoU}$ then
        11. $\triangleright$ generate adversarial noise iteratively with Eq. 11
        12. $\text{noise} \leftarrow \text{NoiseGenerator}(ID_{att}, ID_{scr}, I_t, TrP_t, \hat{TrP}_{t-1}, \text{Tracker}(\cdot), Thr_{iter})$;
        13. $\hat{I}_t \leftarrow \text{Clip}_{[0,1]}(I_t + \text{noise})$; \hspace{1cm} $\triangleright$ clip the adversarial image to $[0,1]$
        14. $\hat{TrP}_t \leftarrow \text{Tracker}(\hat{I}_t, \hat{TrP}_{t-1})$; \hspace{1cm} $\triangleright$ update the adversarial tracklet
        15. $Thr_{IoU} \leftarrow 0$; \hspace{1cm} $\triangleright$ set $Thr_{IoU}$ to 0 if start to attack
   16. end
17. $\hat{V} \leftarrow \hat{I}_t$; \hspace{1cm} $\triangleright$ update the adversarial video
18. end

where $\mathcal{L}_{1}^{\text{smooth}}$ denotes the smooth $L_1$ loss:

$$\mathcal{L}_{1}^{\text{smooth}}(a, b) = \begin{cases} 0.5 \cdot |a - b|^2 & \text{if } |a - b| < 1, \\ |a - b| - 0.5 & \text{else.} \end{cases}$$ \hspace{1cm} (9)

3.5 Crafting the Adversarial Video

Summing up the above, we get the total loss for optimization:

$$\min_{\hat{V}} \text{Loss} = \min_{\hat{V}} \mathcal{L}_{pp} + \mathcal{L}_{cl} + \mathcal{L}_{req}. \hspace{1cm} (10)$$

Then we can calculate the gradient of the total loss with an $L_2$ regularization:

$$\hat{I}_0 = I, \hspace{0.5cm} \hat{I}_{i+1} = \text{Clip}_{[0,1]} \left( \hat{I}_i - \frac{\nabla I \text{Loss}(\hat{I}_i; \theta)}{\| \nabla I \text{Loss}(\hat{I}_i; \theta) \|_2} \right), \hspace{1cm} (11)$$
where \( \hat{I}_i \) denotes the adversarial image at the \( i \)-th iteration.

The algorithm overview of crafting the adversarial videos is shown in Algorithm 1. Firstly, we specify an attack trajectory \( ID_{att} \) in the original tracking video before the attack. For each coming frame, we initialize the tracklet pools, \( TrP_t \) and \( \hat{TrP}_t \), with the original frame, and initialize the adversarial frame \( \hat{I}_t \) as \( I_t \). Next, we conduct a double check to determine whether to attack the current frame: 1) check the object of trajectory \( ID_{att} \) has appeared for more than \( Thr_{frame} \) frames (10 in default) with \( Exist(\cdot) \). This is because the overall attack will make no sense if the attack starts from the first appearance of the attack target; 2) check the tracking of the attack object is the same as the original video with \( CheckFit(\cdot) \). If both conditions are satisfied, we try to find an object that overlaps most with object \( ID_{att} \) as the screener object \( ID_{scr} \) by \( FindMaxIoU(\cdot) \). Then we check whether the IoU between objects \( ID_{att} \) and \( ID_{scr} \) is greater than \( Thr_{IoU} \). If true, the current frame will be attacked, and otherwise not.

For the attacking, we use the \( NoiseGenerator(\cdot) \) to generate adversarial noise by optimizing Eq. 10 iteratively until the tracker makes mistakes in the current frame or the iterations reach \( Thr_{iter} \) (60 in default). Note that during the iteration, point \( c \rightarrow \hat{c}_k \) will leap to the next grid closer to \( c_k \) at certain number of iterations as presented in Sec. 3.4. The noise is then added to the original frame by Eq. 11. We add the noise to the current frame no matter the attack succeeds or not, and the experiments in Sec. 4.3 show that such operation contributes to an easier attack for the following frames. The tracklet pool \( \hat{TrP}_t \) is then re-updated by the adversarial frame \( \hat{I}_t \), and the threshold \( Thr_{IoU} \) is set to zero. In the end, the adversarial frame \( \hat{I}_t \) is added to the adversarial video \( \hat{V} \).

### 4 Experiments

In this section, we first introduce the experiment settings. Then the results of TraSw and the baselines are proposed to demonstrate the effectiveness and efficiency of TraSw. Next we analyze the importance of components in TraSw through the ablation study. Finally, we discuss the influence of \( Thr_{IoU} \) and present the distribution area of adversarial noise.

#### 4.1 Experimental Setup

**Target Models and Datasets.** We choose two representative trackers as the target models: FairMOT [36] and ByteTrack [35]. FairMOT is also used for ablation studies. In particular, we only use \textit{CenterLeaping} to attack ByteTrack, as there is no re-ID branch in ByteTrack. We validate TraSw on the test sets of three benchmarks: 2DMOT15 [18], MOT17 [24] and MOT20 [7].

**Baselines.** TraSw is compared with three classic baselines: 1) random noise perturbation (denoted as RanAt) whose \( L_2 \) distance per frame is limited to [2, 8] randomly; 2) detection attack (denoted as DetAt) which aims to make the attack object invisible to the object detection module [22,34,12,6] (commonly
used in object detection and SOT attacks); 3) *tracker hijacking* attack (denoted as Hijack) [16].

**Evaluation Metric.** Similar to [16], we call a successful attack when the detected objects of the attack trajectory are no longer associated with the original tracklet after the attack. As described in Sec. 3.5, our method attacks when an object overlaps with the attack object. So the attack success rate depends on two factors. Firstly, we need to obtain the number of trajectories satisfying the attack conditions: 1) the trajectory’s object should have appeared for at least $Thr_{frame}$ (10 as the default value) frames; 2) there exists another object overlapping with the attack object, and the IoU should be greater than $Thr_{IoU}$ (0.2 as the default value). Secondly, we need to get the number of successfully attacked trajectories for which the detected bboxes are no longer associated with the original tracklets after the attack. For comparison purposes, the attack conditions of the baselines are the same as TraSw. The effectiveness and efficiency of our method are demonstrated through the attack success rate ($Succ. \uparrow$), attacked frames (#$Fm. \downarrow$), and $L_2$ distance ($L_2 \downarrow$) per track ID of the successful attacks. To make the attack more efficient, the attacked frames are limited to 20 at most.

### 4.2 Adversarial Attack Results

We report the results of TraSw and the baselines on the 2DMOT15, MOT17, and MOT20 test datasets in Tab. 1. Column $IDS_{att}$ is the number of the attackable trajectories on the given tracker and dataset. From the experimental results, we can observe that: 1) compared with the baselines, TraSw achieves the highest $Succ.$ on the three datasets with fewer frames #$Fm.$ and smaller perturbations $L_2$. 2) TraSw outperforms Hijack on the pedestrian multi-object tracking. 3) Generally, in the crowded pedestrian tracking scenarios (MOT20), TraSw can achieve a better success rate than the normal scenarios (2DMOT15 and MOT17) while DetAt and Hijack have no significant improvement. In summary, the results indicate the effectiveness of TraSw in attacking the pedestrian multi-object trackers.

In addition, we also compare the attack results under the constraints of attacked frames and $L_2$ distance (see Fig. 4). Compared with the baselines, TraSw
Table 1: Comparison of the attacks on the MOT-Challenge test datasets

| Dataset  | Tracker | Method | Succ. ↑ (%) | #Fm. ↓ | L₂ ↓ | IDs att |
|----------|---------|--------|-------------|--------|------|--------|
| 2DMOT15  | FairMOT | RanAt  | 5.25        | 8.74   | 43.85| 820    |
|          |         | Hijack | 92.92       | 6.57   | 21.74|        |
|          |         | TraSw  | **93.28**   | **4.25**| **13.99**|        |
|          | ByteTrack | RanAt | 4.80        | 6.83   | 34.14|        |
|          |         | DetAt  | 85.09       | 5.98   | 36.65|        |
|          |         | Hijack | 81.89       | 6.85   | 38.57|        |
|          |         | TraSw  | **89.88**   | **4.09**| **26.49**|        |
| MOT17    | FairMOT | RanAt  | 3.19        | 7.57   | 37.50| 658    |
|          |         | DetAt  | 82.83       | 7.84   | 23.07|        |
|          |         | Hijack | 79.18       | 8.00   | 24.39|        |
|          |         | TraSw  | **91.03**   | **4.74**| **14.40**|        |
|          | ByteTrack | RanAt | 4.26        | 7.73   | 40.14| 705    |
|          |         | DetAt  | 70.35       | 6.04   | 27.81|        |
|          |         | Hijack | 67.53       | 6.95   | 32.31|        |
|          |         | TraSw  | **91.06**   | **4.17**| **24.02**|        |
| MOT20    | FairMOT | RanAt  | 14.53       | 6.77   | 34.17| 1892   |
|          |         | DetAt  | 83.99       | 6.81   | 16.31|        |
|          |         | Hijack | 84.04       | 7.09   | 17.88|        |
|          |         | TraSw  | **96.46**   | **3.94**| **11.67**|        |
|          | ByteTrack | RanAt | 4.27        | 7.46   | 37.07| 1899   |
|          |         | DetAt  | 61.93       | 7.32   | 33.63|        |
|          |         | Hijack | 58.66       | 8.18   | 30.25|        |
|          |         | TraSw  | **94.84**   | **3.46**| **19.54**|        |

Fig. 5: Distances of features and boxes in the original video and adversarial video. $d_{feat}$ and $d_{box}$ represent the cosine similarity of features and the Mahalanobis distance of boxes, as presented in Sec. 3.1. $\hat{T}_i (\hat{O}_i)$ and $T_i (O_i)$ denote the adversarial and original tracklets (objects)

yields higher success rates at certain attacked frames and $L_2$ distance constraints on the three datasets.
To better understand TraSw, we provide analysis on the attacking instance shown in Fig. 1 for the 19-th tracklet (24-th tracklet is the screener object). Specifically, we plot the feature distance and box distance trends in Fig. 5. The blue lines indicate the distance between tracklet 19 and the object with the original ID of 19. The red lines indicate the distance between tracklet 19 and the object with the original ID of 24. The solid lines and the dotted lines indicate the trend curve in the adversarial video and the original video. We can observe that the solid lines coincide with the dotted lines before the attack, after the attack, the red line and the blue line of the adversarial video switch, and tracklet 19 is instead similar to object 24. The original intention of updating tracklets at each time step is to adapt to variations of the tracked objects, but the update mechanism of tracklets allows us to tamper with the tracklets so that the tracker doesn’t realize that it is tracking a completely different object.

Table 2: Attack success rates at certain attack frames. PP, CL and FN represent the PullPush, CenterLeaping and Adding Failure Noise, respectively. \( n_{\#Fm.} \) presents attacking at most \( n \) frames.

| Dataset | Attackers | Attack success rate ↑ (%) |
|---------|-----------|---------------------------|
|         |           | 1\#Fm. | 5\#Fm. | 10\#Fm. | 15\#Fm. | 20\#Fm. |
| 2DMOT15 | TraSw w/o PP | 11.4   | 61.1   | 83.5    | 90.2    | 93.2    |
|         | TraSw w/o CL | 11.2   | 67.1   | 87.6    | 91.6    | 92.8    |
|         | TraSw w/o FN | 17.8   | 66.8   | 82.9    | 85.6    | 87.2    |
|         | TraSw       | 14.4   | **70.5** | **87.8** | **92.1** | **93.3** |
| MOT17   | TraSw w/o PP | 9.7    | 51.1   | 75.0    | 85.9    | 90.0    |
|         | TraSw w/o CL | 8.8    | 38.5   | 64.7    | 74.6    | 81.2    |
|         | TraSw w/o FN | 15.5   | 62.3   | 79.3    | 85.9    | 89.1    |
|         | TraSw       | 14.0   | **63.4** | **81.8** | **89.5** | **91.0** |
| MOT20   | TraSw w/o PP | 13.0   | 64.6   | 86.8    | 93.2    | 94.0    |
|         | TraSw w/o CL | 11.2   | 50.8   | 74.5    | 82.2    | 84.8    |
|         | TraSw w/o FN | 20.9   | 72.3   | 88.8    | 92.9    | 93.3    |
|         | TraSw       | 19.4   | **75.1** | **91.5** | **95.8** | **96.5** |

### 4.3 Ablation Study

Here we discuss the necessity of PushPull, CenterLeaping, and adding failure noises. The comparisons are conducted on FairMOT to analyze the contribution of each component in TraSw. The results are in Tab. 2. We can observe that:

1) The performance of TraSw without PP/CL is much worse than the overall TraSw when only a few frames attacked (i.e., 1\#Fm. and 5\#Fm.), especially on MOT17 and MOT20. It proves that PullPush and CenterLeaping working together can significantly reduce attacked frames. 2) The performance of TraSw without FN is greatly suppressed as the attacked frames increase. It shows that adding failure noise can make the later frames easier to attack, especially on 2DMOT15. 3) On MOT17 and MOT20, the attack success rate of TraSw without CL drops more than without PP, which indicates that CenterLeaping can
further improve the attack success rate in crowded scenarios. It seems that the bounding box matching mechanism of the association algorithm, which is widely used in MOT, is more vulnerable to attack. In summary, the three components significantly contribute to the performance of TraSw.

4.4 Parameter Study on IoU Threshold

One of the TraSw’s attack conditions is that there is another object overlapping with the attack object, and the IoU is greater than the threshold $Thr_{IoU}$. It seems that the attack condition makes TraSw relatively limited (i.e., a few objects satisfy the attack condition). However, according to our analysis in Fig. 6, in most scenarios, the attackable objects occupy 80% under the experimental $Thr_{IoU}$ of 0.2; it is even as high as 90% in the dense scenarios (MOT20). Therefore, our approach can adapt to most pedestrian tracking scenarios.

4.5 Noise Pattern

As shown in Fig. 7, five attack tracklets are randomly selected from five videos to show the distribution area of noise. The blue boxes and red boxes represent the attack and screener objects. The noise mainly focuses on the attack and screener objects, leaving other regions almost not perturbed. It demonstrates that TraSw can focus on key areas and attack them efficiently.

5 Conclusion

This is the first work to study the adversarial attack against pedestrian MOT trackers. The proposed adversarial attack method TraSw, which consists of the PushPull and the CenterLeaping techniques, can efficiently deceive the advanced MOT trackers at a high success rate. Exploiting the update mechanism of tracklets to attack the MOT trackers, TraSw also proves the weakness of the association algorithm in MOT. Empiric experiments on standard benchmarks show that our method outperforms the previous attack methods on object detection. Considering that this is the first adversarial study on pedestrian MOT, we also provide the comparison with tracker hijacker, which is focused on vehicle MOT, for reference.

The study of adversarial examples is of great importance to the model’s robustness. In future work, we will follow up this direction and continue to explore the following problems: 1) MOT often needs to handle the complex data association problem. Instead of attacking the model with a simple negative optimization method, we will explore the vulnerability of MOT models from
different perspectives. 2) We hope that the attack method proposed in this paper could inspire more works in designing robust MOT trackers. We will also continue to design adversarial defense methods to improve the robustness of existing MOT models.

6 Acknowledgements*

This work is supported by National Natural Science Foundation (62076105) and International Cooperation Foundation of Hubei Province, China(2021EHB011).
References

1. Bao, J.: Sparse adversarial attack to object detection. arXiv preprint arXiv:2012.13692 (2020)
2. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). pp. 3464–3468 (2016)
3. Candamo, J., Shreve, M., Goldgof, D.B., Sapper, D.B., Kasturi, R.: Understanding transit scenes: A survey on human behavior-recognition algorithms. IEEE Transactions on Intelligent Transportation Systems 11(1), 206–224 (2009)
4. Chen, L., Ai, H., Zhuang, Z., Shang, C.: Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: 2018 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6 (2018)
5. Chen, S.T., Cornelius, C., Martin, J., Chau, D.H.P.: Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In: Machine Learning and Knowledge Discovery in Databases. pp. 52–68 (2019)
6. Chen, X., Yan, X., Zheng, F., Jiang, Y., Xia, S.T., Zhao, Y., Ji, R.: One-shot adversarial attacks on visual tracking with dual attention. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 10176–10185 (2020)
7. Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L.: MOT20: A benchmark for multi object tracking in crowded scenes. arXiv preprint arXiv:2003.09003 (2020)
8. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: Centernet: Keypoint triplets for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 6569–6578 (2019)
9. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1625–1634 (2018)
10. Fischer, V., Kumar, M.C., Metzen, J.H., Brox, T.: Adversarial examples for semantic image segmentation. In: 5th International Conference on Learning Representations, ICLR 2017, Workshop Track Proceedings (2017)
11. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
12. Guo, Q., Li, Z., Xue, W., Feng, W.: Spark: Spatial-aware online incremental attack against visual tracking. In: European Conference on Computer Vision (ECCV) (2020)
13. Hendrik Metzen, J., Chaithanya Kumar, M., Brox, T., Fischer, V.: Universal adversarial perturbations against semantic image segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2755–2764 (2017)
14. Hu, Y.C.T., Kung, B.H., Tan, D.S., Chen, J.C., Hua, K.L., Cheng, W.H.: Naturalistic physical adversarial patch for object detectors. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7848–7857 (2021)
15. Jia, S., Song, Y., Ma, C., Yang, X.: IoU attack: Towards temporally coherent black-box adversarial attack for visual object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6709–6718 (2021)
16. Jia, Y., Lu, Y., Shen, J., Chen, Q.A., Zhong, Z., Wei, T.: Fooling detection alone is not enough: First adversarial attack against multiple object tracking. In: International Conference on Learning Representations (ICLR) (2020)
17. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. In: Artificial Intelligence Safety and Security, pp. 99–112. Chapman and Hall/CRC (2018)

18. Leal-Taixé, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: Towards a benchmark for multi-target tracking. arXiv preprint arXiv:1504.01942 (2015)

19. Li, B., Wu, W., Wang, Q., Zhang, F., Xing, J., Yan, J.: Siamrpn++: Evolution of siamese visual tracking with very deep networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4282–4291 (2019)

20. Li, B., Yan, J., Wu, W., Zhu, Z., Hu, X.: High performance visual tracking with siamese region proposal network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8971–8980 (2018)

21. Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., Hengel, A.V.D.: A survey of appearance models in visual object tracking. ACM Transactions on Intelligent Systems and Technology (TIST) 4(4), 1–48 (2013)

22. Lu, J., Sibai, H., Fabry, E.: Adversarial examples that fool detectors. arXiv preprint arXiv:1712.02494 (2017)

23. Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Kim, T.K.: Multiple object tracking: A literature review. Artificial Intelligence p. 103448 (2020)

24. Milan, A., Leal-Taixe, L., Reid, I., Roth, S., Schindler, K.: MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016)

25. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1765–1773 (2017)

26. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 427–436 (2015)

27. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 815–823 (2015)

28. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2014)

29. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: International Conference on Learning Representations (ICLR) (2014)

30. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. In: European Conference on Computer Vision (ECCV). pp. 107–122 (2020)

31. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 3645–3649 (2017)

32. Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., Yuille, A.: Adversarial examples for semantic segmentation and object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 1369–1378 (2017)

33. Xu, R., Nikouei, S.Y., Chen, Y., Polunchenko, A., Song, S., Deng, C., Faughnan, T.R.: Real-time human objects tracking for smart surveillance at the edge. In: 2018 IEEE International Conference on Communications (ICC). pp. 1–6 (2018)
34. Yan, B., Wang, D., Lu, H., Yang, X.: Cooling-shrinking attack: Blinding the tracker with imperceptible noises. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 990–999 (2020)
35. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Yuan, Z., Luo, P., Liu, W., Wang, X.: ByteTrack: Multi-object tracking by associating every detection box. arXiv preprint arXiv:2110.06864 (2021)
36. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: Fairmot: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision (IJCV) pp. 1–19 (2021)
37. Zhang, Z., Peng, H.: Deeper and wider siamese networks for real-time visual tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4591–4600 (2019)
38. Zhao, Y., Zhu, H., Liang, R., Shen, Q., Zhang, S., Chen, K.: Seeing isn’t believing: Towards more robust adversarial attack against real world object detectors. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. pp. 1989–2004 (2019)
39. Zhou, X., Koltun, V., Krähenbühl, P.: Tracking objects as points. In: European Conference on Computer Vision (ECCV). pp. 474–490 (2020)